ProAdmin®

Coding Best Practices

(Version July, 2019)

Contents

Testing whether to use late retirement component

Observation: The late retirement component is always equal to the benefit formula before NRD and any early retirement reduction (ERF) is always equal to 1 at NRD and thereafter. Accordingly, it is not necessary to test in the formula whether late retirement should apply or not: the late retirement component can just be used directly, referencing the plan benefit (including ERF) in the component.

Sample Client Coding:

```
BEN_ERF :=      #IF AGE_ACD >= 65

                        #THEN 2 #ROUND LATERET

                        #ELSE 2 #ROUND BENFORM

                        #ENDIF  &
```

Recommended:

```
BEN_ERF :=      2 #ROUND LATERET
```

Temporary Variables in Benefit Definitions for Fulfillment

Observation: Since version 3.03 (released January, 2012) Plan Definitions include a Fulfillment Components topic for specifying the components that need to be calculated for fulfillment needs but might otherwise not be used in the calculations. Prior to that time users had to set up temporary variables in the benefit formula to make certain that a component was available. Even though this need is obsolete, we continue to see it in plans sent in for assistance. Using these variables obfuscates the actual plan formula, making it more time-intensive to understand how a plan has been set up. Moreover, in version 3.13 (released June, 2019), if this “clutter” is removed, the new “Columns” option allows you to easily compare the beginning of the benefit formulas of multiple Benefit Definitions by including “Benefit formula” as a column in the library display.

Sample Client Coding:

```
a:=litoutput&
b:=NRD&
c:=ABmonthly&
d:=ABatBCDmonthly&
e:=Base_a&
f:=Base_b&
g:=ABmonthly_BNormUnlimited&

PlanBen
```

Recommended:

```
PlanBen
```

Use of Annuity Factors for static mortality and interest

Observation: Annuity factors are a "calculation-time-expensive" coding approach in ProAdmin. They are necessary when the mortality is dynamic, interest rates vary over time, or the factor is otherwise "complex", but they are not necessary for static mortality and interest, where the factor is easily calculated and well-documented in a Benefit Formula Component Table.

Sample Client Coding:

AnnFact_1983GAM6Pct_Def / AnnFact_1983GAM6Pct_Imm

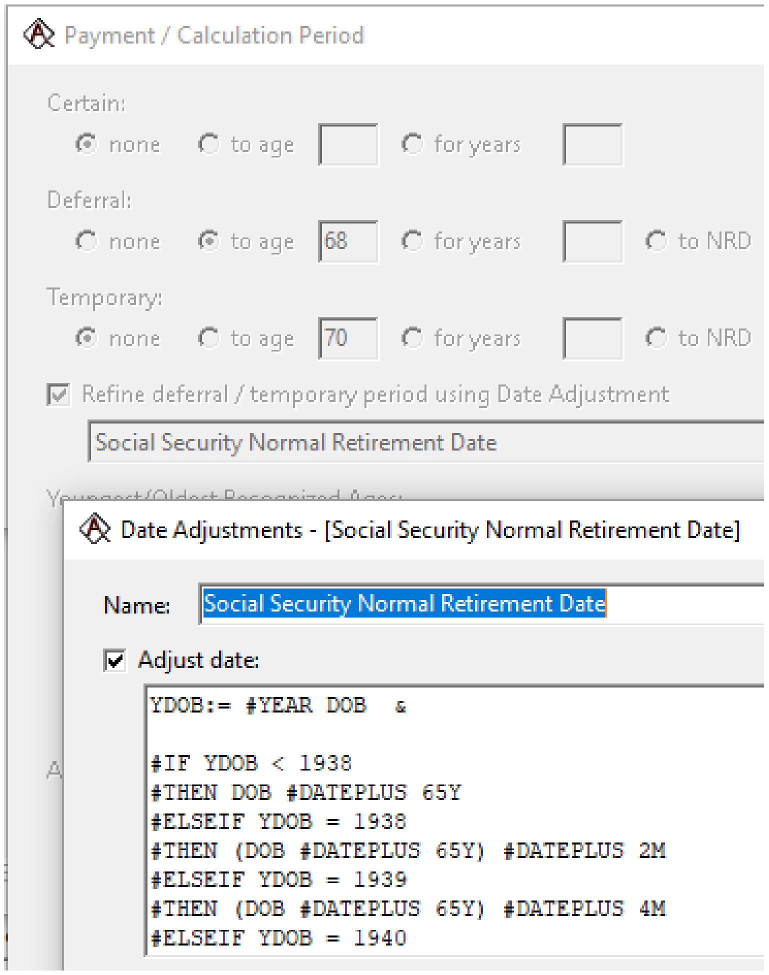
Recommended:

Same construct but use table components instead of annuity factor components. Full documentation is maintained if you create a Benefit Component Table using "Calculate Annuity Factors" under the **Options** tab. When the table is viewed, all details of the annuity factors calculation are retained. (Effective with ProAdmin version 3.12 (released October, 2018), this approach is now automatically suggested when a user attempts to save a "simple" annuity factor component.)

Using a Date Adjustment for SSNRA in Annuity Factors

Observation: #IF #THEN #ELSE is a "calculation-time-expensive" coding approach in ProAdmin. Previously if a user needed an annuity factor deferred or temporary to SSNRA, they used the **Refine deferral/temporary period using Date Adjustment** feature. Effective with ProAdmin version 3.12 (released October, 2018), a new option in annuity factor components allows you to use ProAdmin's built in SSNRA calculation directly.

Sample Client Coding:


Payment / Calculation Period

Certain: [✓ none] to age for years

Deferral: none [✓ to age] 68 for years to NRD

Temporary: [✓ none] to age 70 for years to NRD

[✓ Refine deferral / temporary period using Date Adjustment]

Social Security Normal Retirement Date

Date Adjustments - [Social Security Normal Retirement Date]

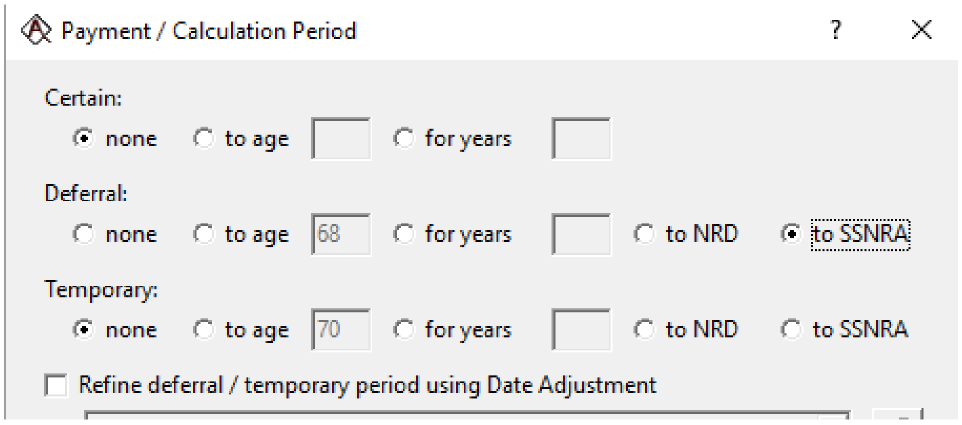
Name: Social Security Normal Retirement Date

[✓ Adjust date:]

```
YDOB:= #YEAR DOB  &

#IF YDOB < 1938
#THEN DOB #DATEPLUS 65Y
#ELSEIF YDOB = 1938
#THEN (DOB #DATEPLUS 65Y) #DATEPLUS 2M
#ELSEIF YDOB = 1939
#THEN (DOB #DATEPLUS 65Y) #DATEPLUS 4M
#ELSEIF YDOB = 1940
```

Recommended:


Payment / Calculation Period

Certain: [✓ none] to age for years

Deferral: none to age 68 for years to NRD [✓ to SSNRA]

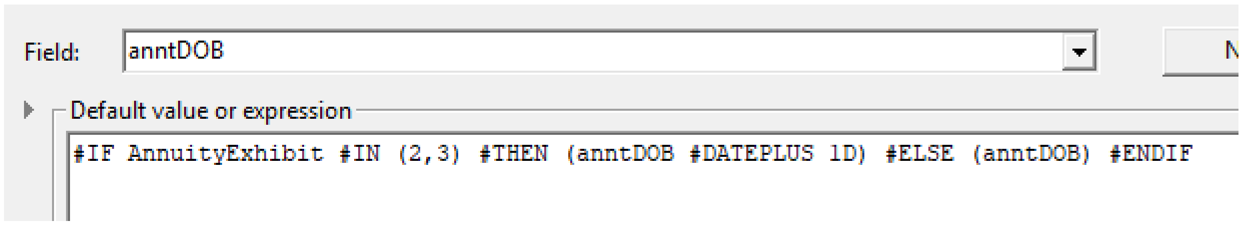
Temporary: [✓ none] to age 70 for years to NRD to SSNRA

Refine deferral / temporary period using Date Adjustment

Modifying source data in Data Defaults (Regression Testing Issue)

Observation: ProAdmin's Data defaults allow data to be modified for "all values, whether or not missing." This is a great feature for defining special data fields, but it can cause an issue for "run and compare" and regression testing if not coded appropriately. "Run and compare" and regression testing are typically done using the saved data to verify that there was no change in results due to new coding or a new version of ProAdmin. If the Data Defaults modify the data each time the calculation is run, the results and data may have spurious differences.

Sample Client Coding:



Field: anntDOB

Default value or expression

```
#IF AnnuityExhibit #IN (2,3) #THEN (anntDOB #DATEPLUS 1D) #ELSE (anntDOB) #ENDIF
```

Recommended:

In the Database or XML Linkage, read in the client data with a different Data Dictionary field name than that which will be used during the calculation, such as "anntDOB_orig" in the example above. Then create the calculation field "anntDOB" from "anntDOB_orig". Then when a calculation is re-run with saved data, anntDOB will not change.

Poorly structured #IF #THEN #ELSEIF

Observation: Typically, if the result of the first #THEN condition and the last #ELSE condition are the same, the expression can be simplified.

Sample Client Coding:

```
#IF #NOT(NQ_only_because_of_VDCP)

#THEN NQEarnings

#ELSEIF ((NQ_only_because_of_VDCP)#AND #NOT (Over_SSTWB_half_Without_VDCP))

#THEN VDCP_Over_SSTWB_half

#ELSE NQEarnings

#ENDIF
```

Recommended:

```
#IF ((NQ_only_because_of_VDCP)#AND #NOT (Over_SSTWB_half_Without_VDCP))

#THEN VDCP_Over_SSTWB_half

#ELSE NQEarnings

#ENDIF
```

#IF #THEN #ELSE conditional used instead of the #MAX Boolean.

Observation: #IF #THEN #ELSE is a "calculation-time-expensive" coding approach in ProAdmin.

Sample Client Coding:

```
#IF (Ben1 > Ben2)

#THEN    Ben1

#ELSE    Ben2

#ENDIF
```

Recommended:

```
Ben1 #MAX Ben2
```

Use of = 1 in Boolean checks

Observation: When using Booleans to zero out a benefit if the field is a 0 (no) or 1 (yes) all that is required is to multiply by the field.

Sample Client Coding:

```
([(LS > 5000) #AND (LS <= 10000)] * LS) * (Restricted = 1) * (.5)
```

Recommended:

```
[(LS > 5000) #AND (LS <= 10000)] * LS *  Restricted  * .5
```

Use of #IN when the list contains one number

Observation: #IN requires a little more effort than = to evaluate.

Sample Client Coding:

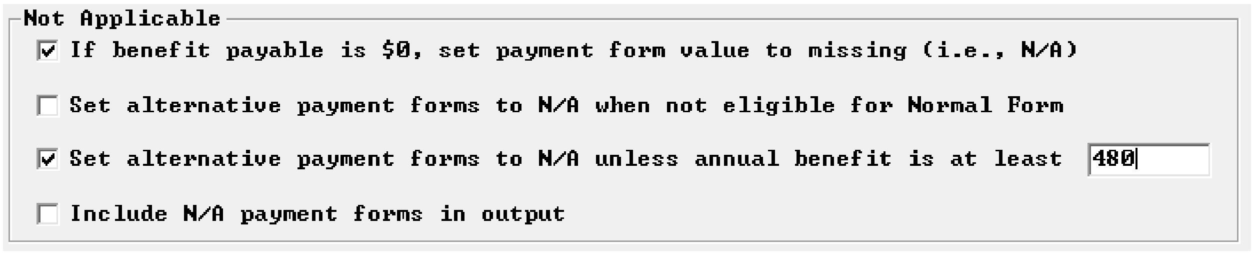
```
(ACD #MONTHDIF (#BEGMTH #DODEC)) #IN (15)
```

Recommended:

```
(ACD #MONTHDIF (#BEGMTH #DODEC)) = 15
```

Separate benefits for each FOA due to minimum monthly benefit

Observation: Effective with Version 3.10 (released in April, 2017), the Plan Definition > Plan Attributes > Miscellaneous Parameters option to **Set alternative payment forms to N/A unless annual benefit is at least** automatically eliminates optional forms of annuity that are not at least the specified minimum annual amount. Note, however, that if the minimum doesn't apply to both the life annuity normal form and the QJSA (albeit reduced) normal form for married participants, then coding in each Benefit Definition *is* currently required.


Not Applicable

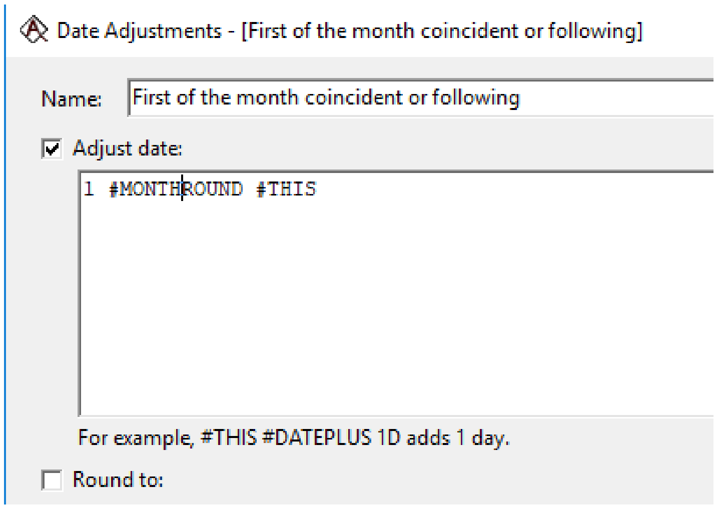
- [✓] If benefit payable is $0, set payment form value to missing (i.e., N/A)
- [ ] Set alternative payment forms to N/A when not eligible for Normal Form
- [✓] Set alternative payment forms to N/A unless annual benefit is at least 480
- [ ] Include N/A payment forms in output

Date Adjustments using formula instead of “Round to”

Observation: Some users use the **Adjust date**: formula on the Date Adjustments dialog box when the **Round to:** parameters would produce the same result more clearly and more efficiently.

Sample Client Coding:

We have seen clients use each of these 3 algorithms to round dates to the first of the month coincident or following: the 1 #MONTHROUND #THIS below, #NEXTBEGMTH #THIS and #BEGMTH #THIS.



Date Adjustments - [First of the month coincident or following]

Name: First of the month coincident or following

☑ Adjust date:

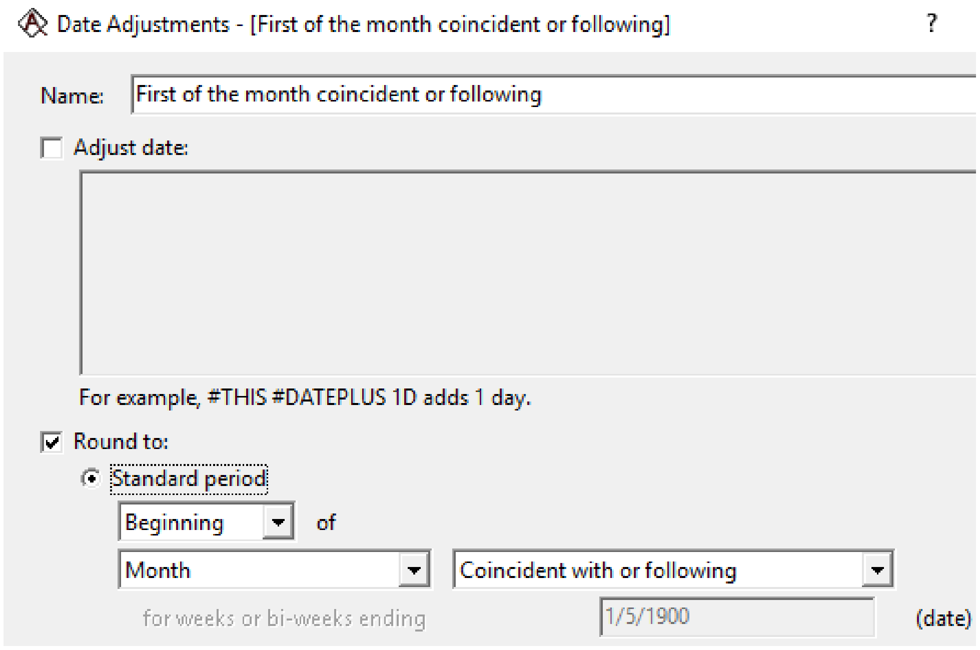
```
1 #MONTHROUND #THIS
```

For example, #THIS #DATEPLUS 1D adds 1 day.

☐ Round to:

Recommended:

Simply using the **Round to:** parameters on the bottom of the dialog is more efficient and better documented.



Date Adjustments - [First of the month coincident or following] ?

Name: First of the month coincident or following

☐ Adjust date:

For example, #THIS #DATEPLUS 1D adds 1 day.

☑ Round to:

● Standard period

Beginning of

Month | Coincident with or following

for weeks or bi-weeks ending 1/5/1900 (date)