Ultimate Software

Using Platform Configurable Fields in Interfaces

Interface Team

Last Updated:
6-21-2017

Contents

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)

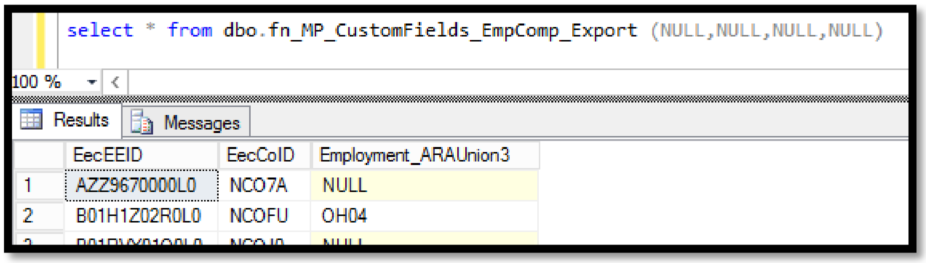
Functional Guide

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).***

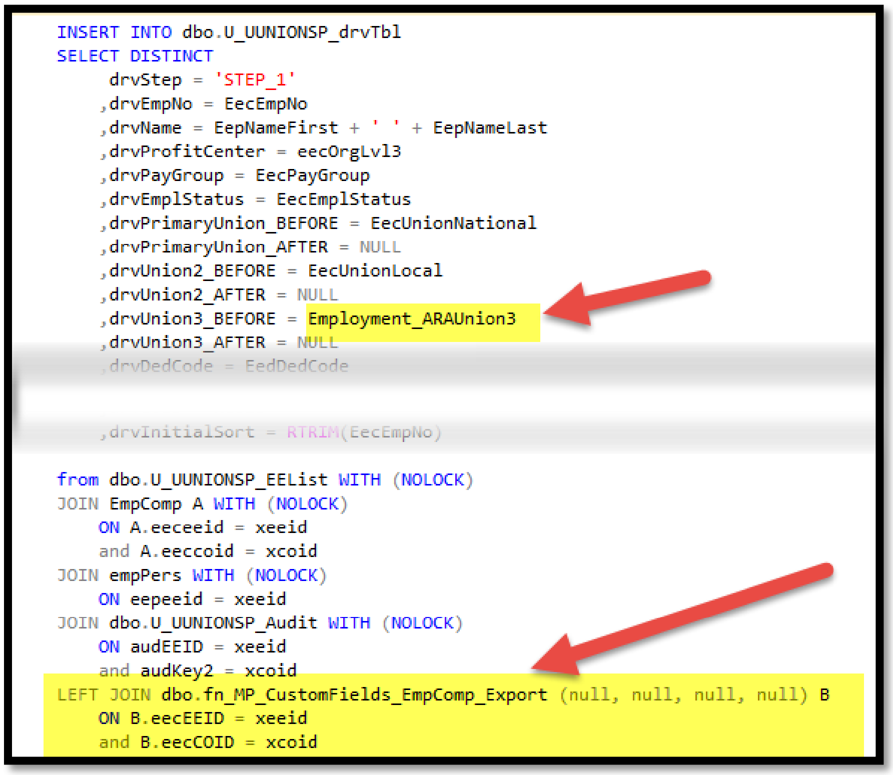


```
select * from dbo.fn_MP_CustomFields_EmpComp_Export (NULL,NULL,NULL,NULL)
```

	EecEEID	EecCoID	Employment_ARAUnion3
1	AZZ9670000L0	NCO7A	NULL
2	B01H1Z02R0L0	NCOFU	OH04
3	[illegible]	[illegible]	[illegible]

Example: PCF Table-Value Function & EmpComp

In the example below, the function ***dbo.fn_MP_CustomFields_EmpComp_Export*** is being joined using EecEEID & EecCOID.



```
INSERT INTO dbo.U_UUNIONSP_drvTbl
SELECT DISTINCT
     drvStep = 'STEP_1'
    ,drvEmpNo = EecEmpNo
    ,drvName = EepNameFirst + ' ' + EepNameLast
    ,drvProfitCenter = eecOrgLvl3
    ,drvPayGroup = EecPayGroup
    ,drvEmplStatus = EecEmplStatus
    ,drvPrimaryUnion_BEFORE = EecUnionNational
    ,drvPrimaryUnion_AFTER = NULL
    ,drvUnion2_BEFORE = EecUnionLocal
    ,drvUnion2_AFTER = NULL
    ,drvUnion3_BEFORE = Employment_ARAUnion3
    ,drvUnion3_AFTER = NULL
    ,drvDedCode = EedDedCode


    ,drvInitialSort = RTRIM(EecEmpNo)

from dbo.U_UUNIONSP_EEList WITH (NOLOCK)
JOIN EmpComp A WITH (NOLOCK)
    ON A.eeceeid = xeeid
    and A.eeccoid = xcoid
JOIN empPers WITH (NOLOCK)
    ON eepeeid = xeeid
JOIN dbo.U_UUNIONSP_Audit WITH (NOLOCK)
    ON audEEID = xeeid
    and audKey2 = xcoid
LEFT JOIN dbo.fn_MP_CustomFields_EmpComp_Export (null, null, null, null) B
    ON B.eecEEID = xeeid
    and B.eecCOID = xcoid
```

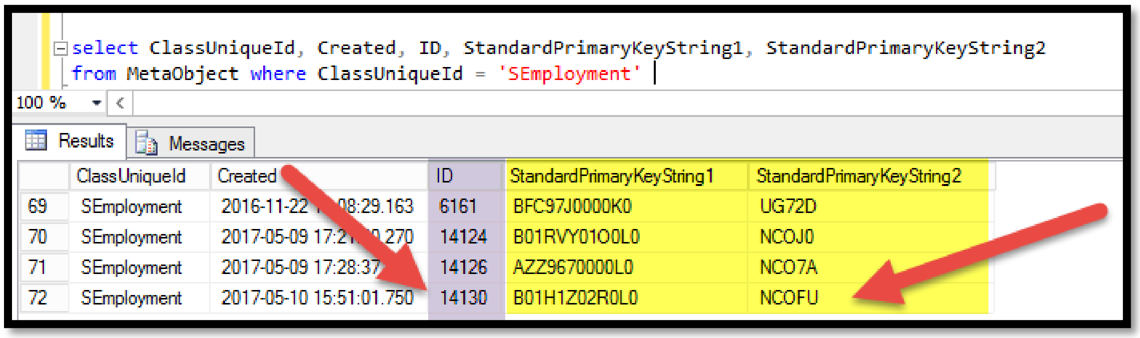
Deeper Dive: What are the underlying system tables accessed by the Table-Value functions?

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.



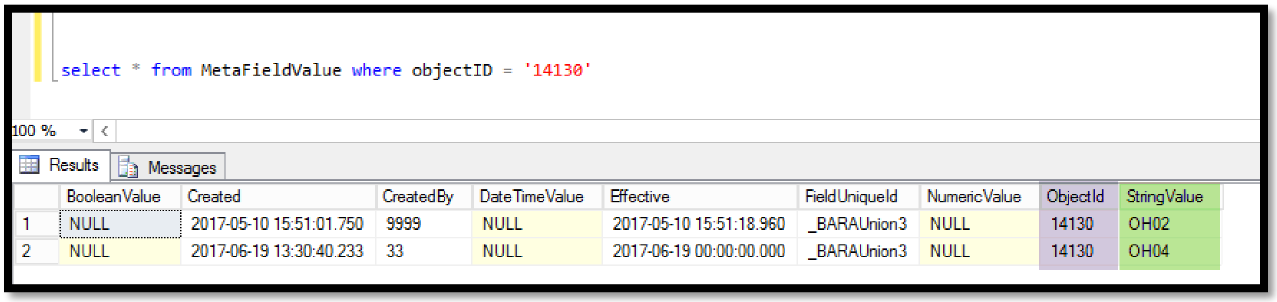
```
select ClassUniqueId, Created, ID, StandardPrimaryKeyString1, StandardPrimaryKeyString2
from MetaObject where ClassUniqueId = 'SEmployment'
```

	ClassUniqueId	Created	ID	StandardPrimaryKeyString1	StandardPrimaryKeyString2
69	SEmployment	2016-11-22 [illegible]8:29.163	6161	BFC97J0000K0	UG72D
70	SEmployment	2017-05-09 17:2[illegible].270	14124	B01RVY01O0L0	NCOJ0
71	SEmployment	2017-05-09 17:28:37[illegible]	14126	AZZ9670000L0	NCO7A
72	SEmployment	2017-05-10 15:51:01.750	14130	B01H1Z02R0L0	NCOFU

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.[*]
  - 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.



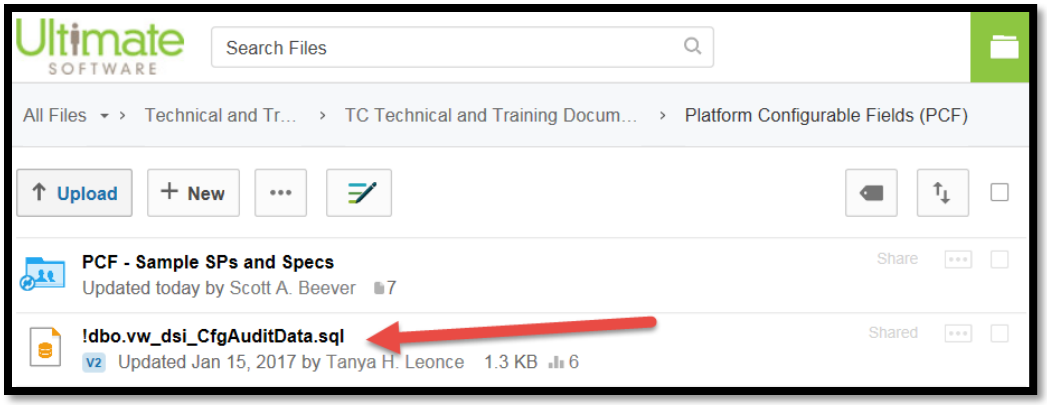
```
select * from MetaFieldValue where objectID = '14130'
```

	BooleanValue	Created	CreatedBy	DateTimeValue	Effective	FieldUniqueId	NumericValue	ObjectId	StringValue
1	NULL	2017-05-10 15:51:01.750	9999	NULL	2017-05-10 15:51:18.960	_BARAUnion3	NULL	14130	OH02
2	NULL	2017-06-19 13:30:40.233	33	NULL	2017-06-19 00:00:00.000	_BARAUnion3	NULL	14130	OH04

[*] 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.

PCF Audit View

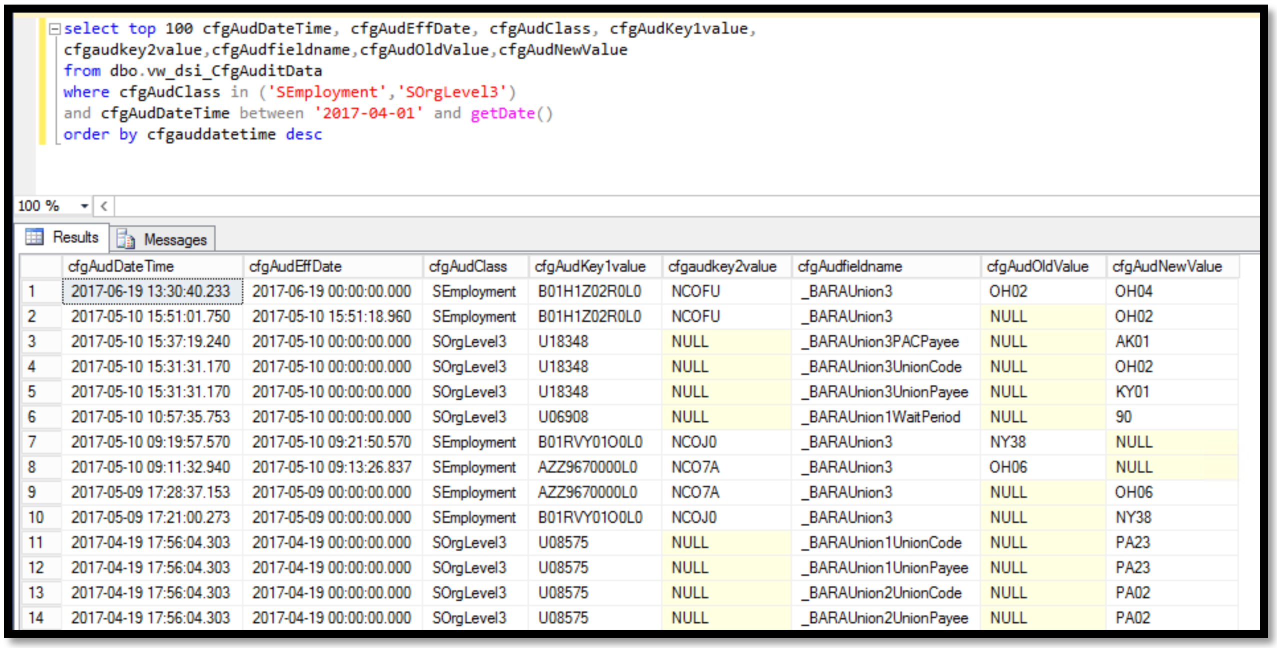
A view was been developed (*vw_dsi_CfgAuditData*) to assist with viewing changes to data in each PCF. The view can be retrieved from the PCF folder in Box & deployed to the customer's environment as needed (https://ulti.box.com/s/expib2vyrdr6h7fyottnnon66zbz84ld).


Search Files
All Files › Technical and Tr... › TC Technical and Training Docum... › Platform Configurable Fields (PCF)
Upload
New
PCF - Sample SPs and Specs
Updated today by Scott A. Beever 7
Share
!dbo.vw_dsi_CfgAuditData.sql
V2 Updated Jan 15, 2017 by Tanya H. Leonce 1.3 KB 6
Shared

- 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.

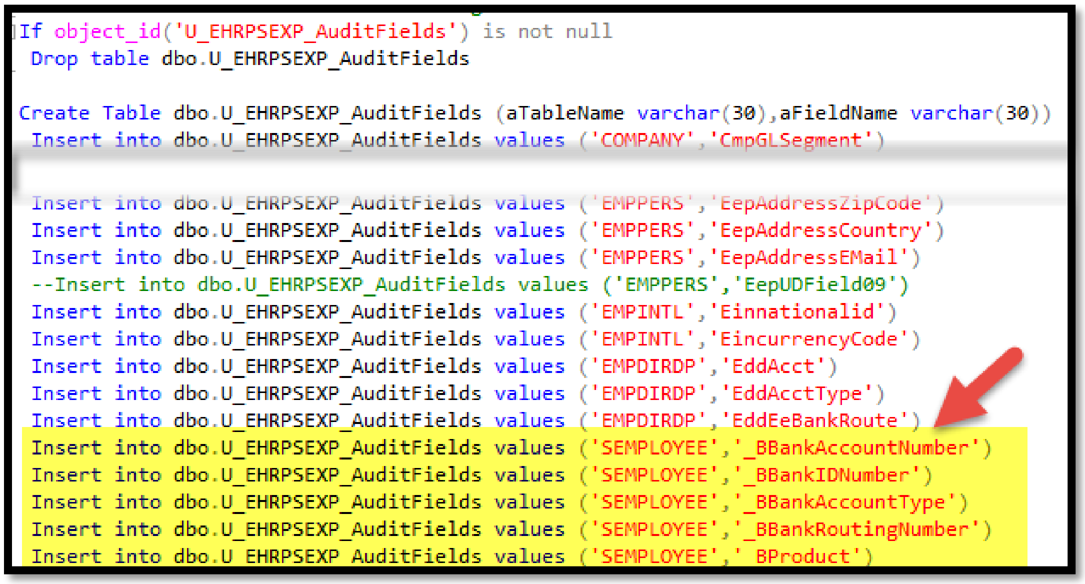


```
select top 100 cfgAudDateTime, cfgAudEffDate, cfgAudClass, cfgAudKey1value,
cfgaudkey2value,cfgAudfieldname,cfgAudOldValue,cfgAudNewValue
from dbo.vw_dsi_CfgAuditData
where cfgAudClass in ('SEmployment','SOrgLevel3')
and cfgAudDateTime between '2017-04-01' and getDate()
order by cfgauddatetime desc
```

	cfgAudDateTime	cfgAudEffDate	cfgAudClass	cfgAudKey1value	cfgaudkey2value	cfgAudfieldname	cfgAudOldValue	cfgAudNewValue
1	2017-06-19 13:30:40.233	2017-06-19 00:00:00.000	SEmployment	B01H1Z02R0L0	NCOFU	_BARAUnion3	OH02	OH04
2	2017-05-10 15:51:01.750	2017-05-10 15:51:18.960	SEmployment	B01H1Z02R0L0	NCOFU	_BARAUnion3	NULL	OH02
3	2017-05-10 15:37:19.240	2017-05-10 00:00:00.000	SOrgLevel3	U18348	NULL	_BARAUnion3PACPayee	NULL	AK01
4	2017-05-10 15:31:31.170	2017-05-10 00:00:00.000	SOrgLevel3	U18348	NULL	_BARAUnion3UnionCode	NULL	OH02
5	2017-05-10 15:31:31.170	2017-05-10 00:00:00.000	SOrgLevel3	U18348	NULL	_BARAUnion3UnionPayee	NULL	KY01
6	2017-05-10 10:57:35.753	2017-05-10 00:00:00.000	SOrgLevel3	U06908	NULL	_BARAUnion1WaitPeriod	NULL	90
7	2017-05-10 09:19:57.570	2017-05-10 09:21:50.570	SEmployment	B01RVY01O0L0	NCOJ0	_BARAUnion3	NY38	NULL
8	2017-05-10 09:11:32.940	2017-05-10 09:13:26.837	SEmployment	AZZ9670000L0	NCO7A	_BARAUnion3	OH06	NULL
9	2017-05-09 17:28:37.153	2017-05-09 00:00:00.000	SEmployment	AZZ9670000L0	NCO7A	_BARAUnion3	NULL	OH06
10	2017-05-09 17:21:00.273	2017-05-09 00:00:00.000	SEmployment	B01RVY01O0L0	NCOJ0	_BARAUnion3	NULL	NY38
11	2017-04-19 17:56:04.303	2017-04-19 00:00:00.000	SOrgLevel3	U08575	NULL	_BARAUnion1UnionCode	NULL	PA23
12	2017-04-19 17:56:04.303	2017-04-19 00:00:00.000	SOrgLevel3	U08575	NULL	_BARAUnion1UnionPayee	NULL	PA23
13	2017-04-19 17:56:04.303	2017-04-19 00:00:00.000	SOrgLevel3	U08575	NULL	_BARAUnion2UnionCode	NULL	PA02
14	2017-04-19 17:56:04.303	2017-04-19 00:00:00.000	SOrgLevel3	U08575	NULL	_BARAUnion2UnionPayee	NULL	PA02

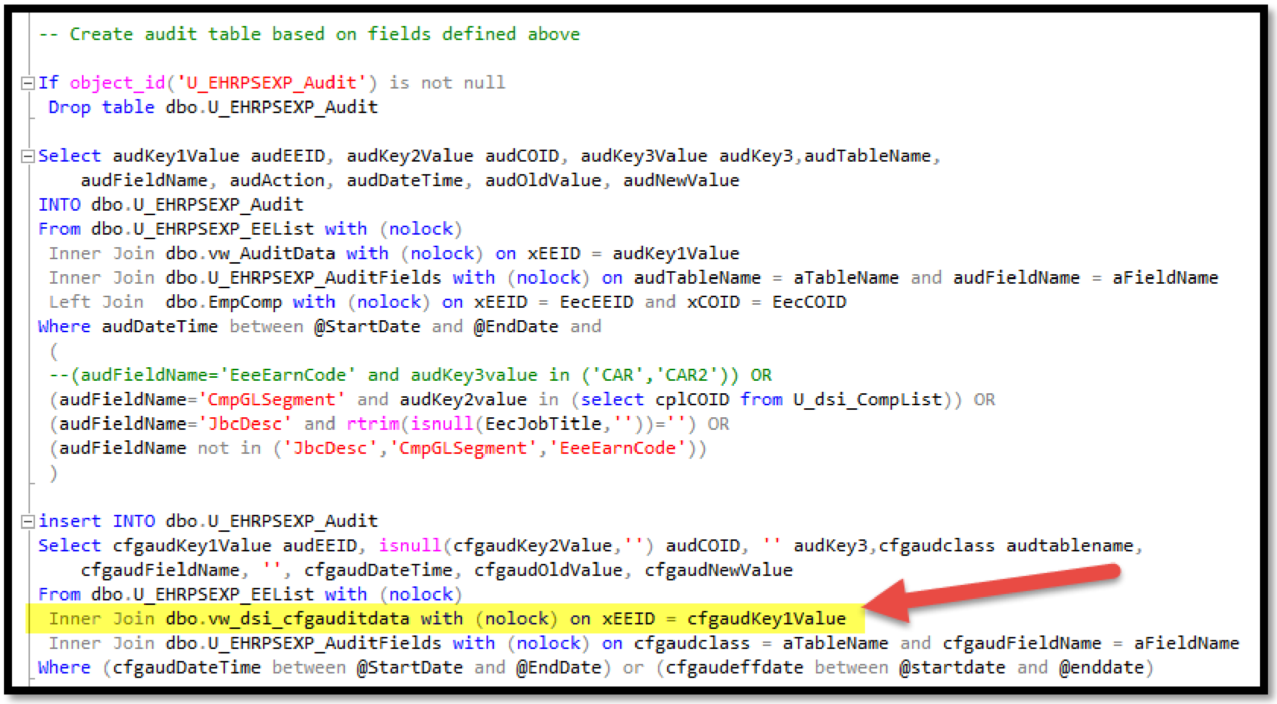
Example: Using the PCF Audit View in a Changes-Only Export

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**".



```
If object_id('U_EHRPSEXP_AuditFields') is not null
 Drop table dbo.U_EHRPSEXP_AuditFields

Create Table dbo.U_EHRPSEXP_AuditFields (aTableName varchar(30),aFieldName varchar(30))
 Insert into dbo.U_EHRPSEXP_AuditFields values ('COMPANY','CmpGLSegment')

 Insert into dbo.U_EHRPSEXP_AuditFields values ('EMPPERS','EepAddressZipCode')
 Insert into dbo.U_EHRPSEXP_AuditFields values ('EMPPERS','EepAddressCountry')
 Insert into dbo.U_EHRPSEXP_AuditFields values ('EMPPERS','EepAddressEMail')
 --Insert into dbo.U_EHRPSEXP_AuditFields values ('EMPPERS','EepUDField09')
 Insert into dbo.U_EHRPSEXP_AuditFields values ('EMPINTL','Einnationalid')
 Insert into dbo.U_EHRPSEXP_AuditFields values ('EMPINTL','EincurrencyCode')
 Insert into dbo.U_EHRPSEXP_AuditFields values ('EMPDIRDP','EddAcct')
 Insert into dbo.U_EHRPSEXP_AuditFields values ('EMPDIRDP','EddAcctType')
 Insert into dbo.U_EHRPSEXP_AuditFields values ('EMPDIRDP','EddEeBankRoute')
 Insert into dbo.U_EHRPSEXP_AuditFields values ('SEMPLOYEE','_BBankAccountNumber')
 Insert into dbo.U_EHRPSEXP_AuditFields values ('SEMPLOYEE','_BBankIDNumber')
 Insert into dbo.U_EHRPSEXP_AuditFields values ('SEMPLOYEE','_BBankAccountType')
 Insert into dbo.U_EHRPSEXP_AuditFields values ('SEMPLOYEE','_BBankRoutingNumber')
 Insert into dbo.U_EHRPSEXP_AuditFields values ('SEMPLOYEE','_BProduct')
```

2. 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.



```sql
-- Create audit table based on fields defined above

If object_id('U_EHRPSEXP_Audit') is not null
 Drop table dbo.U_EHRPSEXP_Audit

Select audKey1Value audEEID, audKey2Value audCOID, audKey3Value audKey3,audTableName,
    audFieldName, audAction, audDateTime, audOldValue, audNewValue
INTO dbo.U_EHRPSEXP_Audit
From dbo.U_EHRPSEXP_EEList with (nolock)
 Inner Join dbo.vw_AuditData with (nolock) on xEEID = audKey1Value
 Inner Join dbo.U_EHRPSEXP_AuditFields with (nolock) on audTableName = aTableName and audFieldName = aFieldName
 Left Join  dbo.EmpComp with (nolock) on xEEID = EecEEID and xCOID = EecCOID
Where audDateTime between @StartDate and @EndDate and
 (
 --(audFieldName='EeeEarnCode' and audKey3value in ('CAR','CAR2')) OR
 (audFieldName='CmpGLSegment' and audKey2value in (select cplCOID from U_dsi_CompList)) OR
 (audFieldName='JbcDesc' and rtrim(isnull(EecJobTitle,''))='') OR
 (audFieldName not in ('JbcDesc','CmpGLSegment','EeeEarnCode'))
 )

insert INTO dbo.U_EHRPSEXP_Audit
Select cfgaudKey1Value audEEID, isnull(cfgaudKey2Value,'') audCOID, '' audKey3,cfgaudclass audtablename,
    cfgaudFieldName, '', cfgaudDateTime, cfgaudOldValue, cfgaudNewValue
From dbo.U_EHRPSEXP_EEList with (nolock)
 Inner Join dbo.vw_dsi_cfgauditdata with (nolock) on xEEID = cfgaudKey1Value
 Inner Join dbo.U_EHRPSEXP_AuditFields with (nolock) on cfgaudclass = aTableName and cfgaudFieldName = aFieldName
Where (cfgaudDateTime between @StartDate and @EndDate) or (cfgaudeffdate between @startdate and @enddate)
```

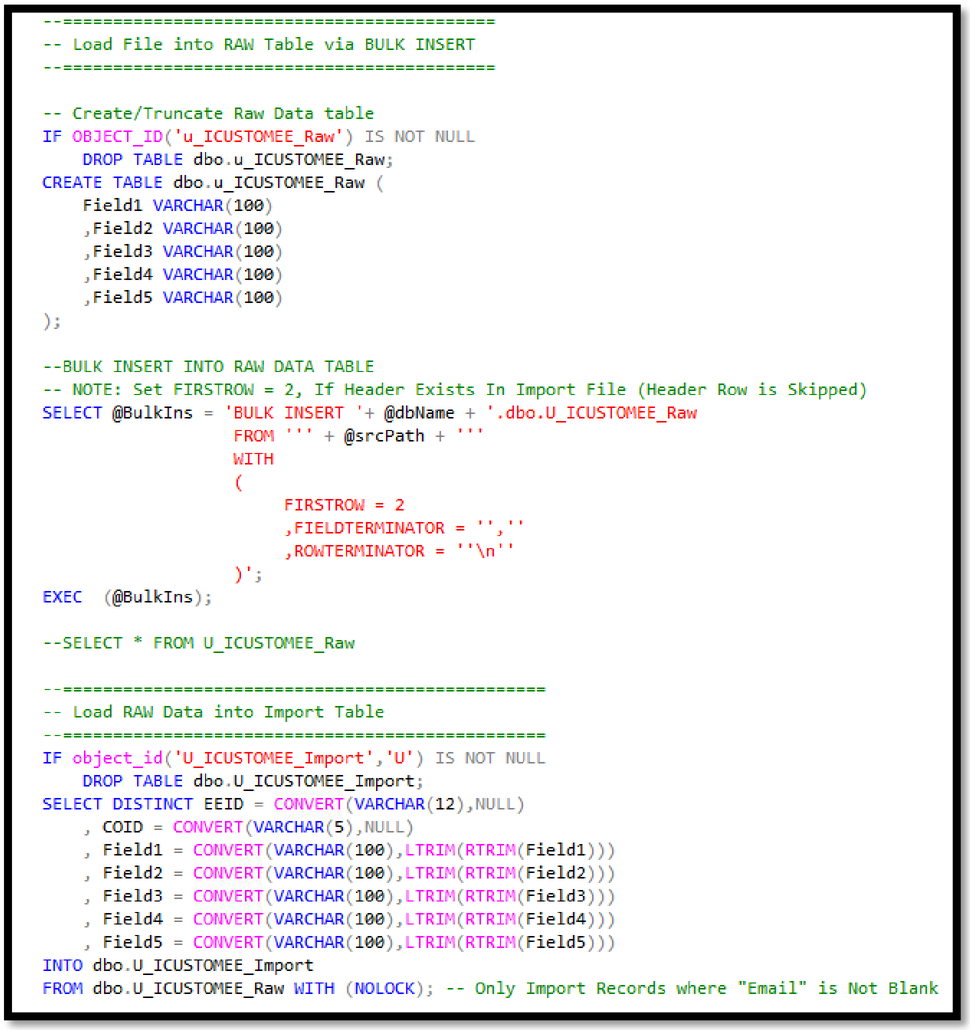
PCF Imports

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

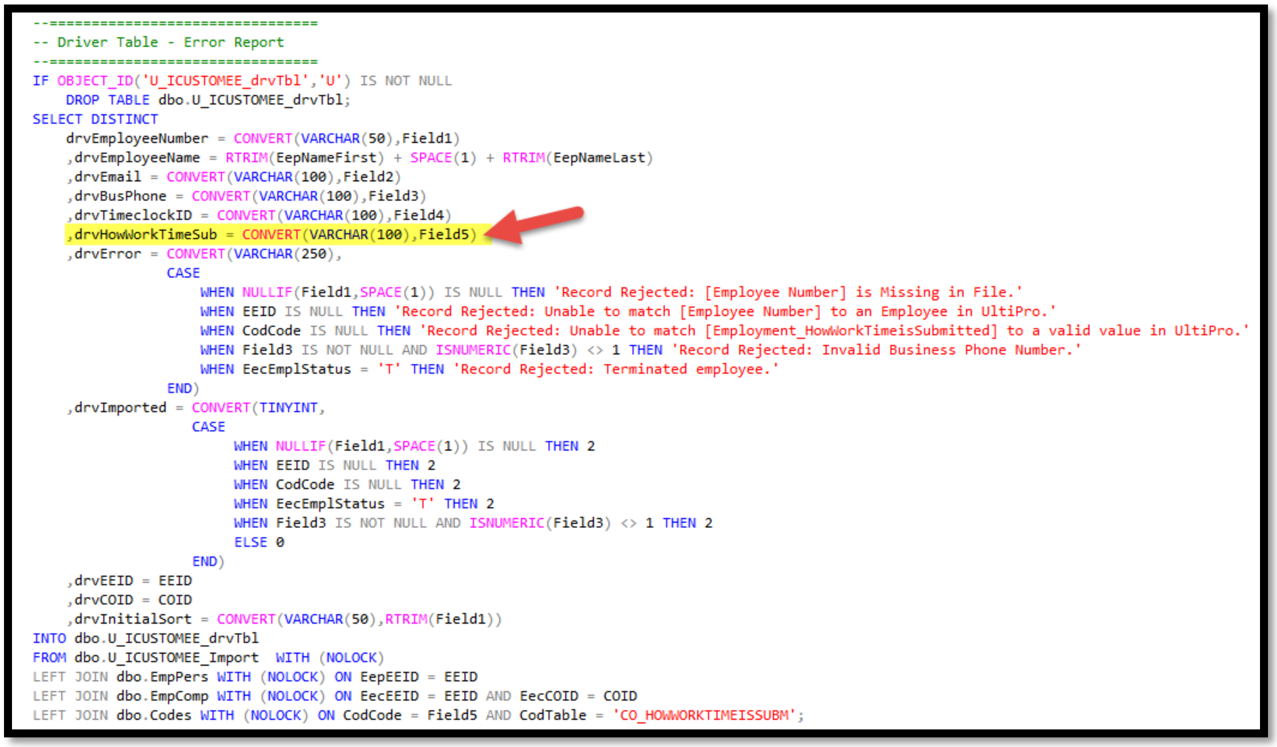
Example: PCF Imports

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.



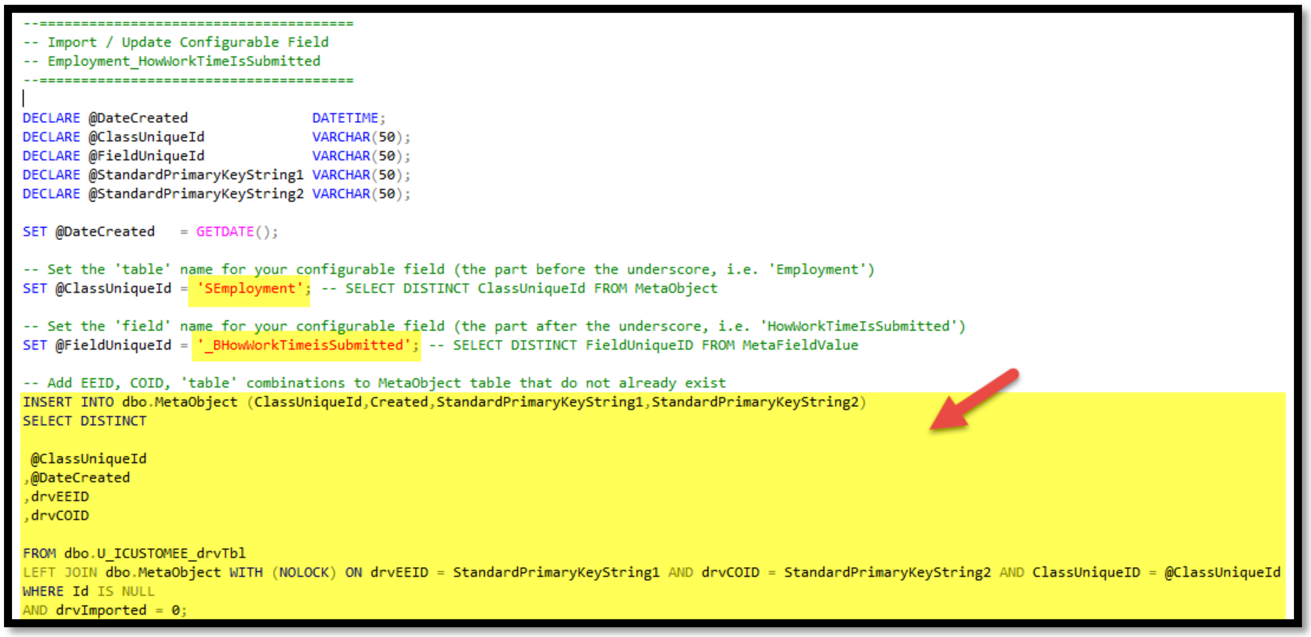
```
--=============================================
-- Load File into RAW Table via BULK INSERT
--=============================================

-- Create/Truncate Raw Data table
IF OBJECT_ID('u_ICUSTOMEE_Raw') IS NOT NULL
    DROP TABLE dbo.u_ICUSTOMEE_Raw;
CREATE TABLE dbo.u_ICUSTOMEE_Raw (
    Field1 VARCHAR(100)
    ,Field2 VARCHAR(100)
    ,Field3 VARCHAR(100)
    ,Field4 VARCHAR(100)
    ,Field5 VARCHAR(100)
);

--BULK INSERT INTO RAW DATA TABLE
-- NOTE: Set FIRSTROW = 2, If Header Exists In Import File (Header Row is Skipped)
SELECT @BulkIns = 'BULK INSERT '+ @dbName + '.dbo.U_ICUSTOMEE_Raw
                   FROM ''' + @srcPath + '''
                   WITH
                   (
                        FIRSTROW = 2
                        ,FIELDTERMINATOR = '','''
                        ,ROWTERMINATOR = ''\n''
                   )';
EXEC  (@BulkIns);

--SELECT * FROM U_ICUSTOMEE_Raw

--==================================================
-- Load RAW Data into Import Table
--==================================================
IF object_id('U_ICUSTOMEE_Import','U') IS NOT NULL
    DROP TABLE dbo.U_ICUSTOMEE_Import;
SELECT DISTINCT EEID = CONVERT(VARCHAR(12),NULL)
    , COID = CONVERT(VARCHAR(5),NULL)
    , Field1 = CONVERT(VARCHAR(100),LTRIM(RTRIM(Field1)))
    , Field2 = CONVERT(VARCHAR(100),LTRIM(RTRIM(Field2)))
    , Field3 = CONVERT(VARCHAR(100),LTRIM(RTRIM(Field3)))
    , Field4 = CONVERT(VARCHAR(100),LTRIM(RTRIM(Field4)))
    , Field5 = CONVERT(VARCHAR(100),LTRIM(RTRIM(Field5)))
INTO dbo.U_ICUSTOMEE_Import
FROM dbo.U_ICUSTOMEE_Raw WITH (NOLOCK); -- Only Import Records where "Email" is Not Blank
```



```sql
--=================================
-- Driver Table - Error Report
--=================================
IF OBJECT_ID('U_ICUSTOMEE_drvTbl','U') IS NOT NULL
    DROP TABLE dbo.U_ICUSTOMEE_drvTbl;
SELECT DISTINCT
    drvEmployeeNumber = CONVERT(VARCHAR(50),Field1)
    ,drvEmployeeName = RTRIM(EepNameFirst) + SPACE(1) + RTRIM(EepNameLast)
    ,drvEmail = CONVERT(VARCHAR(100),Field2)
    ,drvBusPhone = CONVERT(VARCHAR(100),Field3)
    ,drvTimeclockID = CONVERT(VARCHAR(100),Field4)
    ,drvHowWorkTimeSub = CONVERT(VARCHAR(100),Field5)
    ,drvError = CONVERT(VARCHAR(250),
                CASE
                    WHEN NULLIF(Field1,SPACE(1)) IS NULL THEN 'Record Rejected: [Employee Number] is Missing in File.'
                    WHEN EEID IS NULL THEN 'Record Rejected: Unable to match [Employee Number] to an Employee in UltiPro.'
                    WHEN CodCode IS NULL THEN 'Record Rejected: Unable to match [Employment_HowWorkTimeisSubmitted] to a valid value in UltiPro.'
                    WHEN Field3 IS NOT NULL AND ISNUMERIC(Field3) <> 1 THEN 'Record Rejected: Invalid Business Phone Number.'
                    WHEN EecEmplStatus = 'T' THEN 'Record Rejected: Terminated employee.'
                END)
    ,drvImported = CONVERT(TINYINT,
                   CASE
                        WHEN NULLIF(Field1,SPACE(1)) IS NULL THEN 2
                        WHEN EEID IS NULL THEN 2
                        WHEN CodCode IS NULL THEN 2
                        WHEN EecEmplStatus = 'T' THEN 2
                        WHEN Field3 IS NOT NULL AND ISNUMERIC(Field3) <> 1 THEN 2
                        ELSE 0
                   END)
    ,drvEEID = EEID
    ,drvCOID = COID
    ,drvInitialSort = CONVERT(VARCHAR(50),RTRIM(Field1))
INTO dbo.U_ICUSTOMEE_drvTbl
FROM dbo.U_ICUSTOMEE_Import  WITH (NOLOCK)
LEFT JOIN dbo.EmpPers WITH (NOLOCK) ON EepEEID = EEID
LEFT JOIN dbo.EmpComp WITH (NOLOCK) ON EecEEID = EEID AND EecCOID = COID
LEFT JOIN dbo.Codes WITH (NOLOCK) ON CodCode = Field5 AND CodTable = 'CO_HOWWORKTIMEISSUBM';
```

2) Insert records into the **MetaObject** table for objects do not already exist in this table.

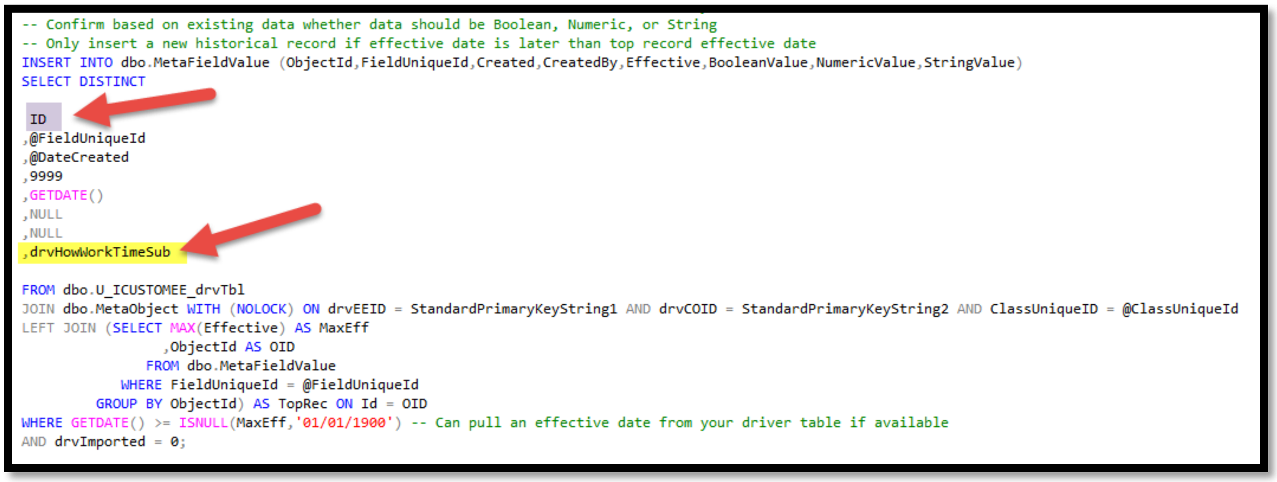
- Remember that the Class names will need to have the prefix "**S**" & the field names will have the prefix "**_B**".
- In this example, **MetaObject** records are being inserted for records where the EEID/COID combination does not already exist in the **Employment** class.



```sql
--======================================
-- Import / Update Configurable Field
-- Employment_HowWorkTimeIsSubmitted
--======================================

DECLARE @DateCreated                 DATETIME;
DECLARE @ClassUniqueId               VARCHAR(50);
DECLARE @FieldUniqueId               VARCHAR(50);
DECLARE @StandardPrimaryKeyString1 VARCHAR(50);
DECLARE @StandardPrimaryKeyString2 VARCHAR(50);

SET @DateCreated    = GETDATE();

-- Set the 'table' name for your configurable field (the part before the underscore, i.e. 'Employment')
SET @ClassUniqueId = 'SEmployment'; -- SELECT DISTINCT ClassUniqueId FROM MetaObject

-- Set the 'field' name for your configurable field (the part after the underscore, i.e. 'HowWorkTimeIsSubmitted')
SET @FieldUniqueId = '_BHowWorkTimeisSubmitted'; -- SELECT DISTINCT FieldUniqueID FROM MetaFieldValue

-- Add EEID, COID, 'table' combinations to MetaObject table that do not already exist
INSERT INTO dbo.MetaObject (ClassUniqueId,Created,StandardPrimaryKeyString1,StandardPrimaryKeyString2)
SELECT DISTINCT

 @ClassUniqueId
,@DateCreated
,drvEEID
,drvCOID

FROM dbo.U_ICUSTOMEE_drvTbl
LEFT JOIN dbo.MetaObject WITH (NOLOCK) ON drvEEID = StandardPrimaryKeyString1 AND drvCOID = StandardPrimaryKeyString2 AND ClassUniqueID = @ClassUniqueId
WHERE Id IS NULL
AND drvImported = 0;
```

3) Insert records into **MetaFieldValue** table for the PCF field that is being updated.

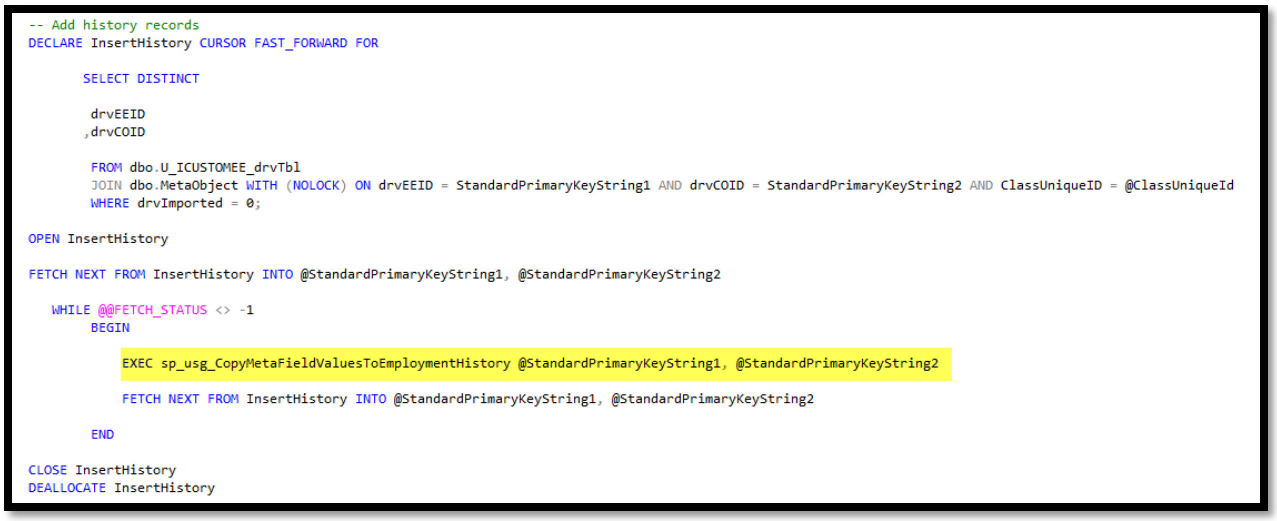
- There is a new record in **MetaFieldValue** for the PCF each time it is changed/updated, so we will always be inserting rows into **MetaFieldValue** with a new effective date, not updating existing rows.
- The data must be inserted into the correct field in **MetaFieldValue**, based on the data type: *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.



```
-- Confirm based on existing data whether data should be Boolean, Numeric, or String
-- Only insert a new historical record if effective date is later than top record effective date
INSERT INTO dbo.MetaFieldValue (ObjectId,FieldUniqueId,Created,CreatedBy,Effective,BooleanValue,NumericValue,StringValue)
SELECT DISTINCT

 ID
,@FieldUniqueId
,@DateCreated
,9999
,GETDATE()
,NULL
,NULL
,drvHowWorkTimeSub

FROM dbo.U_ICUSTOMEE_drvTbl
JOIN dbo.MetaObject WITH (NOLOCK) ON drvEEID = StandardPrimaryKeyString1 AND drvCOID = StandardPrimaryKeyString2 AND ClassUniqueID = @ClassUniqueId
LEFT JOIN (SELECT MAX(Effective) AS MaxEff
                 ,ObjectId AS OID
                FROM dbo.MetaFieldValue
             WHERE FieldUniqueId = @FieldUniqueId
          GROUP BY ObjectId) AS TopRec ON Id = OID
WHERE GETDATE() >= ISNULL(MaxEff,'01/01/1900') -- Can pull an effective date from your driver table if available
AND drvImported = 0;
```

4) For inserts into the **Employment** Class only:

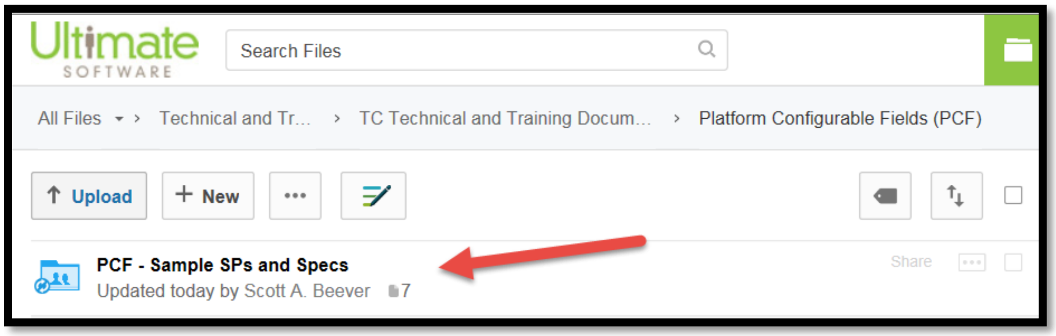
- There is an **EmploymentHistory** class that goes along with the **Employment** Class.
    - **Employment** is similar to **EmpComp**
    - **EmploymentHistory** is similar to **EmpHJob**
- As such, we need to update the **EmploymentHistory** PCF when the **Employment** PCF is updated. Luckily, there 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, …)



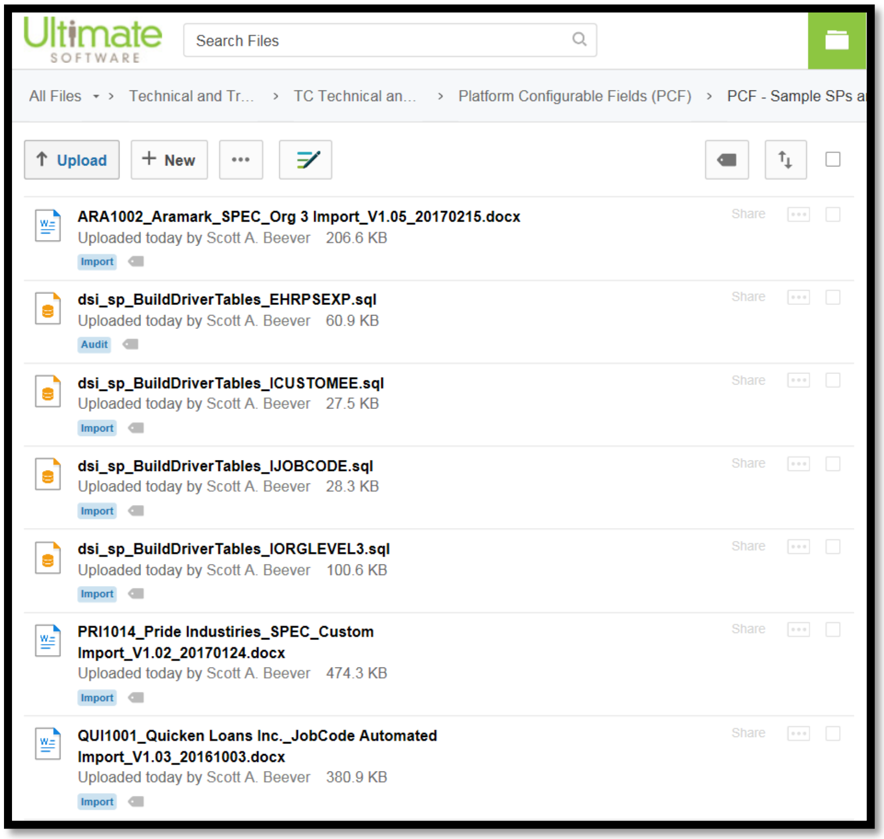
```
-- Add history records
DECLARE InsertHistory CURSOR FAST_FORWARD FOR

       SELECT DISTINCT

        drvEEID
       ,drvCOID

        FROM dbo.U_ICUSTOMEE_drvTbl
        JOIN dbo.MetaObject WITH (NOLOCK) ON drvEEID = StandardPrimaryKeyString1 AND drvCOID = StandardPrimaryKeyString2 AND ClassUniqueID = @ClassUniqueId
        WHERE drvImported = 0;

OPEN InsertHistory

FETCH NEXT FROM InsertHistory INTO @StandardPrimaryKeyString1, @StandardPrimaryKeyString2

   WHILE @@FETCH_STATUS <> -1
        BEGIN

            EXEC sp_usg_CopyMetaFieldValuesToEmploymentHistory @StandardPrimaryKeyString1, @StandardPrimaryKeyString2

            FETCH NEXT FROM InsertHistory INTO @StandardPrimaryKeyString1, @StandardPrimaryKeyString2

        END

CLOSE InsertHistory
DEALLOCATE InsertHistory
```

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



PCF - Sample SPs and Specs
Updated today by Scott A. Beever 7



ARA1002_Aramark_SPEC_Org 3 Import_V1.05_20170215.docx
Uploaded today by Scott A. Beever 206.6 KB
Import

dsi_sp_BuildDriverTables_EHRPSEXP.sql
Uploaded today by Scott A. Beever 60.9 KB
Audit

dsi_sp_BuildDriverTables_ICUSTOMEE.sql
Uploaded today by Scott A. Beever 27.5 KB
Import

dsi_sp_BuildDriverTables_IJOBCODE.sql
Uploaded today by Scott A. Beever 28.3 KB
Import

dsi_sp_BuildDriverTables_IORGLEVEL3.sql
Uploaded today by Scott A. Beever 100.6 KB
Import

PRI1014_Pride Industries_SPEC_Custom Import_V1.02_20170124.docx
Uploaded today by Scott A. Beever 474.3 KB
Import

QUI1001_Quicken Loans Inc._JobCode Automated Import_V1.03_20161003.docx
Uploaded today by Scott A. Beever 380.9 KB
Import

Applications in Interfaces

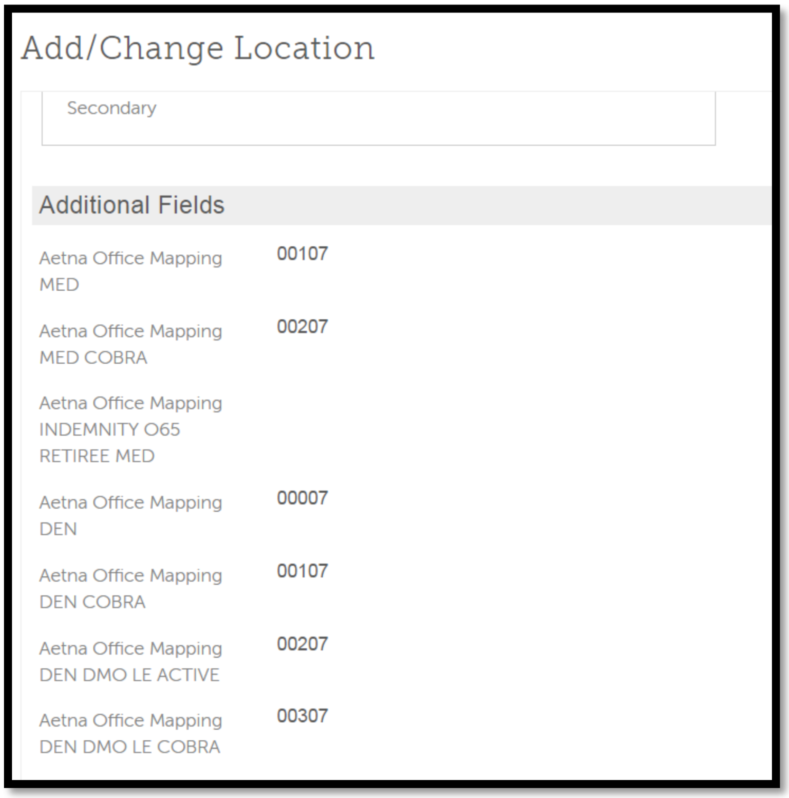
Benefit Plan/Group Mappings

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.

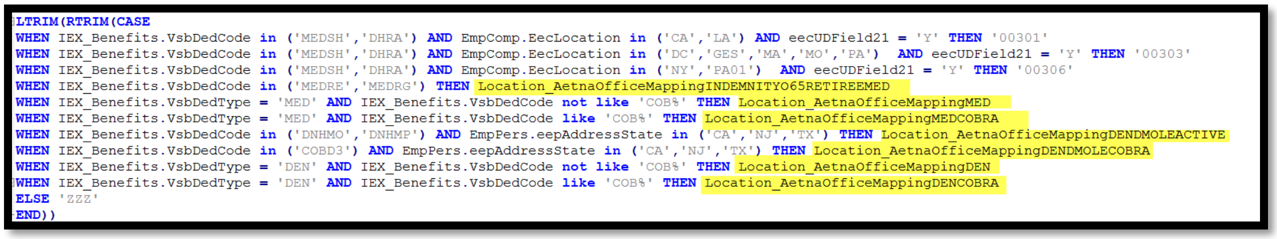
- They decided to add a series of PCFs in the **Location** class, so that these can easily be updated by the end user when a new Location is added or needed to be updated.
- The interface programming is then looking to these PCF values, rather than hardcoding these mappings
  - This will save the customer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


Add/Change Location

Secondary

Additional Fields

Aetna Office Mapping MED	00107
Aetna Office Mapping MED COBRA	00207
Aetna Office Mapping INDEMNITY O65 RETIREE MED	
Aetna Office Mapping DEN	00007
Aetna Office Mapping DEN COBRA	00107
Aetna Office Mapping DEN DMO LE ACTIVE	00207
Aetna Office Mapping DEN DMO LE COBRA	00307



```sql
LTRIM(RTRIM(CASE
WHEN IEX_Benefits.VsbDedCode in ('MEDSH','DHRA') AND EmpComp.EecLocation in ('CA','LA') AND eecUDField21 = 'Y' THEN '00301'
WHEN IEX_Benefits.VsbDedCode in ('MEDSH','DHRA') AND EmpComp.EecLocation in ('DC','GES','MA','MO','PA')  AND eecUDField21 = 'Y' THEN '00303'
WHEN IEX_Benefits.VsbDedCode in ('MEDSH','DHRA') AND EmpComp.EecLocation in ('NY','PA01')  AND eecUDField21 = 'Y' THEN '00306'
WHEN IEX_Benefits.VsbDedCode in ('MEDRE','MEDRG') THEN Location_AetnaOfficeMappingINDEMNITYO65RETIREEMED
WHEN IEX_Benefits.VsbDedType = 'MED' AND IEX_Benefits.VsbDedCode not like 'COB%' THEN Location_AetnaOfficeMappingMED
WHEN IEX_Benefits.VsbDedType = 'MED' AND IEX_Benefits.VsbDedCode like 'COB%' THEN Location_AetnaOfficeMappingMEDCOBRA
WHEN IEX_Benefits.VsbDedCode in ('DNHMO','DNHMP') AND EmpPers.eepAddressState in ('CA','NJ','TX') THEN Location_AetnaOfficeMappingDENDMOLEACTIVE
WHEN IEX_Benefits.VsbDedCode in ('COBD3') AND EmpPers.eepAddressState in ('CA','NJ','TX') THEN Location_AetnaOfficeMappingDENDMOLECOBRA
WHEN IEX_Benefits.VsbDedType = 'DEN' AND IEX_Benefits.VsbDedCode not like 'COB%' THEN Location_AetnaOfficeMappingDEN
WHEN IEX_Benefits.VsbDedType = 'DEN' AND IEX_Benefits.VsbDedCode like 'COB%' THEN Location_AetnaOfficeMappingDENCOBRA
ELSE 'ZZZ'
END))
```

- 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:

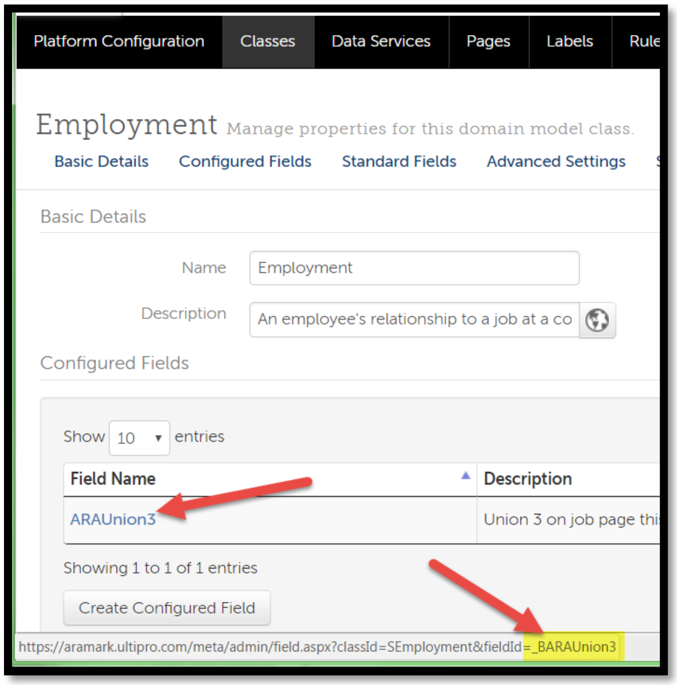
```
select distinct classuniqueID, fielduniqueId
FROM dbo.MetaFieldValue
    JOIN dbo.MetaObject
        ON ObjectID = ID
```

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.



Employment Manage properties for this domain model class.

Basic Details

Name: Employment

Description: An employee's relationship to a job at a co

Configured Fields

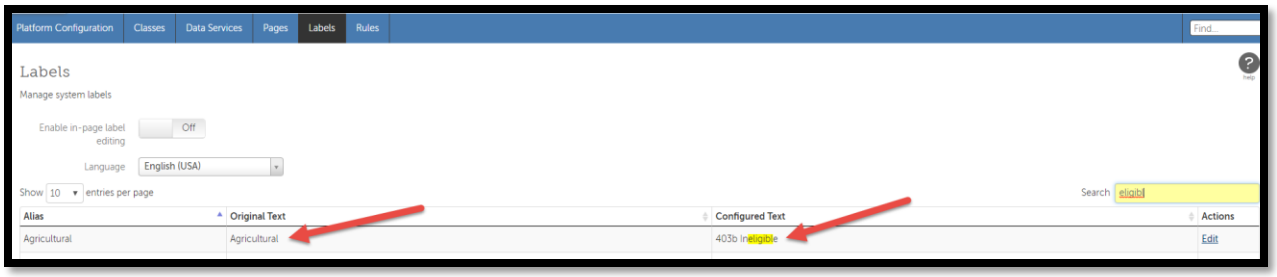
Show 10 entries

Field Name	Description
ARAUnion3	Union 3 on job page thi

Showing 1 to 1 of 1 entries

https://aramark.ultipro.com/meta/admin/field.aspx?classId=SEmployment&fieldId=_BARAUnion3

2) 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.



Labels

Manage system labels

Enable in-page label editing: Off

Language: English (USA)

Show 10 entries per page

Search: eligibl

Alias	Original Text	Configured Text	Actions
Agricultural	Agricultural	403b Ineligible	Edit

3) 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