

ProAdmin version 3.06 introduces a flattened Benefit Definitions interface, the ability to run estimates with a decrement type, new operators #GETTABVAL and #INARRAY, the ability to vary the interest rate for Actuarial Equivalence by coded field, and many other features listed below.

Interface

- ◆ **Flattened benefit interface.** The Benefit Definition topics have been combined into a single dialog box, with several parameters (e.g., the payable party in death benefits) rearranged into a more logical location. This eliminates clicking when setting up plans and allows all pertinent parameters to be easily viewed. Clicking on the triangle in front of the benefit formula or payment forms will expand that section of the dialog box to facilitate editing.

Benefit Definition - [Traditional Plan Retirement - Limited]

Name: XML Code...

Contingency initiating benefits
Contingency:

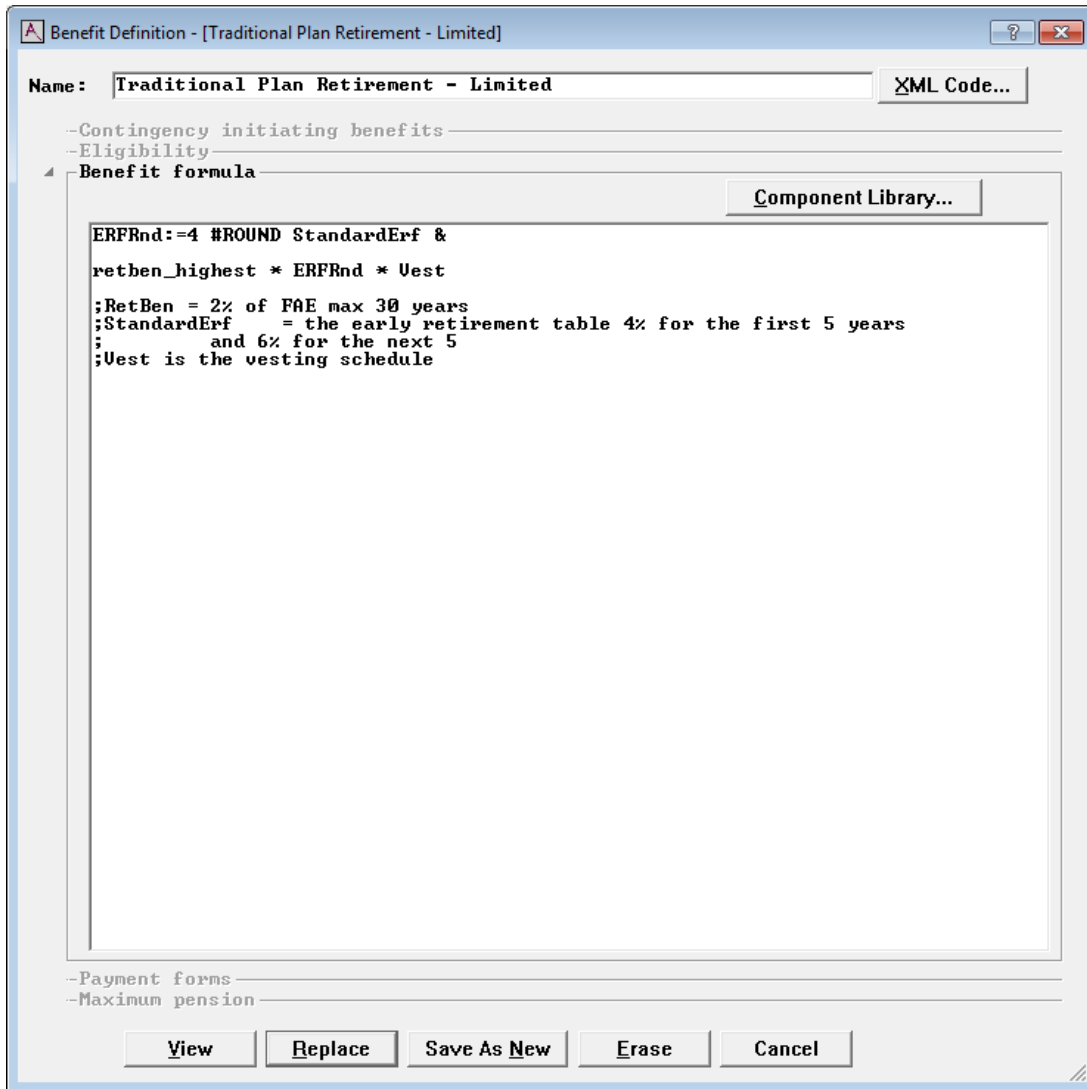
Eligibility
Eligibility:
Using Service:
 Apply alternative eligibility
Alt. Elig.:
Using Service:

Benefit formula

```
ERFRnd:=4 #ROUND StandardErf &  
retben_highest * ERFRnd * Vest  
  
;RetBen = 2% of FAE max 30 years  
;StandardErf = the early retirement table 4% for the first 5 years  
; and 6% for the next 5  
;Vest is the vesting schedule
```

Payment forms
Payment Forms:
10 yr C&L
50% J&S
75% J&S
100% J&S
SS Level Income Option
SS Level Income Option w/ 50% JS
Normal Form:

Maximum pension
Limit: None U.S. 415(b) Canadian ITA
Annuity limit only; no adjustment for payment form.



- ◆ **Scrolling for dialog boxes with many parameters.** Tall dialog boxes can be scrolled.
- ◆ **Reviewing benefits.** To make inputs easier to review:
 - The listing of benefit entries has been condensed, without sacrificing readability. This is to facilitate paper review and documentation.
 - A tabular export to Excel of benefit entries is available to facilitate interactive review with filtering, sorting, etc.

	A	B	C	D	E	F
	Benefit Definition	XML code	XML code override	Contingency	Death benefits paid to	Eligibility Definition
1	Early Retirement Benefit	default		Retirement		Early Retirement Eligibility
2	Vested Retirement Benefit	default		Retirement		Vested Retirement Eligibility
3	Small benefit LumpSum	default		Retirement		Vested Requirement

- ◆ **Decrement Type.** The Estimated Benefit Calculation dialog box has been enhanced to allow for the processing of specific decrement types.

Calculation

Decrement type: **Term./Retirement**

Commencement:

Date(s)	Age(s)
4/01/2017	

One Decrement date:

Multiple decrement/
 commencement dates

Census Specifications

- ◆ **The Beneficiary Data dialog box** has been enhanced to allow for multiple beneficiary type codes to be mapped to the ProAdmin Beneficiary Type codes (None, Non-spouse and Spouse). For example, both Spouse and Domestic Partner beneficiary type codes can be mapped to the ProAdmin Beneficiary Type code "Spouse".

Beneficiary type: **Domestic Partner**

Bentype: **None**

Database Label	ProAdmin Beneficiary Type
No Spouse	None
Beneficiary	Non-spouse
Spouse	Spouse
Domestic Partner	None Spouse Non-spouse

Interest & Annuity Factor Components

- ◆ Interest Factor and Annuity Factor Benefit Formula Components and Accrual Basis Components can now reference generational mortality and age by year of birth mortality improvement scales.

Operators

- ◆ **New operator #INARRAY** is available wherever Data Dictionary fields can be referenced within expressions. This operator allows you to search array fields for specific values. The desired value is the left argument and the array field is the right argument. If the value is found in the array field, 1 is returned; if it is not found, 0 is returned. The left argument can be a number, code, date, or temporary variable. For example, 10 #INARRAY StatusHistory will return 1 if the member ever had a status value of 10.
- ◆ **New operator #GETTABVAL** is available in Service and Salary transformation expressions. This operator will return the value in an external CSV file associated with look-up value(s). The left argument is the name of the CSV file (in quotes; no path or file extension) containing the look-up constraints and associated values. The right argument is a parenthesized list of temporary variables (created by assignment within the expression), used to look up values in the CSV file. If there is no value associated with the set of look-up values, 0 is returned.

For example, a 3 column table of values by date and age might be accessed with the following expression:

```

D_dt := #DATE &
Age := #DATE #YEARDIF DOB &

```

'AgeTable' #GETTABVAL (D_dt, Age)

For more information about #GETTABVAL, please see the article beginning on Page 11.

- ◆ **The #DATE operator** can now be accessed in Benefit Definition benefit formulas, Accrual Definition basis formulas, and Benefit and Accrual Basis subformula components. #DATE offers fine control for defining benefits that vary over time. It is the set of all applicable calculation dates.

Plan Definition

- ◆ A new parameter is available under the Plan Definitions Misc. Parameters... button that sets alternative payment forms to "not applicable" (i.e., missing) when the member is not eligible for the normal form of payment. If the parameter is checked and the member fails the eligibility criteria for the normal form, then all forms of payment will be set to N/A.

The screenshot shows a dialog box titled "Miscellaneous Parameters" with several sections. The "Not Applicable" section is highlighted with a red border and contains the following options:

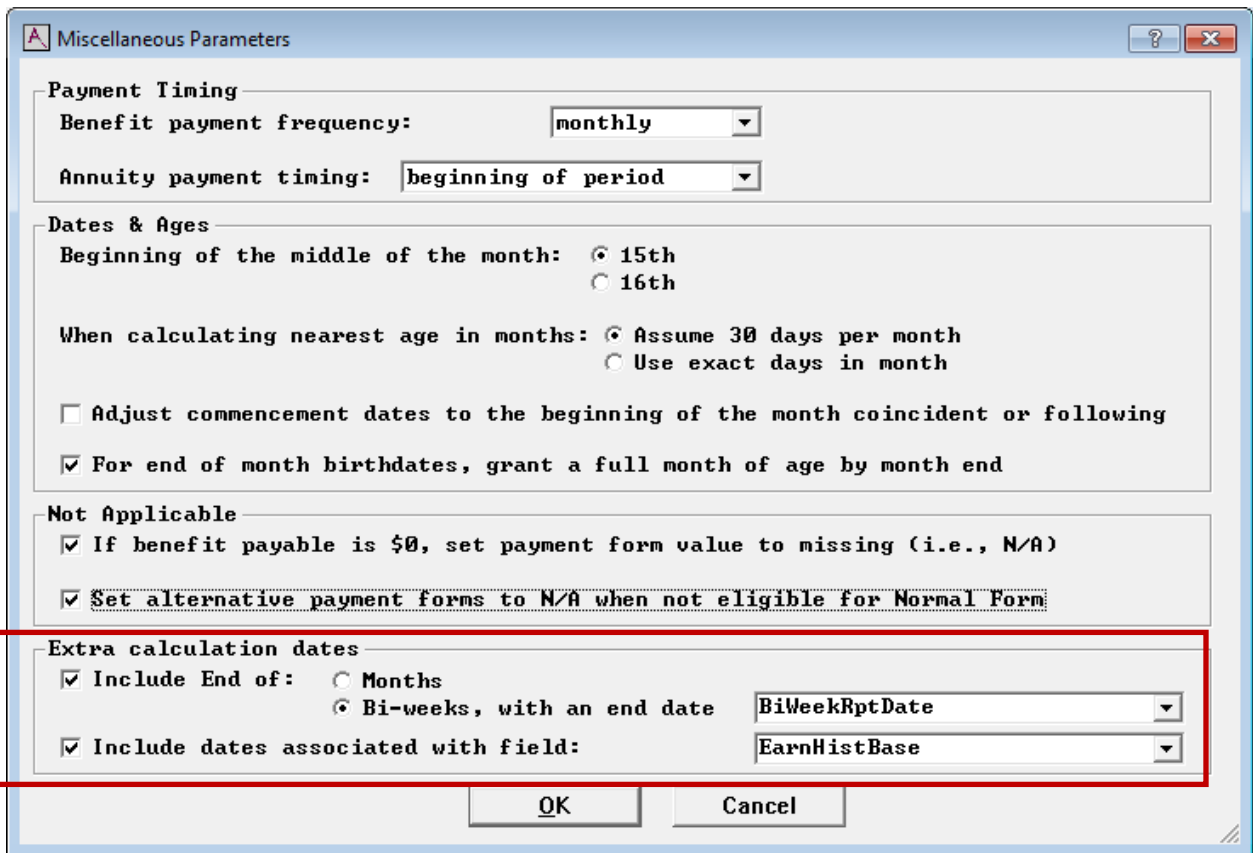
- If benefit payable is \$0, set payment form value to missing (i.e., N/A)
- Set alternative payment forms to N/A when not eligible for Normal Form

Other sections in the dialog include "Payment Timing" (Benefit payment frequency: monthly, Annuity payment timing: beginning of period), "Dates & Ages" (Beginning of the middle of the month: 15th, 16th; When calculating nearest age in months: Assume 30 days per month, Use exact days in month; Adjust commencement dates to the beginning of the month coincident or following; For end of month birthdates, grant a full month of age by month end), and "Extra calculation dates" (Include End of: Months, Bi-weeks, with an end date; Include dates associated with field).

- ◆ **Extra Calculation Dates.** New parameters are available under the Plan Definitions Misc. Parameters... button that optionally add extra calculation dates to the default set of calculation dates. This may be helpful, for example, to force a re-calculation of the accrued benefit at specific dates.

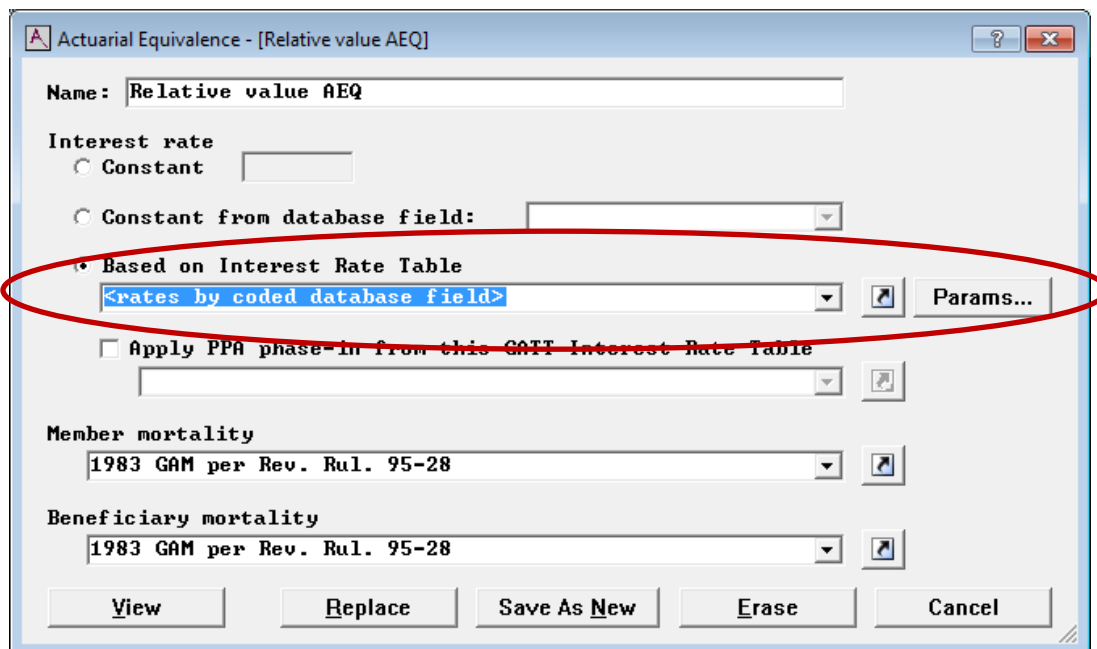
The first new option will include all end of months or end of bi-weeks from the first calculation year to the last calculation year. If you choose end of bi-weeks, you must specify a scalar date field that contains an end of bi-week date; all other bi-weeks are generated from this date.

The second new option will include all the dates associated with a specified field. If the field is a scalar date field, then that single date is added. If the field is an effective date array, then all the effective dates are added; for a start/stop array, all the stop dates are added. If the array field is a date array field, then all the date values will be added.

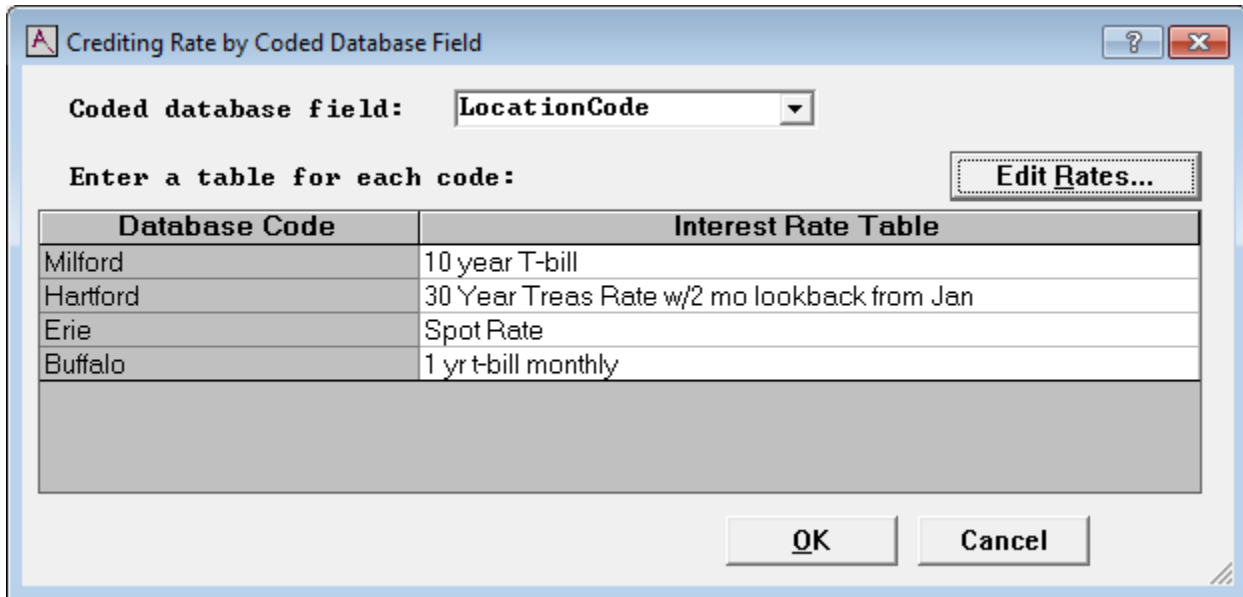


Actuarial Equivalence

- ◆ The interest rate component of actuarial equivalence can now be varied by coded field. If you select the <rates by coded database field> the Params... button will be enabled.



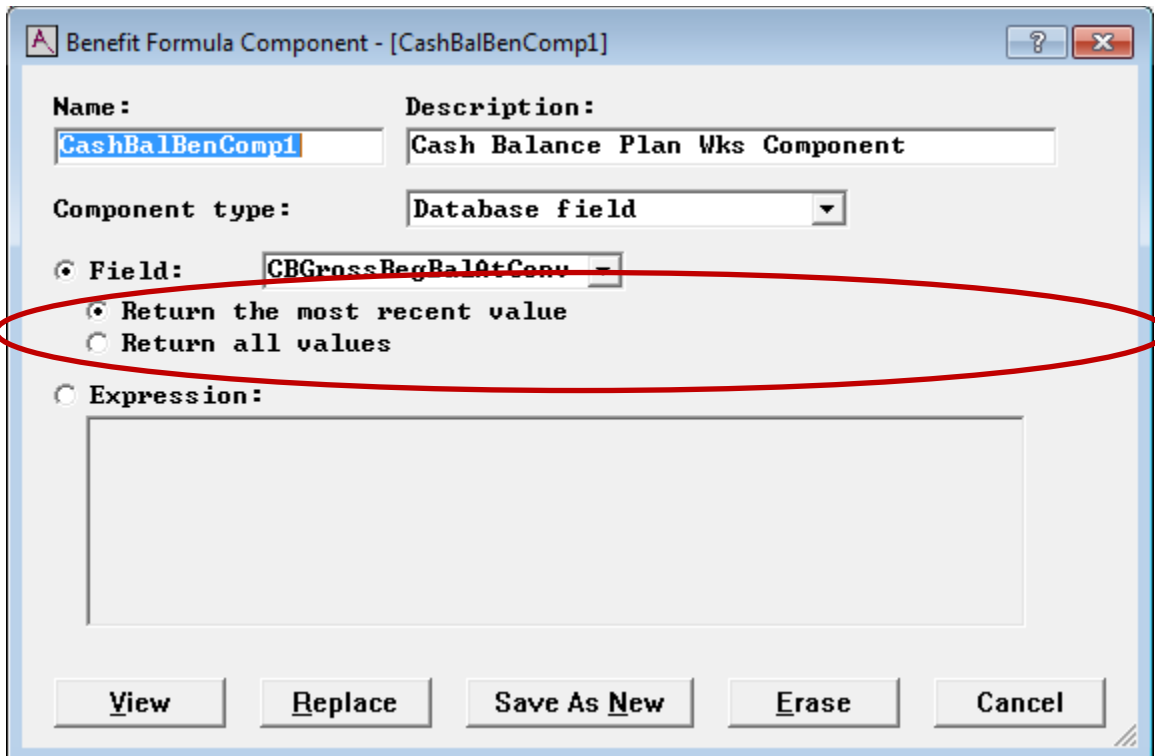
When you select the Params... button, the Crediting Rate by Coded Database field dialog box will display. This is where you select the coded field that is used to determine the interest rate for each of your groups.



- ◆ Generational mortality, age by year of birth mortality, and age by year of birth mortality improvement scales can now be referenced in Actuarial Equivalence library entries. (Note that pre- and post-commencement mortality continues to be available. If referenced, the pre-commencement mortality is only used during deferral periods.)

Database Field Components

- ◆ **Array Values.** The database field type of Benefit Formula Components and Accrual Basis Components has been enhanced to return either the last reported value of a database field (the current default) or the full set of array values from effective or start/stop date fields. This allows a formula, for example, to easily vary based on the location a member was in at different points in time.

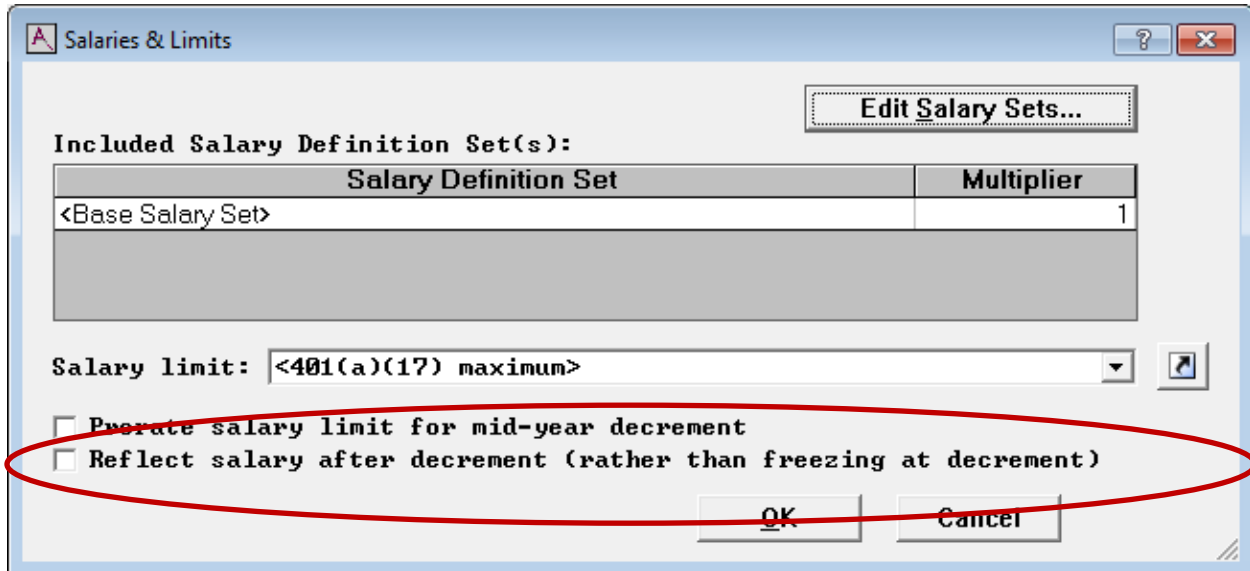


If the field is a start/stop array and the date is in either the start date or the stop date then the value at that date is return; otherwise 0 is returned. If the array is an effective date array, the

value within the effective date range is returned and 0 is returned for any dates prior to the earliest effective date.

Custom Operators

- ◆ **Accruals after decrement.** A Salary Custom Operator can now allow accruals after decrement. This is available for both final and estimate calculations. Note that for this choice to be effective, the referenced Salary Definition Set(s) must also reflect salaries after decrement.



Salaries & Limits

Edit Salary Sets...

Included Salary Definition Set(s):

Salary Definition Set	Multiplier
<Base Salary Set>	1

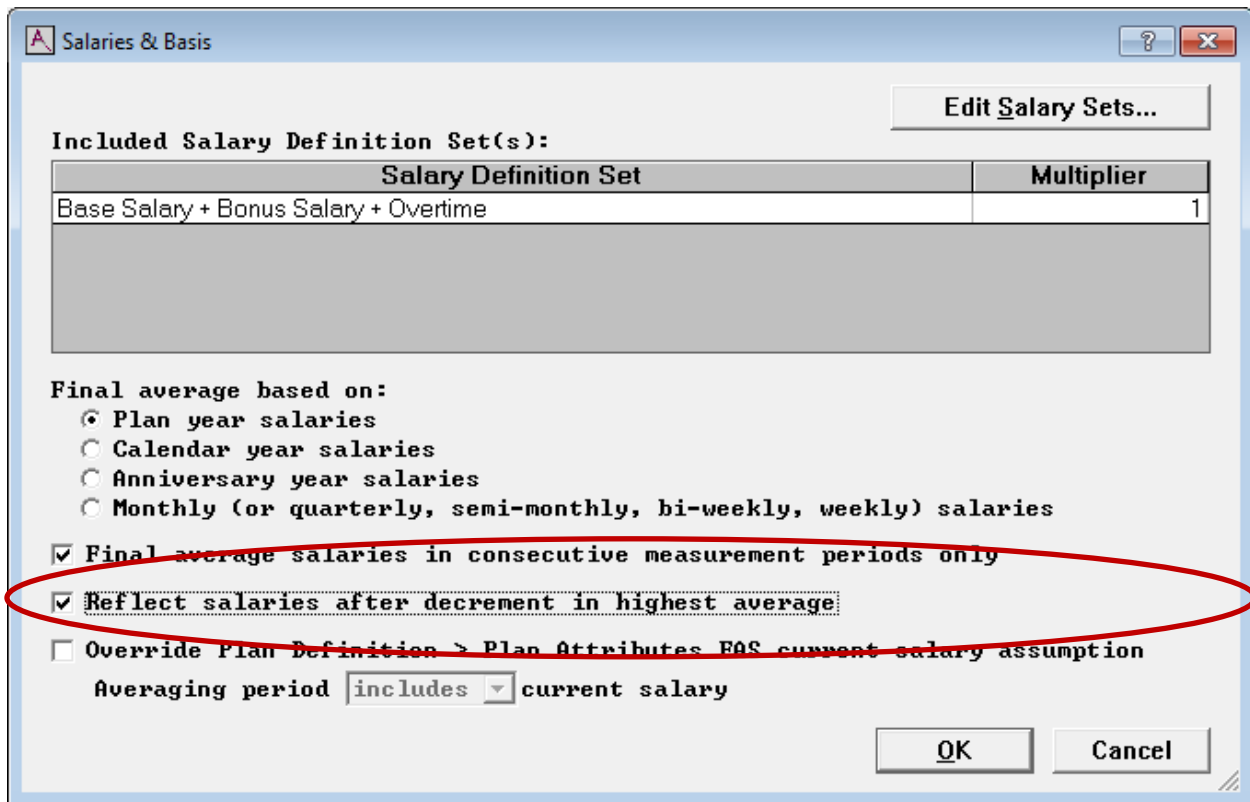
Salary limit: <401(a)(17) maximum>

Prorate salary limit for mid-year decrement

Reflect salary after decrement (rather than freezing at decrement)

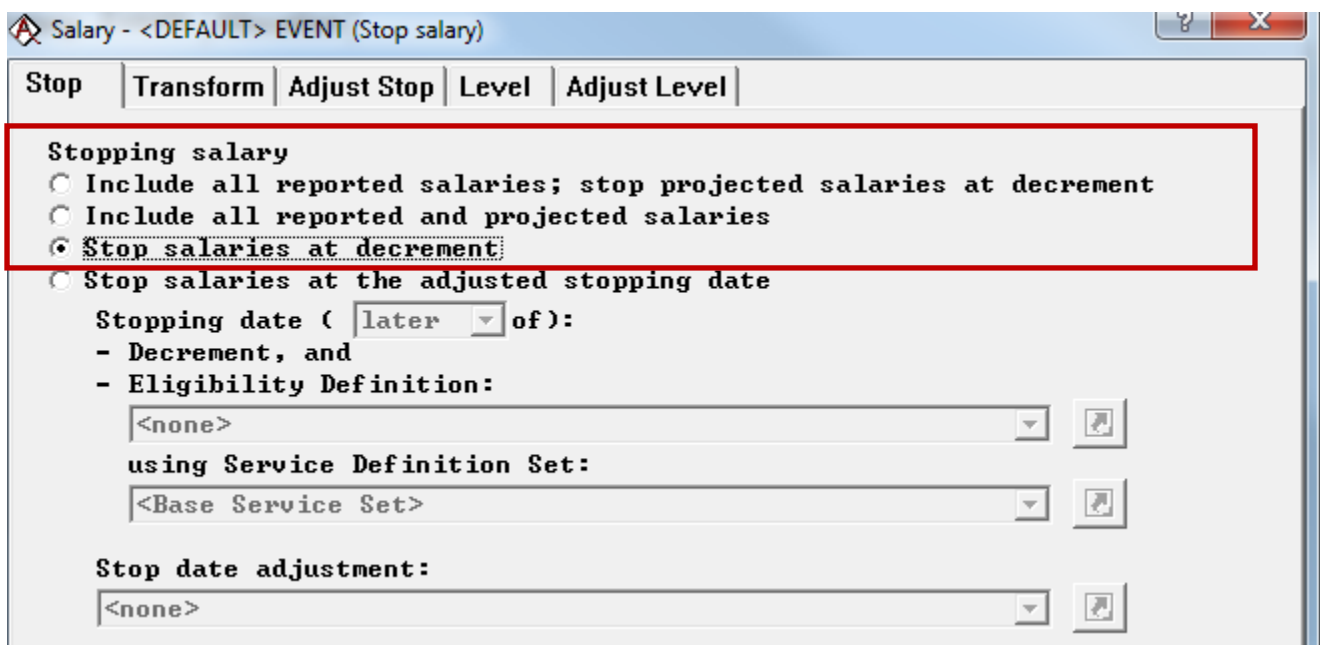
OK Cancel

- ◆ **Salaries after decrement.** The Final Average Salary Custom Operator will now reflect salaries after decrement for estimates. Previously this was only available for final calculations. Note that for this choice to be effective, the referenced Salary Definition Set(s) must also reflect salaries after decrement.



Salary Definitions

- ◆ **The Salary Definition Salary Stop event** dialog box has been enhanced (primarily) to make it easier to use the Salary and FAS custom operator options to reflect salary after decrement. A new option to *Include all reported salaries; stop projected salaries at decrement* has been added. Selecting this option allows you to include reported salaries even if they are reported after decrement, such as unused vacation or sick pay. If a Estimated Benefit Calculation is processed using this parameter and no salary is reported after decrement, projected salaries will be stopped at decrement. The option to *Include all reported and projected salaries* is not new, but has been relabeled for clarity. The new default option, *Stop salaries at decrement*, will ignore any salary reported after decrement.



Output

- ◆ Benefit formula and accrual basis components that have "_DT" (not case sensitive) as the last three letters of the component name will now display in date format in Detailed, Summary, and Output Definitions results.
- ◆ Accrual basis or benefit formula components that reference a date field from the Data Dictionary will now display in date format in Detailed, Summary, and Output Definitions results.
- ◆ Temporary variables and database fields referenced in database field expression Benefit Formula and Accrual Basis Components are now displayed in the Detailed Results, and the expression is included as a footnote to the table.
- ◆ **Re-run results comparison.** (Released via patch in April, 2014) The "Re-run" button for Estimated, Final and Dates/Age/Service calculations now provides more immediately helpful information when the results of the re-run calculation differ from those that were previously saved. The Input Data, External Tables, Summary Results and Output Definition Results are individually compared and any differences (up to a specified MaxDifference amount) are displayed directly in the viewer. New ProAdmin.ini file settings associated with this result comparison are available and will be used if specified. "TestLevel=1" provides the detail mentioned above. "Testlevel=2" adds a summary of changes to the calculation detailed results. Additionally, the ProAdmin.ini file setting "MaxDifference" allows you to indicate how many differences should be captured for each level of comparison noted in the "TestLevel" setting.

ProAdmin

Print... Preview File... Copy Find... Close

Calculation results differ from the prior results:

Run dates:
 - OLD results (5/10/2013 3:08 PM) (User ID: Debbie Benner)
 + NEW results (3/06/2014 3:53 PM) (User ID: Debbie Benner)

External Table differences:
 -[2] C:\PVSRC\RegMaxComp.txt (12/05/2012 11:54 AM)
 +[2] C:\PVSRC\RegMaxComp.txt (11/06/2013 10:57 AM)
 -[6] 30_Year_Treas_Rate.csv (2/08/2013 3:47 PM)
 +[6] 30_Year_Treas_Rate.csv (6/16/2013 10:42 PM)

Summary Result differences:

-[12]	Ret - ALL INTEREST RATES + interest factors	282,072.49	468,081.99	597,364.37	652,272.35
+ [12]	Ret - ALL INTEREST RATES + interest factors	282,072.34	468,081.78	597,364.16	652,272.14
-[19]	INTFAC3	1.0000	1.9148	1.9148	1.9148
+ [19]	INTFAC3	1.0000	1.8541	1.8541	1.8541
-[20]	Intfacs	6.6854	6.6448	6.6448	6.6448
+ [20]	Intfacs	6.5378	6.5008	6.5008	6.5008
-[27]	INTFAC2	2.1610	2.1610	2.1610	2.1610
+ [27]	INTFAC2	2.1044	2.1044	2.1044	2.1044
-[29]	INTFAC4	2.2324	2.1918	2.1918	2.1918
+ [29]	INTFAC4	2.1414	2.1044	2.1044	2.1044
-[42]	Full Lump Sum	282,072.49	N/A	N/A	N/A
+ [42]	Full Lump Sum	282,072.34	N/A	N/A	N/A
-[46]	Full Lump Sum	468,081.99	N/A	N/A	N/A
+ [46]	Full Lump Sum	468,081.78	N/A	N/A	N/A
-[50]	Full Lump Sum	597,364.37	N/A	N/A	N/A
+ [50]	Full Lump Sum	597,364.16	N/A	N/A	N/A
-[54]	Full Lump Sum	652,272.35	N/A	N/A	N/A
+ [54]	Full Lump Sum	652,272.14	N/A	N/A	N/A

Detailed Results are different.

Output Definition Results are identical.

Tools

- ◆ Administrative factors can reference mortality tables with 2D improvement scales.

System

- ◆ Date/time is now formatted using 12 hour (AM/PM) or 24 hour clock per Regional Settings.
- ◆ Timestamps will now be stored relative to UTC rather than local time allowing entries to be in chronological order when work is done in different time zones.
- ◆ "Excel Binary Workbook (*.xlsb)" is an acceptable file type for exporting from or importing to ProAdmin.
- ◆ A new, optional, ProAdmin.ini file setting "UpdateClientFiles" is now available. This new setting controls whether a user has the capabilities to update the ProAdmin client files from a previous version to the new version. If missing or set to "1" (i.e. Yes), a version update can proceed. If the field is set to "0" (i.e. No), the user will be presented with a message informing them that they are not allowed to update client files.
- ◆ A new ProAdmin.ini file setting "SQLFixDupFlds" is now available. This new setting allows you to alias fields that appear more than once in the select statement. The parameter was added because some versions of SQL Server no longer allow a SELECT with duplicate fields.
- ◆ There is a "View" button on backdoor selection boxes with the same functionality as the "View" button on the command bar.
- ◆ **Under 15.** The age 15 minimum age requirement has been lifted. Calculations that previously would not run because the member at some point was under age 15 will now calculate accurate results. Beneficiaries under age 15 will be evaluating using zero mortality rates before age 15.

Changes Log

- ◆ Be sure to read the changes log (see the "Changes Log (ProAdmin).doc" file in the ProAdmin directory) about updates to certain calculations that may change results.



Two Greenwich Office Park
Greenwich, CT 06831

tel: (203) 861-5530
fax: (203) 861-5531
email: support@winklevoss.com
website: www.winklevoss.com

#GETTABVAL Operator

ProAdmin's new #GETTABVAL operator, which is currently only available in salary and service transformation expressions, simplifies the coding and maintenance of complex plans with provisions such as dollar multipliers that change frequently. The operator allows such values to be stored (and updated) externally to the system in a comma separated (i.e., .CSV) file. The #GETTABVAL operator is then used to search the file for a value based on a set of lookup values. If no match is found, the operator returns 0.

The syntax for using the operator is:

a #GETTABVAL b

where:

a is the name, in quotes, of the CSV file containing the table; it does not include a path or file extension (i.e., .CSV).

b is a parenthesized list of temporary variables (created by temporary assignment) that are used to look up a value in the table specified by a

If the set of lookup values (b) corresponds to a value in the specified table (a), then that value is returned; otherwise, 0 is returned.

The CSV file containing the values and their associated lookup values must have the following characteristics:

- The file must be in one of these directories (and this is the search order):
 1. Client directory
 2. User directory
 3. System directory (i.e., directory where ProAdmin is installed)
 4. Historical interest rate directory HIRTDIR (as specified in the proadmin.ini file; only used if HIRTCODE=2)
 5. Regulatory files (per RegPath setting for server calculations)
- The 1st row may contain descriptors (e.g., Date, Plan, Location, Rate)
- The 1st or 2nd row may contain match indicators which define the type of match for a column:
 1. E = exact match (e.g., Location or Plan)
 2. R = range match (e.g., Effective Date). If the date lookup range is 1/1/2000 to 12/31/2000, the indicator for this range is 1/1/2000. If a salary lookup range for a value is from \$20,000 up to but not including \$25,000, the indicator for the \$20,000 value is 20,000 and the indicator for the \$25,000 value is 25,000.
 3. I = ignore this column. Since #GETTABVAL only works with dates and numbers, you might, for example, want to include a column with the location name next to the column with the location code. Use an "I" to tell #GETTABVAL to ignore the column with the descriptor.
 4. V= value. This column contains the values you want #GETTABVAL to retrieve.
- If there is no match indicator row, the first column is assumed to contain a range (R), any intermediate columns are assumed to contain exact matches (E), and the last column is assumed to contain the values (V).
- There can only be one value (V) column.
- After the ignored columns are dropped, any blank lines and rows containing characters are dropped/ignored. This allows you to include comments in the file.

- Each row must contain exactly the same number of values (after ignored columns, blank rows, and rows containing characters are dropped).

Once the processing detailed above is complete, all of the values remaining in the table are numbers, ProAdmin codes (which are numbers), and dates, where the numbers and dates are assumed to be formatted in the Windows local setting format. The table is then evaluated as follows:

- The order of the table columns (after ignored columns, blank rows, and rows containing characters are dropped) must match the order of the list of temporary variables that make up the right argument to #GETTABVAL.
- When looking up values, exact matches are performed first, and then the range values are used.

Some additional pointers when using #GETTABVAL are:

- If #GETTABVAL tries to load a CSV file that happens to be open in Excel, you'll get a File busy (being used by another application or user) message. This is because Excel "locks" a CSV file without allowing read rights. However, if the file was open in NotePad, you won't get this message because NotePad only locks the file when you save it. Thus, opening the file in NotePad will facilitate results checking.
- When you redisplay salary or service transformation expression detailed results for an old saved estimate that used #GETTABVAL, the current version of the CSV file is used, not the version that was available when the original calculation was performed. If the old/new timestamps are different, a warning message will be displayed in the table footnotes.

Range vs Exact Matches

For the discussion which follows, consider these CSV files:

Year,Rate	Year,Limit
R,V	E,V
1990,100	1990,100
2000,115	2000,115
2010,125	2010,125

If you try to look up the value for 2001 in the 1st CSV file where year is treated as a range, 115 will be returned. That's because 2001 is greater than or equal to 2000 and it's strictly less than 2010.

If you try to look up the value for 2001 in the 2nd CSV file where year is treated as an exact match, 0 will be returned. That's because 2001 is NOT equal to 1990, 2000, or 2010. (A match occurs ONLY if 2001 is exactly one of the years, and it isn't.)

Range Order Matches

For the discussion which follows, consider these CSV files:

```
Effdate, Age, Value
R, R, V
1/1/1900, 0, 1.5
1/1/2000, 0, 1.7
1/1/2000, 55, 2.0
1/1/2000, 60, 3.0
1/1/2005, 0, 0.4
1/1/2005, 55, 0.5
```

```
Age, Effdate, Value
R, R, V
0, 1/1/1900, 1.5
```

```
0,1/1/2000,1.7
55,1/1/2000,2.0
60,1/1/2000,3.0
0,1/1/2005,0.4
55,1/1/2005,0.5
```

The difference between these tables is that the Age column is the 2nd column in the first file and the 1st column in the 2nd file. Looking at these tables, you might expect to always get the same value, but you will not. Here is the Age in the 1st CSV file as #GETTABVAL uses it (the range columns are sorted in descending order):

```
Age, Effdate, Value
R, R, V
0, 1/1/1900, 1.5
0, 1/1/2000, 1.7
0, 1/1/2005, 0.4
55, 1/1/2000, 2.0
55, 1/1/2005, 0.5
60, 1/1/2000, 3.0
```

If you try to look up the 12/31/99 value for someone born 3/17/1939 (age 60.8) using the 1st CSV file (where Effdate is the first column), #GETTABVAL returns 1.5. It looks for the appropriate row(s) for 12/31/99 and finds only one: (1/1/1900,0,1.5). Since age 60 is after age 0, 1.5 is returned

If you try to look up the age 60.8 value as of 12/31/99 using the 2nd CSV file (where Age is the first column), #GETTABVAL returns 0. It looks for the appropriate row(s) for age 60.8 and finds only one: (60,1/1/2000,3.0). Since 12/31/99 is before 1/1/2000, there is no value to lookup, and 0 is returned.

Examples

Example 1 – Use the 401(a)(17) maximum compensation limits to limit the salary in a salary transformation

First, save a copy of RegMaxComp.txt as a CSV file called RegMaxComp.CSV because #GETTABVAL only works with CSV files. The file can be saved in the client directory or in the directory containing the original RegMaxComp.txt file. The file should look something like this:

```
1989,200000
1990,209200
1991,222220
1992,228860
1993,235840
1994,150000
1995,153255
1996,157305
1997,161940
1998,165510
1999,168150
2000,172095
2001,178125
2002,200000
2003,203180
2004,207660
2005,213320
2006,221480
2007,228880
2008,234280
2009,246700
2010,246700
2011,246700
```

2012,254780
2013,259100

If you want, you can add column headings and a row containing column types:

```
Year,Limit  
E,V  
1989,200000  
1990,209200  
1991,222220  
...
```

Here is a salary transformation that uses the salary limit:

```
BegPlanYear := #BEGMTH (2 #FSTBUSDAY #DATE) & ; beginning of plan year  
Year := #YEAR BegPlanYear &  
SalLimit := `RegMaxComp' #GETTABVAL (Year) &  
  
#THIS #MIN SalLimit
```

Example 2 – The assumed hours per day varies over time by plan, location and age.

LocationCode and PlanCode are effective date array coded fields that indicate in which locations and plans the member has been. The value in each of these arrays is the numeric ProAdmin code.

LocationCode contains

Date	Code
1/01/1980	1
11/17/1990	2
9/3/1999	1
1/01/2012	2

and PlanCode contains

Date	Code
1/01/1980	1
7/21/1995	2
1/01/2012	3

The HoursByPlanLocAge.CSV file might look like this:

```
Date, Plan, Loc, Age, Value  
R, E, E, R, V  
1/1/2000,1,1,0,8.1  
1/1/2000,1,1,55,8  
1/1/2000,1,1,60,7.9  
1/1/2000,2,1,0,7.7  
1/1/2000,2,1,55,7.6  
1/1/2000,2,1,60,7.9  
1/1/2000,3,1,0,8.1  
1/1/2000,3,1,55,7.7  
1/1/2000,3,1,60,7.8  
...  
1/1/2000,1,2,0,7.8  
1/1/2000,1,2,55,7.7  
1/1/2000,1,2,60,7.7  
...
```

or this:

```

Date, Plan,, Loc,, Age, Value
R, E,I, E,I, R, V
1/1/2000,1,Salaried,1,NYC,0,8.1
1/1/2000,1,Salaried,1,NYC,55,8
1/1/2000,1,Salaried,1,NYC,60,7.9
1/1/2000,2,Hourly,1,NYC,0,7.7
1/1/2000,2,Hourly,1,NYC,55,7.6
1/1/2000,2,Hourly,1,NYC,60,7.9
1/1/2000,3,Admin,1,NYC,0,8.1
1/1/2000,3,Admin,1,NYC,55,7.7
1/1/2000,3,Admin,1,NYC,60,7.8
...
1/1/2000,1,Salaried,2,LA,0,7.8
1/1/2000,1,Salaried,2,LA,55,7.7
1/1/2000,1,Salaried,2,LA,60,7.7
...

```

Here's what an hours transformation might look like

```

D_DT          := #DATE & ; calculation dates
LOC           := LocationCodes &
PLAN         := PlanCodes &
Age           := D_DT #YEARDIF DateOfBirth &
Rates        := 'HoursByPlanLocAge' #GETTABVAL (D_DT, PLAN, LOC, AGE) &
DaysWorked   := 0 #MAX [(1 + #DATE) - DateOfHire] &
NetDaysWorked := 0 #MPNET DaysWorked &
NetHoursWorked := NetDaysWorked * Rates &
HoursInYear   := 1 #MPSUM NetHoursWorked &
#IF HoursInYear < 501
#THEN 0
#ELSEIF HoursInYear < 1000
#THEN .5
#ELSE 1 #MIN (HoursInYear / 1800) #ENDIF

```

If you wanted to freeze the rates in effect at 12/31/2012, but reflect any age, plan and location changes that occur after 12/31/2012, you only need to freeze the date:

```

D_DT          := 12/31/2012 #MIN #DATE & ; freeze calc. dates at 12/31/12
LOC           := LocationCodes &
PLAN         := PlanCodes &
Age           := D_DT #YEARDIF DateOfBirth &
Rates        := 'HoursByPlanLocAge' #GETTABVAL (D_DT, PLAN, LOC, AGE) &
DaysWorked   := 0 #MAX [(1 + #DATE) - DateOfHire] &
NetDaysWorked := 0 #MPNET DaysWorked &
NetHoursWorked := NetDaysWorked * Rates &
HoursInYear   := 1 #MPSUM NetHoursWorked &
#IF HoursInYear < 501
#THEN 0
#ELSEIF HoursInYear < 1000
#THEN .5
#ELSE 1 #MIN (HoursInYear / 1800) #ENDIF

```