

# Getting Started with ProAdmin

## *How to Run a Calculation*

---

*benefit calculation software for defined benefit plans*

**ProAdmin**<sup>®</sup>

Version 3.15

This document was prepared to assist users of WinTech's ProAdmin software system. Its contents may not be used for any other purpose without written permission. The material contained herein is supplied without representation or warranty of any kind. Winklevoss Technologies therefore assumes no responsibility and shall have no liability arising from the supply or use of this document or the material contained herein.

Copyright © 2017 Winklevoss Technologies  
Printed in the United States of America. All rights reserved.  
Unauthorized reproduction is strictly prohibited.

Excel is a registered trademark of Microsoft Corporation.

# Contents

---

- Introduction..... 1**
  - Overview ..... 1
  - Performing the Steps..... 1
  
- Step 1: Open a Client..... 4**
  - Alternative 1: Open a Template Client ..... 4
  - Alternative 2: Create a New Client..... 7
  
- Step 2: Data Concepts ..... 9**
  - Data..... 9
  - Data Dictionary..... 9
  - Database Linkage..... 11
  
- Step 3: Census Specifications..... 15**
  
- Step 4: Plan Benefits..... 16**
  - Plan Definition..... 16
  - Service Definition Sets ..... 18
  - Service Definitions ..... 19
  - Salary Definition Sets..... 20
  - Salary Definitions ..... 21
  - Salary History ..... 23
  - Benefit Definitions..... 23
  - Payment Forms ..... 23
  - Benefit Formulas ..... 24
  - Benefit Formula Components..... 27
  - Final Average Salary Benefit..... 28
  - Hourly Benefit ..... 35
  - Cash Balance Benefit..... 38

**Step 5: Projection Assumptions ..... 46**

**Step 6: Output Definitions ..... 47**

**Step 7: Run an Estimate..... 48**

Set Up an Estimate..... 48  
Inspect the output..... 50

**Appendix A: Expressions ..... 52**

Expression Basics ..... 52  
Expression Help..... 54  
Missing Values ..... 55  
Date Arithmetic ..... 55  
Relational Operations ..... 57  
Searching Character Data ..... 58  
Logical Operations..... 60  
Array Operators ..... 61  
Managing Complicated Expressions ..... 62

**Appendix B: Libraries..... 64**

Libraries..... 64  
Audit Trail ..... 67  
Other Consistency Checks..... 68

**Appendix C: Projects ..... 69**

Projects ..... 69  
Managing Projects ..... 70  
Universe Project ..... 71  
Unhiding Objects..... 71  
Object Descriptions ..... 72

**Appendix D: Shortcuts ..... 73**

Mnemonics ..... 73  
Dialog Box Basics ..... 73  
List Boxes ..... 74  
Drop-down List Boxes..... 75  
Number, Date, and Text Fields..... 76  
Check Boxes ..... 76

Radio Buttons .....	76
Spreadsheet Fields .....	77
From-To Tables .....	78
Tabs .....	79

**Appendix E: XML Linkages..... 81**

XML Database Linkage.....	81
XML Output Linkage .....	87

**Appendix F: Output Definitions..... 97**

**Appendix G: Fulfillment Tool ..... 99**

**Appendix H: ProAdmin Help..... 100**

# Introduction

---

## Overview

This book provides step-by-step instructions for “booting up” plans with ProAdmin. The focus is on running an estimate (or final) calculation starting with data and ending with fulfillment (i.e., forms and letters).

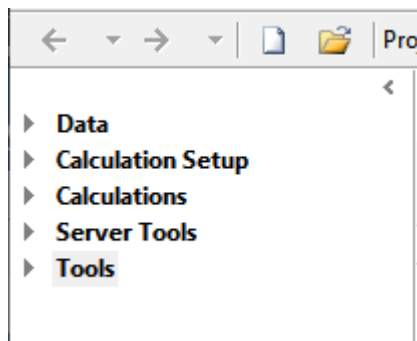
The major steps in the process are:

- |                                  |  |
|----------------------------------|--|
| <b>1. Client</b>                 | Open or create a ProAdmin client                             |
| <b>2. Database Linkage</b>       | Create data fields and link them to a database               |
| <b>3. Census Specifications</b>  | Map linked data to relevant participant parameters           |
| <b>4. Plan Definition</b>        | Enter plan provisions  |
| <b>5. Projection Assumptions</b> | Enter interest rates, salary scale, etc.                     |
| <b>+ 6. Output Definitions</b>   | Define fields for fulfillment merge ( <i>optional step</i> ) |
| <b>7. Run an Estimate</b>        | Calculate benefits   |

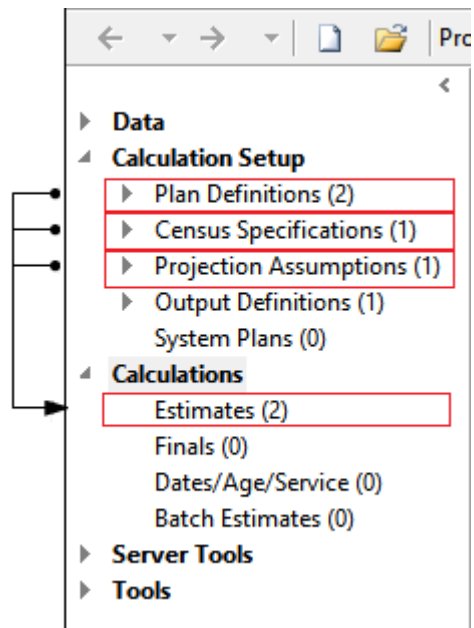
## Performing the Steps

Step 1 is performed through the **File** menu.

Step 2 is created through the **Input** menu, or alternatively using the **Shortcuts** pane:



Steps 3 – 7 create a set of **Census Specifications**, a **Plan Definition**, and a set of **Projection Assumptions** that, when combined, are used to calculate an **Estimated Benefit Calculation**. These steps can be performed using the Input and Execute Menus, or, alternatively, using the Shortcuts pane:



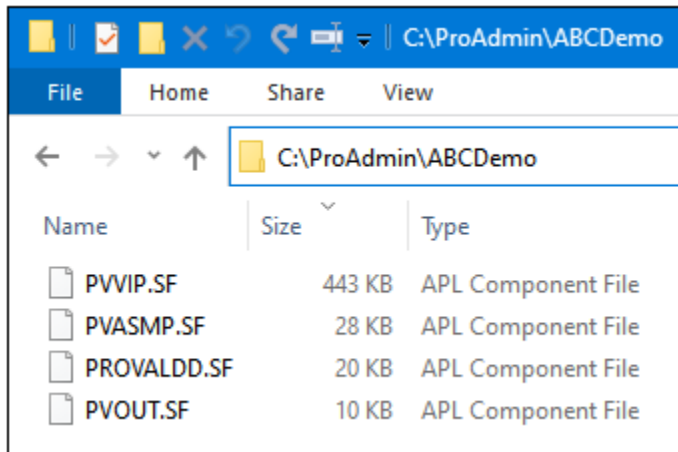




# Step 1: Open a Client

Your work in ProAdmin is stored in “clients”. To open an existing client use **File | Open Client**; to create a new client, use **File | New Client**.

A ProAdmin client is stored in a directory, for example, C:\ProAdmin\ABCDemo, containing:



Library files storing the client’s benefit definitions, projection assumptions, calculation results, etc. The file names are always provaldd.sf, pvasmp.sf, pvout.sf, and pvvip.sf.

Data for the plan participants birthdates, salaries, benefits, etc... is stored outside of ProAdmin. The database type and names are specified by the user.

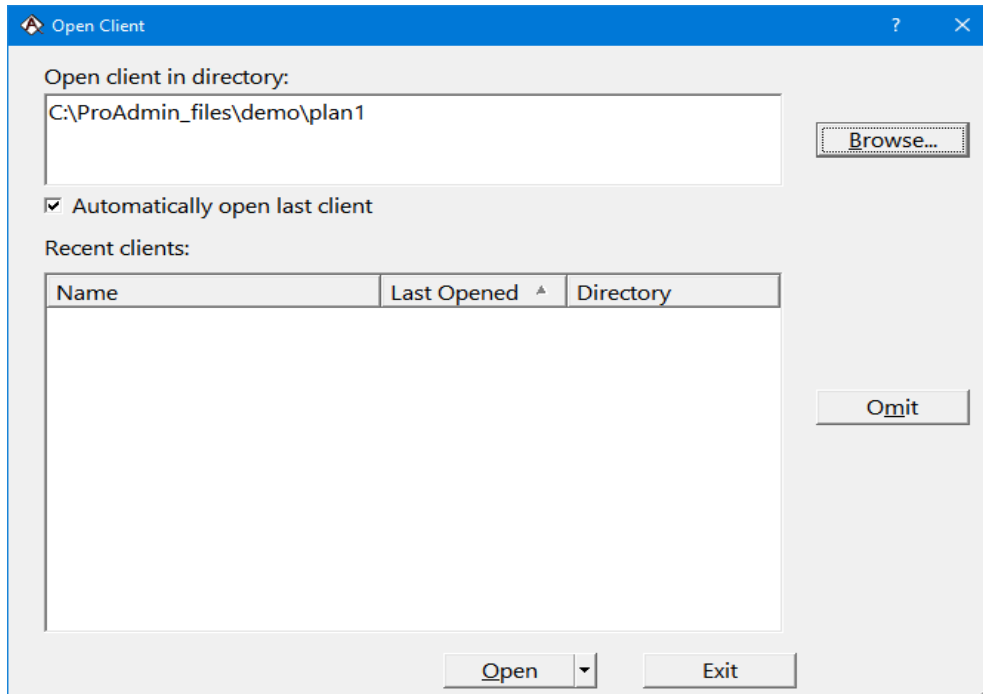
Note: Since the library filenames are always the same, each ProAdmin client must be kept in a separate directory.

## Alternative 1: Open a Template Client

If you have a template client that contains standard data dictionary field names, benefit formula components, etc... or an existing ProVal client that you would like to convert follow these steps. Otherwise, follow the instructions below for Alternative 2: Create a New Client.

- 1 In **Windows Explorer**, make a copy of the template client folder. For example, copy C:\CLIENTS\TEMPLATE to C:\CLIENTS\ABC
- 2 From ProAdmin’s **File** menu, choose **Open Client**.

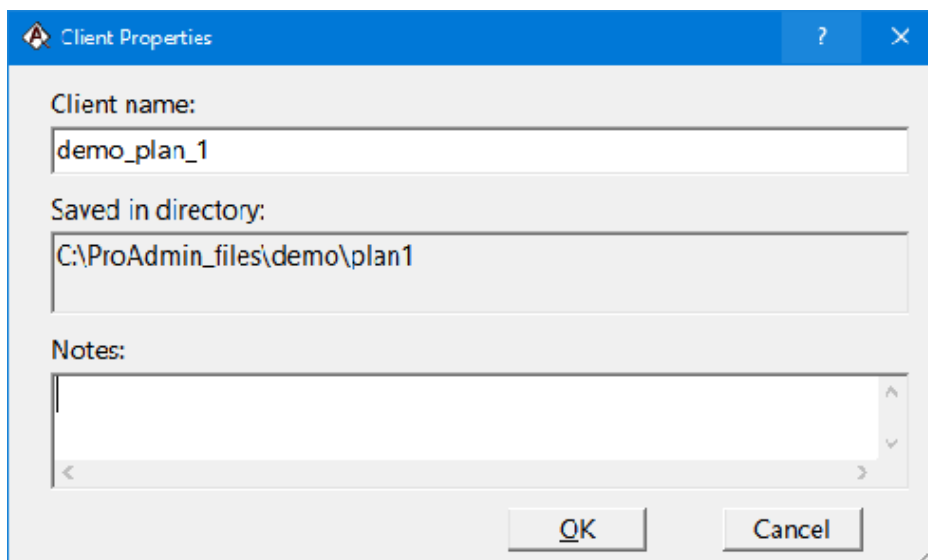
## Step 1: Open a Client



**Browse** to (or type in) the new client **directory** you created in step 1.

Click **Open**.

- 3** If a message appears saying this client “was developed under a previous version of ProAdmin,” click **Yes** to update the client.
- 4** From the **File** menu, choose **Properties**.

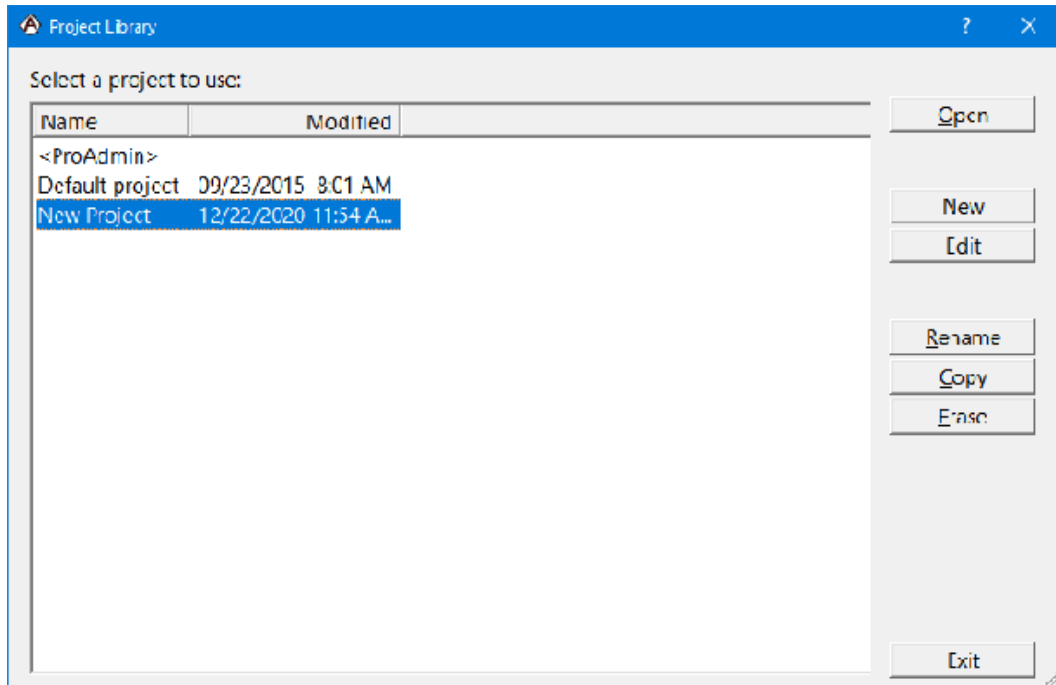


Change the **Client name**.

Click **OK**.

Step 1: Open a Client

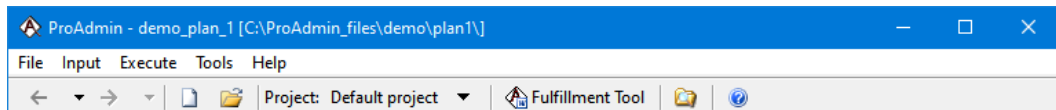
- 5 From the **File** menu, choose **Change Project**.



Under user defined **Projects** may be created to determine which objects are visible within each library. You may double-click (or highlight then click **Open**) to switch to a different project.

For more information, see Appendix C: Projects below.

- 6 The client name and folder should appear in the caption of the main ProAdmin window; while the **Project** appears in the tool bar.

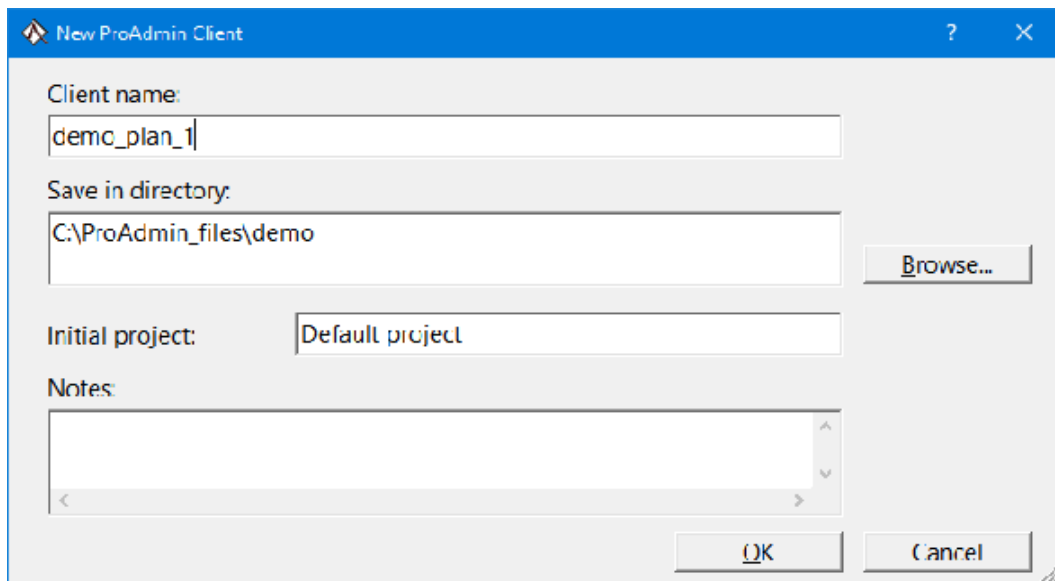


The version number and version date should appear in the status bar at the bottom of the ProAdmin window. You can begin to work, and everything will be saved in your new client folder.



## Alternative 2: Create a New Client

- 1 From the **File** menu, choose **New Client**.

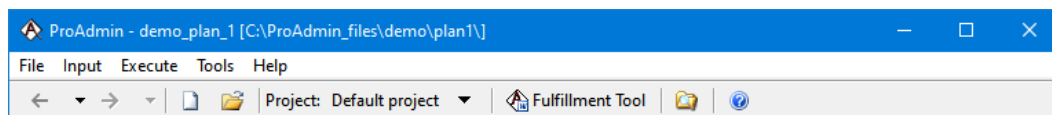


- 2 Provide a **Name** and **Directory** for the client.

For example, “demo\_plan\_1” in directory C:\proadmin\_files\demo\plan1. It is ok to provide a directory that does not exist; ProAdmin will create it for you.

- 3 Provide a name for your **Initial project**. For example, you might create the “Standard Retirement Plan” to hold all of the information for the standard calculations and different projects for QDRO’s or Early Retirement Windows.
- 4 Click **OK**.

Once the client file is initialized and opened, the name folder of the client should appear in the caption of the main ProAdmin window; while the project appears in the toolbar.



The version number and version date should appear in the status bar at the bottom of the ProAdmin window. You can begin to work and everything will be saved in your new client folder.





## Step 2: Data Concepts

### Data

The products in the ProAdmin suite require an external data source for the member data in order to process a calculation. The products connect to the data in two (2) distinct ways. ProAdmin uses Microsoft's ActiveX Data Object (ADO) to connect to and retrieve the member information from the external data source when a calculation is processed. Detailed information on how to create the connection between the Data Dictionary and the data source is described under the Database Linkage topic below. ProAdmin Server and *pensionASAP* expect that the member data will be presented as part of the calculation request in the form of an Extensible Markup Language (XML) document. The steps to create the connection between the Data Dictionary and the XML Schema Document are described in [Appendix E: XML Linkages](#).

### Data Dictionary

ProAdmin can attach to, and accept data in, virtually any structure and has no fixed-format or fixed-data requirements. In order to achieve this flexibility, the system makes use of a **Data Dictionary**. The Data Dictionary gives ProAdmin a user-specified name for each relevant piece of data and describes the type of data. Once a field has been added to the data dictionary, it can be referenced throughout the system as needed for defining benefits, calculating service, etc. The only restrictions on subsequent usage will be based on the data type. For example, when the system is expecting a salary history field, it will only allow numeric array Data Dictionary fields (i.e., where, as in the example below, the **Field type** is numeric, and the **Date dependent array** checkbox is set).

**Data Dictionary** field attributes are illustrated and discussed below.

The screenshot shows a dialog box titled "Field Attributes - [AccBen]". It contains the following fields and options:

- Field name:** AccBen
- Description:** Accrued annual benefit for prior vested members
- Field type:** Numeric (dropdown menu)
- Formatting style:** 999,999,999.99
- Date dependent array**
  - Start / Stop dates
  - Effective date
- Return:** All values (dropdown menu)
- Assign coded labels** (with a "Coded Labels..." button)
- Display first when reviewing data**

Buttons at the bottom: View, Replace, Save As New, Erase, Cancel.

- **Field Name:** The field name may contain letters, digits, and underscore ( \_ ) symbols, but it may not contain other symbols or blanks, and it must begin with a letter. Although field names are displayed in upper and lowercase letters, the field name is not case sensitive. As an example, if the data dictionary contains a field named “BirthDate” the following would all allow you to reference this field “BirthDate”, “birthdate”, “BIRTHdate”, and “BIRTHDATE”.
- **Description:** In addition to the field name, you can provide a one-line description to remind you, or a colleague, what the field represents. The description may include any characters, including spaces.
- **Field Type:** The field type can be numeric, date, or character.
  - **Numeric:** A single number, or array of numbers, such as an employee's salary, which can be either an integer or decimal number.
    - **Scalars:** A field with one distinct value such as a grandfathered benefit.
    - **Date dependent array:** An array of values with a time dimension. The time dimension may be defined using start and stop dates or effective dates. If the **Effective date** option is selected, the date is considered to be the beginning of the period. The **Return** value determines which array values to return : **All values, Most recent value, Largest value, or Smallest value.** An example of date dependent array data would be a member’s historical employment hours.
    - **Coded labels:** A field with distinct values or categories. Each value is identified by both a character *label* and a numeric *code*. Some examples:

<b>Sex</b>	
<b>Label</b>	<b>Code</b>
Male	1
Female	2

<b>Location</b>	
<b>Label</b>	<b>Code</b>
Greenwich, CT	1
Milford, CT	2
Other (NY,CT,MA)	3

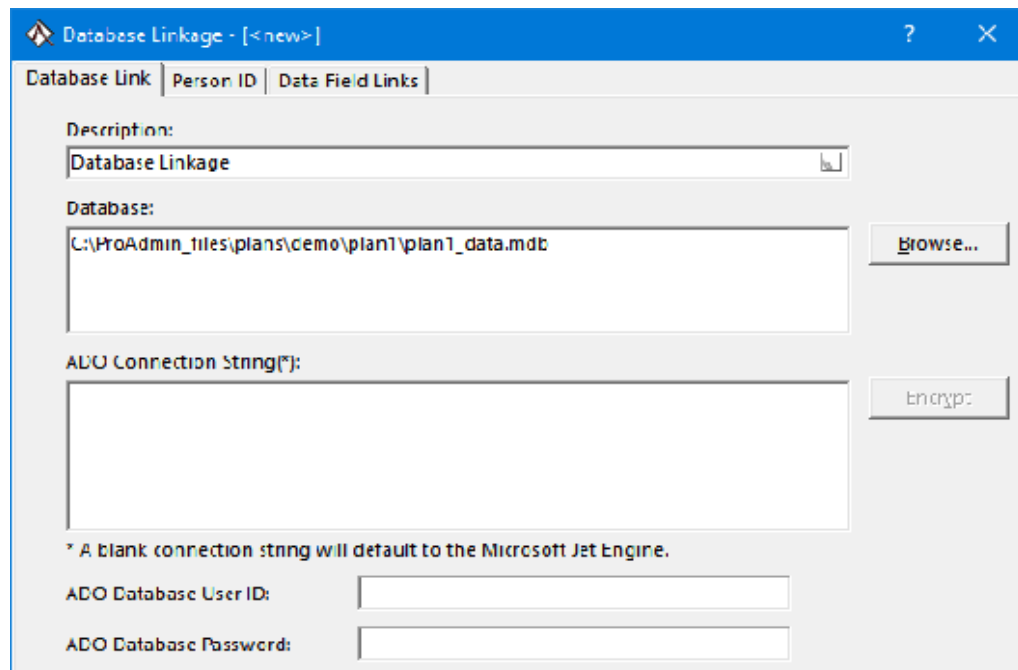
When a coded field is displayed, the character labels are used rather than the numeric codes for ease of use. When writing an expression, e.g., “Location #in (1,3)”, the numeric codes are used.

- **Date:** A date field represents a year, month, and day, such as an employee's date of birth. Internally, a date is represented as an integer giving the number of days from January 1, 1900 to the date in question. (Dates before 1900 are represented as negative numbers). This representation makes ProAdmin year-2000 compliant.
- **Character:** A character string, such as the member's name. The string may have any number of characters and may include spaces.

- **Formatting Style:** For numeric fields, you can specify a format to tell ProAdmin how you want the number displayed. The format is an example of how the number should appear in output, such as “\$9,999.99.” Your example is used to determine the following parameters:
  - How many columns are required to display the field. (This is determined from the number of non-blank characters. In the example above, nine columns would be used.)
  - How many digits are displayed to the right of the decimal point. (In the example above, two decimal digits would be displayed.)
  - Whether or not to insert commas in the number. (Commas will be displayed if your example has any commas.)
  - Whether to put a dollar sign to the left of the number. (A dollar sign will be displayed if your example has one.)

## Database Linkage

Once the Data Dictionary is completed, create a **Database Linkage** to link the fields defined in the **Data Dictionary**. The **Database Linkage** library can be accessed through **Input | Database | Database Linkage** on the menu or from the **Shortcuts** pane. Database Linkage has three main tabs of information to complete: Database Link, Person ID, and Data Field Links.



The screenshot shows a dialog box titled "Database Linkage - [<new>]". It has three tabs: "Database Link", "Person ID", and "Data Field Links". The "Database Link" tab is selected. The dialog contains the following fields and controls:

- Description:** A text box containing "Database Linkage".
- Database:** A text box containing "C:\ProAdmin\_files\plans\demo\plan1\plan1\_data.mdb". To its right is a "Browse..." button.
- ADO Connection String(\*):** An empty text box. To its right is an "Encrypt" button.
- ADO Database User ID:** An empty text box.
- ADO Database Password:** An empty text box.

Below the "ADO Connection String(\*)" field, there is a note: "A blank connection string will default to the Microsoft Jet Engine."

**Database Link** establishes the **Database Path and Name** with which to link, the (Active X Data Object) **ADO Connection String**, **ADO Database User ID**, and **ADO Database Password**. The only required field on this dialog box is the **Database Path and Name**. The **ADO Connection String** is only required if the database is not recognized by the

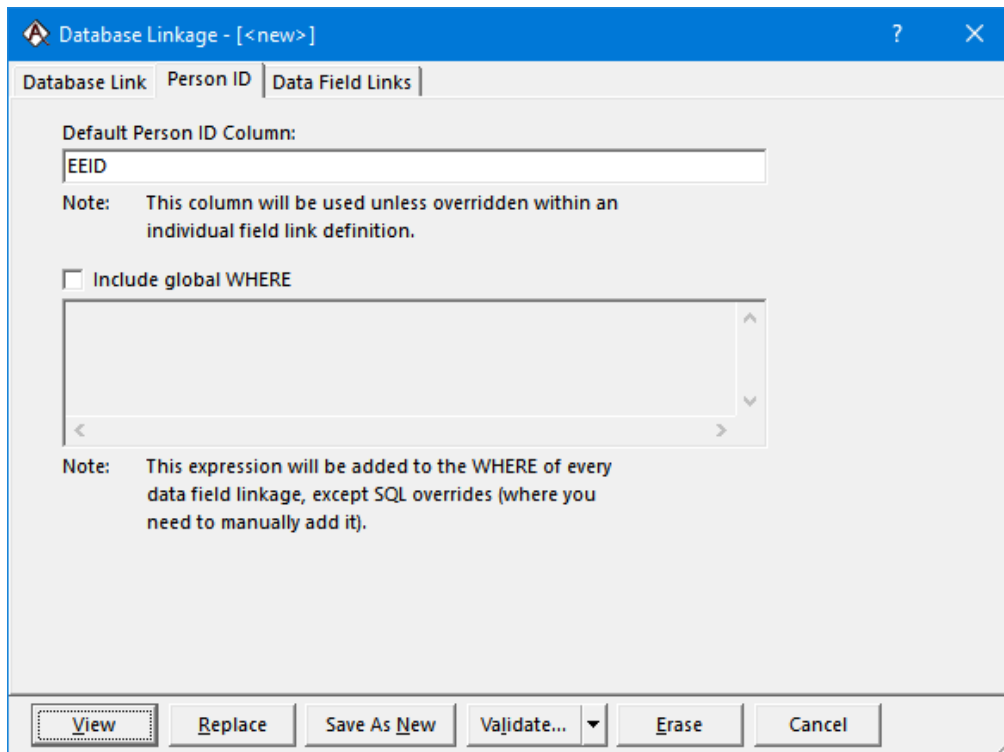


## Step 2: Data Concepts

Microsoft Jet Engine (ex: .mdb). ADO Connection Strings are required when the database is saved in a newer Microsoft Access format (.accdb) or written in SQL Server. The ADO Database User ID and Password are only necessary if the database security requires a User ID and Password to access the information.

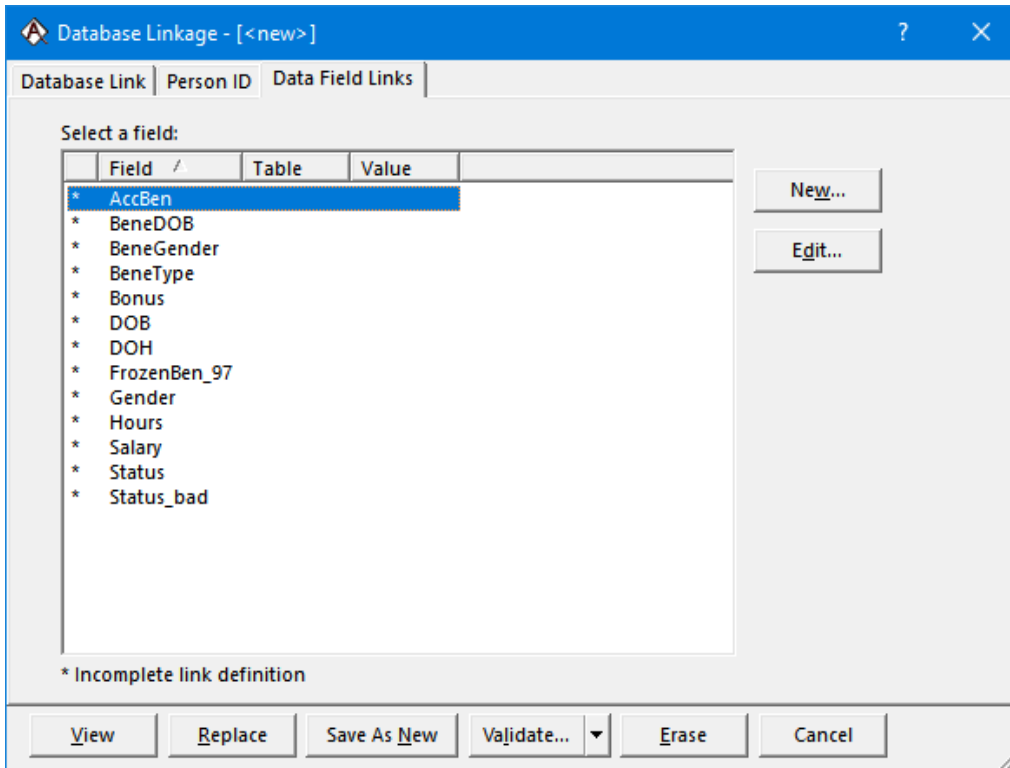
**Password.** The only required field on this dialog box is the **Database Path and Name**. The **ADO Connection String** is only required if the database is not recognized by the Microsoft Jet Engine (ex: .mdb). ADO Connection Strings are required when the database is saved in a newer Microsoft Access format (.accdb) or written in SQL Server. The ADO Database User ID and Password are only necessary if the database security requires a User ID and Password to access the information.

**Person ID** is the default column (or field) used to match records containing individual member data across multiple tables. For example, if all of tables in your database contain a column called EEID you would enter that column name here. This field can be overridden if desired by entering a different column name in the Person ID field on the Data Field Links dialog box. Please note that even though you can change the field name, the value in the field must be the same for all tables so that the data can be retrieved.

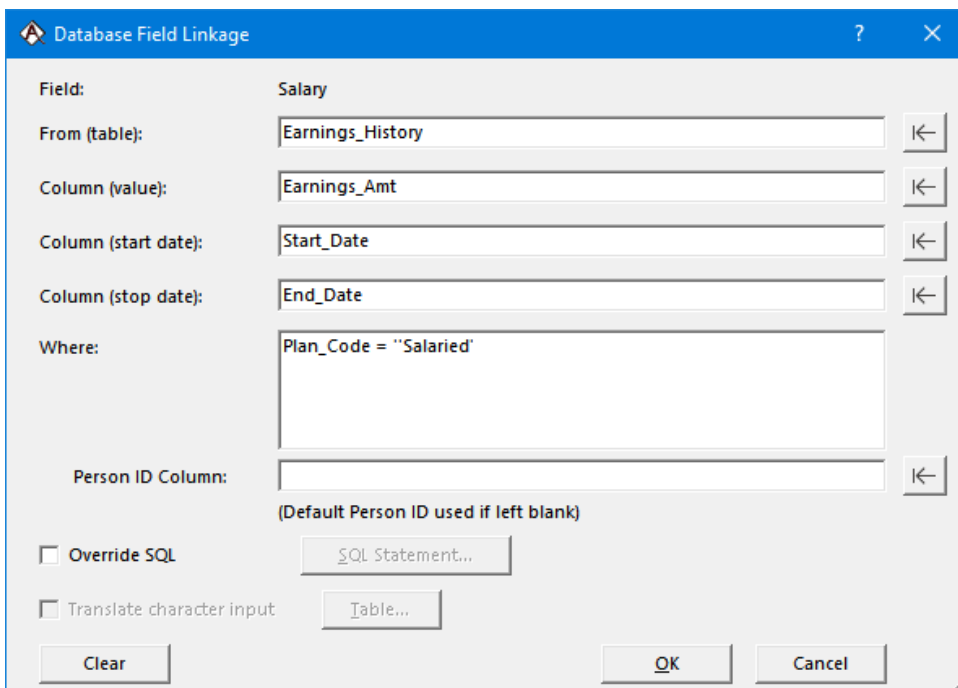



**Data Field Links** maps the ProAdmin fields created in the **Data Dictionary** to the appropriate table and column(s) within the linked database (ie: plandata.mdb). This tab displays all fields in the Data Dictionary. Any entry with a preceding asterisk (\*) indicates that the field does not have a complete database link definition.

## Step 2: Data Concepts



Click on a field name to complete the **Database Field Linkage** definition:



- **From (table)** is the table in the linked database that contains the data for the field. The backdoor button  displays a complete list of tables in the database from which to select.

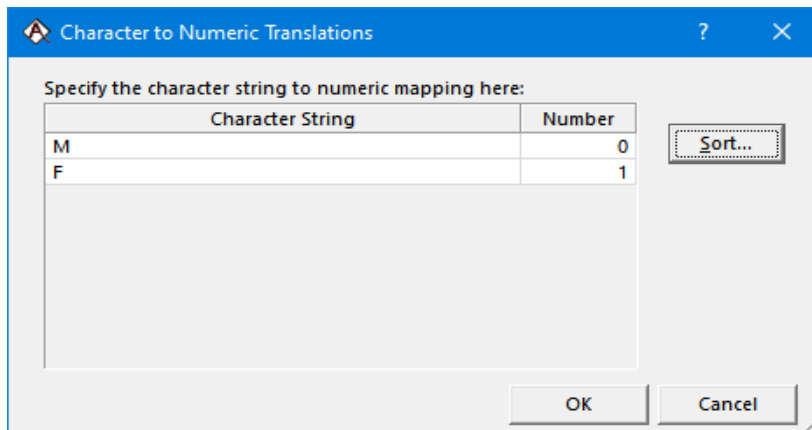
## Step 2: Data Concepts

- **Column (value)** is the column in the referenced table that contains the data value(s).
- **Column (effective date)** is the column in the referenced table that contains the effective date for fields defined as effective date arrays (See Data Dictionary above).
- **Column (start date)** and **Column (stop date)** are the columns in the referenced table that contain the start and stop date, respectively, for fields defined as start/stop date arrays (See Data Dictionary above).
- **Where** is a condition, or filter, which restricts the retrieval of data to only those records (or rows in the table) which satisfy the condition. For example, the referenced table has a field called Plan\_Code to differentiate the different types of earnings stored in the table. To populate the ProAdmin Data Dictionary field called Salary for only those records whose lan\_Code=Salaried, type the following:

```
Plan_Code = 'Salaried'
```

Note: Field names and conjunctions may be case sensitive depending the type of linked database. For example, Plan\_Code ='Salaried' and PLAN\_CODE='SALARIED' return the same value for a Microsoft Access database.

- **Person ID Column** overrides the default column (or field) used to match records containing individual member data across multiple tables for the selected field (See Person ID above). It is required if the default PersonID field is not defined on the referenced table.
- **Override SQL** indicates the data should be retrieved from the linked database using SQL code specified in the **SQL Statement**.
- **Translate character input** allows character values in the linked database to map to a numeric field (as defined in the Data Dictionary). The character values are converted to numeric using the **Table**. For example, ProAdmin requires a numeric coded field to be used to define gender; however, the linked database uses the character values 'M' and 'F' to indicate gender. Then, the character values 'M' and 'F' may be converted to the numeric values 1 and 0 by defining the table as follows



Character to Numeric Translations

Specify the character string to numeric mapping here:

Character String	Number
M	0
F	1

Sort...

OK Cancel

## Step 3: Census Specifications

---


A Census Specification identifies Data Dictionary fields containing important participant data, such as date of birth, hire date(s), historical earnings, etc., required to run a benefit calculation.

**Member Data** determines age, service, earnings, and sex of members.

- **Date of hire** (along with calendar year, plan year, and, if appropriate, measurement period) determines the first date that data is considered for calculations. Earnings and service for durations prior to the date of hire will be ignored.
- The **Service Definition Set** selected for the <**Base Service Set**> **Definition** describes the default calculation for service for any service computation which is not otherwise defined or overridden (ie: plan participation, benefit eligibility, benefit accruals, etc.). For more information on **Service Definition Sets**, see the [Service Definition Set](#) topic under Step 4: Plan Benefits.
- The **Salary Definition Set** selected for the <**Base Salary Set**> **Definition** describes the default calculation for salary, primarily used by the standard system salary operators (ie: #FAS, #SALARY).

**Beneficiary Data** determines age, sex, and type of beneficiary: None, Spouse, Non-spouse.

**Error/Warning Messages** are processed when a calculation runs. Only those entries for which the error/warning applies to census data may be selected. Click the **Add/Omit** button

to add or remove **Errors/Warnings** entries. Click the backdoor button  to create a new, or edit an existing, Error/Warning entry. **Error/Warning Messages** are optional.

**Data Defaults** define default values used for missing data at the time a calculation is executed. Note that these default values merely fill in missing (blank) field values or create new fields; therefore, existing values are not replaced. Note that the data defaults are processed in the order displayed in the list.

- **Default value or expression** can be defined as an expression. For details on writing expressions see [Appendix A: Expressions](#).
- If the field to be defaulted is a numeric array, then enter the rules for determining the associated dates for the array values under **Array Date**. For start and stop arrays, the default Stop Date is the end of the plan year preceding the date of hire and the Start Date is 12 months prior to the Stop Date. For effective date arrays, the Effective Date is the plan year end before hire. These defaults may be overridden using an expression.

## Step 4: Plan Benefits

---

In ProAdmin, plan benefits are defined using six principal building blocks: a **plan definition** (“plan”), **service definition sets** (“service”), **salary definition sets** (“salary”), **benefit definitions** (“benefits”), **benefit formula components** (“components”) and **payment forms** (“payment forms”). A plan is a collection of Benefit Definitions. A Benefit Definition specifies a benefit formula and payment forms. Benefit formulas reference service, salary, benefit formula components and operators.

In this section, we will go into some detail about most of these building blocks. Payment forms will be covered in the appendices.

### *Plan Definition*

The **Plan Definition** collects Benefit Definitions together and defines global plan attributes.

The **Plan Definitions** library can be accessed from **Input | Plan Benefit Definitions | Plan Definitions** on the menu or from the **Shortcuts** pane.

There are four (4) types of **Benefit Definitions** that may be included in a plan: Retirement, Termination, Death, and Disability. An unlimited number of each type of benefit definition may be included in a single plan definition. Add or remove existing Benefit Definitions using the Add/Omit Benefits button.

**Plan Attributes** are plan level characteristics shared by all included Benefit Definitions, such as, plan year, payment frequency, annuity payment timing, plan actuarial equivalence, normal retirement date, etc. Some specific topics to note:

- **Plan’s actuarial equivalence** defines the standard actuarial conversion basis for optional forms of payment. It is the default actuarial equivalence used for all payment forms unless otherwise specified within the payment form definition. For more information, see Appendix G: Payment Forms.
- **Normal Retirement Date (NRD)** determines the member’s Normal Retirement Date using the selected Eligibility Definition and accompanying Service Definition Set. This date is used for deferred and temporary payments to NRD, accrued interest to NRD and reduction of benefits for commencement prior to NRD.
- The **Final Average Salary Calculation** methodology controls all #FAS final average salary calculations. It indicates whether the salary in the measurement period of decrement should be included (**averaging period ends with current salary**) or excluded (**averaging period excludes current salary**). Note that Salary Definitions can be parameterized to effectively override these choices when necessary.
- **Maximum Compensation Law Year** determines the 401(a)17 maximum limit applicable to a measurement period’s salaries. **Social Security Law Year** determines the appropriate historical regulatory data for wage base (#SSWB), covered compensation (#CVCP), PIA (#PIA), and YMPE (#YMPE) determinations. These two parameters are only relevant if the plan year begins in a month other than January.

**Relative Value Calculations** specifies the details for any lump sum equivalent and (optionally) relative value calculations that ProAdmin will perform. In addition to specifying the age and actuarial equivalence assumptions, it is possible to **Use an alternative normal form (denominator)** for the calculation and to **Reflect partial payments (numerator)**.

**Calculated Dates** allows you to calculate dates, such as the date of participation, which will then be referenced during subsequent plan calculations such as to compute service.

**Regulatory Data** defines the historical values used for legislated limits and data.

- **U.S. Maximum Benefits** determines the rules used to calculate the U.S. IRC section 415(b) benefit limits. For more information, from within ProAdmin, go to [Help | Help Topics | Command Reference | Input Menu | Plan Benefit Definitions | Plan Definitions | U.S. Maximum Benefits](#).

**Error/Warning Messages** allows you to specify benefit-related errors and/or warnings that should be generated by the system under specific circumstances. For example, warning that a participant is highly paid or not yet vested.

**Fulfillment Components** identifies objects (components) not directly referenced in the benefit formula but are required, or desired, for reporting purposes (ie: vested percentage).

## Service Definition Sets

Service calculations are handled through **Service Definition Sets**. A **Service Definition Set** is made up of a collection of **Service Definitions**, each defining the rules for calculating service during a specific time interval. An unlimited number of **Service Definitions** may be included within a single **Service Definition Set**; this permits a variety of service calculations for a referenced object (ie: Salary Definitions, Benefit Eligibility Definitions, Accrual Rates, etc.). An alternative **Service Definition Set** may be specified to override the default or **<Base service set>** specified in the **Census Specifications** for any object.

The **Service Definitions Set** library can be accessed from **Input | Service Definitions | Service Definitions Set** on the menu or from the **Shortcuts** pane.

Consider the following example definition for benefit service:

Service is calculated on an elapsed time basis, rounded to the nearest month, for participation from hire through December 31, 1990. For participation between January 1, 1991 through December 31, 1998, service is calculated on an elapsed time basis rounded to the nearest year. For participation from January 1, 1999 and thereafter, a year of service is granted if at least 1,000 hours are worked in the calendar year.

The screenshot shows the 'Service Definition Set - [Benefit Service]' dialog box. The 'Name' field is 'Benefit Service'. The 'Included Service Definition(s):' section contains a table with three rows:

From	To	Service Definition
-	12/31/1994	Elapsed Time from Date Of Hire - nearest month rounding
01/01/1995	12/31/1999	Elapsed Time from Date Of Hire - nearest year
01/01/2000	-	1000 Hours method

Below the table, the 'Service Definition Set rounding rule:' is set to '<none>'. The 'Break-in-Service rule:' has three radio buttons: 'Do not apply' (selected), 'Plan Definition rule', and 'Break-in-Service Definition:'. There is a checkbox for 'Change service only at decrement and end of' which is unchecked. The 'End of measurement period:' section has two radio buttons: 'Date:' (selected) and 'Field:'. At the bottom, there are buttons for 'View', 'Replace', 'Save As New', 'Erase', and 'Cancel'.

For the above example, use a **Service Definition Set** with three (3) distinct **Service Definitions**, one for each mentioned time-period:

## Service Definitions

**Service Definitions** define the rules for service calculations during a specific time interval. Service accruals may be computed based on elapsed time, hours conversion, or an accumulation of service units.

The **Service Definitions** library can be accessed from **Input | Service Definitions | Service Definitions** on the menu or from the **Shortcuts** pane after expanding the **Service Definitions Set** topic.

- **Service rounding rule** applies to the final result of the service calculation.  
However, **Round service earned within each measurement period** indicates the rounding rule is applied to each measurement period instead of the final result. The resulting value is the sum of the individual rounded amounts.
- **Measurement period (used to group and count service periods)** is the increment over which service is accumulated. **End of measurement period** must be defined for bi-weekly or weekly measurement periods.

**Event Definition (used to group and count service)** modifies service according to events (ie: status changes) throughout a participant's career. Examples of events are termination re-hire, disability, layoff, military leave, etc. Multiple events are presented using a coded array field.

Service Definition - [Elapsed Time from Date Of Hire - nearest month rounding]

Name: Elapsed Time from Date Of Hire - nearest month rounding

Calculate service based on:

- Elapsed time
- Reported hours
- Reported service units

Service rounding rule:

Nearest Month

Round service earned within each measurement period

Measurement period (used to group and count service periods):

semi-month

End of measurement period:

- Date: [ ]
- Field: [ ]

Event Definition (used to start and stop service):

<none>

Ignore Events...

Select a topic to edit:

- Elapsed time parameters
- Service eligibility (e.g., 21 and 1)
- Service start adjustment
- Service stop adjustment
- Grandfathered service
- Service multiplier (e.g., part-time adjustment)
- Service transformation expression

Buttons: View, Replace, Save As New, Erase, Cancel



Select and fill out each topic. Topics vary depending on type of service accrual (elapsed time, hours, service unit). Here are a few notes about certain topics:

- **Elapsed time parameters** define the **Calculation method** and **Start date field** for elapsed time service calculations. The **Calculation method** may be date subtraction, calendar days, business days or 360 days per year. The **Start date field**, indicating when service begins, may be any date field or date array field; however, it is ignored when an **Event Definition** is specified above.
- **Hours conversion** defines the rules to convert hours into units of service.

## ***Salary Definition Sets***

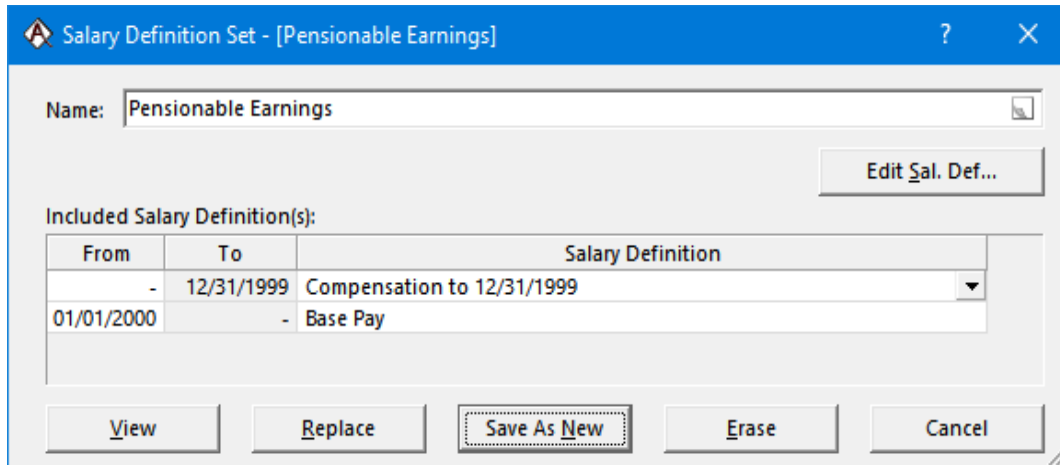
Salary calculations are handled through **Salary Definition Sets**. A **Salary Definition Set** is made up of a collection of **Salary Definitions**, each defining the rules for calculating salary during a specific time interval. **Salary Definitions** reference **Salary Histories** which point to numeric array fields. An alternative **Salary Definition Set** may be specified to override the default or <Base salary set> defined in the **Census Specifications** for any **Custom Operators**.

The **Salary Definitions Set** library can be accessed from **Input | Salary Definitions | Salary Definitions Set** on the menu or from the **Shortcuts** pane.

Consider the following example definition for pensionable earnings:

Pensionable Earnings is defined as (1) and (2) below. (1) For Earnings Computation Periods prior to January 1, 2001: wages, salaries, commissions, bonuses, fringe benefits, reimbursements, and expense allowances to the extent that such amounts are includible in gross income. (2) For Earnings Computation Periods on and after January 1, 2001: base salary.

For the above example, use a **Salary Definition Set** with two (2) distinct **Salary Definitions**, one for each mentioned time period:



## Salary Definitions

**Salary Definitions** determine the salaries recognized and included for salary calculations during a specific time interval.

The **Salary Definitions** library can be accessed from **Input | Salary Definitions | Salary Definitions** on the menu or from the **Shortcuts** pane after expanding the **Salary Definitions Set** topic.

- **Salary History** references the numeric array fields from the data dictionary containing member historical earnings data. For more information, see **Salary History** below.
- **Measurement period (used to group and count salary periods)** is the increment over which salary is accumulated

## Step 4: Plan Benefits

Salary Definition - [Base Pay]

Name: Base Pay

Salary History: SAL - Base Pay

Measurement period (used to group and count salary periods): calendar year

Generate measurement period salaries from less frequent salaries

End of measurement period:  Date:   Field:

Select a topic to edit:

- Initial salary
- Last salary
- Allocate and reflect salary
- Transform salary
- Exclude salary

View Replace Save As New Erase Cancel

Select and fill out each topic. Here are a few notes about certain topics:

- **Allocate and reflect salary** defines salary allocation during the measurement period for both reported and projected salaries.
- **Exclude salary** allows salaries to be ignored in the #SALARY and #FAS custom operators.

## Salary History

**Salary History** references numeric array fields containing reported earnings.

- **Salary Components** are combined based on the dates and the measurement period specified in the **Salary Definition**.
- When **Salary Components are rates of pay**, the **Frequency** for the rate of pay must be defined. Note that if a **Database field (with frequency values)** is used for the pay frequency, the salary values pulled from the database are multiplied by the value in the specified field to determine an annual rate of pay. For example, if the pay value in the database is \$10,000 and the field defining frequency has a value of 12, the annual rate of pay becomes \$120,000.

## Benefit Definitions

A **Benefit Definition** pulls together all of the information about a specific benefit. The benefit definition determines why, when, how much, and in what form payments will be made.

- **Contingency initiating benefits** is the decrement triggering payment of this benefit (retirement, death, disability or termination). When processing an estimate, if a member meets the eligibility requirement for a retirement benefit, all termination benefits will be set to N/A.
- **Eligibility** defines the criteria required for benefit eligibility, where eligibility is determined as of the decrement/termination date. Any service requirements specified in the **Eligibility Definition** are evaluated using the selected **Service Definition Set**. If necessary, you can also choose to **Apply an alternative eligibility** criteria based on a different definition of service.
- **Benefit Formula** describes the calculation of the annual (in the case of annuities) benefit payable. The expression is comprised of Benefit Formula Components, arithmetic operators, and ProAdmin operators. For more information on Benefit Formulas, see Benefit Formulas below.
- **Payment Forms** are the valid payment options for this benefit. They describe when, how and for how long the benefit is to be paid. Multiple Payment Forms may be defined for a single Benefit Definition. The **Normal Form** is the normal form of payment for the actuarial equivalence calculations.

## Payment Forms

The **Payment Forms** library can be accessed from **Input | Plan Benefit Definitions | Payment Forms** on the menu or from the **Shortcuts** pane. For each new **Payment Form**, select the annuity **Type** and fill out each of the topics. Here are a few notes about certain topics:

- **Type** describes the type of annuity payable (ie: single life annuity, joint life annuity, certain only annuity, lump sum, etc.)

- **Basic Form Parameters** control deferral period, temporary period, COLAs, certain period (when applicable), and J&S fractions (when applicable).
- **Conversion from Normal Form** controls the actuarial equivalence assumptions used to convert from the normal payment form to this payment form.

## ***Benefit Formulas***

ProAdmin's benefit formula expression language is both flexible and concise. It accommodates plans of all types: final average pay, career average pay, cash balance, pension equity, floor-offset, etc. Later in this section you'll find step-by-step instructions for setting up several common defined benefit plan types.

A few notes you might find helpful regarding expressions:

- ***ProAdmin starts at the left and works its way to the right, executing operators as they are encountered in the expression.*** There is no precedence among operators. Be mindful of order of operation. For example:

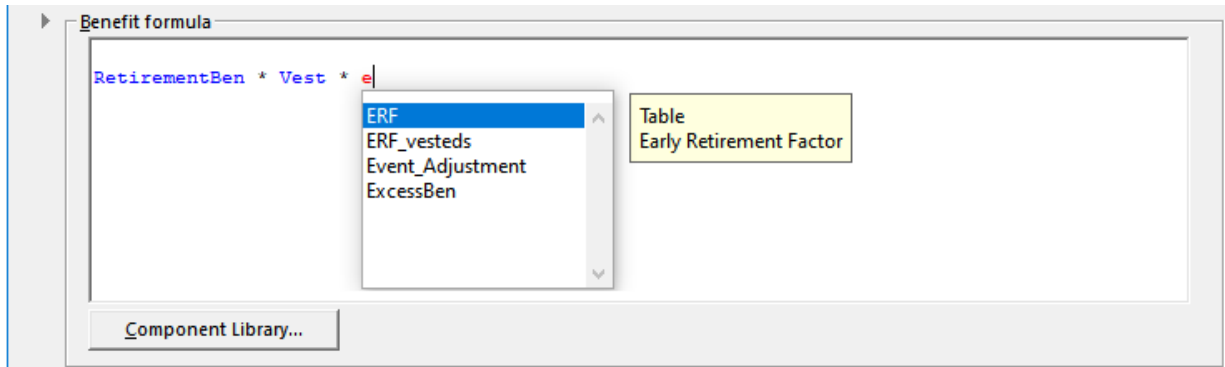
$$1+2*3 = 9$$

- ProAdmin allows any of the following pairs of parenthetical symbols to be used in expressions: ( ), [ ], or { }. Judicious use of these alternative symbols can help clarify expressions containing many nested parentheses. For example:

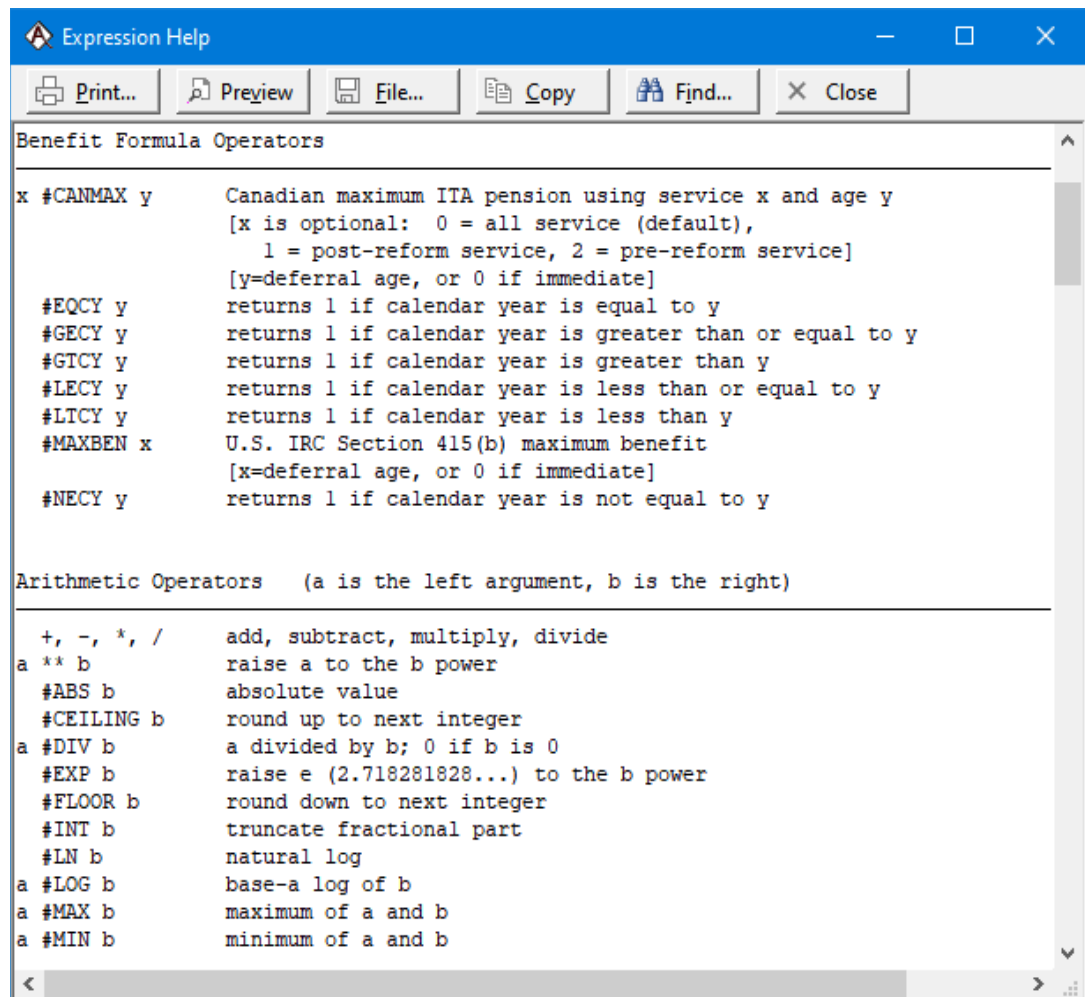
$$1 + (2*3) = 7$$

- Use logical component names and complete descriptions. Clarity is as important as accuracy.
- Capture as much of the complexity of the plan as possible within the components so that the formula itself is clean and easy to understand.
- Let accrual definitions do the work for you. For most pension plans, this type of benefit formula component will be all that you need.
- When you are entering an expression, if you begin typing, predictive text will suggest possible components or operators which you may be looking for. Click once to see the description of the component and twice to select it. Click on a blue component to edit it.

## Step 4: Plan Benefits



- You may also press **F1** to display an expression help window. The help screen describes the components and operators available for use in expressions. The cursor must be in an expression entry field for F1 to display this help message. An example is illustrated below.



This help screen describes the operators available for use in **Benefit Formula** expressions. The cursor must be in an expression entry field for F1 to display this help message. For more information, see Appendix A: Expressions.

## Step 4: Plan Benefits

## ***Benefit Formula Components***

**Benefit Formula Components** are the building blocks of benefit formulas. They can be one of seven different types:

- 1 **Accrual definition** refers to a rate accrual per year of service, age or points (age + service). Depending on the accrual format, (**Final average**, **Career average**, or **Cash balance**), this rate accrual is multiplied in some way by a dollar amount such as salary. All ongoing pension plans will utilize one or more of this type of component.
- 2 **Annuity Factor** calculates a lump sum conversion factor, or annuity factor, based on the mortality and interest assumptions specified.
- 3 **Constant** is a flat constant value which may vary by coded field. It is useful when a constant value is repeated in multiple Benefit Formulas.
- 4 **Database field** refers to the value in a field on the database, such as a grandfathered benefit amount, or an expression using a combination of fields.
- 5 **Interest factor** calculates an interest factor based on the interest rate, mortality and duration assumptions specified.
- 6 **Late Retirement** calculates late retirement increases for an accrued benefit based on the normal retirement date (NRD) and actuarial equivalence assumptions specified.
- 7 **Subformula** is just an expression combining the other types of components. Rather than repeating the expression in multiple benefit formulas, simply refer to the subformula component's name.
- 8 **Table** looks up values from a table based on age and/or service, and may be sex distinct. It may point to a single table or multiple tables which vary by coded field. Examples include: early retirement factors, claims tables, etc.

Rather than describe each of the features of these benefit formula components, we will explore the use of them by way of example in the coding up of some standard pension formulas below.



## Final Average Salary Benefit

The following pages will guide you through setting up an early retirement final average pay benefit formula in ProAdmin using the general expression:

$$(\text{RETIREMENT BENEFIT} + \text{EXCESS BENEFIT}) * \text{ERF}$$

Before you start coding, gather the following information (each *item* in italics is referred to the steps that follow):

Item	Example	Your plan
<i>Eligibility definition</i>	Age 55 with 5 years of participation service	
<i>Salary definition</i>	W2	
<i>Final average salary</i>	Highest 5 out of the last 10	
<i>% of pay</i> <i>(% of pay up to integration level</i> <i>+</i> <i>% of pay over integration level)</i>	1.5% per year of service	
<i>Maximum service</i>	30 years	
<i>Integration level</i>	Covered Compensation	
<i>% of pay over integration level</i>	0.5% per year of service	
<i>Years to apply integration</i>	30 years	
<i>% of pay after integration years</i>	0%	
<i>Early retirement reduction</i>	6.3% per year for the first 5 years 3.6% for the next 5 years	
<i>Payment forms</i>	Life No Death 50% Joint & Survivor Lump sum	

## Step 4: Plan Benefits

We will begin with setting up the RetirementBen component for our benefit formula.

- 1 In the **Benefit Formula Component** library, click **New** and enter a **Name** for the component, for example, RetirementBen. Ensure that the component type is **Accrual – Final Average**.

Benefit Formula Component - [RetirementBen]

Name: RetirementBen Description: .015 \* FAE \* SVC with a max of 30 yrs

Component type: Accrual - Final average

Benefit [basis x (sum of rates)]

Accrual Rates: <unspecified>

Basis Formula: <unspecified>

View Replace Save As New Erase Cancel

- 2 For **Accrual Rates** click on <unspecified>. Select the **Service Definition Set** that calculates your service component, enter the *% of pay* (as a number between 0 and 1). In our example there is a *maximum service* so the rate will change to 0 after 30 years of service. You may also need to change the rate if your plan has a *% of pay after integration years* once service has exceeded *years to apply integration*. Click **OK**.

Accrual Rates

Service Definition Set: Benefit Service

Rate type:

Constant:

Varies by: years of service

with new rates as of:

From	Up to	Rate
0	30	0.015
30	-	0.000

New...

Reflect new rates after decrement

Project & prorate:

Ultimate accrual:

Project to:

Age:

Date field:

Service req'd for ultimate accrual:

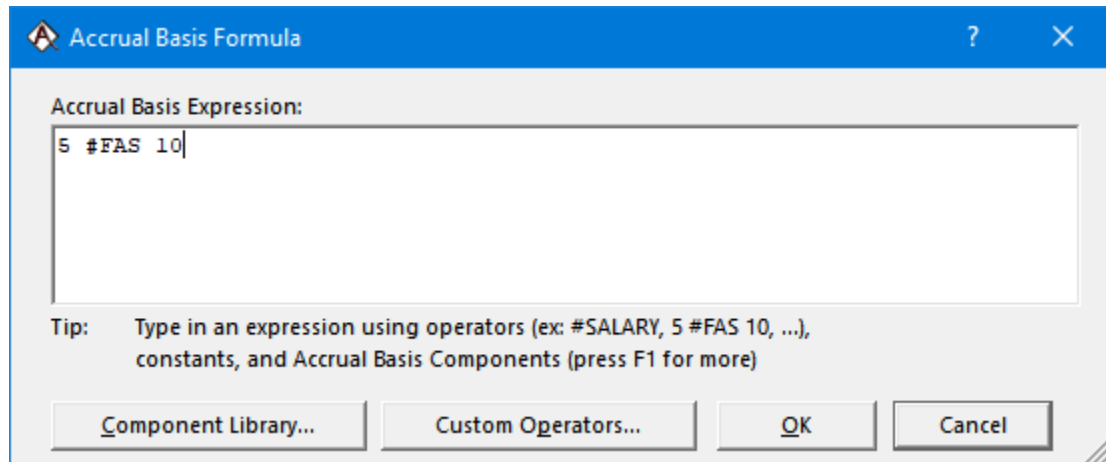
Apply rounding rule: Amount: Direction:

Apply annualized rate to career avg./cash bal. basis formula

Age... OK Cancel

## Step 4: Plan Benefits

- 3 For **Basis Formula** click on `<unspecified>`. Type in the *final average salary* calculation using the syntax “n #FAS m”, for example “5 #FAS 10” for the highest consecutive 5-year average out of the last 10 years. Click **OK**.



Note that #FAS is ProAdmin’s standard final average salary operator. You will create your own **Custom Operators** to be used in place of #FAS if your final average salary isn’t based on annual or consecutive pays.

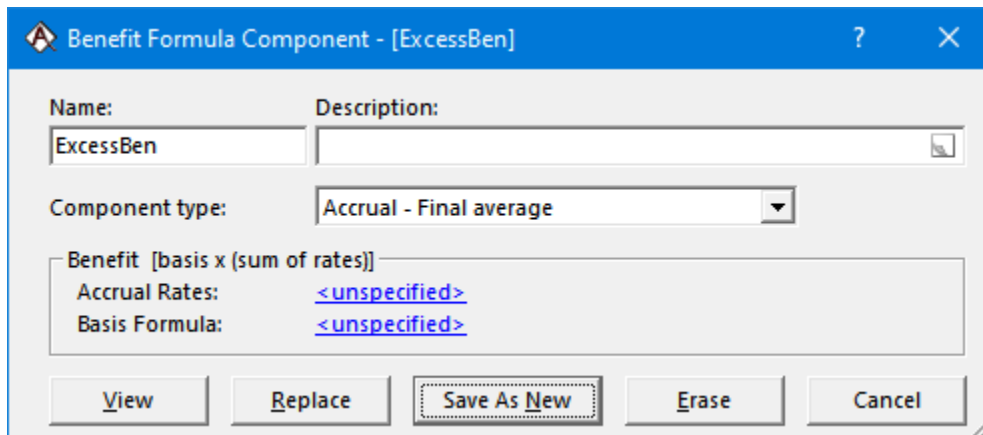
- 4 Click **Save As New**.

## Step 4: Plan Benefits

### EXCESS BENEFIT

Skip this component if your plan doesn't have an *integration level*.

- 1 In the **Benefit Formula Component** library, click **New** and enter a **name** for the component, for example, EXCESS. Ensure that the component type is **Accrual definition** and the accrual format is **Final average**.



Benefit Formula Component - [ExcessBen]

Name: ExcessBen Description:

Component type: Accrual - Final average

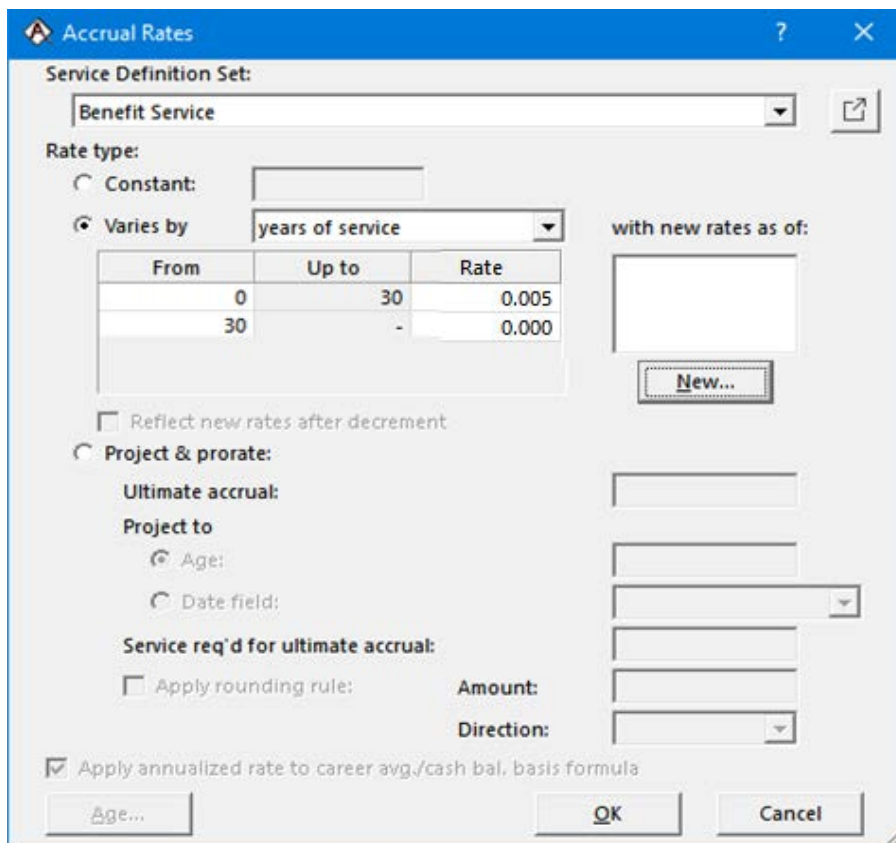
Benefit [basis x (sum of rates)]

Accrual Rates: <unspecified>

Basis Formula: <unspecified>

View Replace Save As New Erase Cancel

- 2 For **Accrual Rates** click on <unspecified>. Select the **Service Definition Set** that calculates your service component, enter the *% of pay over integration level*. In our example there is a *maximum service* so the rate will change to 0 after 30 years of service. You may also need to change the rate if your plan has *years to apply integration*. Click **OK**.



Accrual Rates

Service Definition Set: Benefit Service

Rate type:

Constant:

Varies by: years of service

From	Up to	Rate
0	30	0.005
30	-	0.000

with new rates as of:

New...

Reflect new rates after decrement

Project & prorate:

Ultimate accrual:

Project to:

Age:

Date field:

Service req'd for ultimate accrual:

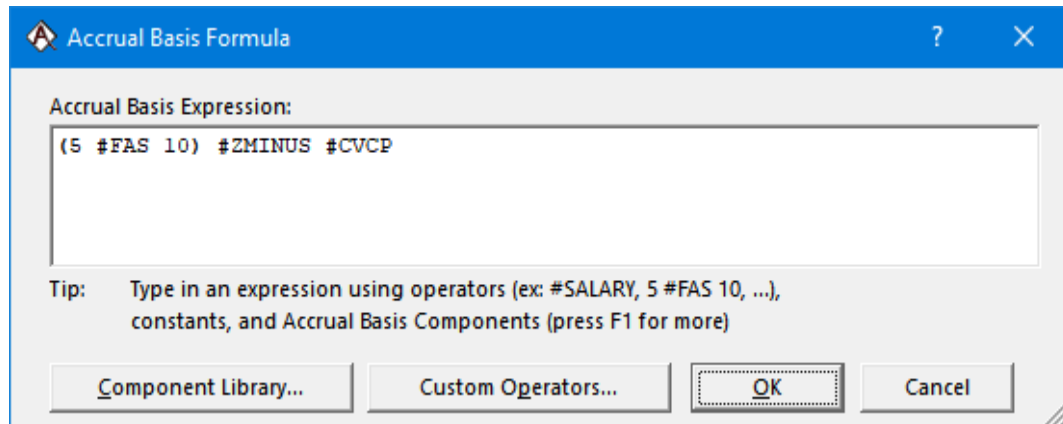
Apply rounding rule: Amount: Direction:

Apply annualized rate to career avg./cash bal. basis formula

Age... OK Cancel

#### Step 4: Plan Benefits

- 3 For **Basis Formula** click on *<unspecified>*. Type in the *final average years* the same way you did for RetirementBen. Then subtract the *integration level*, for example “#CVCP” for covered compensation. Click **OK**.



You will need to use a **Custom Operator** in place of #CVCP if your plan’s definition of covered compensation uses a different averaging period or wage base increases. Alternatively, if your plan defines covered compensation as the value for someone at SSNRA, use #AVGWB 35.

- 4 Click **Save As New**.

## Step 4: Plan Benefits

ERF

- 1 In the **Benefit Formula Component** library, click **New** and enter a **name** for the component, for example, ERF . Select Component type Table

Benefit Formula Component - [ERF]

Name: ERF Description: Early Retirement Factor - .063 for 5 yrs .036 next 5 yr

Component type: Table

Single table for all records:  
ERF - 6.3% for first 5 yrs 3.6% thereafter

Varies by coded field:

Database Code	Table
---------------	-------

Using service from Service Definition Set:  
<Base Service Set>

Age / Interpolation...

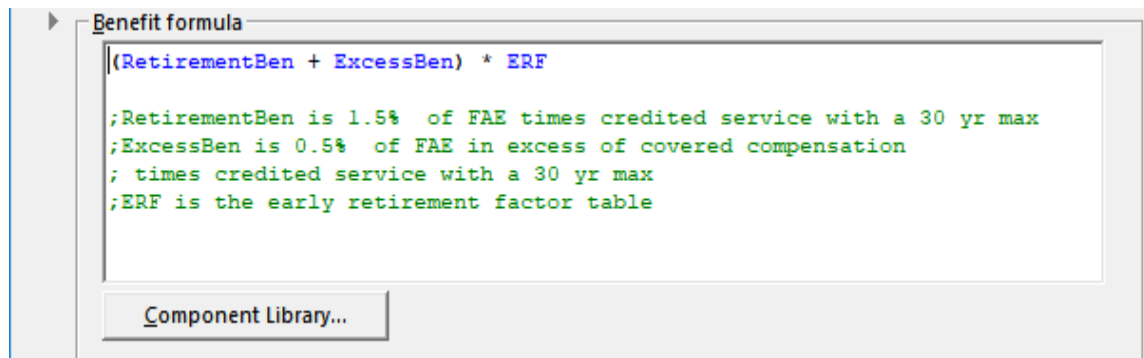
View Replace Save As New Erase Cancel

- 2 Choose **Single table for all records** and select appropriate table. Access the **Benefit Formula Component Table** library through the back door button to edit/create tables. The **Age/Interpolation** parameters determine the age definition for entering the table along with rounding and interpolation assumptions for non-integer ages.
- 3 Click **Save As New**.

## Step 4: Plan Benefits

Now that the **Benefit Formula Components** are set up, create a **Benefit Definition**. The **Benefit Definitions** library may be accessed from **Input | Plan Benefit Definitions | Benefit Definitions** on the menu or from the **Shortcuts** pane.

- 1 Click **New** and enter a **name** for the **Benefit Definition**, for example, “Retirement Benefit.”
- 2 Select Retirement as the **Contingency**.
- 3 In **Eligibility** section, select the appropriate **Eligibility Definition**. Edit/Create **Eligibility Definitions** through the back door button.
- 4 Specify the **Benefit Formula**. If your plan has an *integration level* your formula will be:



(RetirementBen + ExcessBen) \* ERF

If your plan doesn't have an *integration level* your formula will be:

RetirementBen \* ERF

- 5 In the **Payment Forms** section, click on the **Add/Omit** button and select the applicable payment forms. Also select the **Normal Form**.
- 6 Click **Save As New**.

Create additional *Benefit Definitions* for termination, disability, and death benefits.

## Hourly Benefit

The following pages will guide you through setting up a disability hourly benefit using the general expression:

HOURLYBFT

Before you start coding, gather the following information (each *item* in italics is referred to in the steps that follow):

Item	Example	Your plan
<i>Eligibility definition</i>	3 years of service and classified as disabled by Social Security Administration	
<i>Dollar accruals</i>	\$40 / month ( -1989) \$45 / month (1990-2001) \$50 / month (2002-2002) \$55 / month (2003- )	
<i>Accrued benefit</i>	Stored on data (the alternative is to let ProAdmin calculate the accrued benefit)	
<i>Maximum service</i>	30 years	
<i>Payment form</i>	Life no death benefit	

HOURLYBFT

- 1 In the **Benefit Formula Component** library, click **New** and enter a **Name** for the component, for example, HOURLYBFT .

If you have an *accrued benefit* stored on the data, choose **Accrual - Career average** as the component type. If not, choose **Accrual - Final average** as the component type.

- 2 For **Accrual Rates** click on **<unspecified>**. Enter rates equal to the *dollar accruals*. The rate will change to 0 if your plan has a *maximum service*. If the *dollar accruals* change over time, enter the initial amount on the rates screen.



## Step 4: Plan Benefits

Service Definition Set:  
Benefit Service

Rate type:  
 Constant:  
 Varies by: years of service

From	Up to	Rate
0	30	40
30		0

with new rates as of:  
01/01/1990  
01/01/2002  
01/01/2003  
New...

Reflect new rates after decrement

Project & prorate:  
Ultimate accrual:  
Project to:  
 Age:  
 Date field:  
Service req'd for ultimate accrual:  
 Apply rounding rule: Amount:  
Direction:

Apply annualized rate to career avg./cash bal. basis formula

Age... OK Cancel

Enter subsequent amounts by clicking the **New** button and specifying whether the rates apply prospectively only (i.e., **years after the effective date**) or retroactively for **all years**. Click **OK**.

- 3 For **Basis Formula** click on **<unspecified>**. Type in a factor to annualize the *dollar accruals*, generally “1” or “12”. Click **OK**.

Accrual Basis Formula

Accrual Basis Expression:  
12

Tip: Type in an expression using operators (ex: #SALARY, 5 #FAS 10, ...), constants, and Accrual Basis Components (press F1 for more)

Component Library... Custom Operators... OK Cancel

## Step 4: Plan Benefits

- 4 For **Accrued Benefit** click on [<unspecified>](#). Select the **Database field** containing the accrued benefit. Note: this only applies if you have an *accrued benefit* stored on the data and opted to use component type **Accrual - Career average**. Click **OK**.
- 5 Click **Save As New**.

Now that the **Benefit Formula Components** are set up, create a **Benefit Definition**. The **Benefit Definitions** library may be accessed from **Input | Plan Benefit Definitions | Benefit Definitions** on the menu or from the **Shortcuts** pane.

- 1 Click **New** and enter a **Name** for the **Benefit Definition**, for example, “Disability Benefit”.
- 2 Select Disability as the **Contingency**.
- 3 In the **Eligibility** section, select the appropriate **Eligibility Definition**. Edit/Create **Eligibility Definitions** through the back door button.
- 4 In the **Benefit Formula** section, type in the expression: HOURLYBFT
- 5 In the **Payment Forms** section, click on the **Add/Omit** button and select the applicable payment forms. Also select the **Normal Form (for actuarial equivalence calculations)**.
- 6 Click **Save As New**.

Create additional **Benefit Definitions** for retirement, termination, and death benefits.

## Cash Balance Benefit

The following pages will guide you through setting up a retirement cash balance benefit formula using the general expression:

$$(CASHBAL / ANNCONV)$$

Before you start coding, gather the following information (each *item* in italics is referred to in the steps that follow):

Item	Example	Your plan
<i>Eligibility Definition</i>	5 years of service	
<i>Salary definition</i>	W2	
<i>% of pay</i>	1.5% per year of service up to 10 years, 2% for service thereafter	
<i>Beginning balance</i>	Starting point for the Cash Balance calculation	
<i>Interest crediting rate</i>	30 year treasury rates as of November 1 <sup>st</sup> . The stability period of the rate is 1 year.	
<i>Maximum service</i>	30 years	
<i>Payment forms</i>	10 year certain & life Lump sum	

CASHBAL

- 1 In the **Benefit Formula Component** library, click **New** and enter a **Name** for the component, for example, CASHBAL . Select Component type of **Accrual – Cash Balance**.
- 2 For **Accrual Rates** click on **<unspecified>**. Select the Service Definition Set that calculates the service component. Enter the *% of pay* (as a number between 0 and 1).

## Step 4: Plan Benefits

Service Definition Set:  
<Base Service Set>

Rate type:  
 Constant:  
 Varies by: years of service

From	Up to	Rate
0	10	0.015
10	30	0.002
30	-	0

with new rates as of:  
New...

Reflect new rates after decrement

Project & prorate:  
Ultimate accrual:  
Project to:  
 Age:  
 Date field:  
Service req'd for ultimate accrual:  
 Apply rounding rule: Amount:  
Direction:

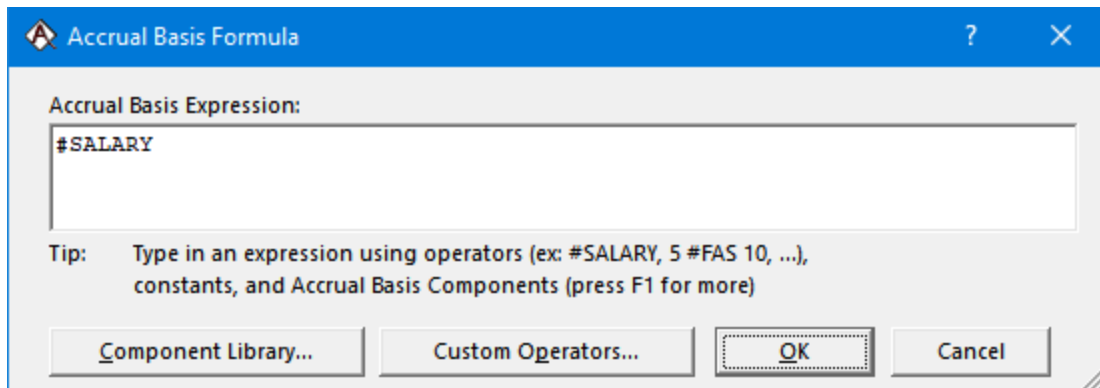
Apply annualized rate to career avg./cash bal. basis formula

Age... OK Cancel

**Apply annualized rate to career avg./cash bal. Basis formula** indicates that an annualized rate is applied as the year's accrual for a participant who earns less than one year of service during the year. For example, for the accrual of 1.5% above, if a person earned  $\frac{1}{2}$  year of service during a year, ProAdmin's default behavior would be to calculate an accrual of 0.75% ( $\frac{1}{2}$  of 1.5%). This is typically appropriate if the salary referenced in the Basis is an annualized rate of pay. Alternatively, if this box is checked, the years' accrual will be 1.5%, where this is typically appropriate if the salary referenced in the Basis is the actual (partial) year's pay.

Click **OK**.

- 3 For the **Basis Formula** click on <unspecified>. Type in the employee's salary using the operator #SALARY. Click **OK**.



You will need to use a **Custom Operator** in place of #SALARY if your *salary definition* differs from the salary definition (in Census Specs).

- 4 For **Accrued Benefit** click on [<unspecified>](#). The accrued benefit can be defined as either an array-type **Database field** (representing the accrued benefit at various points in time) or ProAdmin's calculated **Expected value** based on the rates and basis provided. If a database field is referenced, ProAdmin will begin building the benefit from the most recent crediting period end value provided, and then add on the applicable accrued benefit.

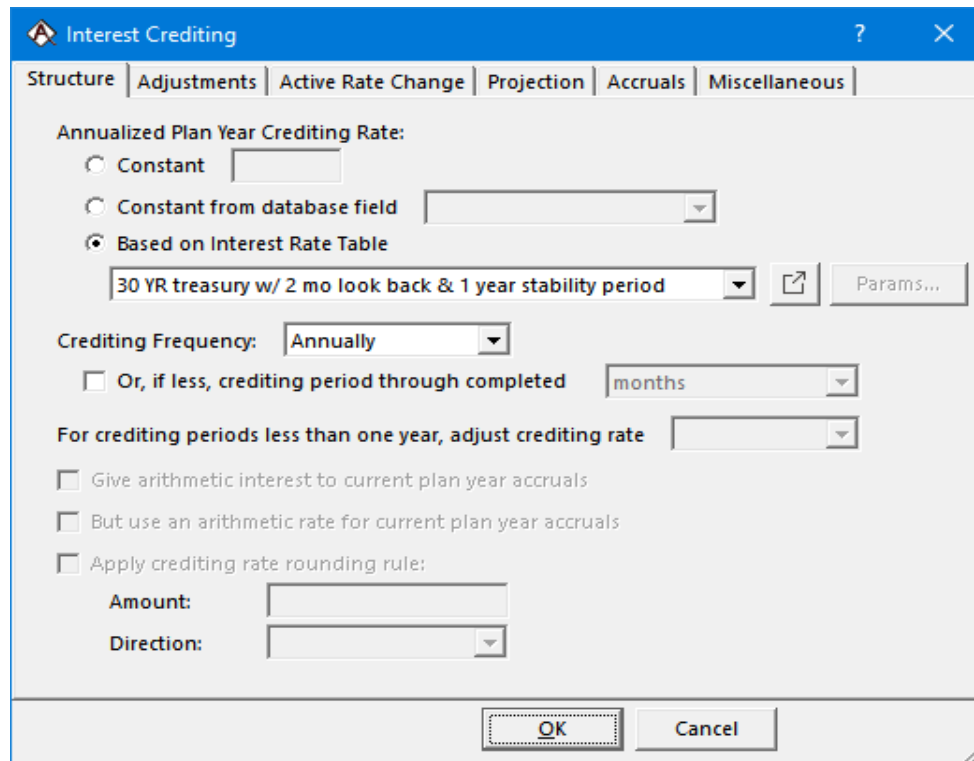
**Zero-out empirical accrued benefit** indicates the accrued benefit is equal to zero and only future accruals are calculated. This avoids double-counting the accrued benefit in complex formulas requiring multiple cash balance components.

Click **OK**.

- 5 The next section is **Interest Crediting**. There are six (6) topics: Adjustments, Active Rate Change, Projection, Accruals, and Miscellaneous.

## Step 4: Plan Benefits

For **Structure** click on [<unspecified>](#); **Annually**.



The screenshot shows the 'Interest Crediting' dialog box with the following settings:

- Annualized Plan Year Crediting Rate:**
  - Constant
  - Constant from database field
  - Based on Interest Rate Table
    - 30 YR treasury w/ 2 mo look back & 1 year stability period
- Crediting Frequency:** Annually
- Or, if less, crediting period through completed (months)
- For crediting periods less than one year, adjust crediting rate
- Give arithmetic interest to current plan year accruals
- But use an arithmetic rate for current plan year accruals
- Apply crediting rate rounding rule:
  - Amount:
  - Direction:

- The **Annualized Plan Year Crediting Rate** can be set to a **Constant**, a **Constant from database field**, or a rate **Based on Interest Rate Table**.
- **Crediting Frequency** specifies how often interest is credited to the cash balance account.

For **Adjustments** click on [<none>](#) or the **Adjustments** tab. This defines fixed rate overrides for historical rate differences from the current rate structure, as well as minimum and maximum credit rates.

For **Active Rate Change** click on [<n/a>](#) or the **Active Rate Change** tab. This controls crediting rate changes upon attainment of an eligibility condition. This is an uncommon feature.

For **Projections** click on [<interest until commencement at last known applicable rate>](#) or the **Projections** tab. This controls interest crediting from decrement or termination through the commencement date or the attainment of an eligibility condition.

For **Accruals** click on [<Parse formula by interest crediting periods>](#) or the **Accruals** tab. This controls allocation for intra-plan year crediting periods:

- If the crediting frequency is other than annual, you have options about how the accrual basis formula is evaluated. **The crediting period accrual basis is determined by \* Parsing the accrual basis formula to the change between Interest crediting periods** applies ProAdmin's default treatment. This assumes that the accrual basis formula components are cumulative plan year values and that the result of the formula needs to be parsed to determine the value attributable to each crediting period. For example, if your accrual basis formula is written as `#SALARY #MIN 15000` and the crediting frequency is set to monthly, ProAdmin

## Step 4: Plan Benefits

will compare the salary earned in the period to 1/12<sup>th</sup> of the 15000 (1250) and return the lesser of the two amounts.

**The crediting period accrual basis is determined by \* Using the accrual basis formula directly** assumes that the user has written an accrual basis formula that is correct for each crediting period and no adjustment (i.e., parsing) should be applied. For example, if your accrual basis formula is written as #SALARY #MIN 15000 and the crediting frequency is set to monthly ProAdmin will compare the salary earned in the period to 15000 and return the lesser of the two amounts.

- For annual crediting, there is an option to **discount current plan year accruals from the end of the crediting period**. This can be thought of as mathematically equivalent to not crediting the accrual until the end of the plan year.
- 6 For **Miscellaneous** click on [<n/a>](#) or the **Miscellaneous** tab. This contains other parameters, such as, days assumed in a year when crediting interest daily.
  - 7 Click **Save As New**.

## Step 4: Plan Benefits

ANNCONV

Skip this component to value the cash balance benefit paid as a lump sum rather than as an annuity.

- 1 In the **Benefit Formula Component** library, click **New** and enter a **Name** for the component, for example, ANNCONV . Change the component type to **Annuity Factor**.
- 2 Select the **Mortality, Interest & Cola** topic to define the basic payment form assumptions. For our example, we will select the mortality table 1983 GAM 50 – 50, a constant rate of 6%, 0 constant COLA increase, monthly payment frequency, and beginning of period payment timing:

**Mortality, Interest & COLAs**

**Mortality**

1983 Group Annuity Mortality 50-50

Use the current applicable mortality for 417(e)

Use zero mortality in the deferral period

**Interest**

Static rate:

Constant: 0.06

Database field:

Segment style rates:

1st 2nd 3rd

Based on Interest Rate Table

Apply PPA phase-in from this GATT Interest Rate Table:

Use alternative interest rate in the deferral period

Constant

Based on Interest Rate Table

**Cost-of-Living Adjustments (COLAs)**

COLA rate during payment period:

Constant: 0

Variable:

COLA rate during deferral period:

Constant: 0

Variable:

Rate type:  Compound  Simple

**Timing parameters**

Payment frequency: Monthly

Payment timing: Beginning of period

Use prior day interest & mortality for 1st plan year day commencement

OK Cancel



## Step 4: Plan Benefits

- 3 Select the **Payment/Calculation Period** topic to control some details about the payment form, including, certain, deferral, and/or temporary period. In our example we want to calculate an annuity factor that will turn our cash balance into a 10 year certain and life annuity:

The screenshot shows the 'Payment / Calculation Period' dialog box. The 'Certain:' section has the 'for years' radio button selected with a value of 10. The 'Deferral:' section has the 'to age' radio button selected with a value of 65. The 'Temporary:' section has the 'none' radio button selected. The 'Refine deferral / temporary period using Date Adjustment' checkbox is unchecked. The 'Youngest/Oldest Recognized Ages:' section has the 'Use age' checkbox selected with a value of 55, and the 'Spot rate method' dropdown is set to 'Start rates at youngest recognized age'. The 'Apply oldest recognized age using:' checkbox is unchecked, with the 'Age' radio button selected and a value of 65. The 'Annuity factor freeze date:' section has the 'Do not freeze' radio button selected.

- 4 Select the **Age/Interpolation** topic to specify how to determine the member's and any potential beneficiary's ages. For our example we will use age nearest.
- 5 Click **Save As New** to save the component.

Now that the **Benefit Formula Components** are set up, create a **Benefit Definition**. The **Benefit Definitions** library may be accessed from **Input | Plan Benefit Definitions | Benefit Definitions** on the menu or from the **Shortcuts** pane.

- 1 Click **New** and enter a **Name** for the benefit definition, for example, "cash balance retirement benefit". Choose Retirement as the **Contingency Initiating Benefit**.
- 2 In the **Eligibility** section, select the appropriate **Eligibility Definition**. Edit/Create **Eligibility Definitions** through the back door button.
- 3 In the **Benefit Formula** section, type in the following expression:  
CASHBAL/ANNCONV
- 4 In the **Payment Forms** topic, click on the **Add/Omit** button and select the applicable payment forms. Also select the **Normal Form** (for actuarial equivalence calculations).
- 5 Click **Save As New**.

#### Step 4: Plan Benefits

Create additional **Benefit Definitions** for disability, termination, and death benefits.

## Step 5: Projection Assumptions

---

**Projection Assumptions** control the assumptions used to project salary, service, regulatory data, and interest rate tables into the future for an **Estimated Benefit Calculation**.

- 1** The **Projection Assumptions** library may be accessed from **Input | Projection Assumptions** on the menu or from the **Shortcuts** pane. Click **New** and enter a **Name** for the library entry, for example, STANDARD ASSUMPTIONS.
- 2** Select the **Inflation and Merit** topic found under **Salary Increases**. For our example, apply a **Constant** 4% to future salaries (input as a decimal). Click **OK**.
- 3** Select the **Salary to Project** topic found under **Salary Increases**. For our example, select the **Most recent or prior year value** for the basis of projected salaries. Check **Most recent value** or **Prior year value** or the maximum of the two (if both are checked). Click **OK**.
- 4** Select the **Hours and Service History Fields** topic found under **Service Increases**. For our example, start the hours projection in the year following the most recent **Reported hours/service (in the year), grossed up to a full year**, and assume an annual accrual of a **Constant** 1500 hours. Click **OK**.
- 5** Select the **Regulatory Data Increase Rates** topic to set increase rates for each type of regulatory data, including the Maximum Benefit, Maximum Compensation, Social Security National Average Wage, Social Security CPI, and the Canada YMPE. For each relevant item, click on the name and set the increase rate to either a constant value, variable from library, or variable rate.
- 6** The **Interest Rate Tables** topic displays all historical interest rate tables referenced in benefit formulas and actuarial equivalence. Open each table and set it to use either the last known historical rate (the default) or specify an assumed rate.
- 7** Click **Save As New**.

## Step 6: Output Definitions

---

ProAdmin does not have an internal letter generation or letter merge facility. To provide the details for the external process ProAdmin has you create an Output Definition. This process allows you to set up field names and indicate which calculation result or data item is mapped to that field.

Estimate and final calculations may be run without an **Output Definition** unless you intend to use ProAdmin's **Fulfillment Tool**. For details on Output Definitions, see: [Appendix H: Output Definitions](#).

# Step 7: Run an Estimate

An estimate calculates benefit definitions, benefit formula components, accrual basis components, and payment forms based on the information set up in steps 2 - 6.

## Set Up an Estimate

Either from **Execute | Estimate** on the menu, or **Estimates** found under **Calculations** section of the **Shortcuts** pane, click **New** and specify the following:

**Estimated Benefit Calculation - [Jones]**

Name: Jones

**Data**

From Database

Linkage: Database Linkage

Person ID: 111-11-1111

From XML

Linkage:

File:

Load data & review  Populate calculation and override items on load

**Calculation**

Decrement type: All

One Decrement date: 03/31/2022

Multiple decrement/commencement dates

Commencement:

Date(s)	Age(s)
04/01/2022	

**Assumptions**

Plan Definition: ProAdmin Online - benefit calculations

Census Specifications: ProAdmin Online version 1

Projection Assumptions: General Assuptions

Override salary inflation assumption with %

Override cash balance interest crediting rate with %

**Output Definition:** SRV - Benefit Calculation Results

Load data & run View... **Replace** Save As New Erase Cancel

## Step 7: Run an Estimate

- In the **Data** section, select the **Data base Linkage** and specify the **Person ID**. Person ID is the identification number in the database that ProAdmin will use to retrieve the member data.

In the **Calculation** section, you can run either **One decrement date** or **Multiple decrement/Commencement dates**. If you select the **One decrement date** option, you must enter the decrement date in the box, enter the date in the format MM/DD/YYYY, and then enter all applicable commencement dates and ages for the run. All dates and ages that you enter are independent of each other and can be entered on any line under the appropriate column. The example above shows an estimate with a termination date of 05/31/2006 and commencement dates of 06/01/2006, 01/01/2007, and ages 55, 62, and 65.

If you select the **multiple decrement/commencement dates** option, you must enter the decrement dates and ages in the date(s) and age(s) spreadsheet. ProAdmin will use the dates and ages entered and the decrement date and then calculate a commencement date of the day after the entered dates and ages.

- Select the appropriate **Plan Definition, Census Specifications, Projection Assumptions** and **Output Definition**.

To execute the estimate, click the **Load data & run** button.

## ***Inspect the output***

There are three (3) steps in reviewing the calculation results. Step one is to look at the results at a summary level. This will give you a high level overview of all of the calculation pieces as of the plan year end preceding termination, dates of termination, and commencement dates. Step Two is required to see how the summary pieces were calculated. In this step you can access the detailed calculation results for each piece. These exhibits will give you details as of each plan year end from date of hire through termination, commencement dates, and in the case of monthly components the value as of each month within a plan year. Step three is to review the output that will be saved for use in the fulfillment process.

### **Summary Results**

- 1** Click the **View** button and select the **Summary Results** output. Note how the results display differ depending on whether you choose to **Show monthly annuities**.
- 2** Click the **View** button to display the **Estimate Calculation Output** report. The report can be printed or saved to a file. Click the **Print** button at the top of the screen to print it. Click on the **File** button at the top of the screen to save to a file. The report may be saved as a text file (comma delimited, text, or text with line draw characters) or as an Excel file.
- 3** Peruse benefit definition values as of the decrement and commencement dates. Inspect the annual payment form values as of the commencement date(s).
- 4** Click the **Close** button and select the **Detailed Results** to dig deeper into the calculation of the individual items displayed in the summary results to investigate any discrepancies.

### **Detailed Results**

- 1** Select the **Detailed Results** and click **View**.
- 2** Use the navigation pane on the left to switch between available reports.
- 3** The **Benefits...** button at the top determines the benefit definitions for which detailed results will be displayed. Click this button to see the list of **Benefit Definitions** applicable for this participant. Check the box in front of an item to include it in the results. Click the **Component Detail** button to select the benefit formula components, the accrual basis components, and, if applicable, the decrement dates for which detailed results are desired. Decrement dates should be entered in MM/DD/CCYY format. Click the **OK**.
- 4** Click **Print** at the top of the screen and select the report(s) to print. Click the **File** button at the top of the screen and select the report(s) to be saved to a file. Report(s) may be saved as a text file (comma delimited, text, or text with line draw characters) or an Excel file.

### **Step three – Output Definitions**

- 1** Select **Output Definition Results** to check the values based on what was mapped in Step 6: Output Definitions above.
- 2** Once you have reviewed the report you can send it to print, save it to a file, or close out of it. If the Output Definition is an Access type (as opposed to XML) and you intend to use ProAdmin's **Fulfillment Tool**, make sure to save the output to an Access database.



# Appendix A: Expressions

---

Expressions are used extensively within ProAdmin. Some examples include:

- creating data defaults;
- creating eligibility definitions;
- creating selection expressions; and
- defining benefit formulas.

This section describes the syntax of ProAdmin expressions.

## Expression Basics

Expressions may involve data dictionary field names, benefit formula component names, numbers, and a wide variety of operators, including the following standard arithmetic operators:

+ add	* multiply
- subtract	/ divide

For example, the expression

$$\text{AnnlzdPayRate} * 0.04$$

might be used in a benefit formula component to indicate that the annual cash balance accrual is four (4) percent of the data element annualized pay rate. Alternatively,

$$\text{Location} = 12$$

might be used in an eligibility definition to limit eligibility to participants in Location 12.

Depending on where the expression is being used in ProAdmin, either field names or benefit formula component names can be used, but not both. Thus, field names are usually used, but when benefits are being defined, benefit formula components are used. (However, if you attempt to use a field name in a benefit definition, ProAdmin will offer to create a benefit formula component for you that references the field.)

ProAdmin provides more than three dozen operators for use in expressions. Using a precedence scheme to decide which operators to execute first (as is done in many languages) would lead to a large and complicated set of rules. To avoid this problem, ProAdmin expressions use a simple, easily remembered rule: *ProAdmin starts at the left and works its way to the right, executing operators as they are encountered in the expression.* There is no precedence among operators.

For example, the expression:

$$\text{Salary2}-\text{Salary1}/\text{Date2}-\text{Date1}$$

is evaluated as:

$$((\text{Salary2}-\text{Salary1})/\text{Date2})-\text{Date1}$$

If the right argument to the division (/) is supposed to be `Date2-Date1`, it needs to be enclosed in parentheses, as in:

$$(\text{Salary2}-\text{Salary1})/(\text{Date2}-\text{Date1})$$

Left arguments don't need surrounding parentheses, but including them sometimes helps clarify the meaning of the expression (as done with `Salary2-Salary1` above).

ProAdmin allows any of the following pairs of parenthetical symbols to be used in expressions: `()`, `[]`, or `{ }`. Judicious use of these alternative symbols can help clarify expressions containing many nested parentheses.

Only a few of the operators available in ProAdmin have symbolic names (like `+` and `*`); most have word names. All non-symbolic operator names are preceded by a pound sign (`#`), as in `#MAX`. This helps you distinguish between operator names and field or benefit component names in your expressions, and it avoids requiring operator names to be reserved words that you cannot use as field or benefit component names.

Unlike many languages, ProAdmin uses an “infix” notation for all operators, even those which have non-symbolic names. For example, in most languages you would compute the larger of two numbers using `Max(Salary1, Salary2)`. In ProAdmin this is written as:

$$\text{Salary1} \#MAX \text{Salary2}$$

Although this may appear strange at first, it is really quite simple: named operators are used in exactly the same way as symbolic operators. The operator name is written *between* the arguments. You may wonder how this notation permits you to compute the maximum of three numbers, something conventionally written as `Max(x, y, z)`. The answer is, the same way as you add three numbers:

sum:	x	+	y	+	z
maximum:	x	#MAX	y	#MAX	z

You can insert spaces in ProAdmin expressions freely without changing the meaning of the expression. On the other hand, you *must* put a space between adjacent names (whether operator names or field names) to prevent ProAdmin from interpreting the result as a single name.

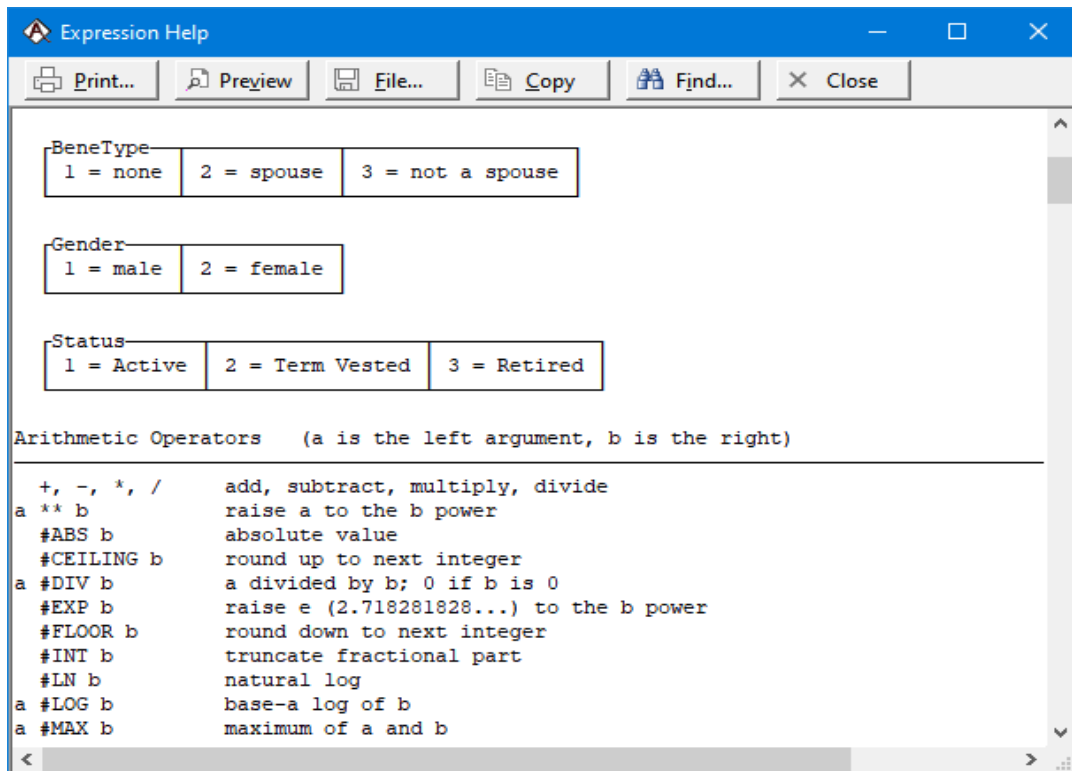
Note also that ProAdmin is *not case-sensitive*. In this manual, operator names are written in uppercase and field and benefit component names in upper- and lowercase, but you can use whatever case you like when entering expressions.

The following is a partial list of the most common arithmetic operators used in ProAdmin.

Operator	Description
$x + y$	adds x and y
$x - y$	subtracts y from x
$x * y$	multiplies x and y
$x / y$	divides x by y
$x ** y$	raises x to the power of y
#ROUND y	rounds y to the nearest integer
#INT y	rounds y down to an integer (truncate)
x #MAX y	maximum (larger) of x and y
x #MIN y	minimum (smaller) of x and y
x #ZMINUS y	x minus y, not less than 0

## Expression Help

When you are entering an expression, the **F1** key can be used to display an expression help dialog box, as illustrated below.



This help screen

- lists the fields defined in the Data Dictionary;
- shows the numeric codes that were specified to represent coded fields; and
- describes the ProAdmin operators available for use in expressions.

The cursor must be in an expression entry field for F1 to display this help message.

## Missing Values

Arithmetic operators treat missing values as a special case. If either or both of the arguments are missing, the result of the operation is missing. This processing is done on a record-by-record basis. For example, if fields A and B have the following values (with “\*” representing missing values), the sum of A and B would be computed as follows:

A	B	A+B
3	4	7
*	4	*
3	*	*
*	*	*
5	3	8

Generally speaking, ProAdmin does not permit missing values in any field that is referenced by a calculation. The exceptions to this rule are twofold. First, data defaults, which are designed to substitute assumptions for missing data, may operate on fields with missing values. Secondly, the operator #OVERRIDE is designed to use a field value if not missing, or otherwise the specified system calculation. Accordingly, if the only calculation reference to a #OVERRIDE field is by #OVERRIDE, the data is allowed to be missing.

## Date Arithmetic

In ProAdmin, dates are represented as integers giving the number of days from January 1, 1900 to the date in question. For example, January 2, 1901 is represented by the number 366, and May 14, 1993 is represented by 34,101 (these numbers are never seen by the user, but are described here for pedagogical purposes). In expressions, dates are treated as ordinary integers, and you can perform various types of date transformations using the standard arithmetic operators. For example “Date+7” adds seven days to a date; “Date1-Date2” returns the number of days between two dates, and the expression

$$\#ROUND[ (Date1+Date2)/2 ]$$

computes the date halfway between two dates.

In an expression, dates can come from the value in a field, or they can be written literally. Within ProAdmin, a **date constant** has the form mm/dd/yyyy, as in 12/31/1991. A constant of this form is converted to the integer that represents the date, and the integer is substituted for the date in the expression. For example, the expression

$$12/31/1991 - Date$$

computes the number of days from “Date” to December 31, 1991. (The result will be negative for dates after December 31, 1991.) The expression

$$1/1/1993 - 1/1/1992$$

computes the number of days between the two dates (which in this case, gives the number of days in 1992).

The system distinguishes between date constants and division of integers by the presence of two slashes within a single “token” (block of characters without a space). For example, 3/4 is interpreted as three divided by four, but 3/4/1992 is a date. If you actually want to divide three-quarters by 1992, use parentheses, as in (3/4)/1992.

Note that if you specify a year using two digits, it is converted automatically to a four-digit year falling in a “sliding dialog box” range such as 1930 to 2029. For example, 1/1/45 becomes 1/1/1945 and 1/1/01 becomes 1/1/2001. The dialog box extends roughly 30 years into the future and 70 years into the past. It remains fixed for 10 years, then jumps ahead a decade. Currently, the dialog box is from 1930 to 2029. On 1/1/2005 it will become 1940 to 2039. To specify a date outside the “sliding dialog box” range, provide the full four digits of the year, as in 1/1/1898.

Two operators provide additional flexibility in comparing dates. The **#YEARDIF** operator subtracts two dates and returns the difference in decimal years. The **#MONTHDIF** operator is similar, but it returns the difference in decimal months. These operators are similar to subtracting dates using minus (-) and dividing by 365 or 30, respectively, however they handle the variable number of days per year and month more precisely. The rounding functions **#ROUND**, **#INT**, and **#CEILING** can be used to convert the results of **#YEARDIF** and **#MONTHDIF** to integers. For example, the expression

```
#INT (1/1/1992 #YEARDIF Date_of_Birth)
```

computes the age of each person in whole years as of January 1, 1992.

At times you may need to add a number of months or years to a date, and it may not be convenient (or even possible) to express the duration as a number of days. For example, adding exactly ten years to a date requires that you add either 3652 or 3653 days, depending on how many leap years fall in the interval. Exact date-shifting can be performed using the **#DATEPLUS** operator, as in the following example:

```
Date #DATEPLUS 1y6m3d
```

This adds 1 year, 6 months, and 3 days to each date in the Date field. The right argument to **#DATEPLUS** is called a **duration**, and it specifies a time period in years, months, and days. The elements of the duration may be written in any order (e.g., 3d1y6m is the same as 1y6m3d), and elements which are zero may be omitted (e.g., 10y for 10 years). The **#DATEPLUS** operator adds the specified duration to the dates in the left argument. A similar operator, **#DATEMINUS**, subtracts a duration from dates.

Durations, like dates, are translated to a (fractional) number representing the approximate number of days in the time period. For example, the interval “1y” becomes the number 365.25. You can use durations with ordinary arithmetic operators if you like; for example, you can add or subtract durations or multiply a duration by a constant. However, if you add or subtract a date and a duration using + or -, the result will be the same as adding or subtracting a number of days, and this will not always be the same as the result produced by **#DATEPLUS** and **#DATEMINUS**. For example, the expression

```
Date + 1y
```

adds 365.25 days to the dates in the Date field. The result, when rounded to the nearest integer, will not be exactly one year later for dates falling in a leap year. Using the #DATEPLUS function instead of + would ensure that all dates are advanced by exactly one year.

There are several other date operators that can be extremely useful in the many data calculations essential to defined benefit administration. #NEXTBEGMTH, for example, rounds a date to the coincident or next following beginning of the month. #DAYOFWEEK returns a code (e.g., 1=Monday) for the day of the week that a date represents, and #LSTBUSDAY returns the date that is the last business day of the specified period.

## Relational Operations

Relational operators are used to compare numbers in expressions. For example, the expression

$$\text{Salary} > 30000$$

compares each record of the Salary field with the number 30,000. The result is expressed using 1s and 0s, with 1 meaning true and 0 meaning false. The result of this expression is 1 in records where Salary is greater than 30,000 and is 0 in records where Salary is 30,000 or less. The result is called a **Boolean** value because it consists of only the values 0 and 1.

One use for Boolean values is in arithmetic expressions. The 0s and 1s returned by a relational expression can be used in the same way as any numeric values, allowing you to write a variety of useful expressions. For example, the following expression classifies salaries by range:

$$(\text{Salary} > 10000) + (\text{Salary} > 25000) + (\text{Salary} > 55000)$$

The result of this expression is 0 for records where Salary is less than or equal to 10,000; 1 where Salary is between 10,000 and 25,000; 2 where Salary is between 25,000 and 55,000; and 3 where Salary is greater than 55,000.

Multiplication by a Boolean value is another useful technique. For example, setting the field Status using the expression:

$$[\text{Status} * (\text{Status} <> 12)] + [99 * (\text{Status} = 12)]$$

changes 12s in Status to 99s.

The following relational operators can be used in ProAdmin expressions:

= equal	<> not equal
< less than	<= less than or equal
> greater than	>= greater than or equal

These operators can be used with numeric fields or date fields, but not with character fields. (See below for how to compare character fields.)

Most relational operators treat **missing values** in the same way that arithmetic operators do: if either or both of the arguments are missing, the result is missing. However, equal (=) and

not equal (<>) handle missing values differently. If one of the arguments is the name #MV (which stands for “missing value”), the result has no missing values, and it simply reports where the other argument is missing. Thus, the expression

```
Sex = #MV
```

identifies those records where the `Sex` field is missing. On the other hand, if neither argument is #MV, the equal and not equal operators behave in the same way as other operators, i.e. the result is missing if either or both arguments are missing.

## Searching Character Data

The relational operators described above can be used only with numeric and date fields. To compare character values you must use the #IN and #NOTIN operators. The #IN operator asks if the left argument (a database field) has any of the values listed in the right argument. The right argument is a quoted string (text surrounded by single quotes) or a parenthesized list of quoted strings separated by commas. For example:

```
LastName #IN 'Smith'
LastName #IN ('Smith', 'Jones', 'Johnson')
```

The first expression selects values in which `LastName` is Smith; in other words, the expression returns 1 where `LastName` is Smith and 0 where it is not. The second expression selects values in which `LastName` is Smith, Jones, or Johnson. The #NOTIN function has similar syntax, but it returns 0 if the value occurs in the list and 1 if it does not. The search conducted by #IN and #NOTIN is not case-sensitive; searching for “Smith” will select values in which the name is spelled Smith, smith, or SMITH.

If the right argument to #IN is empty (just two adjacent single quotes, ' '), the result selects values that are blank. This allows you to select **missing values** of a character field. Note that the double quote character (") cannot be used as a substitute for two adjacent single quotes.

If a target string contains a quote, you must double it in the argument to #IN, as in 'That''s the ticket'. (Double-quote characters are discussed below; they actually have to be quadrupled in strings.)

The #IN and #NOTIN operators can be used with numeric fields as well. In this case the argument should be a parenthesized list of numbers separated by commas. For example:

```
Status #IN (1, 5, 7)
```

selects records in which `Status` is either 1, 5, or 7.

## Patterns

When searching for character values, the items in the right argument to #IN and #NOTIN may be either names or **patterns**. In a pattern, an asterisk (\*) stands for zero or more arbitrary characters. For example, the pattern 'A\*' matches any value that starts with A.

The pattern `'Smi*'` matches the names Smith, Smiley, and Smithers, but not Smuckers. A pattern may contain zero, one, or two asterisks. Here are some examples:

<code>*ers</code>	Values ending in “ers”
<code>*ch*</code>	Values containing “ch”
<code>Jo*s</code>	Values beginning with “Jo” and ending with “s”
<code>J*son*</code>	Values beginning with “J” and containing “son”

Spaces in a pattern are treated in the same way as non-blank characters—for the pattern to match, the value must contain spaces as specified in the pattern. For example, the pattern `'*A B*'` matches values containing the phrase “A <space> B”.

## Advanced Pattern Features

You can also include question mark (?) symbols in a pattern. Each question mark matches exactly one arbitrary character. For example, the pattern `'?BC'` matches ABC and BBC, but not XYZBC. The pattern `'A??'` matches all three-character values that start with A.

To select values that *don't* match a pattern, put a tilde (~) in front of the pattern. For example, `'~A*'` selects values that don't begin with A.

The patterns in a list are processed sequentially, from first to last, and each pattern adds to or removes from the previous selection. This is significant for patterns that start with a tilde. For example, the list (`'AB*'`, `'~??C*'`) first selects all values that begin with AB and then removes values that have C as the third character. This selects ABDEF, AB, and ABZ, but not ABC or ABCDE. If the first pattern in a list begins with a tilde, a tacit `'*'` pattern is added to the front of the list so the search starts by selecting everything, then removing items matching the first pattern.

To search for a value containing the symbols \*, ?, ~, or " (which are “syntactic characters”), you must surround the symbol with double quotes ("). The quotes can go around the entire pattern or around the syntactic character itself. Here are some examples:

<code>"*ABC"</code>	Values equal to <code>*ABC</code>
<code>"*"ABC</code>	Same as above
<code>"X?Y"*</code>	Values that start with X?Y
<code>*" "*"*</code>	Values containing *

If a syntactic character is outside double-quotes, it has its usual syntactic meaning; if it is within double-quotes, it's treated as an ordinary character.

To search for a double-quote character, you must double the double-quote within the surrounding double-quotes, as in:



"D" "Q"      Values equal to D"Q  
 D" " " "Q      Same as above

## Logical Operations

By themselves, relational operators allow you to write simple, single-term expressions. To write more complex expressions, logical operators are used to join together individual comparisons. For example, the expression “male employees whose salary is more than \$30,000” can be written as:

```
(Sex=1) #AND (Salary>30000)
```

In logic, two predicates connected by “and” form a true statement only if both predicates are true. The **#AND** operator returns 1 if both of its arguments are 1. If either or both arguments are 0, the result is 0. Like all operators, it works on a record-by-record basis. The **#AND** operator can be used to select records in which a numeric field falls within a particular range. The statement

```
(10000<=Salary) #AND (Salary<=20000)
```

selects records in which Salary is between \$10,000 and \$20,000, inclusive.

The other logical connective is **#OR**. Two predicates connected by “or” form a true statement if either or both predicates are true. The **#OR** operator returns 1 if either or both arguments are 1. The result is 0 only if both arguments are 0. To select employees who have more than 30 years of service or who are 65 or older, you could use the following statement:

```
(Service>30) #OR (Age>=65)
```

A third logical operator, **#NOT**, reverses logical values. It converts true to false and vice versa. Numerically, it converts 1 to 0 and 0 to 1. Thus, **#NOT(Age<65)** is the same as **Age>=65**.

## Array Operators

Operators that work on arrays are very useful when writing expressions in data defaults or transformation expressions in service and salary definitions. Below is the list of array operators and where they are offered in ProAdmin.

Operator:	Available in:
#DATE	Returns each date at which hours or salaries are evaluated. Available in transformation expressions and often used with #THIS.
#THIS	Provides the date prior to adjusting and rounding when used in Date Adjustments. Provides the numeric value prior to rounding when used in Numeric Rounding. Provides values at each effective date or stop date of the salary, hours or service-units field array being evaluated when used in transformation expressions. The values represent cumulative values within each measurement period.
a #ARRAY b	Creates either effective dated arrays fields or start & stop dated array fields from a list of dates and values. Available in Data Defaults.
a #EFFDATE b	Returns the earliest or most recent date in an array. Available in all expressions.
a #GETVALUE b	Returns the last non-zero value in an array, or the value corresponding to a specified date. Available in Data Defaults and transformation expressions.
a #MPNET b	Subtracts the next previous value within an array on a measurement period basis. Available in Data Defaults and transformation expressions.
a #MPSUM b	Accumulates values within an array on a measurement period basis. Available in Data Defaults and transformation expressions.
a #RANK b	Returns the rank (in integers) of the values within an array, on and before a particular date. Available in transformation expressions.

## Managing Complicated Expressions

As you develop longer and longer expressions, you may find that they become essentially unreadable. There are two ways to avoid this problem: by inserting line breaks and by using temporary variables.

When entering an expression, you can press **Ctrl+Enter** to start a new line. These line breaks do not affect the evaluation of the expression. For example, the following expression (entered as one long line)

```
((OldMultBen #max (FrozBen - Offset)) + NewBenPost98)
#max NewBenAllSvc) #max MinBen
```

could be written as:

```
{ [ (OldMultBen   #max (FrozBen - Offset))
    + NewBenPost98
  ]
  #max NewBenAllSvc
}
#max MinBen
```

Judicious use of line breaks and indentation can make it much easier to identify parenthesized terms and match up opening and closing parentheses.

Another way of avoiding unreadably long expressions is by using multiple statements and assigning intermediate results to temporary variables. Substatements in an expression are separated by ampersand (&), and assignment is denoted by colon-equals (:=). For example, the statement:

```
(A+B)/(1+A+B)
```

could be written as:

```
S:=A+B  &  S/(1+S)
```

The last substatement in an expression is the one that determines the overall result of the expression. The other substatements must define temporary variables.

Using substatements, assignment, and line breaks together offers many opportunities for writing clear expressions. The long expression shown above can be expressed as:

```
OldWearAway := OldMultBen #max (FrozBen - Offset) &
ExtWearAway := (OldWearAway + NewBenPost98)
               #max NewBenAllSvc &
ExtWearAway #max MinBen
```

Note that even when line breaks are used, you still need to put an ampersand between statements.

# Appendix B: Libraries

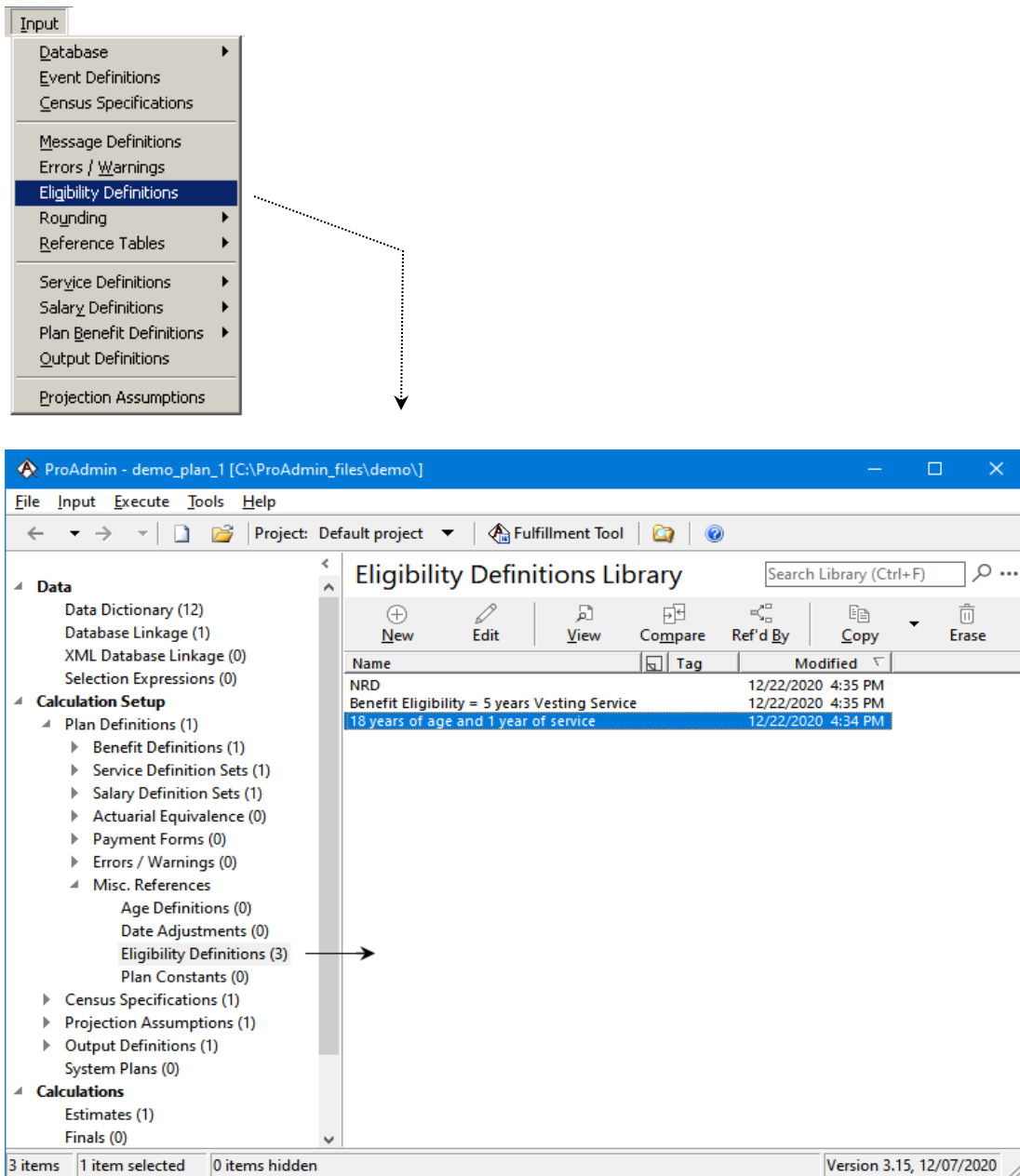
---

## *Libraries*

Within each client, the different types of objects you create are stored in different ProAdmin libraries. For example, one library holds plan definitions, another eligibility definitions, another output definitions, and so forth. Although the libraries hold very different types of data, certain operations and concepts are common to each of them.

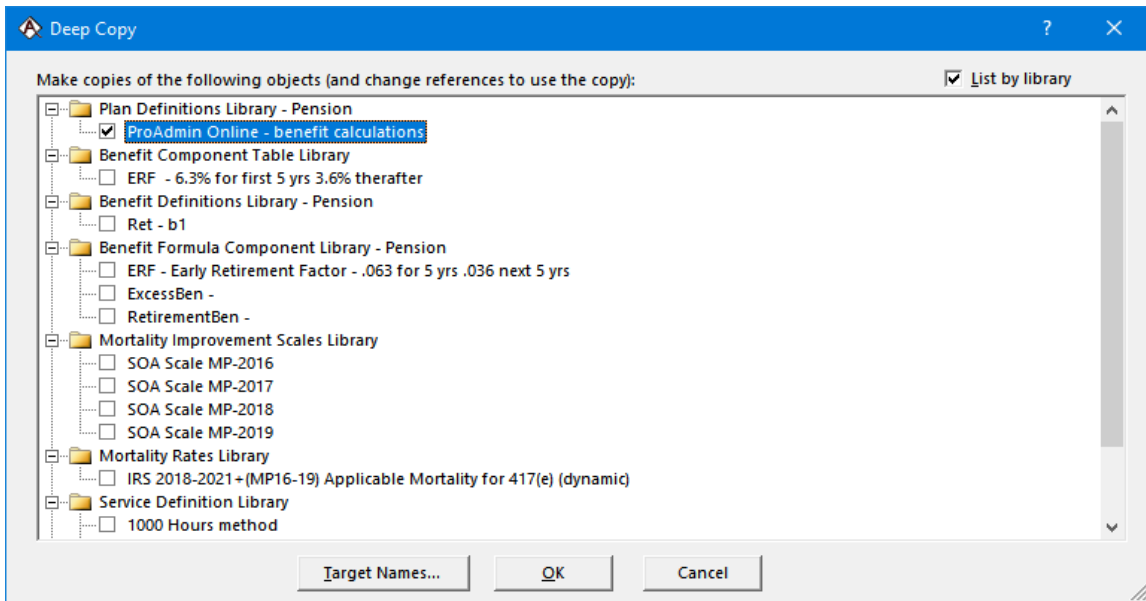
Each library can be accessed through a command on the menu. Libraries found under the **Input**, **Execute** and **Tools** menus can also be accessed through the **Shortcuts** pane. For example, the library shown below can be accessed either through the **Input | Valuation Assumptions** command or through the **Shortcuts** pane. The **Entries** pane displays the contents of the selected library. The tool bar for the **Entries** pane contains buttons for several library entry operations, including New, Edit, Rename, Copy, Erase, Hide, Unhide, Compare, View, Import and Run (where applicable). A typical **Entries** pane is shown below.

## Appendix B: Libraries



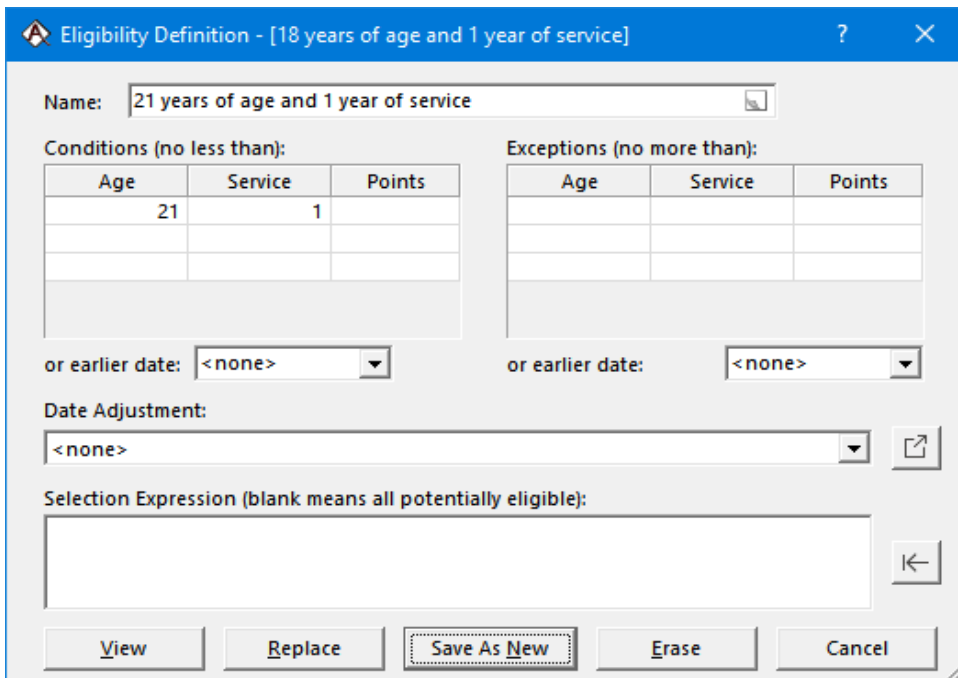
You can create a new library entry in four ways:

1. Click the **New** button found at the top left of the **Entries** pane, or
2. Select an existing entry, click the **Copy** button to make a copy, and **Edit** as necessary, or
3. Select an existing entry, click the **Edit** button, and then exit by clicking the **Save As New** button (see below).
4. Select an existing entry, click the little arrow next to the **Copy** button and select **Deep Copy**. This opens a new dialog box which displays all objects referenced within that object from which you can select other objects to copy, specify **Target Names**, and update references:



This allows revision of a Plan Definition while keeping the original Plan Definition intact without manually creating duplicate benefits and components and updating all of the references.

Selecting one of the library entries (or clicking New) opens up a library **editing dialog box**, which displays the contents of the library entry. You can view the contents or modify them as you wish. For example:

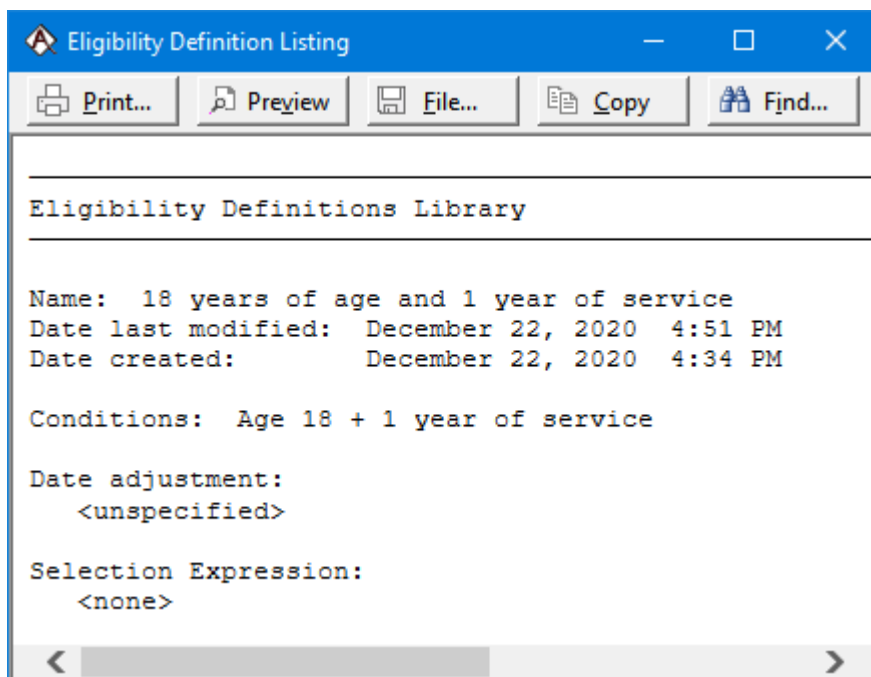


Along the bottom of the editing dialog box are the following four buttons which can be used to exit the dialog:

- **Replace** saves the entry back in the library, retaining any changes that you made and writing over the previous version of the entry.
- **Save As New** makes a copy of the entry you are editing and saves the copy in the library. The original library entry is not altered. You may want to change the name before clicking Save As New so the new entry will have a unique descriptive name. However, if you do not change the name, ProAdmin will suggest a unique name by adding, “#2” to the end of the original name.
- **Erase** erases the library entry. As described in the Appendix C: Projects, this operation is not reversible; ProAdmin will issue a warning and get confirmation from you before erasing the entry.
- **Cancel** exits the dialog without saving any changes that you made. If you have made changes, ProAdmin will warn you of this and get confirmation before exiting.

When you exit the editing dialog, you are returned to the **Entries** pane where you can select a different library entry for editing.

In addition to the cancel buttons, many edit dialogs have a **View** button. Clicking the View button will produce a complete summary of the library entry (see sample below), which can be viewed, printed or saved to a file for reference and documentation.

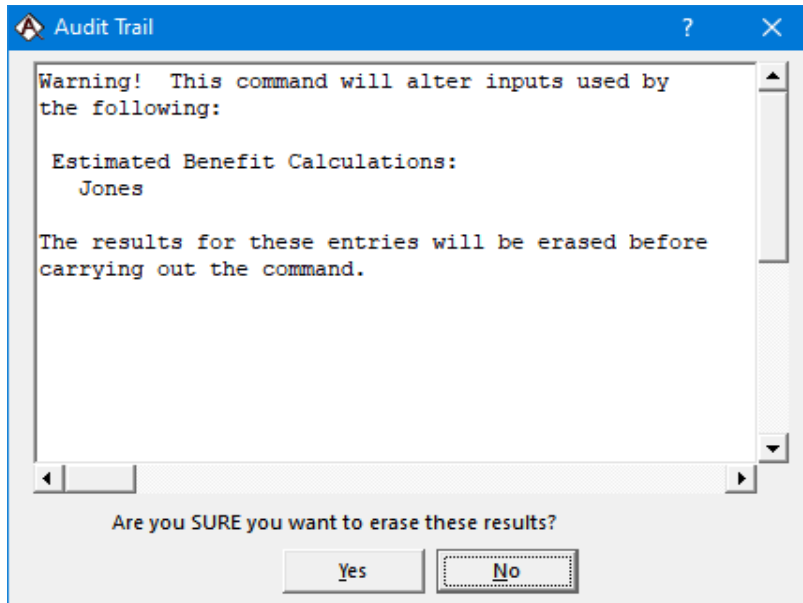


## Audit Trail

When you run an estimate or final calculation, ProAdmin makes note of the inputs that were used to generate the results, and it protects these inputs against later change. This is done so you can determine, with absolute certainty, what inputs were used to produce a given set of output. This protection of inputs is referred to as the **audit trail**. The audit trail protects all library inputs that were used to compute the output.



In practice, you are not actually prevented from making changes to the inputs. Instead, when you attempt to alter an input, ProAdmin displays a warning message listing the calculation runs that will be lost if the associated input is changed and asks if you are sure that you want to modify the input.



If you answer:

- “**Yes**” to change the input, ProAdmin will erase all of the affected output. To produce the revised results, simply go back to the estimate command, for example, and rerun the affected calculations.
- “**No**” to retain the output, you may then choose to save the modified input item as a new entry in the library with a revised name.

## ***Other Consistency Checks***

Some library entries in ProAdmin refer to entries in other libraries. For example, a Plan Definition refers to Benefit Definitions; Benefit Definitions include a formula that refers to Benefit Formula Components; and Benefit Formula Components may refer to Accrual Basis Components. Entries that are referenced by a higher-level entry (such as a Benefit Definition that is referenced by a Plan Definition) are protected against erasure. Before you can erase a referenced low-level entry, you must first remove the reference by the higher-level entry. There is, however, no protection against modification to lower-level entries that are not currently referenced in output; you can modify an object whether or not it is referenced.

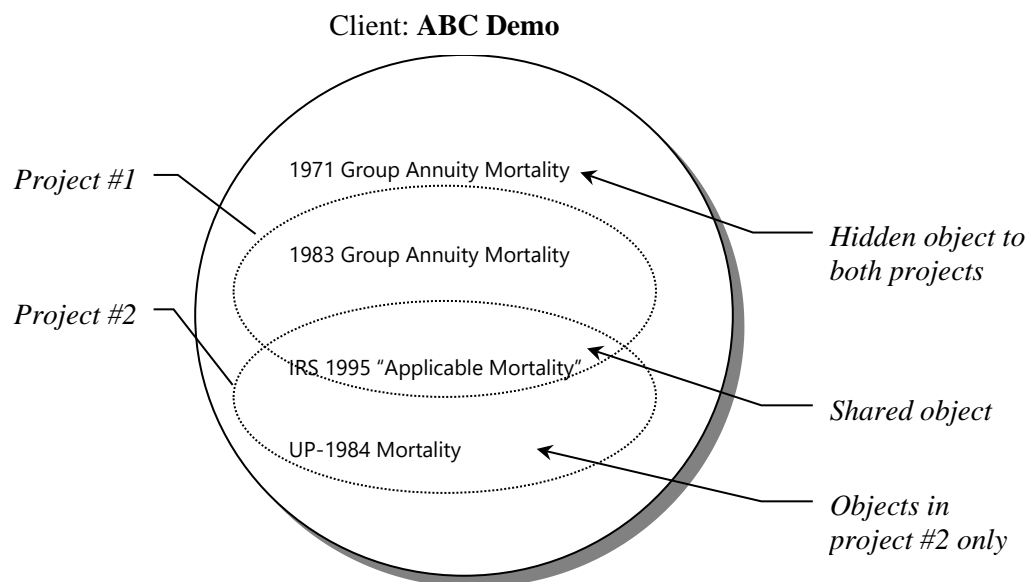
When you replace a library entry that is visible in more than one project, ProAdmin displays a message warning you that other projects will be affected. If you don't wish to alter the entry in the other projects, you can save the modified object as a new library entry. However, if you do so, beware that no higher-level objects will refer to the new object. If there are supposed to be references to it, you will have to manually change the high-level objects to reference the new entry. For example, if you change a benefit definition and save it as a new benefit, you will have to revise plans to include the new benefit.

# Appendix C: Projects

After working in ProAdmin for a while, you may find that your libraries are getting uncomfortably large. A large library means that you have to select from a long list to specify an item in the library. For some users, only a portion of the library may be relevant to any given task. For example, some library entries for the benefit definitions may only be needed when working on an early retirement window for the client. Similarly, library entries for the Salaried Plan may not be needed when working with the Hourly Plan. ProAdmin provides a way of hiding library entries so you can see just the entries that are relevant to your current task. The hiding is implemented by means of projects.

## Projects

You can define any number of projects within ProAdmin, and you can select a project to use as the **current project**. Within a project, you decide which library entries should be visible and which should be hidden. When you create a new library entry, it is automatically made visible in the current project and hidden in other projects. You can **unhide** an entry from another project in the current project, thereby making the entry visible in both projects. When you **erase** an entry, it is permanently erased from the library. There are two safeguards to this type of erasure. First, ProAdmin refuses to erase an object that is referenced by another library, and second, it warns you and requires confirmation before erasing the object.



An object that is visible in more than one project is **shared** by those projects. Only one copy of the object is stored, however, and *if that copy is changed from within one project, it is changed in the other projects as well*. This behavior is especially useful for reference tables. A single copy of a table can be shared by all projects, and when a

correction or update is needed, the change can be entered just once. However, for other types of objects this sharing may not be desirable. The **Save As New** button can be used to make an independent copy of an object so that changes in one project will not affect other projects. Whenever you replace a shared object in a library, ProAdmin warns you that the operation will affect projects other than the current project. You can then decide whether or not you really want a shared object.

## ***Managing Projects***

You can use the **File | Change Project** command or Project / Mode dropdown on the toolbar to open, edit, rename or copy existing projects, create a new project, or erase projects. The name of the current project is displayed in the toolbar at the top of the screen:

- **Open** facilitates switching between projects. This operation opens the highlighted project as the current project. Only objects that are unhidden for the opened (ie: current) project will be displayed for each library.
- **New** brings up a dialog box to create a new project. You must provide a Project Name. Optionally, you may check the box to Unhide All Objects. This option will display all library objects that can be utilized. Otherwise, all objects will be initially hidden in the new project, meaning all libraries will appear to be empty. As you edit the libraries, you can unhide objects as needed from other projects.
- **Edit** displays the library entry for the highlighted project. This operation allows you to edit the entry, create a new entry, or erase the entry. Edits can be made to the entry name, or to unhide all objects. Replace will save the edits to the existing project. Save as new will create a new entry based on the edits. Erase will erase the project (however, no objects associated with that project will be erased). The Universe project cannot be edited.
- **Rename** allows you to edit the name of the highlighted project without entering the Edit dialog box. The Universe project cannot be renamed.
- **Copy** makes a copy of the highlighted project(s). A copy of a project will have the same library objects hidden/unhidden as the project from which it was copied. This operation produces the same result as highlighting a project, clicking Edit, making no changes and clicking Save as new. Highlighting multiple projects creates a single copy of each highlighted project. The Universe project cannot be copied.
- **Erase** erases the highlighted project(s). This operation produces the same result as highlighting a project, clicking Edit and clicking Erase. Multiple projects can be erased at once by highlighting multiple projects. The Universe project cannot be erased. Erasing a project or projects will not erase any objects which are associated with the erased project(s). This prevents other projects from being affected when a project is erased.

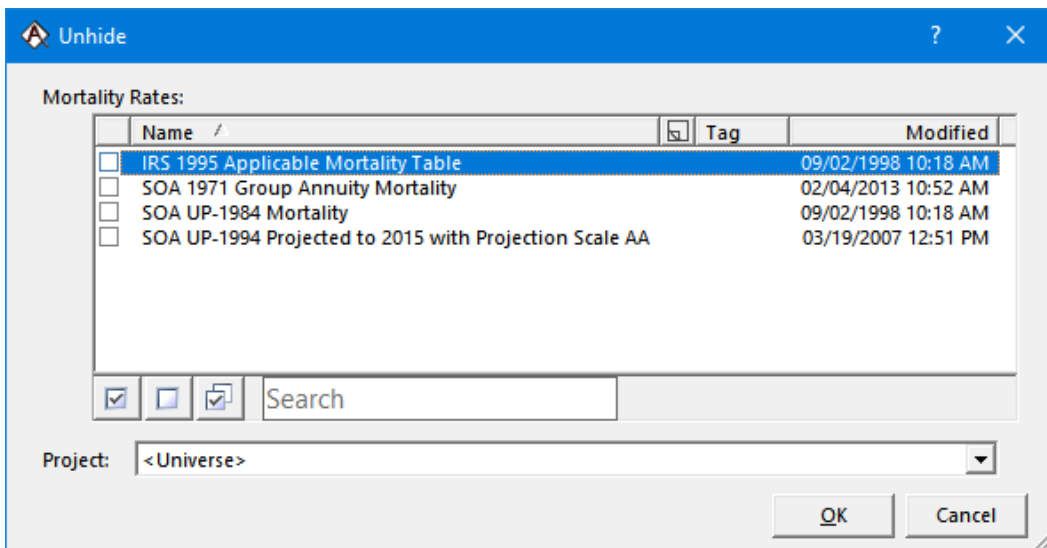
**Note:** After creating a new project, the first thing you should do is use the Database | Data Dictionary command to unhide database field definitions in the new project. The method for doing this is described below.

## Universe Project

It is possible for an object to be visible in no projects (that is, be hidden in every project). This happens, for example, when an object belonging to a single project is erased. The **Universe** project allows you to access such objects. The Universe project contains every object stored in the libraries, regardless of project membership. When you erase an object from a project, the object is still available in the Universe project; this allows you to “unerase” the object by unhideing it from the Universe project as described below. Also, you can select Universe as the current project using the File | Change Project | Mode command. This causes all objects to be visible when you edit a library.

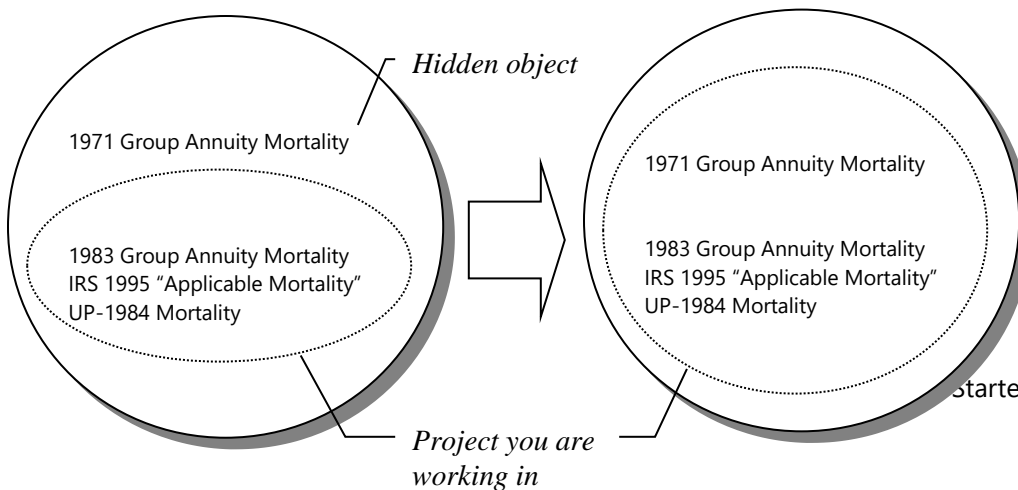
## Unhiding Objects

In most libraries, the entry selection dialog has an **Unhide...** button at the bottom of the dialog. When you push Unhide, ProAdmin displays an unhide dialog, such as the one illustrated below from the Input | Reference Tables | Mortality Rates command.



You may select one or more objects, or you can push the All button to select all objects. When you push the OK button, the selected objects are unhidden in the library you were editing, and you are returned to the entry selection dialog in which you started.

### Unhiding an object



By default, all objects that can be unhidden are displayed (i.e. those in the Universe but not in the current project). You can shorten the list by choosing another project to unhide from. The choices include every project you have created plus the Universe and **Orphanage** projects. The **Orphanage** project contains objects that are not visible in any project other than the Universe. The Orphanage can't be selected as the current project, but it appears as a choice when you are un hiding objects.

To make an independent copy of an object (not shared with another project), first unhide the object as directed above and then edit the object. Enter a new descriptive name for the object and push the **Save As New** button. Then, once again edit the original object and push the **Hide** button to remove the shared copy from the project.

**Note:** It is better to make an independent copy of an object as soon as you unhide it from another project instead of waiting until you revise the object and then using Save As New to make the independent copy. If you wait until later and in the meantime create other library objects that refer to the original (shared) entry, you will have to change those references when you create the new, independent copy.

## ***Object Descriptions***

Objects in the libraries are identified by their descriptive names. Within a given library, each object must have a unique name. This applies even across projects: two objects may not have the same name even if the objects reside in different projects. When you save a new object, you must provide a name that does not yet exist in the library. If you attempt to save an object with the same name as another object in the library, ProAdmin displays a **Duplicate Name** warning and proposes a unique description. The proposal consists of the description you supplied followed by a numeric suffix (for example, “Disability benefit #2”). You can either accept the proposed description or change it to something else.

The formula components used in benefit formulas and accrual basis expressions are handled slightly differently. Each component has both a name (which is used in the formulas) and a description. The names must be unique, but the descriptions need not be unique. If you provide a name that is already in use, ProAdmin does not propose a unique name, but instead shows you the list of names already in use and asks you to provide a new name.

# Appendix D: Shortcuts

---

This chapter describes some keyboard shortcuts to operate ProAdmin. Although ProAdmin operates in an intuitive way that is similar to other Windows-based software packages, ProAdmin has a number of specialized features. For this reason, even experienced software users will benefit from reading this chapter.

## Mnemonics



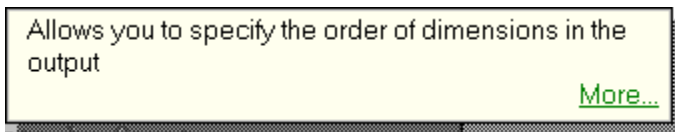
Mnemonics are the underlined letters in menus and buttons. For menus, entering this **letter** executes the menu command. For buttons, pressing **Alt+letter** pushes the button.

## Dialog Box Basics

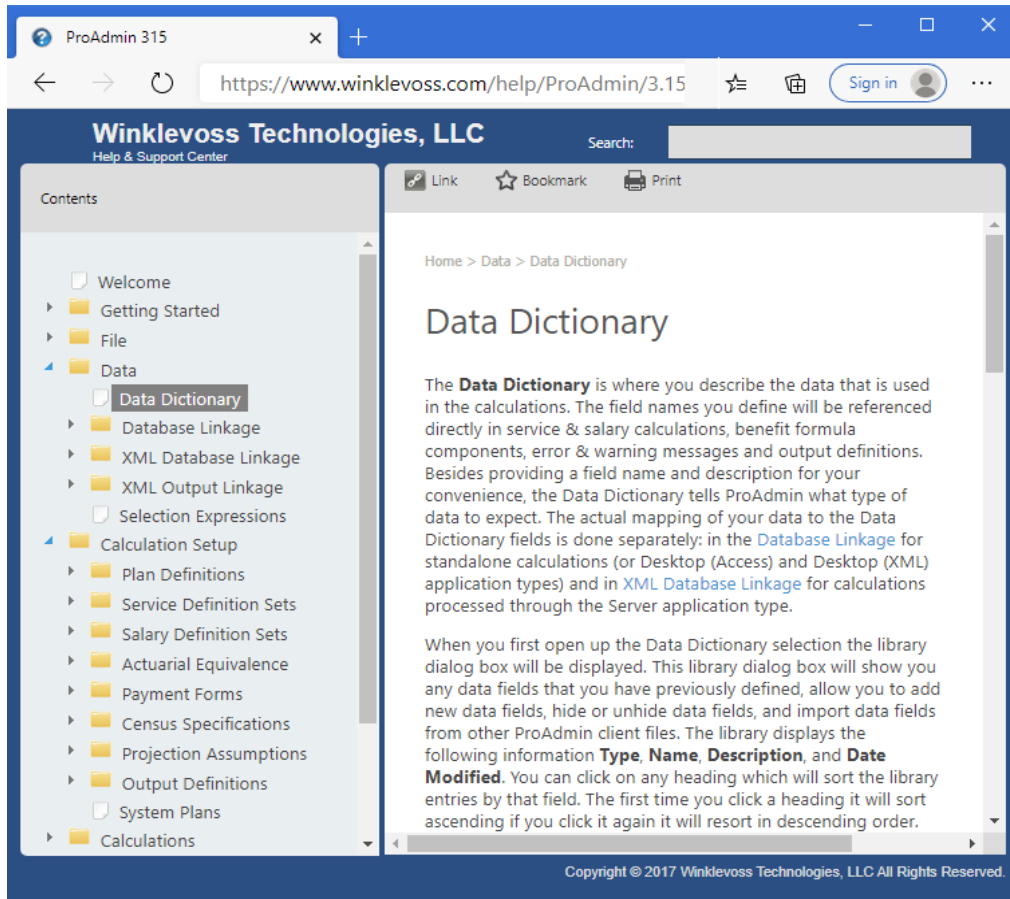
In any dialog, you can use the **Tab** key to move your cursor from field to field. To go backwards, use the **Shift+Tab** key combination.

Clicking the **X** button on the top of any dialog has the same effect as pushing the Cancel or Exit button with the mouse or hitting the **Esc** key.

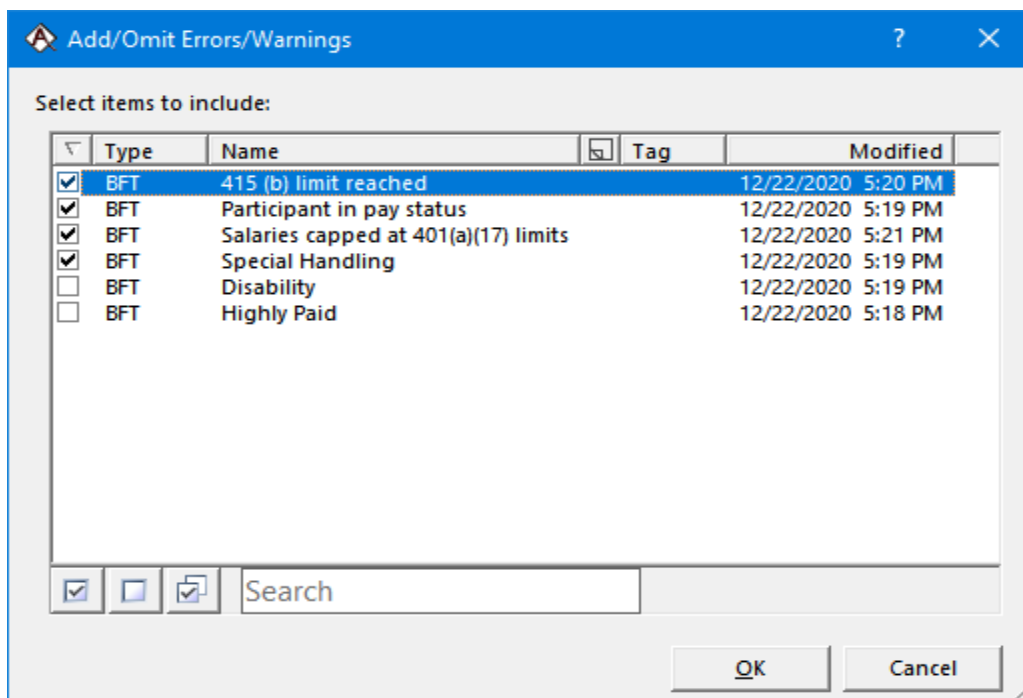
Clicking the **?** button on the top of any dialog and then clicking on a field brings up context-sensitive help to answer questions such as “What is this?” and “Why would I use it?” Alternatively, you can use the **Shift+F1** key combination.



For more help click the More... link in context-sensitive help or press **F1**.



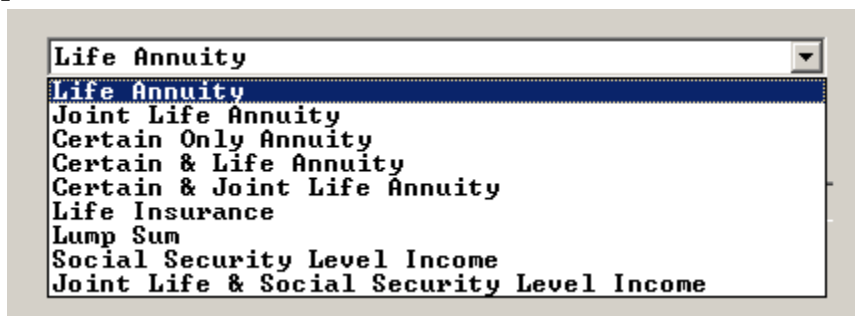
## List Boxes



To select an item in a list box, press **Enter** or the **Spacebar** and a checkmark will appear. List boxes come in two varieties: **single selection** and **multiple selection**. Visually, you can recognize multiple selection lists by the checkbox next to each item (as shown in the figure above). At times, you may want to select all items in a multiple selection list. To do this, simply press **Ctrl+A** (some dialogs have an “All” button that allows you to do this with the mouse). To clear all selections, press **Ctrl+N**. You can also select multiple items by clicking the first item and then holding down the shift key while clicking the last item.

To move the cursor using the keyboard, press the **Up** and **Down** arrows. Alternatively, press a letter key to move the cursor to the next item beginning with that letter (you would press the letter W to move to “W – Age at DOH > 59 Check NRD” in the example above). If several items in the list start with the same letter (“W - Special Handling 01”, “W - Special Handling 02”, “W - Special Handling 03”, and “W - Special Handling 04” in the example above), you may have to press the letter key several times to reach the desired item.

## Drop-down List Boxes



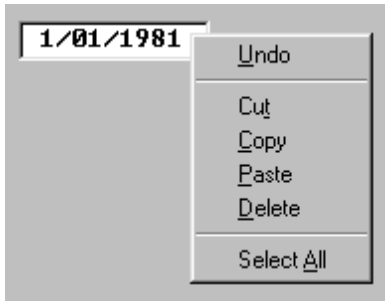
Like a single selection list box, a drop down list box allows you to choose a single item from a list. The difference is that the list is displayed upon demand.

When the list is closed, you can cycle through the choices using the **Up** and **Down** arrows. To display the list, press **F4**, **Alt+Down arrow**, or the **Spacebar**. Then, select an item by pressing **Enter**.

Alternatively, you can press a letter key to select the next item beginning with that letter (you would press the letter S to select “Social Security Level Income” in the example above). If several items in the list start with the same letter, you may have to press the letter key several times to select the desired item.



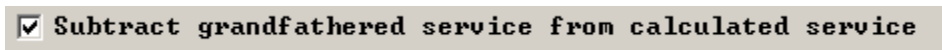
## Number, Date, and Text Fields



Several features are available when entering numbers, dates, and text. These are available by right clicking with the mouse or using the following keystrokes:

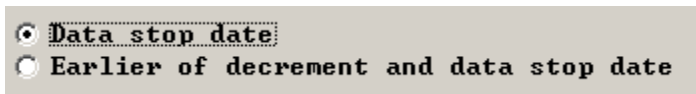
<b>Ctrl+X</b>	cut
<b>Ctrl+C</b>	copy
<b>Ctrl+V</b>	paste
<b>Ctrl+Z or Alt+Backspace</b>	undo

## Check Boxes



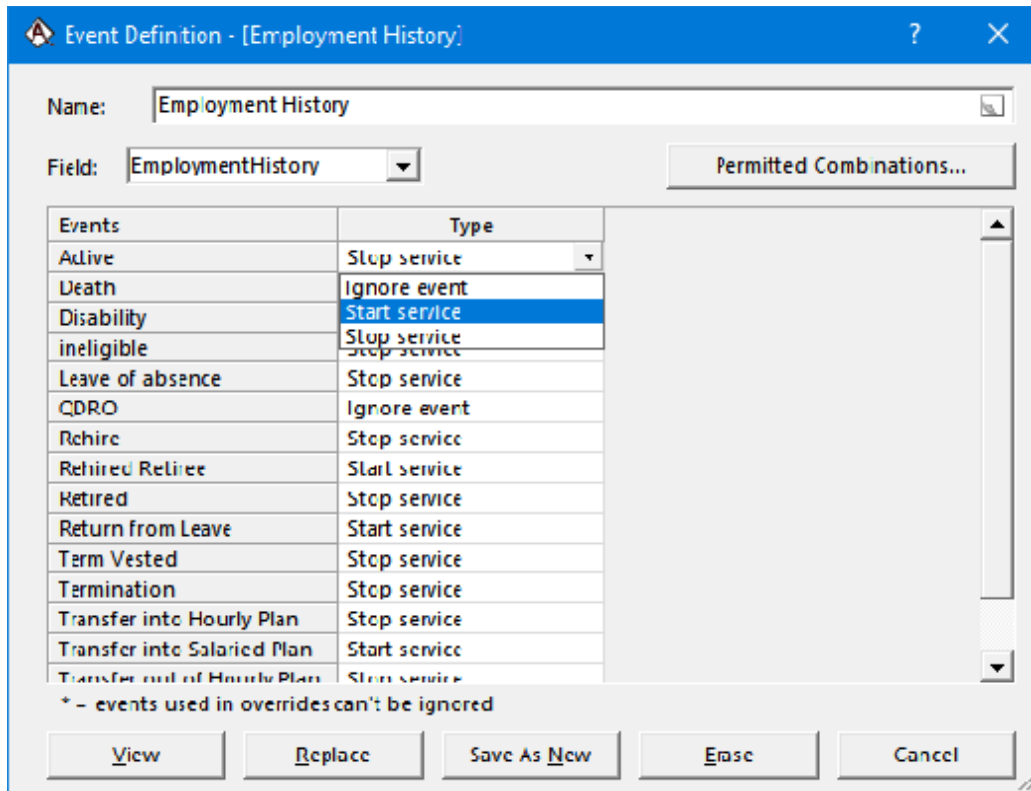
To toggle a check box with the keyboard, use the **Spacebar**.

## Radio Buttons




To change the setting of a radio button with the keyboard, use the **Spacebar** or the **arrow keys**.

## Spreadsheet Fields



A spreadsheet field is a grid of values. They are usually numbers, but may also include dates, characters, and drop-down lists. **Warning:** the shortcuts for drop-down list boxes cannot be used when they appear in spreadsheets!

To edit the value in a cell, press **F2** (or double click with the mouse). Alternatively, you can simply begin typing the replacement value. When you finish entering or revising the value, press **Enter** or **Tab**. The cursor will automatically advance to the next cell down (to the right, if you pressed Tab). Alternatively, you can press an **arrow key** to move in a specific direction.

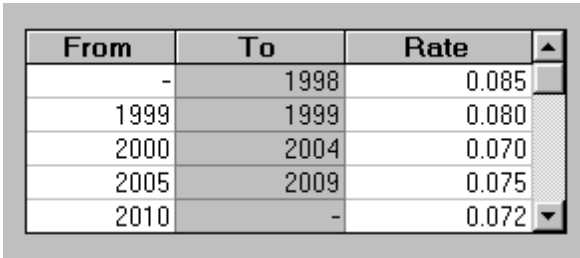
In addition, several other features are available by right clicking with the mouse when you see the  cursor or by using the following keystrokes:

- Ctrl+Ins**      insert a row above the cursor (when available)
- Alt+Ins**      insert a row below the cursor (when available)
- Ctrl+Del**     delete the row the cursor is on (when available)
- Ctrl+D**        duplicates the current cell down
- Ctrl+C**        copy
- Ctrl+V**        paste

Some of the columns of a spreadsheet may contain computed data or data that is displayed for reference purposes only. These columns cannot be changed, and they are displayed in gray.

## ***From-To Tables***

Several ProAdmin commands allow you to enter a quantity, such as an interest rate, that varies with time. These quantities are generally entered using what is called a **From-To Table**. An example is shown below.



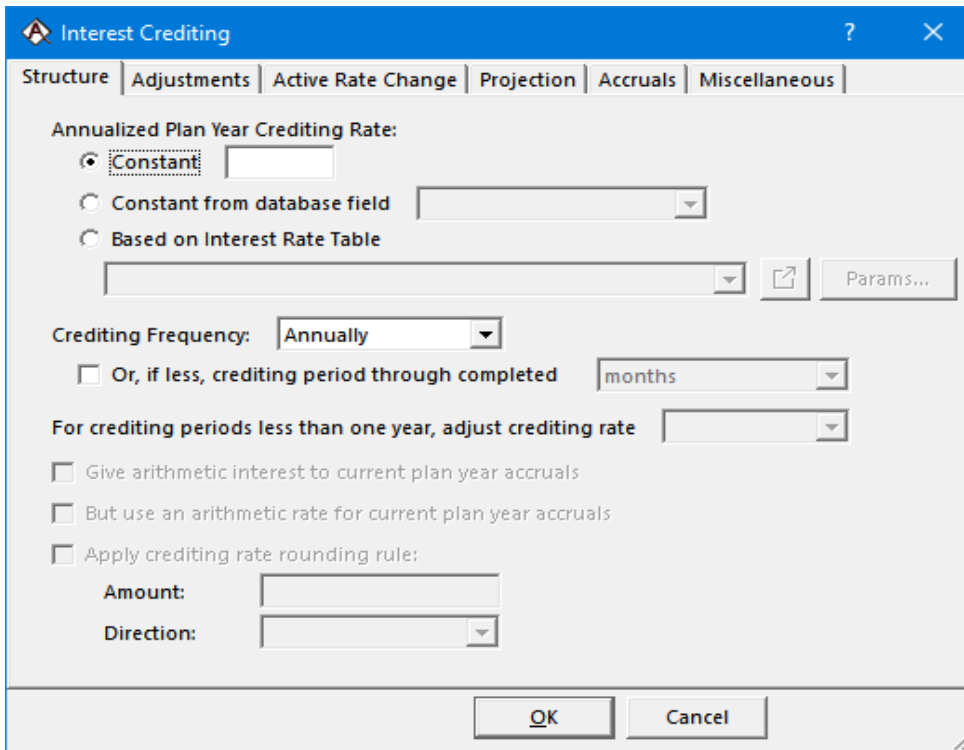
From	To	Rate	
-	1998	0.085	▲
1999	1999	0.080	
2000	2004	0.070	
2005	2009	0.075	
2010	-	0.072	▼

A from-to table is really just a special kind of spreadsheet. All the shortcuts that apply to spreadsheets also apply to from-to tables.

However, entering data in a from-to table is a little different. You enter data only in the first and last columns of the table. The middle “To” column is computed automatically based on your entries in the “From” column. In addition, the hyphens (-) in the first and last rows cannot be changed. The first hyphen (in the From column) represents the beginning of time. The second hyphen (in the To column) represents the end of time.

## Tabs

To move through a series of tabs, as in the example below:



**Ctrl+Tab** or **Ctrl+Page Down**      move forward through tabs

**Ctrl+Shift+Tab** or **Ctrl+Page Up**      move backward through tabs



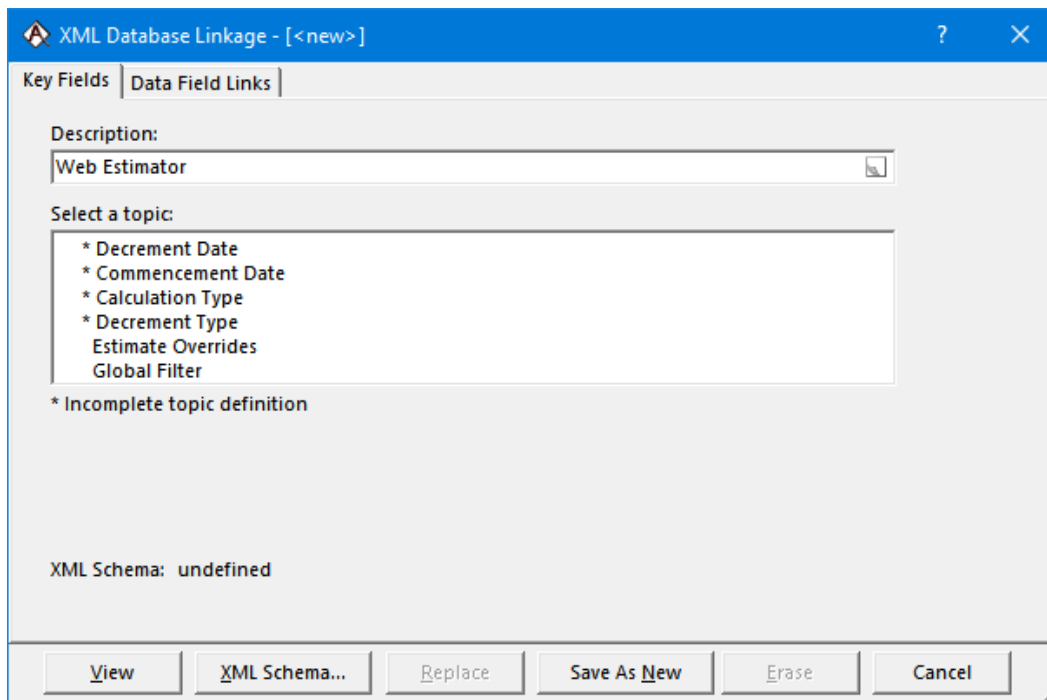
# Appendix E: XML Linkages

## XML Database Linkage

Please note that the XML topics in this chapter (XML Database Linkage and XML Output Linkage) only apply if you are using ProAdmin Server in addition to ProAdmin Desktop.

In ProAdmin Server, the software executes a calculation when it receives an HTTP post. The body of this message contains the member data represented in an Extensible Markup Language (XML) format. In the same manner as was done for ProAdmin through the Database Linkage command, ProAdmin Server needs to have a relationship created between the Data Dictionary and the information contained in the XML document. This is done through the XML Database Linkage command.

To create a new entry in the library, click on the New button to bring up the XML Database Linkage dialog.



The first step in the process of creating a new library entry is to import the XML schema document (XSD). This document defines the layout of the member data. The following is a sample of a partial XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="Calc_Request">
    <xs:complexType>
```

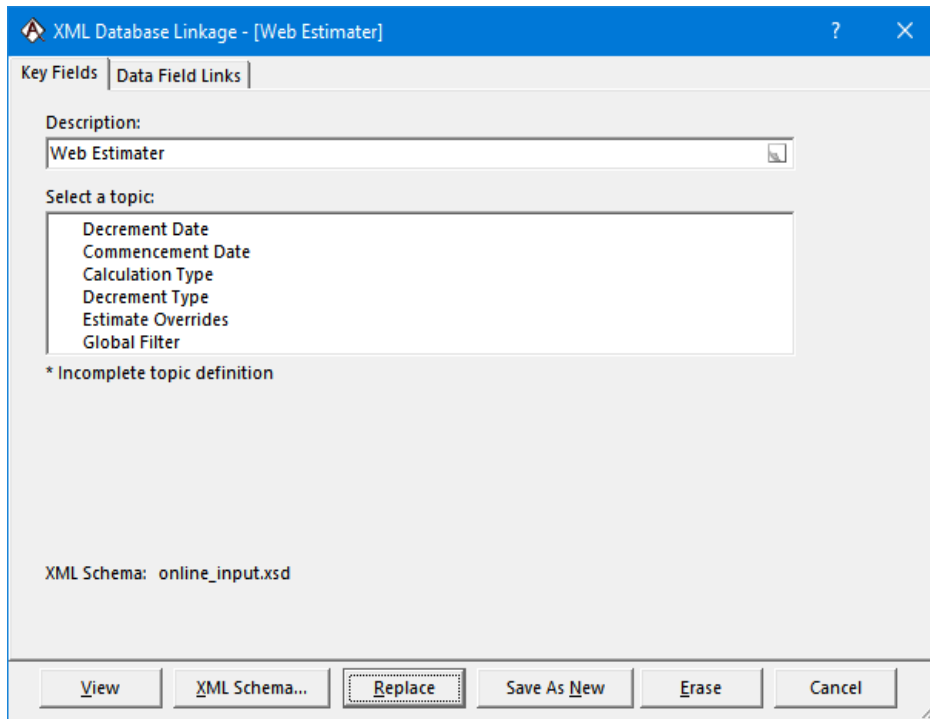
```

<xs:sequence>
  <xs:element name="Header">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Run_ID" type="xs:string"/>
        <xs:element name="Plan_ID" type="xs:string"/>
        <xs:element name="Location_ID" type="xs:string"/>
        <xs:element name="Social_Sec_No" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="User_Input">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Request_Type" type="xs:string"/>
        <xs:element name="Decrement_Type" type="xs:string"/>
        <xs:element name="Termination_Date" type="xs:date"/>
        <xs:element name="Commencement_Date" type="xs:date"/>
        <xs:element name="Beneficiary_Date_Of_Birth" type="xs:date"/>
        <xs:element name="Beneficiary_Type" type="xs:boolean"/>
        <xs:element name="Beneficiary_Gender" type="xs:string"/>
        <xs:element name="Salary_Scale" type="xs:double"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Database_Value">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Indicative">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="DOB" type="xs:date"/>
              <xs:element name="FNAME" type="xs:string"/>
              <xs:element name="LNAME" type="xs:string"/>
              <xs:element name="SEX" type="xs:string"/>
              <xs:element name="MARRIED" type="xs:string"/>
              <xs:element name="DOM" type="xs:date" true"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

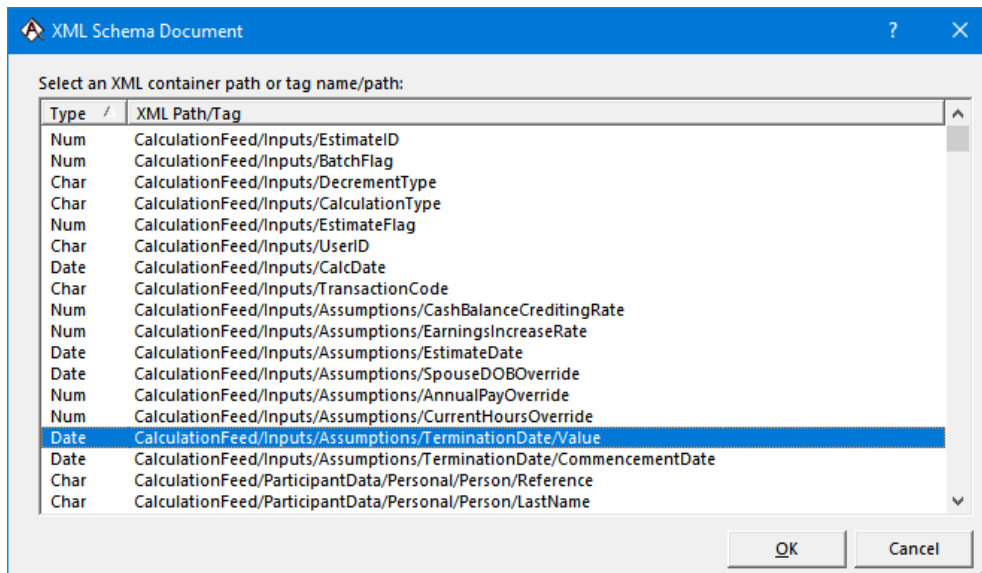
To import the schema, click on the **XML Schema** button and select Import. This will bring up the Import XML schema document dialog box. After either selecting the file or typing in the name, click the Open button. A message will display indicating if the import was successful. After a successful import the bottom of the screen will display the name of the XML schema that was imported and is now associated with this library entry.

There are two tabs on the XML Database Linkage: Data Field Links and Key Fields.



**Description** can be a phrase of any length where all characters are permissible. You should include key information about the use of this library entry

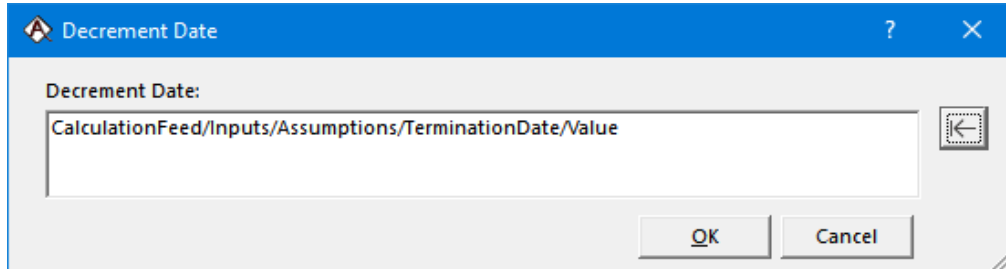
The information on the **Key Fields** tab is the items that are required to process a calculation, where a preceding asterisk (\*) indicates that the item does not have a complete XML link definition. One or more containers must be specified for each item type. You access the XML Data Link definition for an item by clicking on the field name in the list to display the XML Data Link menu. This will display the dialog box for that entry. There are two ways to enter the information required. The first is to type in the path. The second is to click on the arrow button after the field. This displays a dialog box which allows you to select the appropriate path from the XSD that you imported.



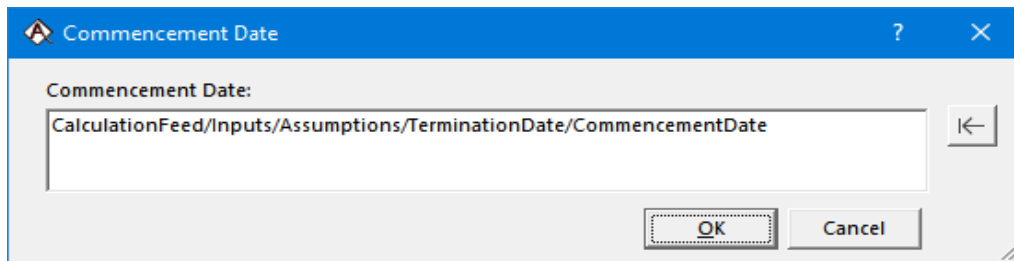


The following items must be linked:

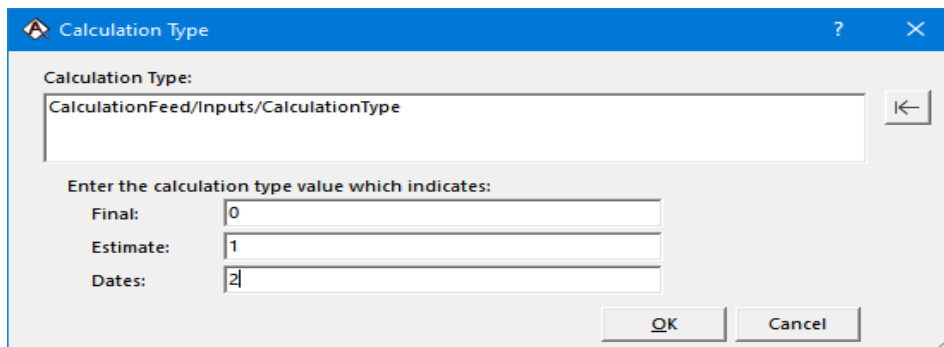
- **Decrement Date** is the date the member either did or is assumed to cease employment. You may either type in the entry or click on the arrow button after the field to select the appropriate container and tag name for the entry.



- **Commencement Date** is the date that payments are assumed to commence. Multiple commencement dates can be specified in the document, each of which will be within the specified XML container. You may either type in the entry or click on the arrow button after the field to select the appropriate container and tag name for the entry.

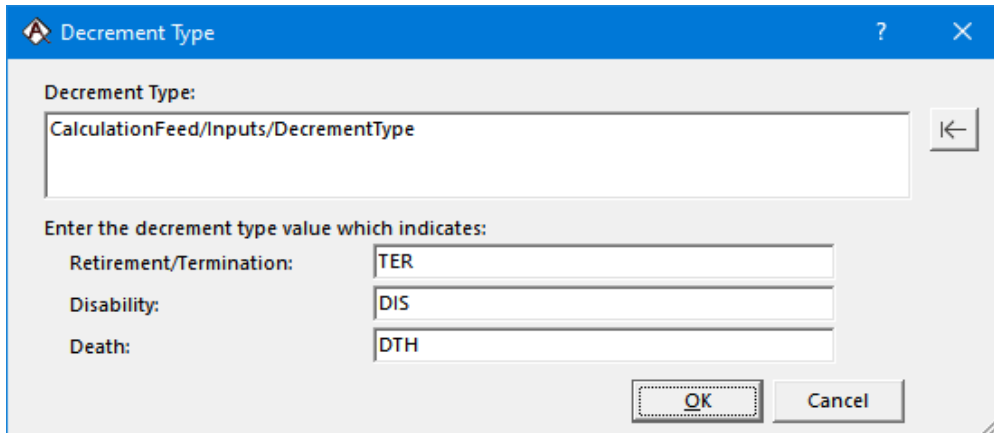


- **Calculation Type** indicates whether the calculation is of the type estimate, final, or date/age/service. In addition to the XML container name for calculation type, you will also enter on this dialog the codes that will indicate whether the request is an estimate, final calculation, or date/age/service. You may either type in the entry or click on the arrow button after the field to select the appropriate container and tag name for the entry.

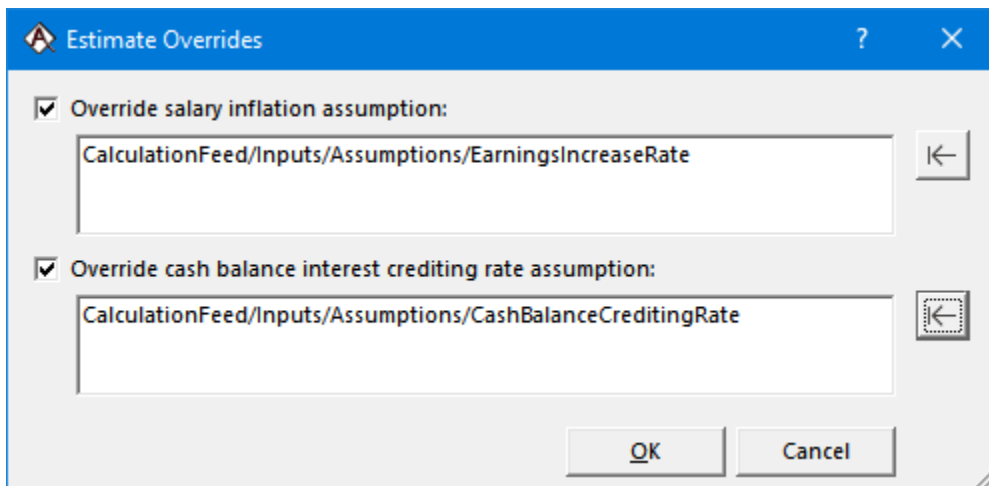


- **Decrement Type** indicates the type of decrement. ProAdmin recognizes termination, retirement, disability and death as valid decrements assuming that

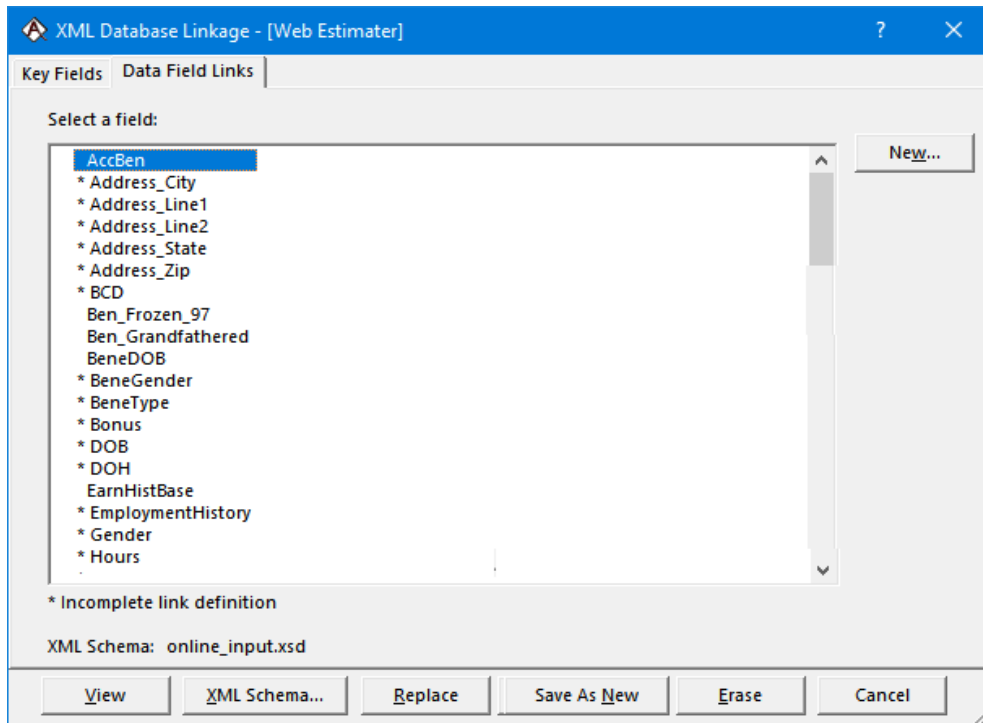
benefits have been defined for them. In addition to the XML container name for decrement type, you will also enter on this dialog the codes that will indicate what the decrement type is, where a single code is used for terminations and/or retirements. Whether or not the member has ceased employment due to termination or retirement is determined by ProAdmin based on the benefit eligibility definitions, where all “termination” benefits are ignored once the member is eligible for a “retirement” benefit.



- **Estimate Overrides** indicates potential overrides to the salary inflation assumption and the cash balance crediting rate assumptions used in the estimate calculations. Containers are only entered for these values if you choose to make the option available by checking the respective boxes.

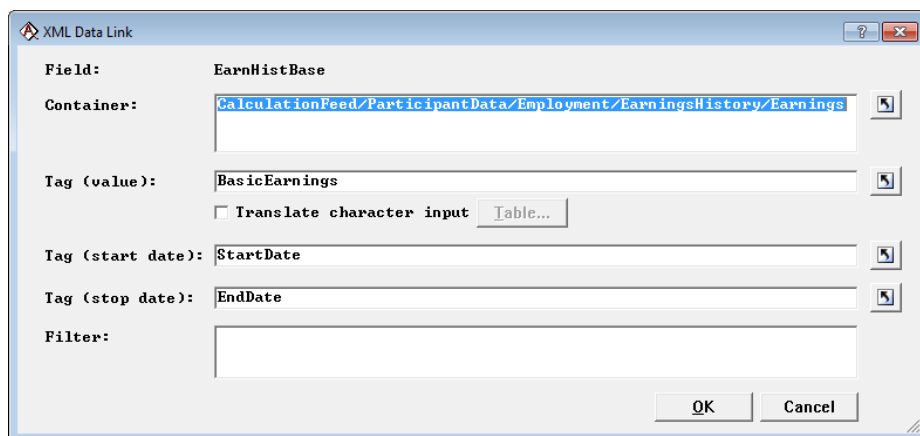


**Data Field Links** is where you associate the fields in the Data Dictionary with the XML data.



The list displayed is all fields in the Data Dictionary, where a preceding asterisk (\*) indicates that the field does not have a complete XML link definition. You access the Data Link definition for a field by clicking on the field name in the list to display the XML Data Link menu.

The XML Data Link menu is where you specify the XML container, XML tag, and the optional Filter information. This is the information that defines where this data is located in the XML document. The information that is required changes based on the definition of the field in the Data Dictionary.



- **Container** is the path to the tags that contain the required information. You select the container by clicking on the arrow button after the container field. This will display all containers as specified within the XML schema that was imported.

- **Tag** these fields are the names of the tags that surround the data. The screen shows all required tags and in parentheses what field the tag is for. You select the tag by clicking on the arrow button after the tag field. This will display all tags contained within the container that was selected above. As an example in the screen above there are three tags that need to be set for the Data Dictionary field EarnHistBase. They are for the actual earnings value, the start date, and the stop date for the time dimension of the array.
- **Filter** is used to set a condition that will limit the data that can be used to populate this field. As an example the Data Dictionary has a field called Base Pay. The XML document has a container called Earnings that contains all of the member earnings that are reported by the client. To differentiate between the different earnings amounts that are reported, the Container has an Element called EarningsType. The amounts in the Container that are for Base Pay have an EarningsType element with a value equal to BP. For this scenario the Container would specify Earnings and the Filter box would contain the following: EarningsType = 'BP'. Thus, only those items with an Earnings Type = BP from the XML input document would populate the Data Dictionary field Base Pay. This works in the same manner as the Where condition on the Data Link Dialog box under Database Linkage.

Be aware that the XML document is by nature a text document, and this makes the Filter case sensitive. If the data inside of the XML tags is lower case and you enter a Filter condition in upper case, the condition will not be met.

## ***XML Output Linkage***

ProAdmin Server returns calculation results in an XML format. The XML Output Linkage allows you to set the structure of this return document and create the fields that are referenced in the Output Definitions. Output Definitions is where you will map ProAdmin calculation results. When a calculation is processed ProAdmin will create an XML document based on the structure set up in the XML Output Linkage and containing the results from the Output Definitions.

To create a new entry you click on the New button. This will bring up the XML Output Linkage menu. The first step in the process of creating a new library entry is to import the XML schema (XSD). The schema is a document that defines the layout of the XML document (XML tag names, data types, number of occurrences, and location of the data items). To import the schema, you click on the **Import Schema** button. This will bring up the XML schema document dialog box. After either typing in the name or selecting the file, click the Open button. A message will display indicating if the import was successful. After a successful import the bottom of the screen will display the name of the XML schema that was imported and is now associated with this XML Output Linkage library entry.

While ProAdmin allows for you to create the names and the attributes of the fields that are returned it does expect that the schema will have a certain structure. This structure is that a decrement date will have one or more commencement dates and that one commencement date will have one or more payment forms. You also have the option to set up a Benefit Level within the Decrement. If you use this feature the commencement

level is within the benefit level. Here is the hierarchy with the optional benefit level being used.

Decrement – one

(One for each termination date)

Benefit – many

(One for each eligible benefit definition at that decrement)

Commencement – many

(One for each commencement date)

Payment Forms – many

(One for each eligible payment form within the benefit definition)

The XML Output Linkage menu has four tabs: Schema Structure, Plan Dependent, Input Pass Thru, and Standard.

- **Description** which can be a phrase of any length where all characters are permissible. You should include key information about the use of the XML Output Linkage in the description.
- **Decrement level** this sets the container that holds the fields that are associated with a decrement date.
- **Benefit level (within Decrement)** this optional field allows you to keep all information at a benefit level within the decrement level.
- **Commencement level (within Benefit or Decrement)** this sets the container that holds the fields that are associated with a commencement date. This container must reside within a decrement or if you have selected the benefit within the benefit level.
- **Payment form level (within Commencement)** this sets the container that holds the fields that are associated with a payment form.
- **Select an output topic**
- **Key Fields** this is where you will set the actual tags that hold the decrement date, optional benefit level detail, commencement date, and payment form details. Click on the Key Fields entry in the Select an output topic to display the Key Fields Dialog box.

- **Decrement Level Detail** contains the decrement date. To set the decrement date click the lookup button following the decrement date field. This will display a list of XML tags from the Decrement Level that you set. Click on the correct tag for this field.
- **Benefit Level Detail** contains the Code, Description, and Normal form code for the optional benefit level. The code that will be returned is either the default code or the user set override code associated with the benefit. The description is the descriptive name by which the benefit is known. The normal form code is the code associated with the payment form that has been set as the normal form for the benefit under the Payment Forms selection of the Benefit Definition. You may set any or all of these three entries if you have selected the check box on the Schema Structure tab. To set a tag for one of the fields click the check box in front of the field to activate the field and lookup button. Click on the lookup button after the field to display a list of all tag names within the benefit level set on the Schema Structure tab. Select the appropriate field from the list.
- **Commencement Level Detail** contains the commencement date and the optional error indicator field. To set the commencement date click the lookup button following the commencement date field. This will display a list of XML tags from the Commencement Level that you set. Click on the correct tag for this field. The optional error indicator is a field that will hold a value notifying you that there were errors at this commencement level. To set this field you click the check box in front of the field to activate the error indicator field. Once the field is activated you can click on the lookup button following the field. This will display a list of XML tags from the Commencement Level that you set. Click on the correct tag for this field.
- **Payment Form Detail** allows you to set up the fields that will help you determine the form of payment. These will be output with every item that is

set up in the Output Definitions that is at a payment form level. The fields that can be set up are Code, Percent, and Units. The code is the default code or the user override code set up in the Output Type Code. The percent is the value set up for the Fraction of Joint & Survivor benefit received when only the beneficiary is alive. The units is the value set up in the Certain period (years) field.

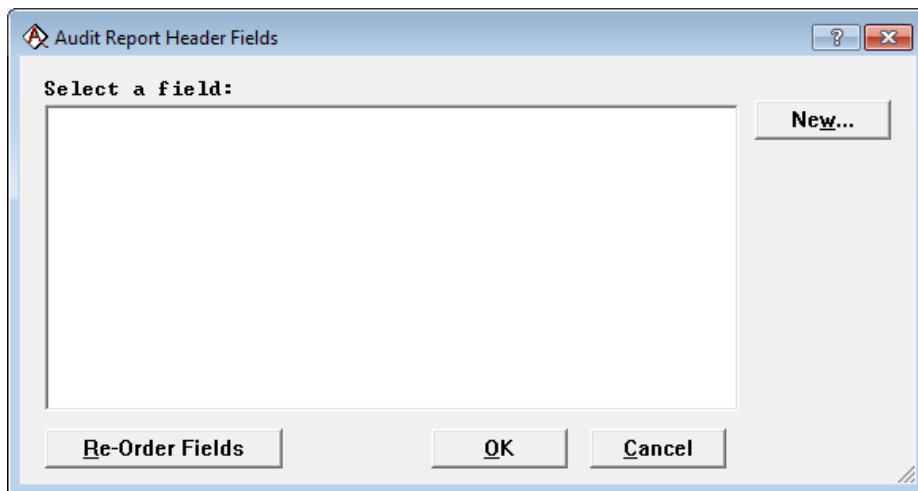
- **Audit Report** this is where you will set the tag information that indicates you want the optional text audit report produced. This report gives you details about the calculation that was processed. It contains the member data used in the calculation, the final result of the benefit formula components, accrual basis components, benefit amounts, payment amounts, and details about the Final Average Salary components. Click on the Audit Report entry in the Select an output topic list to display the Audit Report dialog box.

- Check the **Write an Audit Report if a value is passed in the file name** check box to activate the other fields on the dialog box.
- **Audit Report file name tag** click on the lookup button to set the tag that contains the file name for the audit report. The lookup button will display all of the tags from the schema associated with the XML Database Linkage set on the Input Pass Through tab. If you use the same report name for all requests the system will concatenate the output to the end of the report. If you specify a unique name it will create a new report.

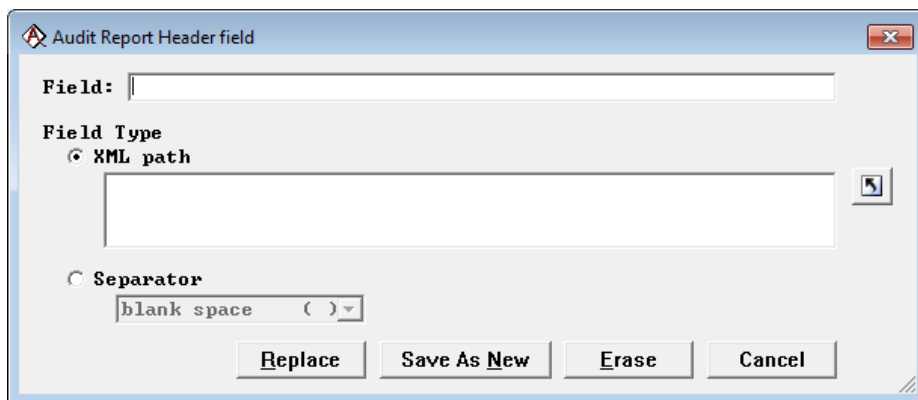
The tags should only contain the file name as the path for the file is set in the INI file for the calculation engine.

- **Person ID (tag to be inserted into the audit report)** click on the look up button to set the tag that contains the person id that will be displayed on the first line of the report. The lookup button will display all of the tags from the schema associated with the XML Database Linkage set on the Input Pass Through tab.

- **Report Header** button allows you to set up additional fields that will be written on the first line of the report. These fields will be added after the Person ID field. You can set up two types of fields to appear in the header. You can set a field from the XML input document or a separator to appear between each field. This information helps you search through the audit report if you are concatenating the results of many requests to one report. When you click on this button you will see the Audit Report Fields Header dialog box.



- The **Re-Order Fields** button allows you to set the order of the fields that will follow the Person ID field.
- To add a field you click on the **New** button. To edit an existing entry click on that field. This will display the Audit Report Header Field dialog box.

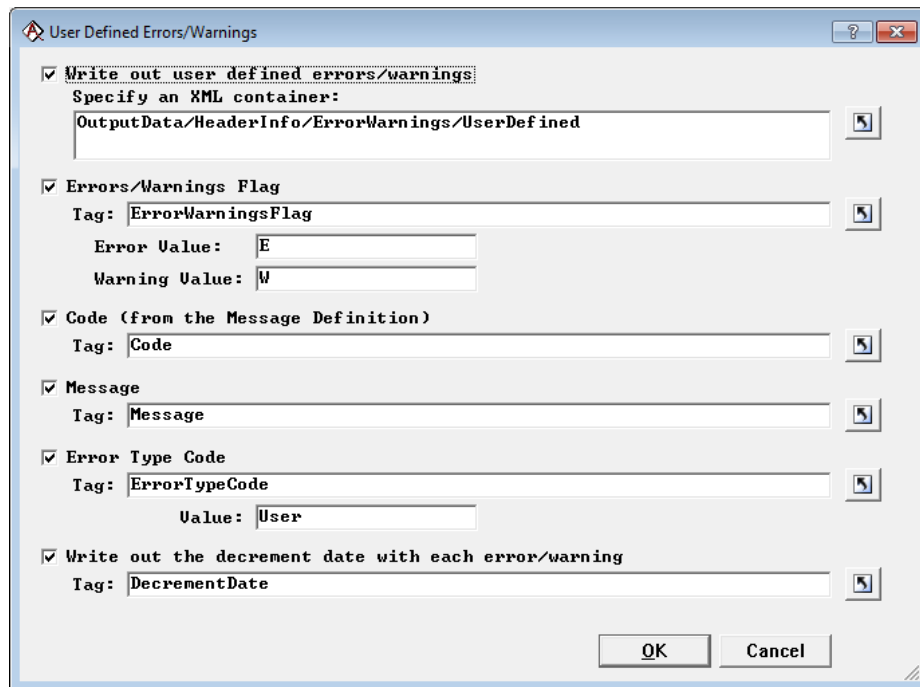


- The **Field** box is a description of the field and can be of any length where all characters are permissible. You should include key information about the use of this field.
- **XML path** are the fields from the XML input document that you want to appear in the header. To set up one of these fields click on the radio button in front of the field to activate the field. You then click on the look up button to set the tag that contains the data for the field that you want displayed on

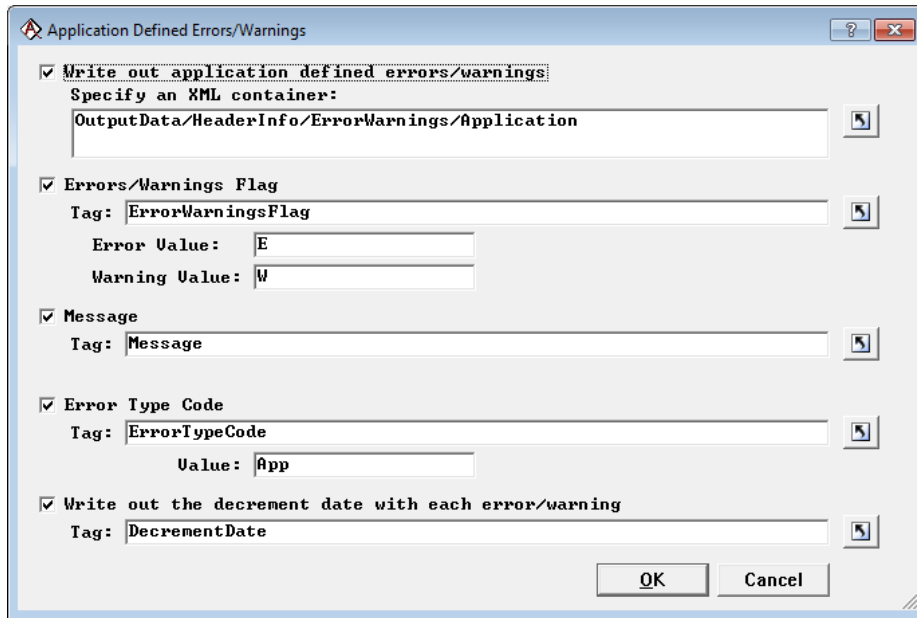


the first line of the report. The lookup button will display all of the tags from the schema associated with the XML Database Linkage set on the Input Pass Through tab.

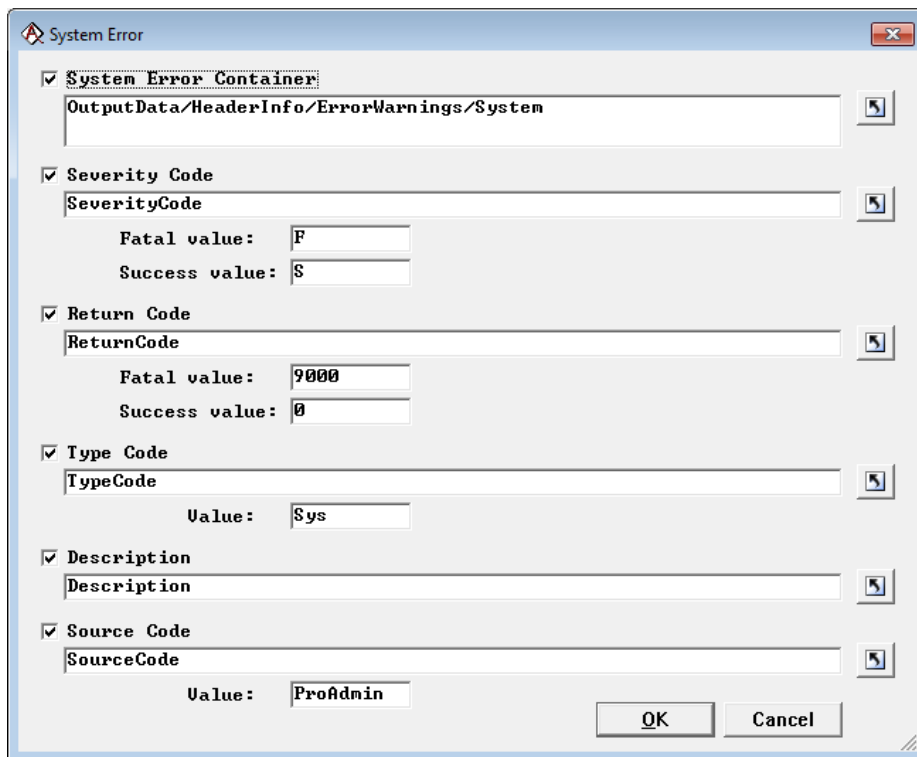
- **Separator** allows you to set up fields that will be inserted between the Person ID and any of the fields that you have set up. To set up this field click on the radio button to activate the field. Then click on the drop down menu to select the separator choices.



- **User Defined Errors/Warnings** this is where you set the tags for the data and benefit error and warning conditions that you created. Click on the User Defined Error/Warnings entry in the Select and output topic list to display the



- **Application Defined Errors/Warnings** this is where you set the tags for ProAdmin's internal error and warning conditions.



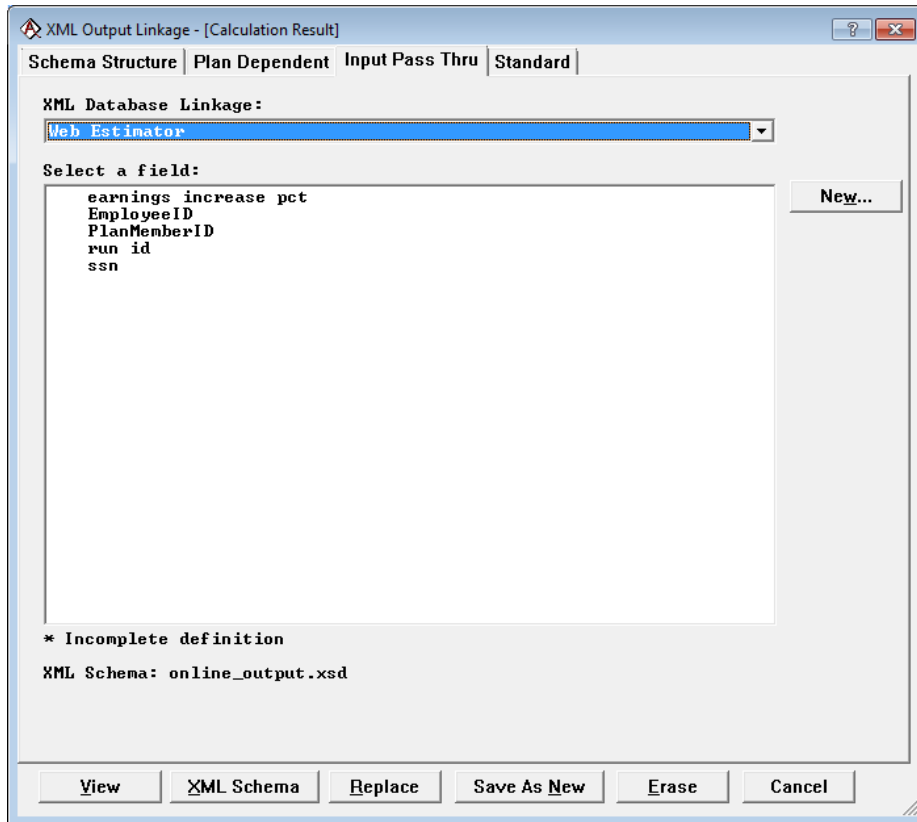
- **System Errors** this is where you set the tags for errors that indicate problems with the application.

The **Plan Dependent** tab is where you define the XML tags that are created as the result of the plan rules and formulas. This tab describes generic output containers such as

“accrued benefit” and “final average salary”. Later, in the Output Definitions command, you will link Plan Benefits and Benefit Formula Components to the generic output containers based on the provisions of the specific plan(s). The Plan Dependent tab contains:

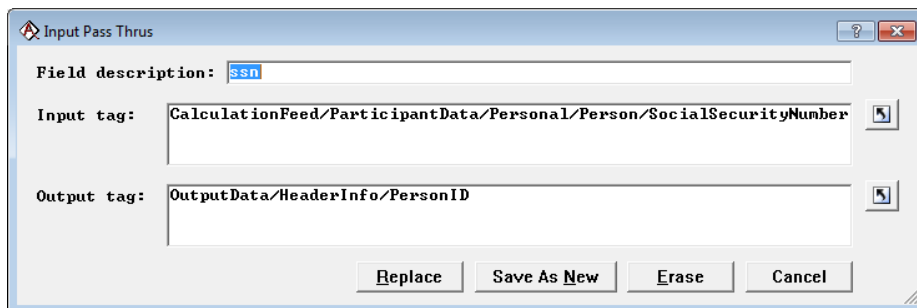
- A list of output fields, where an asterisk indicates an incomplete description. Output fields are user-defined, and are added to the list by pressing the **New** button.

- The information required for each desired output field is as follows:
  - **Field Description** can be a phrase of any length where all characters are permissible. You should include key information about the use of the field in the description as this is what you and other users will see in the Output Definitions.
  - **Varies by**, which may be <none> (e.g., SSN), decrement (e.g., final average earnings at decrement), commencement (e.g., early retirement reduction factor at commencement) or payment form (e.g., member benefit). This will set the container that the tags can be selected from.
  - **Tag**, which is the descriptor that will surround this piece of output in the XML document. To set the tag click the look up button that follows the field and select the entry that you want.
  - There are also some optional tags that can be set with the specified output and the system will automatically populate them when it creates the return document. These optional output items are generally applicable only for data that varies by payment forms. They include the deferred commencement date, temporary stop date, beneficiary amount, post-Social Security level income amounts, and information about lump sum equivalent and relative value.

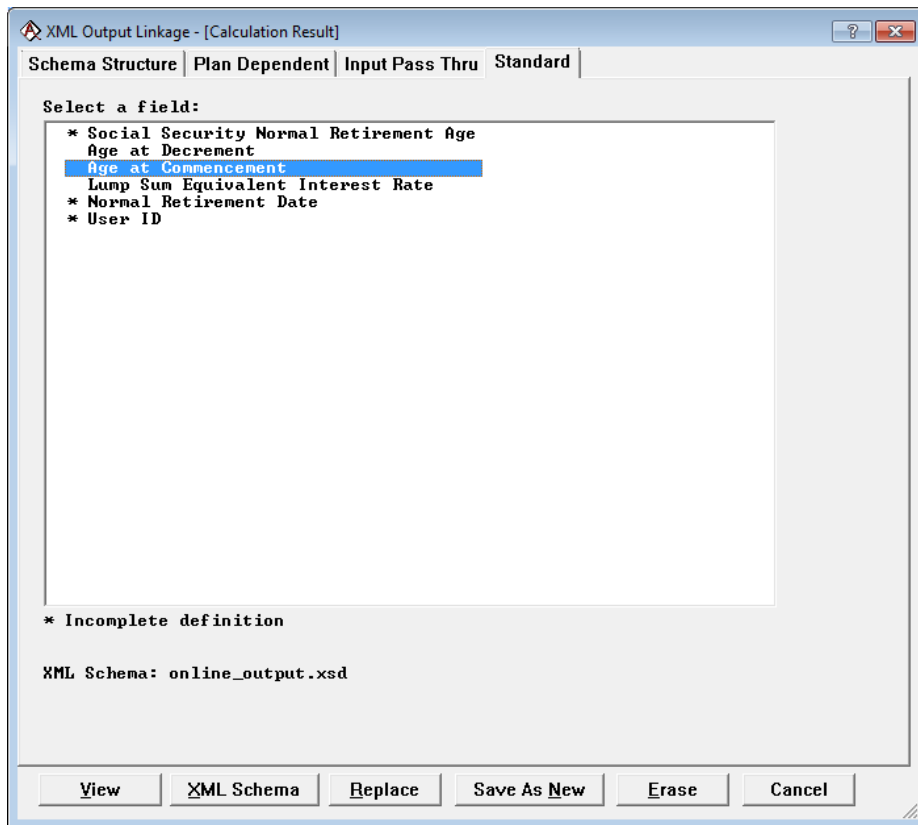


The **Input Pass Thru** tab is where you set up the fields that you want returned from the input XML document. These fields are items that are not created as a result of a ProAdmin calculation. Potential examples of these fields are a system generated run id, the system plan key fields, and member Social Security Number. There is no limit to the number of fields that can be read off of the input document and passed back on the output document. The tab contains:

- **XML Database Linkage** is where you select the library entry that defines the input document. This entry determines which fields you can select to pass through to the output document.
- A list of output fields, where an asterisk indicates an incomplete description. Output fields are user-defined, and are added to the list by pressing the **New** button.



- **Field Description** can be a phrase of any length where all characters are permissible. You should include information about the field in the description.
- **Input tag** you click on the look up button after the field to display all of the fields on the XML Database Linkage entry that you selected.
- **Output tag** you click on the look up button after the field to display all of the fields on the schema document that you imported.



The **Standard** tab allows you to set up XML output tags for the standard calculations performed by the system. There are currently four choices on the standard output tab: Social Security normal retirement age, age at decrement, age at commencement, and lump sum equivalent interest rate. These fields are optional and do not need to have XML information setup for them. If you do choose to have XML output of one or more of these fields, click on the desired output item and enter the desired output tag.

# Appendix F: Output Definitions

**Output Definitions** allow you to set up the output you want to collect from an estimate or final calculation. Those using the Server application type must first create an XML output linkage as discussed in [Appendix E: XML Linkages](#). Those using the stand-alone Desktop application type may go directly to the next step as described below.

Click the **Add** button to include a new entry to the **Output Fields**. Select the appropriate type for the new output item: Benefit Detail, Date/Age/Service, Input Pass Thru, or Standard Result.

- **Benefit Detail** may contain calculation results from across all benefit definitions, from selected benefit definitions, from a type of benefit (ie: retirement/termination), or from individual components used in the benefit formula. This type is useful for obtaining benefit calculation details required for fulfillment, such as, individual salaries used to calculate the final average salary. Benefit Formula Components and Accrual Basis Components must be either part of an executed benefit formula or in the list of components under the Fulfillment Components topic of the Plan Definition for them to be included as an output item.
- **Date/Age/Service** is a date, an age, or a service amount at either decrement or after meeting a specified eligibility condition (ie: Normal Retirement Date, Normal Retirement Age, Vesting Services, etc...).
- **Input Pass Thru** is an input item, selected from the Data Dictionary, which does not change as a result of a calculation (ie: first name, last name, date of birth, social security number, etc.). This type of Output Definition is not available to the Server application type.

- **Standard Result** is only available for the Desktop (Access) application type. There are currently six standard results: **Social Security normal retirement age**, **Age at decrement**, **Age at commencement**, **Lump sum equivalent interest rate**, **Normal Retirement Date**, and **User ID**. Lump sum equivalent interest rate is the rate used to determine relative value.

Each new **Output Definition** item requires a Description and an Output Field Name. **Description** is any descriptive name by which this entry will be known. **Output Field Name** is the unique field name to identify this item in the output. Output Field Names cannot contain spaces or dashes (-).

# Appendix G: Fulfillment Tool

---

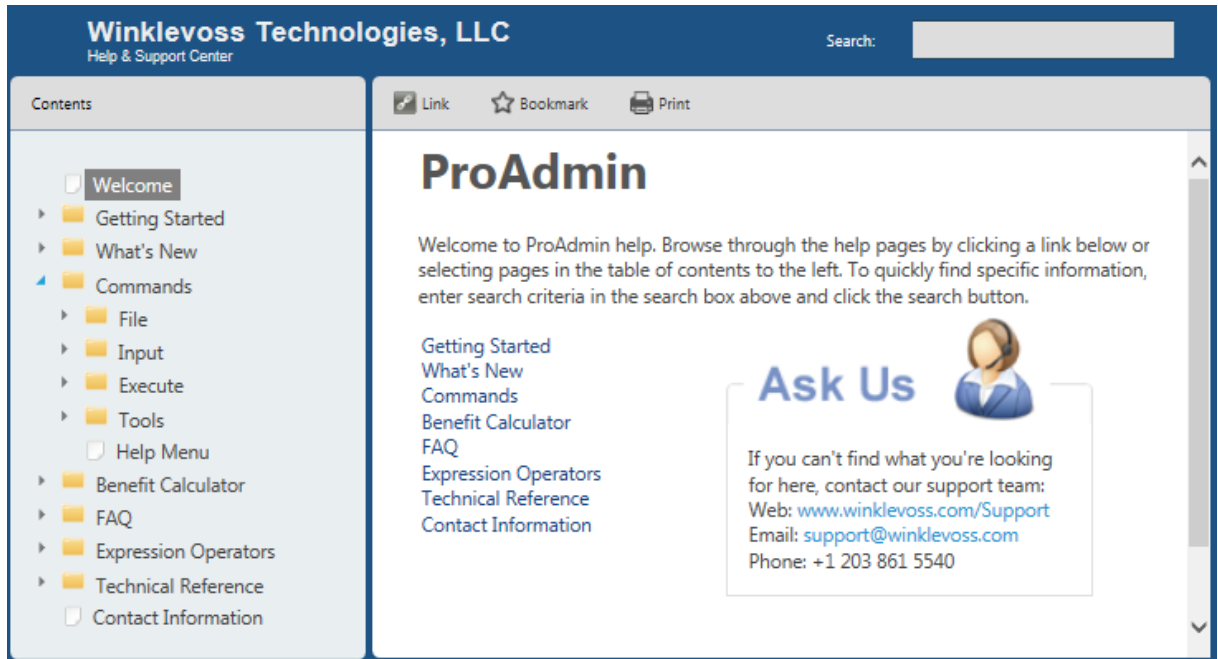
ProAdmin's **Fulfillment Tool**, in conjunction with **Output Definitions** and Word templates, can be used to create annual statements for plan participants as well as letters and forms (i.e. fulfillment) for terminating and retiring members. Below are the basic steps in this process.

- 1) Create/design the Word template leaving space for individual member items. Different templates will be used for different purposes; annual statements and letters to terminating non-vested members are examples of separate templates. At this point in the process there is no need to insert Word merge fields.
- 2) Inspect the template for required fields such as date of birth, member name, accrued benefit, vested percentage, etc.
- 3) Design the Output Definition so required fields are written to the output. Output Definitions have several different types of **Output Fields**. You can make use of the **Benefit Detail** type for computed amounts, the **Date/Age/Service** type for specific dates, ages and service amounts, the **Input Pass Thru** type for items such as member name, location name, etc. and the **Standard Result** type for items that ProAdmin automatically computes such as Normal Retirement Date.
- 4) Create a **Package Code** as one of the required Output Definition fields. This code is used to select the Word template and is determined within ProAdmin. It can be an Input Pass Thru, a computed value or anything that can be used to specify which Word template is to be used.
- 5) Run a batch calculation or an individual calculation through **ProAdmin Desktop**. If an individual calculation is run, the Output Definition results must be written to an Access database. Running a batch calculation automatically writes the output an Access database. This database is different from the database which houses member information that is used for ProAdmin benefit calculations.
- 6) If needed, complete any queries within the Access database for additional computed values.
- 7) Launch the Fulfillment Tool and open the database within the Tool. Complete the Client Information dialog box.
- 8) If needed, complete the Query Mapping.
- 9) Open the Word template which is to be selected for the calculation that was run above.
- 10) Insert Word merge fields in the template(s) to capture the items written to the output and complete the Field Mapping within the Fulfillment Tool so the items in the output are written to the correct Word merge field. Remove any unnecessary spaces on the Word document.
- 11) Complete the Package Mapping.
- 12) Preview the completed document.



# Appendix H: ProAdmin Help

In ProAdmin, you can get additional information by selecting **Help Topics** from the Help menu.



- **Getting Started** provides access to this document.
- **What's New** provides a list of what's new in the latest ProAdmin release.
- **Command Reference** explains the questions and options on every screen.
- **Benefit Calculator** explains Calculator mode procedures.
- **Frequently Asked Questions** contains answers to some of the most commonly asked questions.
- **Expression Operators** explains the syntax and purpose of each ProAdmin operator used in expressions, such as #MAX or #YEARDIF.
- **Technical Reference** documents certain ProAdmin calculations, such as Primary Insurance Amount.