Data Services Benefits Deduction Module Guide
Version 1.30

Contents

Overview 5
Quick Start 6
IMPORTANT – DO NOT IN ANY WAY MODIFY ANY OF THE BDM OBJECTS 7
FAQ – What if I need to clean the EELIST 7
FAQ – I need to only report employees with changes to their Deduction 7
FAQ – What if I need to reference the effective Dated Tables in my driver SP 8
Build a Single Consolidated Table for Just Your Formatcode 9
What if I want to add additional custom fields to the consolidated table? 9
Selecting Deduction Codes 10
What if the customer wants to select deductions by type rather than dedcode? 10
What if I'm selecting deduction types, but just need to exclude a few deduction codes 10
The customer wants all deductions that start with MM, DD, and if the run is on a Sunday during a lunar eclipse in an odd-numbered year during a debate on the debt ceiling, ZZ. 11
I need to change the deduction type. (Used for FSA plans to give different types to Health and Dependent Care) 13
I only need employee deductions. 13
I only need dependent deductions. 13

I need both, but the employee and dependent deductions are not the same. 13
The module returns only valid deduction per type, but my export needs valid deductions within the same type. 14
Need an Open Enrollment Run? 15
What if they're on .NET? 15
What if I want to exclude termed deductions from Open Enrollment? 15
Excluding termed deductions from Open Enrollment is super but… My client has Severance employees with Terminated employment statuses and Active deductions. Can I drop OE deductions that have been stopped only? 16
The customer is bypassing the normal OE process and just using the pending tables, or a third-party loaded the OE data into the pending tables. What now? 16
Domestic Partners 17
Some of my deductions are domestic partner deductions. 17
For employees, the module discards the DP deduction and processes the regular deduction, but for this customer it's reversed: they want to process the DP deduction for employees. 17
Future-Dated Stops and Starts 18
The vendor can't take any records with future-dated start dates. 18
They can take future-dated start dates, but only within a certain range. 18
The vendor can't take any records with future-dated stop dates. 18
They can take future-dated stop dates, but only within a certain range. 18
Benefit Change Reasons and Change Dates 19
I need to know the latest benefit change reason from EmpHDed. 19
The vendor wants to know the date the employee entered the benefit tier, e.g. the date they went from employee-only to employee-plus-one 19
They actually want the above date as the Eed/Dbn start date field. Should I copy it over? 19
I need two separate runs for my export, but the BDM wipes out my first run. 20

How can I tell the difference between my runs? They both have the same formatcode. 20
Counting Dependents 21
It would be nice if I could count the types of dependents associated with each employee deduction. 21
Miscellaneous Parameters 22
It would also be nice if I could limit the module to one or two employees for testing. 22
I need the module to return only terms 22
My export is based on deduction stop dates, not benefit stop dates 22
I have quite a few bad dependent deductions… active where the employee is terminated, or the dependent has a deduction the employee doesn't 23
Programmer-defined Fields 24
Cobra 25
To set the Cobra type and trigger the Cobra module 25
To set the Cobra event date 26
Option: Pull COBRA Reasons/Dates from Platform Configurable Fields, rather than Back Office Fields 27
Specify the Cobra Reasons to Use 28
Specify the Cobra Reasons Not to Use 28
Specify the COBRA Reasons where the Dependent can be identified as PQB 28
Indicate that Employee Termination Reasons should override Cobra Reasons 28
Manually Populate the Cobra Reasons Table 29
Determine Whether to Pull DedCodes Based on DedIsCobraCovered = Y 29
Calculate the PQB 30
Calculate Multiple PQBs (more than one member under an employee) 31
Must Set the Relationships 31

Validate employees for Cobra if there's a dependent on Cobra 32
New Enrollees 33
Set the New Enrollee type and trigger the New Enrollee module: 33
Allow only spouses and/or domestic partners: 33
Don't include dependent enrollees if the employee is valid 34
Validate employees as new enrollees if there's a dependent enrollee 34
Some Standard BDM Queries 35
Error Checking 36
Appendix: BDM Version History 37
BDM v1.30 Changes/Enhancements Summary 37
BDM v1.29 Changes/Enhancements Summary 37
BDM v1.28 Changes/Enhancements Summary 37
BDM v1.27 Changes/Enhancements Summary 38

Overview

The benefits deduction module (BDM) is designed to standardize and simplify how benefit deductions are processed in exports. It has the following functionality:

1. Return employee deductions, dependent deductions, or both.
2. BDM deduction tables include fields from EmpDed, DepBPlan and DedCode tables.
3. Select deduction based on dedcodes, dedtypes, or a custom selection
4. Select terms based on audit, on Eed/DbnBenStopDate within two dates, or just return all actives
5. Automatically return the most current deduction in case of company transfer
6. Handles waives correctly
7. Process active or passive Open Enrollment (OE)
8. Process ASP or .net Open Enrollment
9. Calculate the benefit option date, i.e. the date the employee first moved into a certain benefit tier
10. Process OE using pending sessions (e.g. PENDING_BI) if the customer wishes to load their OE using pending transactions
11. Select a separate subset of dependent deductions
12. Process domestic partner deductions automatically; TC can choose whether to keep the pre- or post-tax deduction
13. Exclude any deductions with future-dated start dates, or with start dates outside a certain range
14. Set all deductions with future-dated stop dates to active, or with future-dated stop dates within a certain range
15. Count dependents for each deduction: spouses, children, domestic partners, and domestic partner children
16. Flags invalid deductions (e.g. waived, termed before date range) with the reason they're invalid, making troubleshooting easier.
17. Flag Cobra deductions
18. Flag newly-enrolled deductions

Please note - The BDM is for benefit election exports. If you are creating a payroll contribution export, you want to be sure to include all of the contributions for the pay period as per the statement of work. If you use the BDM and an employee stopped coverage but still had a contribution, the ValidForExport flag could be set to 'N' and their record would not be sent to the vendor.

Quick Start

Paste the following code into your stored procedure:

```
DELETE FROM dbo.U_dsi_bdm_Configuration WHERE FormatCode = @FormatCode

INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'DedCodes', '<your deduction codes>')
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'StartDateTime', GETDATE() - 7)
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'EndDateTime', GETDATE())
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'TermSelectionOption', 'AuditDate')

EXEC dbo.dsi_bdm_sp_PopulateDeductionsTable @FormatCode
```

1. Set `<your deduction codes>` to your comma-delimited deduction codes, e.g. `'HMO, PPO'`
2. Set the start and end date times as necessary for an on-demand or scheduled run.
3. Set the TermSelectionOption to:
    a. `AuditDate` if you want all terms keyed within the range (most common)
    b. `StopDate` if you want all terms with Eed/DbnBenStopDates within the range
    c. `ActiveOnly` if you want active deductions only

Build your driver tables and add the following join for employees. Set the string below in the Where clause.

```
JOIN dbo.U_dsi_bdm_EmpDeductions ON EedCOID = xCOID AND EedEEID = xEEID

Where EedFormatCode = @FormatCode AND EedValidForExport = 'Y'
```

Add the following join for dependents:

```
JOIN dbo.U_dsi_bdm_DepDeductions ON DbnEEID = xEEID AND DbnDepRecID = ConSystemID

Where DbnFormatCode = @FormatCode AND DbnValidForExport = 'Y' AND
```

After running the BDM, execute the SP below to see what data-related issues may be affecting the output.

```
EXEC dbo.dsi_bdm_sp_ErrorCheck 'YOUR FORMATCODE'
```

That's it.

IMPORTANT – DO NOT IN ANY WAY MODIFY ANY OF THE BDM OBJECTS

Often times, the BDM objects will be installed with product releases; therefore any modifications you make to any objects within the install script will be overwritten.

FAQ – What if I need to clean the EELIST

The BDM module will use the EELIST. If you need to do any cleanup for the EELIST, you should do it prior to calling the module. Then data will only be returned for the employees that were in the EELIST.

Example:

```
Delete
  From dbo.U_[Your FormatCode]_EEList
 Where xCOID <> dbo.dsi_fn_GetCurrentCOID(xEEID)
   and xEEID in (Select xEEID From dbo.U_[Your FormatCode]_EEList Group By xEEID Having Count(*) > 1)
```

FAQ – I need to only report employees with changes to their Deduction

You can use the BDM module to bring back the employee and dependent data, and then within your driver table join to audit to determine which of those records you actually want to use. You could update the ValidForExport flag in the BDM tables and then pull where ValidForExport = 'Y', or join to a table that had the list of employees and ded-codes that were changed.

FAQ – What if I need to reference the effective Dated Tables in my driver SP

If you need to reference any of the effective dated tables in you code prior to calling the dsi_bdm_sp_PopulateDeductionsTable, please be sure to add this block of code at the beginning of your code. It will set the Connection Date so that for .NETOE, you can see the correct data as the BDM would.

The effective dated tables include – DEDTYPE, DEDCODE, BENPROG, EMPDED, DEPBPLAN, BNFBPLAN

```
   -- Set the effective date for .Net OE since dedcodes are being manually populated
if @exportcode like '%OE%'
BEGIN
  if (SELECT CASE WHEN isnull(dbo.dsi_bdm_fn_Trim(a.PrkProdCode),'') = 'NETOE' THEN '1' ELSE '2' END
     FROM Compmast
     LEFT JOIN RbsProductKeys a on a.PrkProdCode = 'NETOE') = '1'

  BEGIN
      EXEC MASTER.dbo.CXN_SET_SESSION_DATE @EndDate
   END
END
```

This code only needs to be included if you are accessing effective dated tables prior to calling BDM module. An example would be if you are building the dedcode tables in your driver sp.

Build a Single Consolidated Table for Just *Your* Formatcode

By default the BDM populates two separate tables for employee and dependent deductions. The TC needs to select from both tables and remember to pick up only valid deductions for his/her formatcode. To make things a little easier, the BDM offers an option to create a single consolidated table of all valid employee and dependent deduction records for your formatcode only.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'BuildConsolidatedTable', 'Standard')
```

This tells the BDM to build a table named u_dsi_bdm_<YourFormatCode>, e.g. **u_dsi_bdm_ECIGNA**.

The table includes the most-used fields for employees and dependents. You can also build the table any time in your code by issuing the following command:

```
EXEC dbo.dsi_bdm_sp_BuildConsolidatedTable 'yourformatcode', 'Standard'
```

What if I want to add additional custom fields to the consolidated table?

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'BdmAddField', '<bdmYourNewField>')
```

This tells the BDM to add a custom field with the name that has been provided.

- The prefix 'bdm' will be automatically added to the field name if it was not specified
- All custom fields are added with the data type **varchar(500)**.
- Multiple custom fields can be added by inserting multiple `'BdmAddField'` rows into `U_dsi_bdm_Configuration`.
- BDM will check to make sure the field names to be added do not already exist in the consolidated table. If they already exist, the fields with duplicate names will be skipped and a warning message is displayed.
- If the TC is running the BDM multiple times with the consolidated table in the same SP, the custom fields must be defined in the final BDM run for the SP. Otherwise the custom fields defined in earlier calls to the BDM would be lost in the final BDM execution for the SP.

Selecting Deduction Codes

What if the customer wants to select deductions by type rather than dedcode?

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'DedTypes', '<your deduction types>')
```

This will tell the BDM to select all dedcodes within the dedicated deduction types, e.g. `'MED, DEN'`

Note: if you use the **`DedCodes`** and **`DedTypes`** parameters, the BDM will select from both. This allows you to mix and match dedcodes and dedtypes within an export.

Note #2: the BDM previously used the **`SelectUsingDedTypes`** parameter, which told it to use the values in the DedCodes parameter as deduction types. This is still supported for backwards compatibility, but is deprecated. Use the **`DedTypes`** parameter instead.

What if I'm selecting deduction types, but just need to exclude a few deduction codes

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'ExcludeDedCodes', '<excluded deduction codes>')
```

This will tell the BDM to exclude the dedcodes in your list. Normally only used when deduction types are being selected and a few dedcodes need to be excluded.

The customer wants all deductions that start with MM, DD, and if the run is on a Sunday during a lunar eclipse in an odd-numbered year during a debate on the debt ceiling, ZZ.

First, delete or comment out the DedCodes and/or DedTypes parms in your code:

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'DedCodes', '<your deduction codes>')
```

Then set ManualDedCodes to Y:

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'ManualDedCodes', 'Y')
```

This tells the module you'll be manually populating the U_dsi_bdm_DeductionCodes table using your own code. Note that this table must have deduction codes in it or the module will error out. The following fields must be populated:

- UdcFormatCode : your export's format code
- UdcDedTarget: EMP for employee, DEP for dependent. Note that if you're pulling both employee and dependent dedcodes you must have an EMP entry and a DEP entry for each code.
- UdcDedCode : the deduction code
- UdcDedType: the deduction type from the DedType table
- UdcIsDomPartnerDed: Y indicates the deduction is a post-tax DP deduction. See the domestic partner section for more information.

The code on the next page was used to manually populate the deduction codes for an export, as the client needed the Medical deductions for the provider grouped by provider group #. When an employee switched from one group to another they needed a stop transaction for old group plan and the start for the new group plan. The group number was in the dednotes column and multiple codes could share a group in which only the active plan within a group was to be sent.

```
--Populate Deduction Codes

INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'ManualDedCodes', 'Y')

DELETE FROM dbo.U_dsi_bdm_DeductionCodes WHERE UdcFormatCode = @FormatCode

Insert INTO dbo.u_dsi_bdm_DeductionCodes (udcformatcode,udcdedtarget, udcdedcode, udcdedtype,
udcisdompartnerded)
 Select distinct
      @formatcode,
      'EMP',
      deddedcode,
      deddedtype,
      'N'
 From dbo.DedCode
 Where dedbenplanprovider = 'BCBSRI'

Insert INTO dbo.u_dsi_bdm_DeductionCodes (udcformatcode,udcdedtarget, udcdedcode, udcdedtype,
udcisdompartnerded)
 Select distinct
      @formatcode,
      'DEP',
      deddedcode,
      deddedtype,
      'N'
 From dbo.DedCode
 Where dedbenplanprovider = 'BCBSRI'

EXEC dbo.dsi_bdm_sp_PopulateDeductionsTable @FormatCode
```

*** **Remember** since you are referencing an effective dated table before calling the BDM, you must have the logic from page 6 prior to this blocks of code.

I need to change the deduction type. (Used for FSA plans to give different types to Health and Dependent Care)

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'UpdDedType', 'FSA')
```

The above statement will set the deduction type for all deductions with the FSA type equal to the deduction code

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'UpdDedCodeType', 'LTD,LTD50')
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'UpdDedCodeType', 'LTDPT,LTD50')
```

The above statements will set the dedtype for the dedcodes LTD and LTDPT to LTD50. The first element is the dedcode and the second is the dedtype you want the deduction code to be.

I only need employee deductions.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'TableType', 'EMP')
```

I only need dependent deductions.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'TableType', 'DEP')
```

I need both, but the employee and dependent deductions are not the same.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'DedCodesDep', '<comma-delimited list of dependent dedcodes')
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'DedTypesDep', '<comma-delimited list of dependent dedtypes')
```

The module will select employee deductions by using the standard DedCodes and/ or DedTypes parameters and dependent deductions using the DedCodesDep and/or DedTypesDep parameters.

The module returns only valid deduction per type, but my export needs valid deductions within the same type.

By default the BDM only returns the latest deduction within each type, e.g. medical. Occasionally we will need valid deductions within the same type, e.g. FSA or LTD. There are two ways to do this. The first is to set the below parm:

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'AllowMultiDedType', 'Y')
```

This will allow valid deductions within the same type.

The other way is to use the ManualDedCodes parameter that was previously discussed to load the deduction code table manually and override the type yourself. For example, say a customer has dedcodes LTD and LTDBU and both are type LTD. The module will return only the latest valid deduction, but the customer needs both. When you load the U_dsi_bdm_DeductionCodes table you can override the LTDBU type to LTDBU (or whatever you want). The module will then treat the dedcodes as two separate types.

Note that if the export has a mix of dedcodes, meaning some should only return the latest within type and some should allow multiple dedcodes within a type, you have to load the table manually and override the types for the applicable deductions.

Need an Open Enrollment Run?

Paste this code into your stored procedure:

```
--Set if OE
If @exportcode like '%PASSIVE%'
BEGIN
      INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'OEType', 'PASSIVE')
END

If @exportcode like '%ACTIVE%'
BEGIN
      INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'OEType', 'ACTIVE')
END
```

As long as you set up your OE sessions with PASSIVE or ACTIVE in the session, the export session run will control whether an Active or Passive OE run is to be used. We will not have to change from year to year to switch from Active to Passive or vice-versa

What if they're on .NET?

The module will automatically check Ultipro for the OEType and adjust the run accordingly; no action is needed.

What if I want to exclude termed deductions from Open Enrollment?

Paste this code into your stored procedure:

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'IncludeOEDrops', 'N')
```

All terminated deductions and employees will be dropped from the export.

Excluding termed deductions from Open Enrollment is super but… My client has Severance employees with Terminated employment statuses and Active deductions. Can I drop OE deductions that have been stopped only?

Paste this code into your stored procedure:

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'OERemoveTermEmps', 'N')
```

This feature will only work in accordance with the above parameter (`IncludeOEDrops`) being set to N. In this case Stop-Coverage records will be dropped and active coverage records will be sent regardless of the employee's status.

The customer is bypassing the normal OE process and just using the pending tables, or a third-party loaded the OE data into the pending tables. What now?

Paste this code into your stored procedure:

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'OESessions', '<customer's session codes, comma-delimited>')
```

Setting this parameter will tell the module to take the transactions directly from the pending tables using the supplied codes. If the customer just data-entered the pending transactions, the session code will be PENDING_BI. If the data was manually loaded, e.g. by a third party or an SC, the session codes might vary. Contact the customer. Multiple session codes can be entered in comma-delimited format.

Domestic Partners

Some of my deductions are domestic partner deductions.

For domestic partners the Ultipro standard is to assign the employee two deduction codes, a pre-tax "regular" deduction and a post-tax domestic partner deduction; the domestic partner only gets the post-tax domestic partner deduction. This means the employee will have two active deductions. By setting the DP deductions we tell the module to discard the employee's DP deduction and process just the regular deduction.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'DedCodesDP', '<comma-delimited list of domestic partner dedcodes')
```

Note: all deductions, including domestic partner deductions, must be entered in the DedCodes parameter. DP deductions will therefore be entered twice, once in the DedCodes parameter and once in the DedCodesDP parameter.

For employees, the module discards the DP deduction and processes the regular deduction, but for this customer it's reversed: they want to process the DP deduction for employees.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'DiscardNonDPDeduction', 'Y')
```

Future-Dated Stops and Starts

The vendor can't take any records with future-dated start dates.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'ExclFutureDatedStartDates', 'Y')
```

If set to Y, any deduction with a start date after the EndDateTime is discarded

They can take future-dated start dates, but only within a certain range.

Set the below variable and any start date sixty dates after the EndDateTime is discarded.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'FutureDatedStartDateDays', '60')
```

The vendor can't take any records with future-dated stop dates.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'ExclFutureDatedStopDates', 'Y')
```

If set to Y, any deduction with a stop date after the EndDateTime is set to active.

They can take future-dated stop dates, but only within a certain range.

Set the below variable and any stop date sixty dates after the EndDateTime is set to active.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'FutureDatedStopDateDays', '60')
```

Benefit Change Reasons and Change Dates

I need to know the latest benefit change reason from EmpHDed.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'GetChangeReason', 'Y')
```

This will tell the module to populate the EedChangeReason field in the U_dsi_bdm_EmpDeductions table.

The vendor wants to know the date the employee entered the benefit tier, e.g. the date they went from employee-only to employee-plus-one.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'CalcBenOptionDate', '1')
```

This will tell the module to populate the EedBenOptionDate field in the U_dsi_bdm_EmpDeductions table. If you need the date automatically copied to the DbnBenOptionDate field in the U_dsi_bdm_DepDeductions table, set it to 2:

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'CalcBenOptionDate', '2')
```

They actually want the above date as the Eed/Dbn start date field. Should I copy it over?

You can, or you can tell the module to do it for you:

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'OverrideBenStartDate', 'Y')
```

Creating Multiple Runs

I need two separate runs for my export, but the BDM wipes out my first run.

By default the BDM creates one run per formatcode, e.g. if you run it and then run it again with the same formatcode, your first run will be wiped out. In certain circumstances you might want to create multiple runs with different sets of parameter, e.g. a Cobra run and a Newly Enrolled run for the same export.

The BDM allows you to do this by preserving the previous run. So on run #2 you'd enter this parameter:

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'AddToPreviousRun', 'Y')
```

When run #2 is finished both your old and new runs will be in the BDM tables, including the consolidated table if you tell the BDM to build it. Note that you're limited to four preserved runs; any more than that and the BDM will generate an error.

How can I tell the difference between my runs? They both have the same formatcode.

You can assign a Run ID to any BDM run; it will be populated in the Eed/DbnRunID field. In the example above, you might assign the first run the ID Cobra:

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'RunID', 'Cobra')
```

Then you might assign a different ID to your second run:

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'RunID', 'New Enrollees')
```

Counting Dependents

It would be nice if I could count the types of dependents associated with each employee deduction.

First, tell the BDM to count dependents:

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'CountDependents', 'Y')
```

Then tell the BDM the codes for each kind of dependent you want to count, e.g . 'CHL, STC':

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'RelationshipsSpouse', '<relationship codes>')
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'RelationshipsChild', '<relationship codes>')
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'RelationshipsDomPartner', '<relationship codes>')
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'RelationshipsDPChild', '<relationship codes>')
```

The module will populate the below fields with the number of each type of dependent. Note that these fields also need to be entered for Cobra exports to calculate the PQBs.

- EedNumSpouses, EedNumChildren, EedNumDomPartners, EedNumDPChildren

BDM will now calculate the number of dependents on dependent records when the consolidated table is built and the 'CountDependents' parameter is used.

- BdmNumSpouses, BdmNumChildren, BdmNumDomPartners, BdmNumDPChildren

This may be helpful for COBRA when a dependent is PQB and the benefit option translation depends on the number of dependents are losing coverage. The dependent counts include the dependent who is PQB, if applicable. Counts on dependent records are only provided within the consolidated table.

Miscellaneous Parameters

It would also be nice if I could limit the module to one or two employees for testing.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'TestEEID', '<EEID>')
```

Enter the test EEIDs in the above parm. You can enter more than one; just keep inserting new TestEEID insert statements.

Don't forget to remove these parms when you're done testing.

I need the module to return only terms

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'TermsOnly', 'Y')
```

If this is set the module will invalidate all active deductions and return only terms. This is rare, and might only be used when an export needs both termed and active deductions for the same employees.

My export is based on deduction stop dates, not benefit stop dates

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'UseDedStopDate', 'Y')
```

- Notes: Setting UseDedStopDate to Y will tell the module to process deductions based on EedStopDate/DbnStopDate rather than Eed**Ben**StopDate/Dbn**Ben**StopDate for **ALL** Deduction Codes in that run. Use of this parameter trumps all other BDM logic.
- When then "UseDedStopDate" parameter is not used or is <> 'Y', the BDM will evaluate for each Deduction Code whether DedIsBenefit = 'Y'. If DedIsBenefit <> 'Y', it will automatically use eedStopDate (rather than eedBenStopDate).

I have quite a few bad dependent deductions… active where the employee is terminated, or the dependent has a deduction the employee doesn't

Normally the TC would point this out and advise the client to clean up the data. (The BDM error report will flag these errors.) In certain cases, however, a tight deadline or thousands of errors might not allow a timely cleanup. Setting the InvalidateBadDeps switch will cause the BDM to invalidate the bad dependent deductions so they don't appear on the export. This parm should be used with caution.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'InvalidateBadDeps', 'Y')
```

Programmer-defined Fields

The module tables contain a few user-defined fields that can be used by TCs for whatever.

U_dsi_bdm_EmpDeductions

```
EedUSGField1 – VARCHAR(256)
EedUSGField2 – VARCHAR(256)
EedUSGDate1  - DATETIME
EedUSGDate2  - DATETIME
```

U_dsi_bdm_DepDeductions

```
DbnUSGField1 – VARCHAR(256)
DbnUSGField2 – VARCHAR(256)
DbnUSGDate1  - DATETIME
DbnUSGDate2  - DATETIME
```

U_dsi_bdm_YourFormatCode (consolidated table)

```
BdmUSGField1 – VARCHAR(256)
BdmUSGField2 – VARCHAR(256)
BdmUSGDate1  - DATETIME
DbnUSGDate2  - DATETIME
```

Cobra

The BDM is capable of creating Cobra data and calculating whether an employee/dependent is the PQB.

To set the Cobra type and trigger the Cobra module

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'CobraType', '1')
```

1. Pull Cobra reasons from EepCobraReason and ConCobraReason
2. Pull Cobra reasons from EmpHDed and ConCobraReason. Also set **CHGRP**[1] as a valid change reason
3. Pull Cobra reasons from EmpHDed and ConCobraReason. Consider **CHGRP** as a valid change reason only if the employee's new DedGroup is NOT linked to any plans of type: MED, DENT, VIS, FSA.
4. Pull Cobra reasons using the following logic. Consider **CHGRP** as a valid change reason only if the employee's new DedGroup is NOT linked to any plans of the same DedType as the DedType being terminated[2]: MED, DENT, VIS, FSA.

   Employees:
   - If EepCobraReason is not NULL, then eepCobraReason.
   - ELSE if edhChangeReason from most recent EmpHDed record is COBRA Qualifying, then pull that edhChangeReason.
   - ELSE if there is a COBRA Qualifying edhChangeResaon that is that exists where EdhEffDate >= @StartDate, then pull the most recent edhChangeReason that is COBRA Qualifying.

   Dependents:
   - Pull Cobra Reasons from ConCobraReason.

The reasons are populated in the Eed/DbnCobraReason fields.

---

[1] **Note**: A benefit change reason will be set as **CHGRP** by UltiPro when the benefit change (normally a coverage stop event) is the result of a failure to pass the current eligibility criteria of the Deduction Group assigned. *i.e.* the employee has changed deduction groups and the new deduction group is not eligible for one or more plans in which the employee was actively enrolled. *Example*: An employee transitioned from FT to PT.

[2] For example, if an employee loses VIS coverage with a Change reason of **CHGRP** and the employee's new DedGroup does not have any VIS plans, the employee would be considered COBRA eligible for VIS, even though he/she may still maintain MED or DENT coverage in his/her new DedGroup

To set the Cobra event date

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'CobraDate', '1')
```

1. Pull Cobra dates from EepDateofCobraEvent and ConDateofCobraEvent
2. Pull Cobra dates from benefit stop dates
3. Pull Cobra dates using the following logic:

    Employees:
    - If EepDateofCobraEvent is not NULL then EepDateofCobraEvent
      ELSE EedBenStopDate

    Dependents:
    - If ConDateofCobraEvent is not NULL then ConDateofCobraEvent
      ELSE DbnBenStopDate

The dates are populated in the Eed/DbnCobraDate fields.

Option: Pull COBRA Reasons/Dates from Platform Configurable Fields, rather than Back Office Fields

Cobra Reasons & Cobra Dates can be maintained in Platform Configurable Fields (PCF) that have been setup by the customer. Insert only the BDM Configuration lines applicable to the customer setup.

The PCFs setup for `'EepCobraReasonPCF'` or `'EepDateOfCobraEventPCF'` must be setup in the **Employee** Class in the customer's environment.
The PCFs setup for `'ConCobraReasonPCF'` or `'ConDateOfCobraEventPCF'` must be setup in the **Contacts** Class in the customer's environment.

BDM will confirm that the specified PCF exists, before proceeding:

- If the specified PCF exists in the customer's environment, then that PCF is used exclusively. Any data in the corresponding Back Office COBRA field will then be ignored by the BDM Cobra module.
- If the specified PCF cannot be found, BDM will display a warning message to the TC and will default back to the corresponding Back Office COBRA field.

`'EepCobraReasonPCF'` can only be used with CobraType 1 or 4.
`'EepDateOfCobraEventPCF'` can only be used with CobraDate 1 or 3.
`'ConCobraReasonPCF'` can be used with any Cobra Type (1, 2, 3, or 4).
`'ConDateOfCobraEventPCF'` can only be used with CobraDate 1 or 3.

```
INSERT INTO dbo.U_dsi_BDM_Configuration VALUES (@FormatCode,'EepCobraReasonPCF','<SQL PCF Name>');
INSERT INTO dbo.U_dsi_BDM_Configuration VALUES (@FormatCode,'ConCobraReasonPCF', '<SQL PCF Name>');
INSERT INTO dbo.U_dsi_BDM_Configuration VALUES (@FormatCode,'EepDateOfCobraEventPCF', '<SQL PCF Name>');
INSERT INTO dbo.U_dsi_BDM_Configuration VALUES (@FormatCode,'ConDateOfCobraEventPCF', '<SQL PCF Name>');
```

Client facing setup instructions for COBRA PCFs are available in the following document:

COBRA Client Setup_Platform.docx

Specify the Cobra Reasons to Use

Note: if the reason parameters are left blank the BDM will just use the benefit change reasons in the BenChRsn table where the BchIsCOBRAQualifiedEvent = Y.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'CobraReasonsEmp', 'LEVNT3, 200, etc.')
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'CobraReasonsDep', '2001, 201, 202, etc.')
```

Specify the Cobra Reasons Not to Use

Setting these parameters will tell the BDM to ignore particular reasons, for example Employee Death

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'InvalidCobraReasonsEmp', 'LEVNT3, 200, etc.')
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'InvalidCobraReasonsDep', '2001, 201, 202, etc.')
```

Specify the COBRA Reasons where the Dependent can be identified as PQB

CobraReasonsDepPQB was added to allow the TC to specify which Reason Codes should qualify for inclusion in the file as a dependent-only loss of coverage.

- For example, EE death (term reason '203') is may be identified as an Invalid COBRA reason for the EE, but the dependent may still be to be flagged as PQB based on ChangeReason = '210'.

```
INSERT INTO dbo.U_dsi_BDM_Configuration VALUES (@FormatCode,'CobraReasonsDepPQB','201,204,210,LEVNT3,LEVNT4');
```

Indicate that Employee Termination Reasons should override Cobra Reasons

Setting these parameters will tell the BDM to invalidate records with particular *employee* **termination** reasons. For example: Employee Death may be a valid Cobra reason (200 above) but we wouldn't send the employee record in that case.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'InvalidTermReasonsEmp', '605, 200, etc.')
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'InvalidTermReasonsDep', 'A23, 605, 202, etc.')
```

Manually Populate the Cobra Reasons Table

If the TC needs to populate the reasons programmatically (e.g. all reasons that start with '2'), setting this parameter will tell the BDM to use whatever reasons it finds in the U_dsi_bdm_ChangeReasons table. The TC needs to populate U_dsi_bdm_ChangeReasons before the BDM runs.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'ManualCobraReasons', 'Y')
```

Determine Whether to Pull DedCodes Based on DedIsCobraCovered = Y

The default for a regular run is that all dedcodes in the defined list will be pulled; the default for a Cobra run is to also pull only deduction codes with DedIsCobraCovered = Y. This parameter allows you to override this. One example where you might use it is a Cobra general notices export; although it's not technically a Cobra run (only new deductions will be pulled) you might only want Cobra-covered dedcodes. (You can accomplish the same thing by manually loading the dedcode table; this just makes it a little easier.)

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'UseCobraCoveredDeds', 'Y')
```

X will tell the module to use deduction codes or deduction types specified where DedIsCobraCovered = Y.

Y will tell the module to ignore deduction codes or deduction types specified and use deductions where DedIsCobraCovered = Y. If no Dedcodes and/or types are specified, then it will return all deductions where DedIsCobraCovered = Y

N will tell the module to ignore deduction codes or deduction types specified and use deductions where DedIsCobraCovered = N. If no Dedcodes and/or types are specified, then it will return all deductions where DedIsCobraCovered = N

A will tell the module to use deduction codes or deduction types specified regardless of DedIsCobraCovered indicator assigned to DedCodes.

Blank or not set will pull all deductions. (Remember for Cobra runs, the above defaults to Y unless you override it.)

Calculate the PQB

By default the BDM sets the IsPQB flag first for one member of an employee's family (Employee, Spouse or a dependent). The PQB designation is first assigned to the employees and then for spouses without employees, then for the oldest child(ren) if no employee or spouse is dropping their coverage. Setting the parameter below tells the BDM to calculate the IsPQB flag differently when it is being assigned to dependents (non-spouse).

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'CobraPQBType', '1')
```

1> *All* children without employee or spouse dropping are flagged as PQBs, not just the oldest child.

Calculate Multiple PQBs (more than one member under an employee)

Some cobra vendors ask that more than one PQB designation be assigned when employees and their dependents have cobra events within the same period. When the parameter below is set, the BDM will select all members (under an employee EEID/COID) with a Cobra qualified event and set the Eed/DbnIsPQB = 'Y' for the member with the highest "rank" per each valid cobra reason and deduction code combination within the members.

Example 1: An employee has stopped their FSA coverage. The same employee's spouse and children stopped their medical benefit coverage within the date range with a reason of "Divorce". The employee kept her medical coverage. Because the employee had the highest "rank" for the FSA deduction within the members (Employee > Spouse > Children) she will be set as a PQB. Because the spouse had the highest "rank" among the members whom had stopped the medical coverage he will also be designated as a PQB.

Example 2: A Spouse dropped their dental coverage with a reason code of "Divorce". He has two sons. One had also stopped his dental coverage with a reason of "Divorce" and the other dropped his dental coverage with a reason code of "Exceeded maximum age". The spouse will be designated as a PQB because they had a higher rank when compared to the youngest son. The older son, whom had aged-out of coverage, will be set as a PQB because his cobra reason differs from that of the spouse.

If a '1' or a '2' value is set as the parameter, the PQB indicators will be updated per above and, in addition, a user-defined BDM field will be set for dependents whom are to roll-up or "link" to a PQB. This user-defined fiedld assigments is meant to aid when sorting/grouping records in an export.

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'CobraPQBMulti', '1')
```

0> Do not populate DbnUSGField1 nor DbnUSGField2 with any values. Only set the PQB indicators accordingly.

1> Set the **DbnUSGField1** column, in the U_dsi_bdm_DepDeductions table, to be the DbnDepRecID of the PQB in which the dependent is to "roll-up-to".

2> Set the **DbnUSGField2** column, in the U_dsi_bdm_DepDeductions table, to be the DbnDepRecID of the PQB in which the dependent is to "roll-up-to".

Must Set the Relationships

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'RelationshipsSpouse', '<relationship codes>')
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'RelationshipsChild', '<relationship codes>')
```

Validate employees for Cobra if there's a dependent on Cobra

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'CobraIncludeEmps', 'Y')
```

If set to Y, if there's a valid Cobra dependent the employee's deduction will set to valid even if they're not a valid Cobra deduction. This might be used when the export requires employee records for new Cobra dependents.

New Enrollees

The BDM is capable of determining whether an employee/dependent is newly enrolled in a deduction. The determination is based on whether the Eed/DbnBenStartDate has been entered in audit within the BDM's timeframe. Only newly-enrolled deductions will be flagged as valid; all other deductions will be flagged as invalid even if they're active.

Set the New Enrollee type and trigger the New Enrollee module:

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'NewEnrolleeType', '1')
```

1. Anyone newly enrolled in any deduction is valid
2. All new enrollees with no previous plan since their Date of Last Hire (eecDateOfLastHire)
3. All new enrollees with no previous plan in all history.
4. All new enrollees & re-enrollees with no active plan since their Date of Last Hire (eecDateOfLastHire). Re-enrollments into the same DedCode are considered under this option. For example, the employee may have been enrolled in **MED1** during his/her original employment and then was rehired at a later date and re-enrolled in **MED1**. NewEnrolleeType = '4' will consider this re-enrollment scenario as valid in the New Enrollee Module.

Allow only spouses and/or domestic partners:

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'NewEnrolleeDeps', '1')
```

All dependents are included by default. Setting this parm will allow only spouses or domestic partners along with the employees. Note that the `RelationshipsSpouse` and (if necessary) the `RelationshipsDomPartner` parms must also be set.

1. Include only spouses
2. Include only spouses and domestic partners

Don't include dependent enrollees if the employee is valid

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'NewEnrolleeExcludeDeps', '1')
```

This will invalidate any dependent enrollees if the employee is valid.

1. Exclude each dependent dedcode where the same valid employee dedcode exists
2. Exclude ALL dependent dedcodes where a valid employee dedcode exists

Validate employees as new enrollees if there's a dependent enrollee

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'NewEnrolleeIncludeEmps', 'Y')
```

If set to Y, if there's a valid dependent enrollee the employee's deduction will set to valid even if they're not newly enrolled. This might be used when the export requires employee records for new dependent enrollees.

Some Standard BDM Queries

Main tables

```
Select EedFormatCode, EedValidForExport, EedDedRowStatus, EedEEID, EedCOID, EedDedCode, DedDedType,
EedBenOption, EedBenStartDate, EedBenStopDate, EedBenStatus

From dbo.U_dsi_bdm_EmpDeductions
Where EedEEID = ''
Order By EedEEID, EedCOID, EedDedCode


Select DbnFormatCode, DbnValidForExport, DbnDedRowStatus, DbnEEID, DbnDepRecID, DbnDedCode, RTRIM(ConNameLast) +
', ' + RTRIM(ConNameFirst) 'Name', ConRelationship, DedDedType, DbnDedCode, DbnBenOption, DbnBenStartDate,
DbnBenStopDate

From dbo.U_dsi_bdm_DepDeductions
JOIN Contacts (NOLOCK) on ConEEID = DbnEEID AND ConSystemID = DbnDepRecID
Where DbnEEID = ''
Order By DbnEEID, DbnDepRecID, DbnDedCode
```

Consolidated table

```
Select BdmRecType, BdmCOID, BdmEEID, BdmDepRecID, BdmRunID, BdmDedType, BdmDedCode, BdmBenOption,
BdmBenStartDate, BdmBenStopDate, BdmBenStatus

From dbo.U_dsi_bdm_ECERCOBRA
Order By BdmEEID, BdmRecType Desc
```

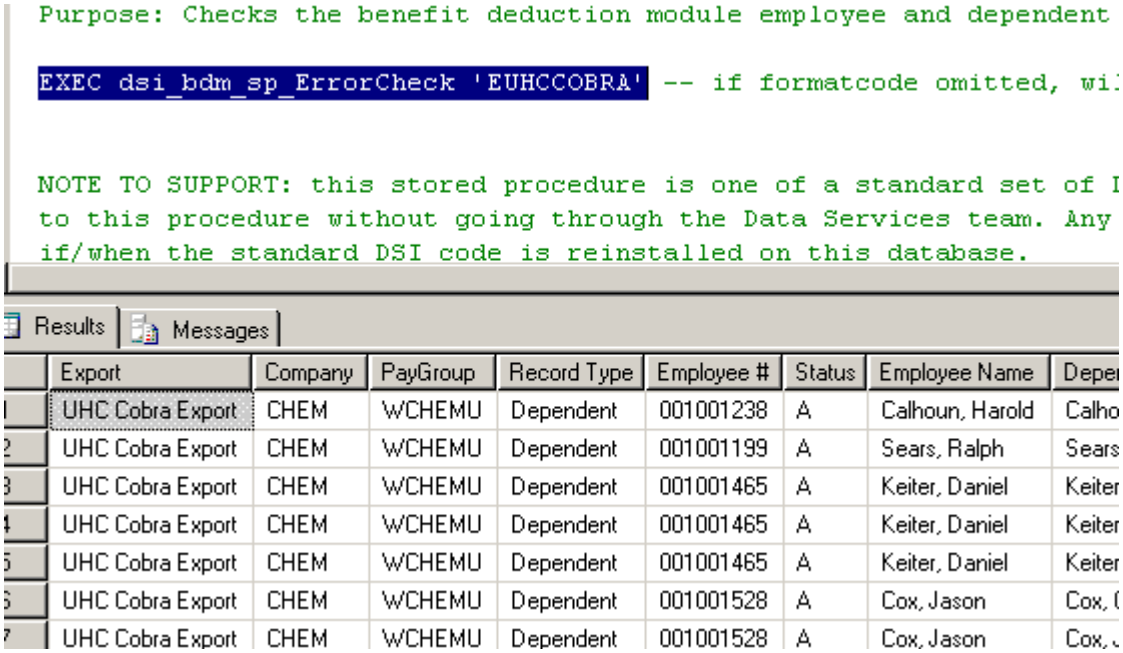
Error Checking

The BDM comes with an error-checking SP that lists common data issues, e.g. multiple active deductions within the same deduction type, dependent has an active deduction that the employee does not, etc. This can be quite helpful in tracking down export issues, many of which will be data-related. The selection is designed to be pasted into an Excel spreadsheet or email and sent to the customer.

To run:

```
EXEC dbo.dsi_bdm_sp_ErrorCheck 'YOUR FORMATCODE'
```

If you omit the formatcode it will run the error check for all deductions in the BDM deduction tables.



```
Purpose: Checks the benefit deduction module employee and dependent

EXEC dsi_bdm_sp_ErrorCheck 'EUHCCOBRA' -- if formatcode omitted, wi

NOTE TO SUPPORT: this stored procedure is one of a standard set of I
to this procedure without going through the Data Services team. Any
if/when the standard DSI code is reinstalled on this database.
```

Results | Messages

	Export	Company	PayGroup	Record Type	Employee #	Status	Employee Name	Depe
1	UHC Cobra Export	CHEM	WCHEMU	Dependent	001001238	A	Calhoun, Harold	Calho
2	UHC Cobra Export	CHEM	WCHEMU	Dependent	001001199	A	Sears, Ralph	Sears
3	UHC Cobra Export	CHEM	WCHEMU	Dependent	001001465	A	Keiter, Daniel	Keiter
4	UHC Cobra Export	CHEM	WCHEMU	Dependent	001001465	A	Keiter, Daniel	Keiter
5	UHC Cobra Export	CHEM	WCHEMU	Dependent	001001465	A	Keiter, Daniel	Keiter
6	UHC Cobra Export	CHEM	WCHEMU	Dependent	001001528	A	Cox, Jason	Cox, (
7	UHC Cobra Export	CHEM	WCHEMU	Dependent	001001528	A	Cox, Jason	Cox, J

Appendix: BDM Version History

BDM v1.30 Changes/Enhancements Summary

1) New SP added to BDM (`dbo.dsi_bdm_sp_PopulateDeductionsTable_v2`).
    a. This new version provides for better SQL optimization & shorter run times for customers with large datasets.
    b. At this time, TCs are encouraged to continue to use the existing version of the SP (`dbo.dsi_bdm_sp_PopulateDeductionsTable`), unless long run times are a concern, since additional optimization for audit processing is anticipated for BDM releases in the near future.
        i. Future optimization will be isolated to `dbo.dsi_bdm_sp_PopulateDeductionsTable_v2`, to ensure existing integrations that currently use `dbo.dsi_bdm_sp_PopulateDeductionsTable` remain unaffected by these changes.

BDM v1.29 Changes/Enhancements Summary

1) Added .dbo prefix to object names throughout BDM functions & SPs.
2) Added semicolon terminators throughout BDM functions & SPs.
3) Suppress print statements in SQL console when "Testing' flag for FormatCode in dbo.U_dsi_Configuration = 'N'

BDM v1.28 Changes/Enhancements Summary

1) Added ability for clients to maintain Cobra Reasons & Cobra Dates in Platform Configurable Fields (PCF), rather than the Back Office COBRA fields.
2) CobraType '4' has been added to support a standard COBRAType logic that can be used across UD & UDE as a standard "best practice".
3) CobraDate '3' has been added to support a standard COBRADate logic that can be used across UD & UDE as a standard "best practice".
4) CobraReasonsDepPQB was added to allow the TC to specify which Reason Codes should qualify for inclusion in the file as a dependent-only loss of coverage. For example, EE death (term reason '203') is may be identified as an Invalid COBRA reason for the EE, but the dependent may still be to be flagged as PQB based on ChangeReason = '210'.
5) Moved @CobraIncludeEmps logic to the end of the BDM Cobra Module Logic, to ensure all dependent COBRA logic has been completed before determining EEs that should be included since they have at least 1 valid dependent.
6) Added NewEnrolleeType '4'. All new enrollees & re-enrollees with no active plan since their Date of Last Hire (eecDateOfLastHire). Re-enrollments into the same DedCode are considered under this option. For example, the employee may have been enrolled in **MED1** during

his/her original employment, was rehired at a later date and then re-enrolled in **MED1**. NewEnrolleeType = '4' will consider this re-enrollment scenario as valid in the New Enrollee Module.

BDM v1.27 Changes/Enhancements Summary

1) Update to *CountDependents* functionality:
   - BDM will now calculate the number of dependents on dependent records when the consolidated table is built and the `'CountDependents'` parameter is used.
     - BdmNumSpouses, BdmNumChildren, BdmNumDomPartners, BdmNumDPChildren
   - This may be helpful for COBRA when a dependent is PQB and the benefit option translation depends on the number of dependents are losing coverage. The dependent counts include the dependent who is PQB, if applicable. Counts on dependent records are only provided within the consolidated table.
2) Ability to add custom (TC Defined) fields to the consolidated table:

```
INSERT INTO dbo.U_dsi_bdm_Configuration VALUES (@FormatCode, 'BdmAddField', '<bdmYourNewField>')
```

   - This tells the BDM to add a custom field with the name that has been provided.
   - The prefix 'bdm' will be automatically added to the field name if it was not specified
   - All custom fields are added with the data type **varchar(500)**.
   - Multiple custom fields can be added by inserting multiple `'BdmAddField'` rows into `U_dsi_bdm_Configuration`.
   - BDM will check to make sure the field names to be added do not already exist in the consolidated table. If they already exist, the fields with duplicate names will be skipped and a warning message is displayed.
   - If the TC is running the BDM multiple times with the consolidated table in the same SP, the custom fields must be defined in the final BDM run for the SP. Otherwise the custom fields defined in earlier calls to the BDM would be lost in the final BDM execution for the SP.
3) Updated string variables that contain DedCode/DedType lists to VARCHAR (8000), to avoid previous field length limitations.
4) Correction to PQB rules for domestic partners & domestic partners children, to ensure these relationships are accurately reflected as PQB, if applicable.
5) Correction to NewEnrolleeDeps = '2' logic. The *RelationshipsSpouse* relationship list was referenced twice in previous versions. Corrected to reference the *RelationshipsSpouse* & *RelationshipsDomPartner* relationship lists, instead.
6) Updated *dsi_BDM_sp_ErrorCheck* to not flag error for an employee with > 1 valid deduction code within a Deduction Type, when the DedType allows for multiple deductions.

7) Added error check to *dsi_BDM_sp_ErrorCheck* to check for Deduction Codes where DedIsCobraCovered = 'Y', but DedIsBenefit = 'N' when *CobraType* or *UseCobraCoveredDeds* parameter is specified.