



ProVal API Users' Guide

Last updated: 4/10/2026



This document was prepared to assist users of Winklevoss Technologies' ProVal 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, LLC therefore assumes no responsibility and shall have no liability arising from the supply or use of this document or the material contained herein.

Copyright © 2026 Winklevoss Technologies, LLC
Printed in the United States of America. All rights reserved.
Unauthorized reproduction is strictly prohibited.

Windows®, Visual Basic for Applications® and Access® are registered trademarks of Microsoft Corporation.

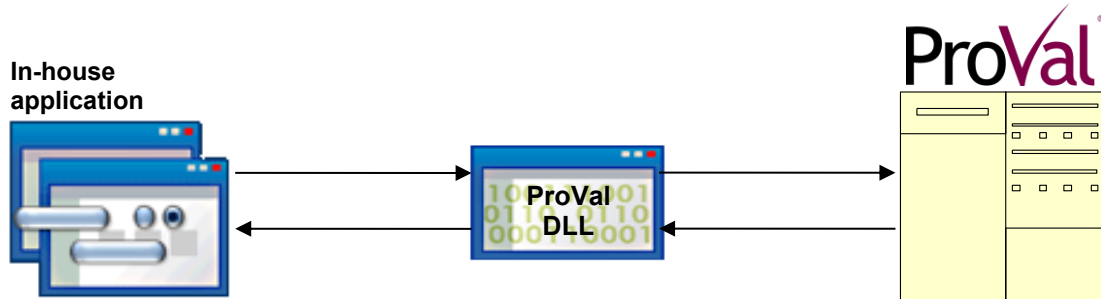
Contents

OVERVIEW.....	5
Licensing Requirements	5
API SETUP (32-BIT)	6
Registration of ProVal API	6
Calling the ProVal API	8
API SETUP (64-BIT)	9
Registration of ProVal API	9
Calling the ProVal API	11
Disposing of the PVAPI64 Instance	12
API FUNCTIONS.....	13
System Related	13
FileClose	13
FileOpen	14
ListLibraries.....	17
QueryLibrary	18
Database Related	20
CreateNewDatabase	20
EraseDatabase.....	21
GetRecordLayout	22
ImportCSVData.....	24
ListDatabases	26
MergeUpdate	27
PrintData	29
QueryDatabase	30
ViewDataDictionary	31
RunDataScript	32
ImportDataCorrections (New 3.22).....	34
Plan Definition	36
GetPlanConstants.....	36
SetPlanConstants	37
Valuation Assumptions.....	39
GetValAssum	39
GetValAssumDoc.....	40
SetValAssum	41
GetTable	43
SetTable	44
Valuation	45
GetValInputs	45
SetValInputs	47
RunVal.....	48
GetValResults	51
GetValResultsDoc.....	54
Valuation Set.....	55
GetValSetInputs.....	55
SetValSetInputs.....	55
GetValSetEventDoc	56
RunValSet	57
GetValSetExhibitResults	57

Core Projection	59
RunCore.....	59
GetCoreResults	61
Asset & Funding Policy	63
GetAFP	63
GetAFPDdoc	63
SetAFP.....	63
Deterministic Forecast	65
RunDetFore	65
Report Writer	67
ExportValSetExhibits	67
GetRWDataSets	69
GetRWReconOfAssets	70
GetRWStmntOfAssets	71
SetRWReconOfAssets	72
SetRWStmntOfAssets	73
Batch (New 3.22).....	74
BatchGetJobs.....	74
BatchGetWorkerStatus.....	75
BatchRunVal.....	76
BatchRunValSet	77
BatchRunCore.....	77
BatchRunDetFore	78
Administration Factors	78
GetRUNADFACTARGS	79
RunADFACT	79
ADDITIONAL METHODS	82
System Related	82
eng.ChangeWorkspace.....	82
APPENDIX A: ERRORS	83

Overview

ProVal API is designed to expose certain functions in ProVal to other programs via an apartment threaded ActiveX DLL. The API facilitates seamless integration of ProVal within existing in-house applications allowing for greater automation of repetitive functions.



ProVal API may be configured to use a local or a network installation of ProVal. This guide assumes that ProVal is already set up correctly – for instructions on setting up ProVal, refer to the ProVal installation guide (readme.doc) in the ProVal installation folder.

Licensing Requirements

A ProVal license server (for details see the ProVal License Server Guide in the ProVal installation guide) or site license must be available for the API to function. Additionally, at least a Pension-Only or OPEB license must be available, depending on the mode used within the API (as expected, a Full license serves either case).

API Setup (32-bit)

Registration of ProVal API

NOTE: The PVAPI.dll (VB6 version) is sunsetted in ProVal 3.23, and is being replaced by PVAPI32.dll. The former dll will continue to work with existing functionality, but only the newer PVAPI32.dll will be maintained going forward.

In order to use the ProVal API, you must:

- Register the PVAPI32.dll
- Register the ProVal application for usage by the ProVal API

Any registration statements must be executed with "administrator rights" on the computer. Below are examples.

Navigate to the Framework folder by entering the following:

```
CD "C:\Windows\Microsoft.NET\Framework\v4.0.30319\"
```

Register PVAPI32.DLL using the REGASM command. Examples:

```
RegAsm.exe /codebase "C:\Program Files (x86)\WinTech\ProVal\PVAPI32.dll" /tlb
```

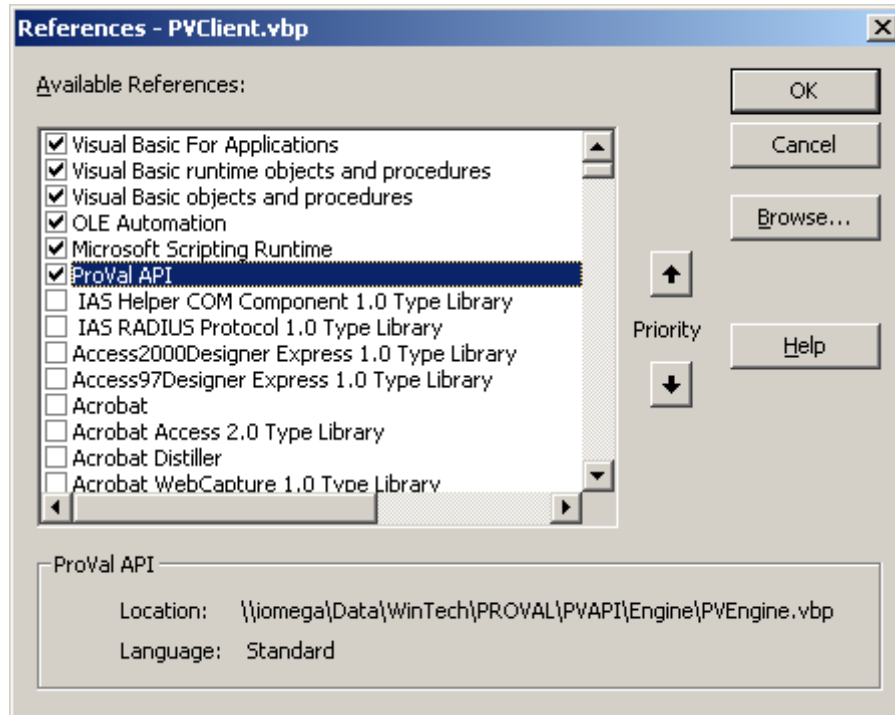
```
RegAsm.exe /codebase "N:\Apps\PVAPI32.dll" /tlb
```

Register the ProVal application for usage by the ProVal API. Specify the Proval.exe and Proval.ini to be used by the ProVal API. The folder containing the Proval.ini file will be considered the "user" directory and must allow "write" access. Examples:

```
"C:\Program Files (x86)\WinTech\ProVal\Proval.exe" "C:\ProvalUser\Proval.ini" /regserver
```

```
"N:\Apps\Proval.exe" "C:\ProvalUser\Proval.ini" /regserver
```

DEVELOPERS ONLY - From the list of available reference libraries select "ProVal API", as shown below:



ProVal API is now ready to be called from other applications.

Calling the ProVal API

This guide uses Visual Basic for Applications to provide examples; the procedure using other languages is similar.

The API contains one main class - PVAPI. Every function exposed by the API is called in the following manner:

- **ProVal functions are exposed using only one PVAPI method, PVCall, which accepts two arguments:**
 - **[0] The ProVal function name (case-insensitive String).** Examples of such function names would be FILEOPEN, FILECLOSE, etc.
 - **[1] Parameter List (Variant).** NOTE: If parameters are required, all arguments must be supplied for the functions to operate correctly.
- **PVCall always returns a Variant Array.**

A generic Visual Basic for Applications code snippet may resemble the following:

```
Dim eng as New PVAPI
Dim RetVal as Variant
Dim xArgList()
xArgList = Array(param1,param2,1,0)
RetVal = eng.PVCall("ProValFuncName", xArgList)
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
```

Of course, it is reasonable to assume that the eng object defined above will typically be parked in a module and available publicly in production applications, so that different instances are not invoked for each function call.

NOTE:

- 1.** Database filenames are accepted with or without the .SF extension, when passed as arguments in ProVal API functions.
- 2.** All database file ties are released after database related functions are executed i.e. there is no concept of a "current database" within the API.

API Setup (64-bit)

Registration of ProVal API

In order to use the ProVal API, you must:

- Register the PVAPI64.dll
- Register the ProVal application for usage by the ProVal API

Any registration statements must be executed with "administrator rights" on the computer. Below are examples.

Navigate to the Framework64 folder by entering the following:

```
CD "C:\Windows\Microsoft.NET\Framework64\v4.0.30319\"
```

Register PVAPI64.DLL using the REGASM command. Examples:

```
RegAsm.exe /codebase "C:\Program Files (x86)\WinTech\ProVal\PVAPI64.dll" /tlb
```

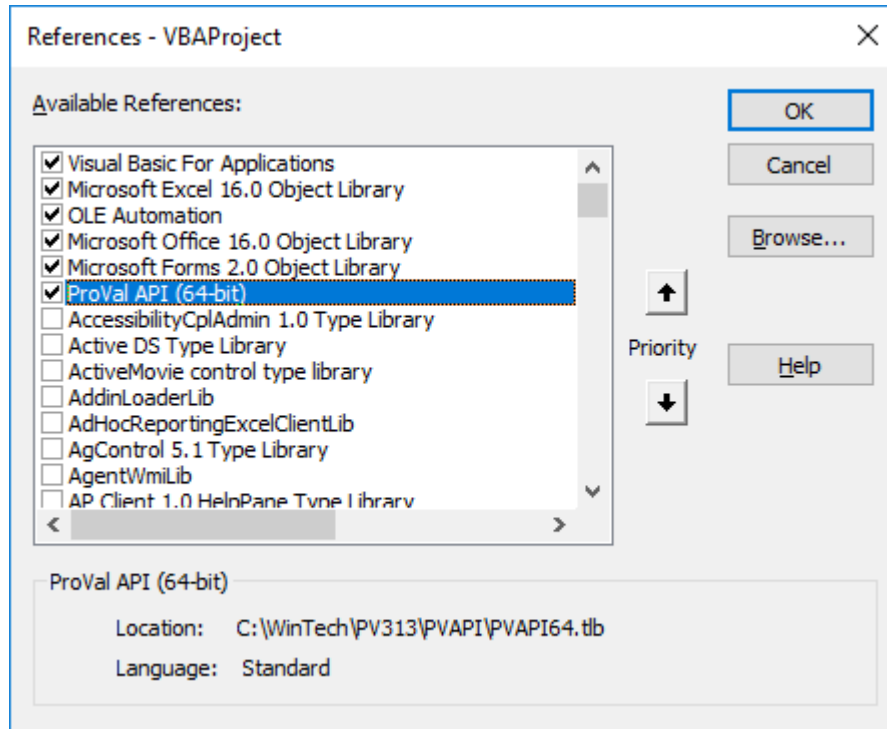
```
RegAsm.exe /codebase "N:\Apps\PVAPI64.dll" /tlb
```

Register the ProVal application for usage by the ProVal API. Specify the Proval.exe and Proval.ini to be used by the ProVal API. The folder containing the Proval.ini file will be considered the "user" directory and must allow "write" access. Examples:

```
"C:\Program Files (x86)\WinTech\ProVal\Proval.exe" "C:\ProvalUser\Proval.ini" /regserver
```

```
"N:\Apps\Proval.exe" "C:\ProvalUser\Proval.ini" /regserver
```

- **DEVELOPERS ONLY** - From the list of available reference libraries select "ProVal API", as shown below:



- ProVal API is now ready to be called from other applications.

Calling the ProVal API

The API contains one main class – PVAPI64. Every function exposed by the API is called in the following manner:

- ProVal functions are exposed using only one PVAPI64 method, PVCall, which accepts two arguments:
 - [0] The ProVal function name (case-insensitive String). Examples of such function names would be FILEOPEN, FILECLOSE, etc.
 - [1] Parameter List (Variant). NOTE: If parameters are required, all arguments must be supplied for the functions to operate correctly.
- PVCall always returns a Variant Array.

A generic Visual Basic for Applications code snippet may resemble the following:

```
Dim eng as New PVAPI64.PVAPI64
```

```
' note the extra .PVAPI64 for the 64-bit version
```

```
Sub FuncName()
```

```
Dim RetVal as Variant
```

```
Dim xArgList()
```

```
xArgList = Array(param1,param2,1,0)
```

```
Set eng = New PVAPI64.PVAPI64
```

```
' note the above line should be before the first use of eng
```

```
RetVal = eng.PVCall("ProValFuncName", xArgList)
```

```
Dim ErrCode as Long, ErrDesc as String
```

```
ErrCode = RetVal(0)
```

```
ErrDesc = RetVal(1)
```

```
End Sub
```

Of course, it is reasonable to assume that the eng object defined above will typically be parked in a module and available publicly in production applications, so that different instances are not invoked for each function call.

NOTE:

1. Database filenames are accepted with or without the .SF extension, when passed as arguments in ProVal API functions.
2. All database file ties are released after database related functions are executed i.e. there is no concept of a "current database" within the API.

Disposing of the PVAPI64 Instance

Disposing of the ActiveX object by calling the Dispose method.

A generic Visual Basic for Applications code snippet may resemble the following:

eng.Dispose

Set eng = Nothing

Note, closing your program will automatically dispose of the object as well.

API Functions

System Related

FileClose

Description:

Closes the current client folder.

Arguments

<None>

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string)

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
RetVal = eng.PVCall("FILECLOSE")
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
```

FileOpen

Description:

Opens a ProVal client folder. Closes current one if one currently open, regardless of whether the new file was successfully opened.

Argument (Variant Array)	
[0]	UpdateFolderIfNecessary (long: 1 = yes, 0 =No)
[1]	ClientFolder – Path to folder (string)
[2]	ComputationMode (long) Code that indicates ProVal's computation mode: <ol style="list-style-type: none">1. = U.S. Qualified Pension,2. = Retiree Medical,3. = Public Pension,4. = Nonqualified,5. = Canadian Pension,6. = German Pension,7. = United Kingdom Pension
[3]	OpenMode (optional; string) – If specified, a string that describes the manner in which to open the client. Valid values are: 'MUSTWRITE' open client normally only; do not open as "read only" if client is in use (default) 'READWRITE' try to open client normally first, but then open as "read only" if client is in use 'READONLY' open client as "read only"

Return Value (Variant Array)	
[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 31 = Client Folder doesn't exist 32 = Client Folder is an invalid path 33 = Client Folder isn't a valid ProVal client folder 34 = Client Folder in use 35 = Client Folder needed updating, but UpdateFolderIfNecessary is False 36 = Unknown error opening client 37 = Client Folder has been updated by a newer version of ProVal
[2]	ProVal version and date (only returned for errors 35 or 37; string) Example: "3.19 7/1/2023". Date formatted per the machine's Regional Settings, e.g., MM/DD/YYYY, DD/MM/YYYY, etc. (new in 3.19)

Visual Basic for Applications Code Snippet

```
Dim eng as New PVAPI ' This only has to be done once in a module

Sub FileOpen()
Dim RetVal As Variant
Dim FilArr()
'      Open Client in U.S. Qualified Mode, with READWRITE access,
'      and update if necessary
FilArr = Array(1, "C:\MyClient", 1, "READWRITE")
RetVal = eng.PVCall("FILEOPEN", FilArr)
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
End Sub
```


ListLibraries

Description:

Lists all available ProVal libraries.

Arguments

<None>

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string)
[2]	List of short library names - used as argument in QueryLibrary (string array)
[3]	List of descriptive library names (string array)

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
RetVal = eng.PVCall("LISTLIBRARIES")
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
' Check for errors before continuing
If RetVal(0) = 0 Then
    ' Declare arrays to contain long and short library names
    Dim ShName(), DescName() As String
    ShName = RetVal(2)
    DescName = RetVal(3)
    ' View each entry using a messagebox
    For i = LBound(ShName) To UBound(ShName)
        MsgBox ts.WriteLine ShName(i) + " | " + DescName(i)
    Next
End If
```

QueryLibrary

Description:

Returns information about all the entries in a library. Use ListLibraries for a list of short and full library names.

Arguments

Short Library Name (e.g., "RECLAYOUT", as returned by ListLibraries).

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string)
[2]	List of components/fieldnames (string array)
[3]	List of descriptions (string array)
[4]	List of modification timestamps (string array)
[5]	ID# of element

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
RetVal = eng.PVCall("QUERYLIBRARY", "RECLAYOUT")
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
' Check for errors before harvesting data
If RetVal(0) = 0 Then
    Dim FldArr(), DescArr(), TimeStampArr(), IDNumArr() As Variant
    FldArr = RetVal(2)
    DescArr = RetVal(3)
    TimeStampArr = RetVal(4)
    IDNumArr = RetVal(5)
    ' Parse the entries
    For i = LBound(FldArr) To UBound(FldArr)
        MsgBox ts.WriteLine FldArr(i) + " | " + DescArr(i) + _
            " | " + TimeStampArr(i) + " | " + IDNumArr(i)
    Next
End If
```

Sample Output (separated by |)

```
FAEBEN | 2%*SVC*FAE | 10/24/1994 13:41:57 | 4
INTCASHBAL | Integrated, age graded, cash balance | 10/24/1994
14:50:15 | 3
GROSSBEN | Gross Benefit | 10/24/1994 13:42:10 | 2
PIA | Social Security Offset | 10/24/1994 13:42:51 | 1
PROJBEN1 | Project & Prorate benefit, piece 1 | 10/24/1994
13:43:02 | 5
PROJBEN2 | Project & Prorate Benefit, piece 2 | 10/24/1994
13:43:13 | 6
```

Database Related

CreateNewDatabase

Description:

Create a new database.

Arguments

	Name of database to create (string)
--	-------------------------------------

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string)

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
RetVal = eng.PVCall("CREATENEWDATABASE", "NEWDATAFILE")
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
```

EraseDatabase

Description:

Erases an existing database.

Arguments

	Name of database to erase (string)
--	------------------------------------

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string)

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
RetVal = eng.PVCall("ERASEDATABASE", "NEWDATAFILE")
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
```

GetRecordLayout

Description:

Returns a record layout's definition. For names of record layouts, use QueryLibrary with "RECLAYOUT" argument.

Arguments

Name of record layout (string)

Return Value (Variant Array)

- | | |
|------|---|
| [0] | Error code (long) |
| [1] | Error description (string)
Notable Errors (also see Appendix A):
70 = Record layout does not exist |
| [2] | Record format (integer)
1 = fixed width, 2 = delimited (CSV), 3 = Excel |
| [3] | Delimiter type (integer)
1=comma, 2=semicolon, 3=tab, 4=space, 5=other |
| [4] | Delimiter "other" character (string) |
| [5] | List of field names (string array) |
| [6] | List of starting columns for fixed width files (integer array) |
| [7] | List of ending columns for fixed width files (integer array) |
| [8] | List of numeric field multipliers (double array) |
| [9] | List of date formats (string array)
Example: "MM/DD/YYYY". |
| [10] | List of character labels found in the import file (array of arrays), an array for each field (string array)
Examples: "Act" "Ret" "Term" or "Male" "Female" |
| [11] | List of field codes (array of arrays); an array for each field (long array)
Examples (corresponding to [9] above): (1 2 3) or (1 2) |
| [12] | List of century breakpoints for two digit years in date fields (array of arrays); an array for each field (integer vector)
[0] code indicating type of century breakpoint:
1=no breakpoint, all years are 1900
2=years greater than breakpoint are in 1800s, otherwise 1900s
3=years less than breakpoint are in 2000s, otherwise 1900s
[1] breakpoint value |
| [13] | List of file positions for delimited or Excel files (integer array) |

[14]	List of case matching flags (integer array). 1= yes, 0=no
-------------	---

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant  
RetVal = eng.PVCall("GETRECORDLAYOUT", "Layout for mydata.xls")
```

ImportCSVData

Description:

Imports data from a CSV file.

Argument (Variant Array)	
[0]	Name of database to import into (string)
[1]	CSVFile Name of CSV File (string)
[2]	File Handling Options (string): A = Append R = Replace Q = Quit
[3]	Selection Expression (string)
[4]	Log File Name (string) Default log file name is PROVALAPI1.TMP, PROVALAPI2.TMP, etc.
[5]	Error Code Overrides - Error codes separated by spaces under which the user wants processing to continue instead of aborting (string). Alternatively, an empty string must be used if the intention is to halt processing on all errors. NOTE: Errors cannot be overridden unless specified.
[6]	Record Layout - Name of Record Layout to be used during import (optional). If a record layout is not provided, then the first row of the CSV data file must contain the appropriate field names.

Return Value (Variant Array)	
[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 41 = Invalid Record Layout specification 42 = CSV file does not exist 43 = Import validation error 44 = Data Import is not allowed into encrypted database files 301 = Problems with Import Schema (can override and writes to message log) 302 = Error processing import file (can override and writes to message log) 303 = Ambiguous coded field values that are similar to existing values (can override and writes to message log)

[2]

311 = Running this command will erase the results of other objects
(can override and writes to message log)

Log File Name (string)

NOTE: If log file is not written out, this contains an empty string.

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
X = Array("DATAFILE", "C:\IMPDATA.CSV", "R", "", "", "", "")
RetVal = eng.PVCall("IMPORTCSVDATA", X)
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
```

ListDatabases

Description:

Returns a list of database in currently open client folder.

Arguments

<None>

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string)
[2]	List of database names (string array)

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
RetVal = eng.PVCall("LISTDATABASES")
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
' Check for errors
If RetVal(0) = 0 Then
    Dim DBListArr() As String
    DBListArr = RetVal(2)
    ' Parse the list
    For i = LBound(DBListArr) To UBound(DBListArr)
        MsgBox DBListArr(i)
    Next
End If
```

MergeUpdate

Description:

Performs a Merge/Update

NOTE: Assumes target fields = source fields.

Argument (Variant Array)	
[0]	CurrentDatabase Name of main database (string)
[1]	UpdateDatabase Name of Update database (string)
[2]	KeyFields Name of field(s) used as the key in matching the two databases separated by spaces (string)
[3]	SourceFields Name of field(s) to update separated by spaces ('ALL' to indicate all fields) (string)
[4]	RecordsInBoth Determines what to update when record found in both databases (long: 1=update missing and non-missing values, 2=missing values only, 3=do not update existing records, 4=update missing and non-missing values (allowing values to be overwritten with missing values))
[5]	RecordsInUpdate Determines what to do if record found in UpdateDatabase only (long: 1=append, 2=ignore)
[6]	RecordsInCurrent Determines what to do if record in CurrentDatabase only (long: 1=keep existing records, 2=delete existing records)
[7]	CreateMergeStatus - Determines whether the MergeStatus flag should be created or not (long: 1=yes,0=no)
[8]	SelectionExpression (string)
[9]	MatchCaseKeyFields - Determines whether key fields are case sensitive (long: 1=yes,0=no)
[10]	Log File Name (string) Default log file name is PROVALAPI1.TMP, PROVALAPI2.TMP, etc.
[11]	Error Code Overrides - Error codes separated by spaces under which the user wants processing to continue instead of aborting (string). Alternatively, an empty string must be used if the intention is to halt processing on all errors. NOTE: Errors cannot be overridden unless specified.

Return Value (Variant Array)	
[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 45 = Invalid Merge/Update key field

[2]	46 = Duplicate keys found in the update file
	201 = Duplicate keys found in the database file that is being updated (can override)
	311 = Running this command will erase the results of other objects (can override and writes to message log)
	Log File Name (string) NOTE: If log file is not written out, this contains an empty string.

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
X = Array("SRCDB", "UPDTDB", "KEYFLD1 KEYFLD2", "FLD1 FLD2 FLD3", 1, 1, 1, 1, 1, "")
RetVal = eng.PVCall("MERGEUPDATE", X)
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
```

PrintData

Description:

Exports database information into a specified file.

NOTE: There is a difference in behavior while selecting column headings between ProVal and the API. Within the API, the actual field name is used as the column heading, while within ProVal, the custom column title (specified within the Data Dictionary Field Attributes) is used as the heading.

Argument (Variant Array)

[0]	Name of database to print (string)
[1]	SourceFields (Fields to be included separated by spaces, use 'ALL' for All) (string)
[2]	FileOut – name of output file with .TXT or .CSV extension (string)
[3]	Selection Expression (string)
[4]	File Handling Options (string): A = Append R = Replace Q = Quit <i>Meaningful when file already contains data</i>

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string)

Visual Basic for Applications Code Snippet

```
Dim X() As Variant
X = Array("DATABASE", "FLD1 FLD2", "OUT.TXT", "", "R")
RetVal = eng.PVCall("PRINTDATA", X)
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
```

QueryDatabase

Description:

Queries a ProVal database and returns the number of records, number of fields and the fieldnames in the database.

Arguments

	Name of database to query (string)
--	------------------------------------

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string)
[2]	Number of fields (long)
[3]	Number of records (long)
[4]	List of fieldnames (string array)

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
RetVal = eng.PVCall("QUERYDATABASE", "ABCDemo2000")
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
If RetVal(0) = 0 Then
    Dim NumFields, NumRecs As Long, FldArr() As String
    NumFields = RetVal(2)
    NumRecs = RetVal(3)
    FldArr = RetVal(4)
    ' Parse the field array
    For i = LBound(FldArr) To UBound(FldArr)
        MsgBox FldArr(i)
    Next
End If
```

ViewDataDictionary

Description:

Returns a matrix containing information regarding the data dictionary.

Arguments

<None>

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string)
[2]	List of field names (string array)
[3]	List of descriptions (string array)
[4]	List of field types (long array) 1 = numeric 2 = character 3 = coded field 4 = date 5 = Social Security No.
[5]	List of display widths for character fields (long array)
[6]	List of formatting styles for numeric fields (string array) e.g. 999,999.99.
[7]	List of column titles (string array)
[8]	List of field codes (array of arrays); an array for each field (long array)
[9]	List of field labels (array of arrays); an array for each field (string array)

RunDataScript

Description:

Executes a previously defined Data Script.

Argument (Variant Array)	
[0]	Name of Data Script to execute (string)
[1]	Log file name (string) Default log file name is PROVALAPI1.TMP, PROVALAPI2.TMP, etc.
[2]	Error Code Overrides - Error codes separated by spaces under which the user wants processing to continue instead of aborting (string). Alternatively, an empty string must be used if the intention is to halt processing on all errors. NOTE: Errors cannot be overridden unless specified. 214 - Screen data errors will be Replaced or Removed if no new errors 215 - Screen data errors will be Merged 216 - Screen data errors will be Canceled and not logged 217 - Screen data does not have new errors but there are existing errors in the error log of the database. Would you like to remove (code 214) or keep (code 216)?
[3]	Run Type - "" or 0 = Run normally, 1 = Validate

Return Value (Variant Array)	
[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 84 = Data Script library entry does not exist 87 = Data Script contains missing or incomplete inputs and cannot be run 88 = Data Script aborted while running 201 = Duplicate keys found in the database file that is being updated (can override) 302 = Error processing import file (can override) 303 = Ambiguous coded field values that are similar to existing values (can override)
[2]	Log File Name (string) NOTE: If log file is not written out, this contains an empty string.
[3]	Data Script Output (variant) Summary of Data Script run. One row for each step in the script.

	[;0] 0=success, 1=error, MV=not executed
	[;1] Name of the Data Script step
	[;2] Error message (when the first column equals 1=error)
	[;3] Full output message
	[;4] 1-line execution summary (e.g., 100 records imported from [xyz.xls])

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
X = Array("Data Script Name", "", "302", "")
RetVal = eng.PVCall("RUNDATASCRIPT", X)
' Now process return values
Dim ErrCode as Long, ErrDesc as String, DScriptResults as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
DScriptResults = RetVal(3)
```

ImportDataCorrections (New 3.22)

Description:

Imports data corrections from an excel spreadsheet generated by the data questions tool within ProVal.

Argument (Variant Array)

[0]	Name of database to import into (string)
[1]	Corrections File Pathway of file containing the answered data questions to be imported as corrections (string)
[2]	Log File Name (string) Default log file name is PROVALAPI1.TMP, PROVALAPI2.TMP, etc.
[3]	Error Code Overrides - Error codes separated by spaces under which the user wants processing to continue instead of aborting (string). Alternatively, an empty string must be used if the intention is to halt processing on all errors. NOTE: Errors cannot be overridden unless specified.

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 12 = Invalid ProVal database specified. 43 = Import validation error 44 = Data Import is not allowed into encrypted database files 47 = Data corrections file does not exist. 48 = Problem adding a comment from the data corrections file. 49 = Excel workbook does not appear to have been generated by ProVal's Data Questions command. 311 = Running this command will erase the results of other objects (can override and writes to message log). 312 = Database is not the same as the one that generated the data questions (can override and writes to message log). 313 = Problems importing from a sheet in the data corrections file (can override and writes to message log).
[2]	Log File Name (string) NOTE: If log file is not written out, this contains an empty string.

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
X = Array("DATAFILE", "C:\IMPDATA.CSV", "", "")
RetVal = eng.PVCall("IMPORTDATACORRECTIONS", X)
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
```

Plan Definition

GetPlanConstants

Description:

Returns information about Plan Constants from a Plan Definition.

Note: This function is only available via the API; it is not used by the ProVal interface.

Arguments

	Name of Plan Definition (string)
--	---

Return Value (Variant Array).

[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 84 = Plan Definition entry does not exist
[2]	Plan Constants in use indicator (long) 1 = yes, 0 = no
[3]	Names of Plan Constant coded fields (string for each field) [coded fields] [string]
[4]	Coded field labels (string for each field in [3] and field code in [5]) [coded fields] [codes] [string]
[5]	Coded field codes (long array for each field in [3]) [coded fields] [codes]
[6]	Plan Constant names (string for each field in [3] and plan constant in [6]) [coded fields] [plan constants] [string]
[7]	Plan Constant values (long for each field in [3], plan constant in [6], and field code in [5]) [coded fields] [plan constants] [codes]

Visual Basic for Applications Code Snippet

```
Dim RetVal as Variant
RetVal = eng.PVCall("GETPLANCONSTANTS", "MYPLAN")
```

SetPlanConstants

Description:

Save Plan Constant values to a Plan Definition.

Note: This function is only available via the API; it is not used by the ProVal interface.

Argument (Variant Array)	
[0]	Name of original Plan Definition (string)
[1]	New Plan Definition name for "Save As" or empty string ("") to "Replace" in Plan Definition library (string)
[2]	Plan Constant values. Number of values must match the existing number of values. See GetPlanConstants[7] (variant)
[3]	Log file name (string) Default log file name is PROVALAPI1.TMP, PROVALAPI2.TMP, etc.
[4]	Error Code Overrides - Error codes separated by spaces under which the user wants processing to continue instead of aborting (string). Alternatively, an empty string must be used if the intention is to halt processing on all errors. NOTE: Errors cannot be overridden unless specified.

Return Value (Variant Array)	
[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 75 = Incorrect number of values, expecting value for each Coded Field, Plan Constant, and Field Code 86 = New name for Plan Definition is already in use 311 = Running this command will erase the results of other objects (can override and writes to message log)
[2]	Log File Name (string) NOTE: If log file is not written out, this contains an empty string.

Visual Basic for Applications Code Snippet

```
Dim PCArray, RetVal as Variant
PCArray = eng.PVCall("GETPLANCONSTANTS", "MYPLAN")
If PCArray(0) <> "" Then
    PCValues = PCArray(7) ' plan constant values

    ' @EarlyAge
    PCValues(0)(0) = 55
    PCValues(0)(1) = 53
    ' @NormalAge
    PCValues(1)(0) = 63
    PCValues(1)(1) = 67
    ' @Service
    PCValues(2)(0) = 10
    PCValues(2)(1) = 12

    X = Array("MYPLAN", "", PCValues, "", "")
    RetVal = eng.PVCall("SETPLANCONSTANTS", X)
End If
```

Valuation Assumptions

GetValAssum

Description:

Returns the Valuation Assumptions.

Note: This function is only available via the API; it is not used by the ProVal interface.

Arguments	
	Name of Valuation Assumptions (string)

Return Value (Variant Array).	
[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 64 = Valuation Assumptions do not exist
[2]	Assumption type (integer) 1 = funding, 2 = accounting
[3]	Law type (integer) US: 1=pre-PPA, 2=post-PPA, 3=pre-PPA and post-PPA, 4=multiemployer UK: 1=PPF, 2=Ongoing OPEB: 0=not LTD, 1=LTD
[4]	Valuation assumptions vector (variant). For further documentation, call <i>GetValAssumDoc</i>

GetValAssumDoc

Description:

Returns documentation for the result of **GetValAssum**.

Note: This function is only available via the API; it is not used by the ProVal interface.

Arguments

<None>

Return Value (string).

[0] Documentation for result of **GetValAssum**, newline-delimited string

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
Dim myArray As Variant
Dim i As Long
RetVal = eng.PVCall("GETVALASSUMDOC")

myArray = Split(CStr(RetVal(0)), vbCr)
' print the GetValAssum Documentation to your worksheet
For i = LBound(myArray) To UBound(myArray)
    ActiveSheet.Cells(i, 1) = myArray(i)
Next
```


SetValAssum

Description:

Sets the Valuation Assumptions.

Note: This function is only available via the API; it is not used by the ProVal interface.

For use in the following sequence:

1. **GetValAssum.** Return the Valuation Assumptions.
2. **SetValAssum.** Revise the Valuation Assumptions.
3. **RunVal.** Recalculate contributions and expense with revised Valuation Assumptions.

Argument (Variant Array)	
[0]	Name of original Valuation Assumptions (string)
[1]	New Valuation Assumptions name for "Save As" or empty string ("") to "Replace" in Valuation Assumptions Library (string)
[2]	Valuation Assumptions. See GetValAssum return value for format (variant)
[3]	Log file name (string) Default log file name is PROVALAPI1.TMP, PROVALAPI2.TMP, etc.
[4]	Error Code Overrides - Error codes separated by spaces under which the user wants processing to continue instead of aborting (string). Alternatively, an empty string must be used if the intention is to halt processing on all errors. NOTE: Errors cannot be overridden unless specified.

Return Value (Variant Array)	
[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 65 = New name for Valuation Assumptions is already in use 85 = Incorrect number of items for library entry 121 = Library Entry error (writes to message log) 311 = Running this command will erase the results of other objects (can override and writes to message log)
[2]	Log File Name (string) NOTE: If log file is not written out, this contains an empty string.

Visual Basic for Applications Code Snippet

```
Dim SetValAssumArr As Variant
Dim GetValAssumArr As Variant
Dim X As Variant, Y As Variant
Dim ValAssum As String, ValAssumSaveAs As String

ValAssum = "Funding 2016" ' Val assumptions to use
ValAssumSaveAs = "Funding 2016 From API" 'Save as new

' Check to see if a client is open
If eng.CurrClient <> "" Then
    GetValAssumArr = eng.PVCall("GETVALASSUM", ValAssum)
    If GetValAssumArr(0) = 0 Then 'Received assumptions successfully
        ' ***** Example *****
        ' * Changing the PPA minimum funding segment rates *
        ' *****
        GetValAssumArr(4)(168)(0) = 0.03
        GetValAssumArr(4)(168)(1) = 0.04
        GetValAssumArr(4)(168)(2) = 0.05

        ' ***** Example *****
        ' * Changing the PPA max tax segment rates *
        ' *****
        GetValAssumArr(4)(133)(0) = 0.05
        GetValAssumArr(4)(133)(1) = 0.055
        GetValAssumArr(4)(133)(2) = 0.06

        Y = Array(GetValAssumArr(2), GetValAssumArr(3), GetValAssumArr(4))
        X = Array(ValAssum, ValAssumSaveAs, Y, "", "")
        SetValAssumArr = eng.PVCall("SETVALASSUM", X)
    ' Now process return values
    If SetValAssumArr(0) = 0 Then 'Successfully saved revised assumptions
        MsgBox ("Completed successfully!")
    Else
        MsgBox ("Error code: " & SetValAssumArr(0) & vbCr & _
            "Error description: " & SetValAssumArr(1))
    End If
End If
Else
    MsgBox("You must open a ProVal client first.")
End If
```

GetTable

Description:

Returns a table.

Note: This function is only available via the API; it is not used by the ProVal interface.

Arguments

[0]	Name of the Table library (string) Library names: MORTRATE – mortality rate table DISABRATE – disability rate table TERMRATE – termination rate table RETRATE – retirement rate table SALSCALE – salary merit scale ABTABLE – benefit component table POSTDECPROB – election probability COLA – COLA rate table FRACMARRIED – fraction married table SPAGEDIFF – age difference table MORTIMP – mortality improvement scale LAPSE – lapse probability table
[1]	Name of Table entry (string)

Return Value (Variant Array).

[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 71 = Protected tables cannot be modified 72 = Dynamic mortality tables cannot be accessed through the API
[2]	Table type (long)
[3]	Starting index of first dimension (long)
[4]	Starting index of second dimension (long)
[5]	Shape of array containing table values (variant)
[6]	Array of table values (variant)

SetTable

Description:

Sets a Table.

Note: This function is only available via the API; it is not used by the ProVal interface.

Argument (Variant Array)	
[0]	Library name (string)
[1]	Name of original Table entry (string)
[2]	New Table name for "Save As" or empty string ("") to "Replace" in Valuation Assumptions Library (string)
[3]	Table values. See GetTable return value for format (variant)
[4]	Log file name (string) Default log file name is PROVALAPI1.TMP, PROVALAPI2.TMP, etc.
[5]	Error Code Overrides - Error codes separated by spaces under which the user wants processing to continue instead of aborting (string). Alternatively, an empty string must be used if the intention is to halt processing on all errors. NOTE: Errors cannot be overridden unless specified.

Return Value (Variant Array)	
[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 85 = Incorrect number of items for library entry 311 = Running this command will erase the results of other objects (can override and writes to message log)
[2]	Log File Name (string) NOTE: If log file is not written out, this contains an empty string.

Valuation

GetValInputs

Description:

Gets the Valuation input parameters from a previously defined Valuation.

Argument (Variant Array)

[0]	Name of Valuation (string)
------------	-----------------------------------

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string)
[2]	Valuation Date (string) [3] [1] Year [2] Month [3] Day
[3]	Name of Census Database File (string)
[4]	Name of Census Specification (string)
[5]	Use Data Defaults (bool) [1=Yes; 0=No]
[6]	Selection Expression (string)
[7]	Name of Plan Definition (string)
[8]	Name of Funding Valuation Assumptions (string)
[9]	Name of Accounting Valuation Assumptions (string)
[10]	Sensitivities (variant) [10] [1] Run Funding Sensitivities [2] Run Accounting Sensitivities [3] Which sensitivities to run? [5] [1] interest rates (bool) [2] salary inflation (bool) [3] COLAs (bool) [4] increase and crediting rates (bool) [5] lump sum & optional forms (bool) [4] Sensitivity set (group) of each sensitivity type [7] (1=misc.; 2=interest; 3=inflation; 4=trend; 5=mortality) [1] interest rates sensitivity set (long) [2] salary inflation sensitivity set (long) [3] COLAs sensitivity set (long) [4] increase and crediting rates sensitivity set (long)

	[5] lump sum & optional forms sensitivity set (long)
	[6] valuation mortality sensitivity set (long)
	[7] mortality in conversion factors sensitivity set (long)
	[5] Allow negative values for economic sensitivities [5] (1=yes,0=no)
	[1] interest rates (bool)
	[2] salary inflation (bool)
	[3] COLAs (bool)
	[4] Increase and crediting rates (bool)
	[5] lump sum & optional forms (bool)
	[6] Funding economic assumptions sensitivity inputs
	[1] Low (double)
	[2] High (double)
	[7] Accounting economic assumptions sensitivity inputs
	[1] Low (double)
	[2] High (double)
	[8] Mortality sensitivities to run (1=yes,0=no)
	[1] valuation mortality (bool)
	[2] mortality in conversion factors (bool)
	[9] Mortality assumptions sensitivity inputs for funding [4;2]
	[1;] Male setback (long)
	[2;] Female setback (long)
	[3;] Male factor (double)
	[4;] Female factor (double)
	[;1] Low
	[;2] High
	[10] Mortality assumptions sensitivity inputs for accounting [4;2]
	[1;] Male setback (long)
	[2;] Female setback (long)
	[3;] Male factor (double)
	[4;] Female factor (double)
	[;1] Low
	[;2] High
[11]	Subtotal Fields (variant) [#Fields]
[12]	Code for calculation of projected headcounts & benefits by subtotal (long) [0=do not calculate, 1=PBO/PUCAL only, 2=calculate for all bases]
[13]	Name of Individual Results File (string)
[14]	Name of Scaling Factor to apply (string)
[15]	Checkbox for Individual results (1=Run, 0=Do no run) (bool)

SetValInputs

Description:

Saves Valuation Input parameters into a previously defined Valuation.

Argument (Variant Array)	
[0]	Name of Valuation (string)
[1]	Valuation Date (string) [3] [1] Year [2] Month [3] Day
[2]	Name of Census Database File (string)
[3]	Name of Census Specification (string)
[4]	Use Data Defaults (bool) [1=Yes; 0=No]
[5]	Selection Expression (string)
[6]	Name of Plan Definition (string)
[7]	Name of Funding Valuation Assumptions (string)
[8]	Name of Accounting Valuation Assumptions (string)
[9]	Sensitivities (variant) [10] [1] Run Funding Sensitivities [2] Run Accounting Sensitivities [3] Which sensitivities to run? [5] [1] interest rates (bool) [2] salary inflation (bool) [3] COLAs (bool) [4] increase and crediting rates (bool) [5] lump sum & optional forms (bool) [4] Sensitivity set (group) of each sensitivity type [7] (1=misc.; 2=interest; 3=inflation; 4=trend; 5=mortality) [1] interest rates sensitivity set (long) [2] salary inflation sensitivity set (long) [3] COLAs sensitivity set (long) [4] increase and crediting rates sensitivity set (long) [5] lump sum & optional forms sensitivity set (long) [6] valuation mortality sensitivity set (long) [7] mortality in conversion factors sensitivity set (long) [5] Allow negative values for economic sensitivities [5] (1=yes,0=no) [1] interest rates (bool) [2] salary inflation (bool) [3] COLAs (bool)

	[4] Increase and crediting rates (bool)
	[5] lump sum & optional forms (bool)
	[6] Funding economic assumptions sensitivity inputs
	[1] Low (double)
	[2] High (double)
	[7] Accounting economic assumptions sensitivity inputs
	[1] Low (double)
	[2] High (double)
	[8] Mortality sensitivities to run (1=yes,0=no)
	[1] valuation mortality (bool)
	[2] mortality in conversion factors (bool)
	[9] Mortality assumptions sensitivity inputs for funding [4;2]
	[1;] Male setback (long)
	[2;] Female setback (long)
	[3;] Male factor (double)
	[4;] Female factor (double)
	[;1] Low
	[;2] High
	[10] Mortality assumptions sensitivity inputs for accounting [4;2]
	[1;] Male setback (long)
	[2;] Female setback (long)
	[3;] Male factor (double)
	[4;] Female factor (double)
	[;1] Low
	[;2] High
[10]	Subtotal Fields (variant) [#Fields]
[11]	Code for calculation of projected headcounts & benefits by subtotal (long) [0=do not calculate, 1=PBO/PUCAL only, 2=calculate for all bases]
[12]	Name of Individual Results File (string)
[13]	Name of Scaling Factor to apply (string)
[14]	Checkbox for Individual results (1=Run, 0=Do not run) (bool)

Return Value (Variant Array)	
[0]	Error code (long)
[1]	Error description (string)

RunVal

Description:

Executes a previously defined Valuation.

Argument (Variant Array)	
[0]	Name of Valuation to execute (string)
[1]	New valuation name for "Save As" or empty string ("") to "Replace" in Valuation Library (string)
[2]	Name of Census Database File or "" to use name saved with Valuation (string)
[3]	Name of Individual Results File to use or "" to use name saved with Valuation (string)
[4]	Hide RecIDs from the processing messages displayed in the log file? (long: 1=yes, 0=no)
[5]	Log file name (string) Default log file name is PROVALAPI1.TMP, PROVALAPI2.TMP, etc.
[6]	Error Code Overrides - Error codes separated by spaces under which the user wants processing to continue instead of aborting (string). Alternatively, an empty string must be used if the intention is to halt processing on all errors. NOTE: Errors cannot be overridden unless specified.
[7]	Selection Expression or "" to use selection expression saved with Valuation (string)
[8]	Name of Funding Valuation Assumptions or "" to use name saved with Valuation (string)
[9]	Name of Accounting Valuation Assumptions or "" to use name saved with Valuation (string)
[10]	Valuation Date or "" to use valuation date saved with Valuation. Date formatted per the machine's Regional Settings, e.g., MM/DD/YYYY, DD/MM/YYYY, etc.
[11]	Run Type - "" or 0 = Run normally, 1 = Validate , 3 = Run Individual Results For Changed Records (German Mode ONLY).

Return Value (Variant Array)	
[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 13 = Hide RecIDs indicator must be 1 or 0 60 = Valuation does not exist 61 = The Valuation cannot be run until you complete the following inputs 63 = Invalid date 66 = Valuation Assumptions are mismatched by funding/accounting type 111 = Valuation error (writes to message log)

[2]

211 = Valuation has already been run (can override)
212 = Requested valuation name is already in use (can override)
213 = Running this command could permanently change the Individual Results database (can override)
311 = Running this command will erase the results of other objects (can override and writes to message log)

Log File Name (string)**NOTE:** If log file is not written out, this contains an empty string.**Visual Basic for Applications Code Snippet**

```
Dim RetVal As Variant
X = Array("VAL 2006", "", "", "", 1, "", "", "", "", "", "1/1/2006", "")
RetVal = eng.PVCall("RUNVAL", X)
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
```

GetValResults

Description:

Returns the Valuation results.

Note: This function is only available via the API; it is not used by the ProVal interface.

Arguments (Variant Array)

[0]	Name of Valuation (string)
[1]	Assumption type (integer) 1 = funding, 2 = accounting
[2]	Type of result (optional; integer) 0 = totals, or row number within matrix of coded field values for subtotals (see Return Value below)
[3]	Name of Output Style (optional; string)

Return Value (Variant Array).

	The format of the Return Value depends on whether an (optional) Output Style was specified in the Arguments. Each format is shown separately below.
	<u>If an Output Style was not specified</u>
[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 60 = Valuation does not exist 62 = Valuation has not been run 68 = Row number for subtotals is invalid
	Output Arrays for Pension:
[2]	OUTC1 (variant). For further documentation, call GetValResultsDoc .
[3]	OUTC2 (variant). For further documentation, call GetValResultsDoc .
[4]	OUTC3 (variant). This is no longer used in version 3.18.
[5]	OUTC4 (variant). For further documentation, call GetValResultsDoc .
[6]	OUTC5 (variant). For further documentation, call GetValResultsDoc .
[7]	OUTC6 (variant). For further documentation, call GetValResultsDoc
	Additional Output for Pension:
[8]	Active benefit detail.
[9]	Inactive benefit detail.

[10]	Data fields subject to summation.
[11]	Field names used to define subtotals, if any. Character matrix.
[12]	Numeric matrix of coded field values for subtotals, with a row for each subtotal and column for each subtotal field in [11]. [subtotal;fields].
Output Array for Medical:	
[2]	OUTM1 (variant). For further documentation, call <i>GetValResultsDoc</i> .
Additional Output for Medical:	
[3]	Field names used to define subtotals, if any. Character matrix.
[4]	Numeric matrix of coded field values for subtotals, with a row for each subtotal and column for each subtotal field in [3]. [subtotal;fields].
<u>If an Output Style was specified</u>	
The number of elements in the Return Value will depend on what variables, subtotals, and sensitivities were selected in the Output Style.	
[0]	Error code (long)
[1]	Error description (string)
[2]	The number I of tables that will be returned (integer) For each table, 3 elements will be returned. For $i \in (0, 1, 2, \dots, I-1)$:
[3+i+0]	Number m of rows in table i (integer)
[3+i+1]	Number n of columns in table i (integer)
[3+i+2]	Table i (m by n array)

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
X = Array("VAL 2006", 1)
RetVal = eng.PVCall("GETVALRESULTS", X)
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
If RetVal(0) = 0 Then ' Successfully retrieved results
' select which of the OUTC arrays you would like to get results from
Dim OUTC1Arr As Variant, OUTC2Arr As Variant, OUTC3Arr As Variant
Dim OUTC4Arr As Variant, OUTC5Arr As Variant, OUTC6Arr As Variant

OUTC1Arr = RetVal(2)
OUTC2Arr = RetVal(3)
OUTC3Arr = RetVal(4)
OUTC4Arr = RetVal(5)
OUTC5Arr = RetVal(6)
OUTC6Arr = RetVal(7)

' see GETVALRESULTSDOC for more information on where to find results
MsgBox ("Headcount for active members: " & OUTC1Arr(0, 0))

End If
```

GetValResultsDoc

Description:

Returns documentation for the Valuation results.

Note: This function is only available via the API; it is not used by the ProVal interface.

Arguments

<None>

Return Value (string).

Documentation for Valuation results in **GetValResults**, newline-delimited string. The output differs between OPEB and (all other) pension modes:

Return Position	Label in documentation (Pension modes)	Label in documentation (OPEB mode)
[2]	OUTC1_DEF	OUT_MED_DOC
[3]	OUTCX2_DEF	
[4]	No longer used (3.18)	
[5]	OUTCX4_DEF	
[6]	OUTCM5_DEF	
[7]	OUTC6_DEF	

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
Dim myArray As Variant
Dim I As Long
RetVal = eng.PVCall("GETVALRESULTSDOC")

myArray = Split(CStr(RetVal(0)), vbCr)
' print the GetValResults Documentation to your worksheet
For I = LBound(myArray) To UBound(myArray)
    ActiveSheet.Cells(I,1) = myArray(I)
Next
```

Valuation Set

GetValSetInputs

Description:

Get Valuation Set Inputs.

Argument (Variant Array)

[0]	Name of Valuation Set (string)
------------	---------------------------------------

Return Value (Variant Array)

[0]	Error code (long)
------------	--------------------------

[1]	Error description (string)
------------	-----------------------------------

Notable Errors (also see [Appendix A](#)):

54 = Valuation Set does not exist

64 = Asset/Funding Policy does not exist

[2]	Event matrix. The first row of this matrix is the Baseline Gain or Loss Event. The subsequent rows define additional events. For details of event matrix, see GetValSetEventDoc.
------------	--

[3]	Apply Scaling Factors (bool)
------------	-------------------------------------

[4]	Name of Asset/Funding Policy (string)
------------	--

[5]	Funding Assumption Sensitivity 1=baseline, 2=misc. high, 3=misc. low, 4=interest high, 5=interest low, 6=inflation high, 7=inflation low, 8=trend high, 9=trend low, 10=mortality high, 11=mortality low
------------	--

[6]	Accounting Assumption Sensitivity 1=baseline, 2=misc. high, 3=misc. low, 4=interest high, 5=interest low, 6=inflation high, 7=inflation low, 8=trend high, 9=trend low, 10=mortality high, 11=mortality low
------------	---

SetValSetInputs

Description:

Set Valuation Set Inputs.

Argument (Variant Array)

[0]	Name of Valuation Set to execute (string)
------------	--

[1]	Event matrix. The first row of this matrix is the Baseline Gain or Loss Event. The subsequent rows define additional events. For details of event matrix, see GetValSetEventDoc.
------------	--

[2]	Apply Scaling Factors (bool)
[3]	Name of Asset/Funding Policy (string)
[4]	Funding Assumption Sensitivity 1=baseline, 2=misc. high, 3=misc. low, 4=interest high, 5=interest low, 6=inflation high, 7=inflation low, 8=trend high, 9=trend low, 10=mortality high, 11=mortality low
[5]	Accounting Assumption Sensitivity 1=baseline, 2=misc. high, 3=misc. low, 4=interest high, 5=interest low, 6=inflation high, 7=inflation low, 8=trend high, 9=trend low, 10=mortality high, 11=mortality low

Return Value (Variant Array)	
[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 54 = Valuation Set does not exist 64 = Asset/Funding Policy does not exist 211 = Valuation Set has already been run (can override) 212 = Requested Valuation Set name is already in use (can override) 213 = Running this command could permanently change the Individual Results database (can override) 311 = Running this command will erase the results of other objects (can override and writes to message log)
[2]	Log File Name (string) NOTE: If log file is not written out, this contains an empty string.

GetValSetEventDoc

Description:

Returns documentation for the result of **GetValSetInputs[2]** (*Event Matrix*).

NB! This function is only available via the API; it is not used by the ProVal interface.

Arguments
<None>

Return Value (string). See also <i>Return Value on Error</i> .	
[0]	Documentation for result of GetValSetInputs[2] (<i>Event Matrix</i>), newline-delimited string

RunValSet

Description:

Executes a previously defined Valuation Set.

Argument (Variant Array)	
[0]	Name of Valuation Set to execute (string)
[1]	New Valuation Set name for "Save As" or empty string ("") to "Replace" in Valuation Set Library (string)
[2]	Name of Asset/Funding Policy or "" to use name saved with Valuation Set (string)
[3]	Log file name (string) Default log file name is PROVALAPI1.TMP, PROVALAPI2.TMP, etc.
[4]	Error Code Overrides - Error codes separated by spaces under which the user wants processing to continue instead of aborting (string). Alternatively, an empty string must be used if the intention is to halt processing on all errors. NOTE: Errors cannot be overridden unless specified.

Return Value (Variant Array)	
[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 54 = Valuation Set does not exist 64 = Asset/Funding Policy does not exist 211 = Valuation Set has already been run (can override) 212 = Requested Valuation Set name is already in use (can override) 213 = Running this command could permanently change the Individual Results database (can override) 311 = Running this command will erase the results of other objects (can override and writes to message log)
[2]	Log File Name (string) NOTE: If log file is not written out, this contains an empty string.

GetValSetExhibitResults

Description:

Get Valuation Set Inputs.

Argument (Variant Array)

[0]	Name of Valuation Set (string)
[1]	Exhibits List of exhibit codes (listed below) separated by spaces or 'ALL' (string)
[2]	Display results by valuation event? (long: 1=yes,0=no)
[3]	Rounding method (long: 0, 1, 100, 1000, 10000, 100000, 1000000)

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 54 = Valuation Set does not exist 64 = Asset/Funding Policy does not exist
[2]	Exhibit data. This data is in a nested array structure. The Exhibit name is displayed and the matrix that follows it is the corresponding data values for that Exhibit.

Core Projection

RunCore

Description:

Executes a previously defined Core Projection.

Argument (Variant Array)	
[0]	Name of Core Projection to execute (string)
[1]	New Core Projection name for "Save As" or empty string ("") to "Replace" in Core Projection Library (string)
[2]	Name of Census Database File or "" to use name saved with Core Projection (string)
[3]	Name of Individual Results File to use or "" to use name saved with Core Projection (string)
[4]	Hide RecIDs from the processing messages displayed in the log file? (long: 1=yes, 0=no)
[5]	Log file name (string) Default log file name is PROVALAPI1.TMP, PROVALAPI2.TMP, etc.
[6]	Error Code Overrides - Error codes separated by spaces under which the user wants processing to continue instead of aborting (string). Alternatively, an empty string must be used if the intention is to halt processing on all errors. NOTE: Errors cannot be overridden unless specified.
[7]	Selection Expression or "" to use selection expression saved with Core Projection (string)
[8]	Name of Funding Valuation Assumptions or "" to use name saved with Core Projection (string)
[9]	Name of Accounting Valuation Assumptions or "" to use name saved with Core Projection (string)
[10]	Valuation Date or "" to use valuation date saved with Core Projection (string). Date formatted per the machine's Regional Settings, e.g., MM/DD/YYYY, DD/MM/YYYY, etc.
[11]	Name of Projection Assumptions or "" to use name saved with Core Projection (string)
Return Value (Variant Array)	
[0]	Error code (long)

[1]	Error description (string) Notable Errors (also see Appendix A): 13 = Hide RecIDs indicator must be 1 or 0 60 = Core Projection does not exist 63 = Invalid date 66 = Valuation Assumptions are mismatched by funding/accounting type 111 = Core Projection error (writes to message log) 211 = Core Projection has already been run (can override) 212 = Requested Core Projection name is already in use (can override) 213 = Running this command could permanently change the Individual Results database (can override) 311 = Running this command will erase the results of other objects (can override and writes to message log)
[2]	Log File Name (string) NOTE: If log file is not written out, this contains an empty string.

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
X = Array("CP 2006", "", "", "", 1, "", "", "", "", "", "1/1/2006", "")
RetVal = eng.PVCall("RUNCORE", X)
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
```

GetCoreResults

Description:

Returns the Core Projection results.

Note: This function is only available via the API; it is not used by the ProVal interface.

Arguments (Variant Array)

[0]	Name of Core Projection (string)
[1]	Sensitivity type (integer) 1 = Baseline 2 = Low interest 3 = High interest 4 = Low inflation 5 = High inflation 6 = Low interest, Low inflation 7 = Low interest, High inflation 8 = High interest, Low inflation 9 = High interest, High inflation 10 = Low lump sum interest 11 = High lump sum interest 12 = Low asset benchmark 13 = High asset benchmark
[2]	Type of result (integer) 0 = totals, or row number within matrix of subtotal permutations
[3]	Output Style (optional) (new in 3.19)

Return Value (Variant Array).

	The format of the Return Value depends on whether an (optional) Output Style was specified in the Arguments. Each format is shown separately below.
	<u>If an Output Style was not specified</u>
[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 60 = Core projection does not exist 62 = Core projection has not been run 68 = Row number for subtotals is invalid 69 = Row number for sensitivity is invalid
	Output Arrays for Pension:

[2]	OUTC1 (variant). For further documentation, call <i>GetValResultsDoc</i> .
[3]	OUTC2 (variant). For further documentation, call <i>GetValResultsDoc</i> .
[4]	OUTC3 (variant). This is no longer used in version 3.18.
[5]	OUTC4 (variant). For further documentation, call <i>GetValResultsDoc</i> .
[6]	OUTC5 (variant). For further documentation, call <i>GetValResultsDoc</i> .
[7]	OUTC6 (variant). For further documentation, call <i>GetValResultsDoc</i>
	Additional Output for Pension:
[8]	Active benefit detail.
[9]	Inactive benefit detail.
	Other Output:
[10]	Matrix of field names used to define subtotals, if any.
[11]	Permutations of internal coded values that correspond to defined subtotal fields and identify the stored subtotals.
<hr/>	
	Output Array for Medical:
[2]	OUTM1 (variant). For further documentation, call <i>GetValResultsDoc</i> .
	Other Output:
[3]	Matrix of field names used to define subtotals, if any.
[4]	Permutations of internal coded values that correspond to defined subtotal fields and identify the stored subtotals.
	<u>If an Output Style was specified</u>
	The number of elements in the Return Value will depend on what variables, subtotals, and sensitivities were selected in the Output Style.
[0]	Error code (long)
[1]	Error description (string)
[2]	The number I of tables that will be returned (integer) For each table, 3 elements will be returned. For $i \in (0, 1, 2, \dots, I-1)$:
[3+i+0]	Number m of rows in table i (integer)
[3+i+1]	Number n of columns in table i (integer)
[3+i+2]	Table i (m by n array)

Asset & Funding Policy

GetAFP

Description:

Returns the asset & funding policy vector. Does NOT reflect any adjustments made by previous calls to **SetAFP**.

Arguments

[0]	Name of Asset & Funding Policy (string)
-----	---

Return Value (Variant Array). See also Return Value on Error.

[0]	Error code (long)
[1]	Error description (string)
[2]	Asset & Funding Policy vector (variant). For further documentation, call GetAFPDoc

GetAFPDoc

Description:

Returns documentation for the result of **GetAFP**.

NB! This function is only available via the API; it is not used by the ProVal interface.

Arguments

<None>

Return Value (string). See also Return Value on Error.

[0]	Documentation for result of GetAFP , newline-delimited string
-----	--

SetAFP

Description:

Sets the Asset & Funding Policy.

NB! This function is only available via the API; it is not used by the ProVal interface. Intended for clients who want to manipulate the Asset & Funding Policy through the API. Great care must be taken to adjust all of the relevant items in a consistent manner.

For use in the following sequence:

1. **GetAFP**. Return the Asset & Funding Policy.
2. **SetAFP**. Revise the Asset & Funding Policy.
3. **RunDetFcst** or **RunStoFcst**. Recalculate contributions and expense with revised Asset & Funding Policy.
4. (optional) **GetValSetExhibitResults**
5. Repeat last three steps as needed.

Argument (Variant Array)

[0]	Name of Asset & Funding Policy (string)
[1]	New Asset & Funding Policy name for "Save As" or empty string ("") to "Replace" in AFP Library (string)
[2]	Asset & Funding Policy. See GetAFP return value for format.
[3]	Log file name (string) Default log file name is PROVALAPI1.TMP, PROVALAPI2.TMP, etc.
[4]	Error Code Overrides - Error codes separated by spaces under which the user wants processing to continue instead of aborting (string). Alternatively, an empty string must be used if the intention is to halt processing on all errors. NOTE: Errors cannot be overridden unless specified.

Return Value (Variant Array). See also Return Value on Error.

[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 85 = Incorrect number of items for library entry 311 = Running this command will erase the results of other objects (can override and writes to message log)
[2]	Log File Name (string) NOTE: If log file is not written out, this contains an empty string.

Deterministic Forecast

RunDetFore

Description:

Executes a previously defined Deterministic Forecast.

Argument (Variant Array)

- | | |
|-----|--|
| [0] | Name of Deterministic Forecast to execute (string) |
| [1] | New Deterministic Forecast name for "Save As" or empty string ("") to "Replace" in Deterministic Forecast Library (string) |
| [2] | Name of Asset/Funding Policy or "" to use name saved with Deterministic Forecast (string) |
| [3] | Name of Deterministic Assumptions or "" to use name saved with Deterministic Forecast (string) |
| [4] | Log file name (string)
Default log file name is PROVALAPI1.TMP, PROVALAPI2.TMP, etc. |
| [5] | Error Code Overrides - Error codes separated by spaces under which the user wants processing to continue instead of aborting (string).
Alternatively, an empty string must be used if the intention is to halt processing on all errors.
NOTE: Errors cannot be overridden unless specified. |

Return Value (Variant Array)

- | | |
|-----|--|
| [0] | Error code (long) |
| [1] | Error description (string)
Notable Errors (also see Appendix A):
64 = Deterministic Forecast does not exist
64 = Asset/Funding Policy does not exist
64 = Deterministic Assumptions does not exist
211 = Deterministic Forecast has already been run (can override)
212 = Requested Deterministic Forecast name is already in use (can override)
213 = Running this command could permanently change the Individual Results database (can override)
311 = Running this command will erase the results of other objects (can override and writes to message log) |
| [2] | Log File Name (string) |

NOTE: If log file is not written out, this contains an empty string.

Report Writer

ExportValSetExhibits

Description:

Exports a set of valuation set exhibits to a Microsoft Access database.

Argument (Variant Array)	
[0]	ValSet Exact name of the valuation set (string)
[1]	FileOut Access database filename where output is to be sent (string)
[2]	Exhibits List of exhibit codes (listed below) separated by spaces or 'ALL' (string)
[3]	File Handling Options (string): A = Append R = Replace Q = Quit
[4]	Display results by valuation event? (long: 1=yes,0=no)
[5]	Rounding method (long: 0, 1, 100, 1000, 10000, 100000, 1000000)

Exhibit Code	Exhibit Description
14	Schedule of Active Participant Data
13	Schedule B - Schedule of Active Participant Data
25	Distribution of Inactive Benefits Inforce
21	Development of Actuarial Assets
12	Summary of Employer Contribution
1	Summary of Minimum Required Contribution Limits
2	Summary of Maximum Tax Deductible Contribution Limits
3	Funding Amortization Bases, Minimum Basis
4	Funding Amortization Bases, Maximum Basis
5	Additional Funding Charge
6	Schedule of Employer Contributions
7	Development of Full Funding Limits (or Canadian Surplus Threshold)
11	Development of Normal Cost
22	Development of Market-Related Assets
8	Development of Pension Expense
15	Accounting Amortization Bases

9	Reconciliation of Funded Status / Balance Sheet Entries
10	Statement of PVAB and Reconciliation from Prior Year
18	Development of Annual Required Contribution (ARC)
19	Development of Annual OPEB Cost & Net OPEB Obligation
20	Development of +1%/-1% Trend Sensitivities
16	Development of Solvency Deficiency
26	Development of PBGC Premium
30	Development of Solvency Assets
31	Development of Estimated Reorganization Index
32	Development of Reserve & General Account

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string)
	Notable Errors (also see Appendix A):
	50 = Valuation Sets have NOT been run in this project
	51 = Asset & Funding Policy problems while validating Valuation Set
	52 = Valuation Event problems while validating Valuation Set
	53 = Error creating Valuation Set Exhibits
	54 = Valuation Set does not exist
	55 = Invalid valuation exhibit number
	56 = Schedule of active data and/or distribution of inactive benefits not available
	57 = Problems trying to write to the Access database file

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
X = Array("GROUPED DATA", "OUTDATA.MDB", "14 13", "", 0, 1000)
RetVal = eng.PVCall("EXPORTVALSETEXHIBITS", X)
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
```

GetRWDatasets

Description:

Returns Data Sets mapping for a Report Definition in a Report Writer database

Argument (Variant Array)

[0]	File Access database filename (string)
[1]	ReportNo Report Definition number. Corresponds to ReportNo field in PVRW_RptDefs or PVRW_Reports table (long).

Return Value (Variant Array)

[0]	Error code (long) (see Appendix A)
[1]	Error description (string)
[2]	List of data set types (string array) Types: "Valuation Set", "Deterministic Forecast", "Gain & Loss", or "Descriptive Statistics"
[3]	List of Labels (string array) Examples: "Current Year", "Prior Year 1", etc.
[4]	List of data set descriptions (string array) Example: "01/01/2015 -- xyz"
[5]	List of data set RunIDs (integer array) Corresponds to ProVal_ValSet_Params.RunID field
[6]	List of group codes (integer array)
[7]	List of alternate labels (string array) Specified for exhibits if label is longer than 19 characters.

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
X = Array("c:\temp\ReportData.accdb", 1)
RetVal = eng.PVCall("GETRWDATASETS", X)
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
```

GetRWReconOfAssets

Description:

Returns "Reconciliation of Assets" data for a Report Definition in a Report Writer database.

Argument (Variant Array)

[0]	File Access database filename (string)
[1]	ReportNo Report Definition number. Corresponds to ReportNo field in PVRW_RptDefs or PVRW_Reports table (long).

Return Value (Variant Array)

[0]	Error code (long) (see Appendix A)
[1]	Error description (string)
[2]	AssetData . A matrix with a row for each line in the exhibit and columns holding: [;0] Outline level: 0=Special, 1=Top, 2, 3 [;1] Row type [;2] Full outline for FieldSpec -- e.g., '(1)(c)(ii)' [;3] Text -- without outline prefix and formula suffix [;4] Value for data set #1 [;5] Value for data set #2 ... [;n+3] Value for data set #n

Valid row types are:

0=Special
1=Normal
8=Total Income
9=Disbursements
10=Plan Transfers
11=Net Increase/(Decrease)
12=Market Value of Assets for Valuation
13=Return on Market Value of Assets

The first two rows of the matrix are special. They must be:

[0;] 0 0 " 'Data Set Label' 'data set name #1' 'data set name #2' ...
[1;] 0 0 'ValDate' 'Valuation Date' 'mm/dd/yyyy' 'mm/dd/yyyy' ...

Note: The rows and columns of matrices are transposed in VB.

GetRWStmntOfAssets

Description:

Returns "Statements of Assets" data for a Report Definition in a Report Writer database.

Argument (Variant Array)

[0]	File Access database filename (string)
[1]	ReportNo Report Definition number. Corresponds to ReportNo field in PVRW_RptDefs or PVRW_Reports table (long).

Return Value (Variant Array)

[0]	Error code (long) (see Appendix A)
[1]	Error description (string)
[2]	AssetData. Same structure as AssetData for GetRWReconOfAssets , except valid row types are: 0=Special 1=Normal 8=Trust Fund Market Value of Assets 9=Receivables 10=Payables 11=Market Value of Assets for Valuation 12=Actuarial Value of Assets 13=Difference in Market Value and Actuarial Value of Assets 14=Actuarial Value of Assets as a Percentage of Market Value

SetRWReconOfAssets

Description:

Saves "Reconciliation of Assets" data for a Report Definition in a Report Writer database.

Argument (Variant Array)	
[0]	File Access database filename (string)
[1]	ReportNo Report Definition number. Corresponds to ReportNo field in PVRW_RptDefs or PVRW_Reports table (long).
[2]	AssetData . Same structure as AssetData for GetRWReconOfAssets .

Return Value (Variant Array)	
[0]	Error code (long) Notable Errors (also see Appendix A): 90 = Asset data is not shaped correctly 91 = Asset data has invalid first two rows 92 = Asset data requires exactly one row of types X in column 2 93 = Asset data has an incorrect outline string in column 3 94 = Asset data has invalid values in column 4 95 = Asset data has invalid values in column 5
[1]	Error description (string)

SetRWStmntOfAssets

Description:

Saves "Statements of Assets" data for a Report Definition in a Report Writer database.

Argument (Variant Array)

- | | |
|-----|---|
| [0] | File Access database filename (string) |
| [1] | ReportNo Report Definition number. Corresponds to ReportNo field in PVRW_RptDefs or PVRW_Reports table (long). |
| [2] | AssetData . Same structure as AssetData for GetRWStmntOfAssets . |

Return Value (Variant Array)

- | | |
|-----|---|
| [0] | Error code (long)
Notable Errors (also see Appendix A):
90 = Asset data is not shaped correctly
91 = Asset data has invalid first two rows
92 = Asset data requires exactly one row of types X in column 2
93 = Asset data has an incorrect outline string in column 3
94 = Asset data has invalid values in column 4
95 = Asset data has invalid values in column 5 |
| [1] | Error description (string) |

Batch (New 3.22)

BatchGetJobs

Description:

Get all the jobs queued on a Batch Inbox.

Argument (Variant Array)

[0]	Batch Inbox Queue name of the Batch Inbox Queue (string)
------------	---

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A):
[2]	Client path in UNC format (string)
[3]	Project id # (0 for universe projects) (string)
[4]	Project name (string)
[5]	Computational mode (string)
[6]	Type: 1 = valuations, 2 = valuation sets, 3 = cores, 4 = deterministic forecasts, 5 = stochastic forecasts, 6 = gain & loss, 7 = experience study (string)
[7]	Library entry ID # (string)
[8]	Library entry name (string)
[9]	User name from proval.ini file (string)
[10]	Start time (0=immediate, 1=1:00AM, ..., 17=5:00PM) (string)
[11]	Submitted time - file created timestamp (string)
[12]	Actual start time (string)
[13]	Completed time (string)
[14]	Processing messages (string)
[15]	Start time (original when submitted) (string)
[16]	Client name (string)
[17]	Server that is processing the request (string)
[18]	Status: 1 = pending, 2 = ready, 3 = running, 4 = done, 5 = canceled, 6 = aborted (string)
[19]	Batch request file name (string)

BatchGetWorkerStatus

Description:

Get the status message of the workers on the Batch Server.

Argument (Variant Array)	
[0]	Batch Inbox Queue name of the Batch Inbox Queue (string)

Return Value (Variant Array)	
[0]	Error code (long)
[1]	Error description (string)
	Notable Errors (also see Appendix A):
[2]	Worker GUID – unique identifier (string)
[3]	Status message (string) - % complete of the current job it is running.

BatchRunVal

Description:

Executes a previously defined Valuation.

Argument (Variant Array)

[0]	Name of Valuation to execute (string)
[1]	Name of Batch Inbox queue.
[2]	Start time of Batch job (0-23, 1=1am, 0=12am, 12=12pm) or '' to start immediately (string)
[3]	Log file name (string) Default log file name is PROVALAPI1.TMP, PROVALAPI2.TMP, etc.
[4]	Error Code Overrides - Error codes separated by spaces under which the user wants processing to continue instead of aborting (string). Alternatively, an empty string must be used if the intention is to halt processing on all errors. NOTE: Errors cannot be overridden unless specified.

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A):
[2]	Log File Name (string) NOTE: If log file is not written out, this contains an empty string.

Visual Basic for Applications Code Snippet

```
Dim RetVal As Variant
X = Array("VAL 2025", "G:\WinTech\ProVal\BatchInbox", "", "", "")
RetVal = eng.PVCall("BATCHRUNVAL", X)
' Now process return values
Dim ErrCode as Long, ErrDesc as String
ErrCode = RetVal(0)
ErrDesc = RetVal(1)
```

BatchRunValSet

Description:

Executes a previously defined Valuation Set.

Argument (Variant Array)	
[0]	Name of Valuation Set to execute (string)
[1]	Name of Batch Inbox queue.
[2]	Start time of Batch job (0-23, 1=1am, 0=12am, 12=12pm) or '' to start immediately (string)
[3]	Log file name (string) Default log file name is PROVALAPI1.TMP, PROVALAPI2.TMP, etc.
[4]	Error Code Overrides - Error codes separated by spaces under which the user wants processing to continue instead of aborting (string). Alternatively, an empty string must be used if the intention is to halt processing on all errors. NOTE: Errors cannot be overridden unless specified.

Return Value (Variant Array)	
[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A):
[2]	Log File Name (string) NOTE: If log file is not written out, this contains an empty string.

BatchRunCore

Description:

Executes a previously defined Core.

Argument (Variant Array)	
[0]	Name of Core to execute (string)
[1]	Name of Batch Inbox queue.
[2]	Start time of Batch job (0-23, 1=1am, 0=12am, 12=12pm) or '' to start immediately (string)
[3]	Log file name (string) Default log file name is PROVALAPI1.TMP, PROVALAPI2.TMP, etc.
[4]	Error Code Overrides - Error codes separated by spaces under which the user wants processing to continue instead of aborting (string).

Alternatively, an empty string must be used if the intention is to halt processing on all errors.

NOTE: Errors cannot be overridden unless specified.

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A):
[2]	Log File Name (string) NOTE: If log file is not written out, this contains an empty string.

BatchRunDetFore

Description:

Executes a previously defined Determination Forecast.

Argument (Variant Array)

[0]	Name of Determination Forecast to execute (string)
[1]	Name of Batch Inbox queue.
[2]	Start time of Batch job (0-23, 1=1am, 0=12am, 12=12pm) or '' to start immediately (string)
[3]	Log file name (string) Default log file name is PROVALAPI1.TMP, PROVALAPI2.TMP, etc.
[4]	Error Code Overrides - Error codes separated by spaces under which the user wants processing to continue instead of aborting (string). Alternatively, an empty string must be used if the intention is to halt processing on all errors. NOTE: Errors cannot be overridden unless specified.

Return Value (Variant Array)

[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A):
[2]	Log File Name (string) NOTE: If log file is not written out, this contains an empty string.

Administration Factors

GetRUNADFACTARGS

Description:

Returns the default values for an RunADFACT call based on the named Administration Factor Tool library entry (from the opened client).

Note: This function is only available via the API; it is not used by the ProVal interface.

Arguments

[0]	Name of Administration Factor (string)
-----	---

Return Value (Variant Array).

[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 84 = Message Definition does not exist
[2]	RunADFACT vector (variant). See RunADFACT arguments in its documentation below.
[3]	Empty string or vector of payment form names
[4]	Empty string or name of the normal form
[5]	Empty string or vector of valid commutative function names

RunADFACT

Description:

Returns annuity factor(s), conversion factor(s), or commutation function for the named Administration Factor (that exists in the client's Administration Factor Tool Library).

Note: This function is only available via the API; it is not used by the ProVal interface.

Arguments

[0]	Administration Factor name (string) The name of the Administration Factor (in the Administration Factors Library) used to calculate annuity factors, conversion factors, or commutation functions. {i.e., the assumptions}
[1]	Calculation Type (long): 1=annuity factors; 2=conversion factors; 3=commutation functions. Note: If you request a calculation type that the Administration Factor wasn't setup for, then you'll get an appropriate error message.
[2]	Annuity Factor or commutation function name (string). For an annuity factor calculation type, annuity factors will be returned for the named

	annuity factor (which must be defined in the named Administration Factor). For the conversion factor type, conversion factors will be returned for this named annuity factor using the Normal Form specified in the Administration Factor. For a commutation functions, it must be one of these names: Dx, Dy, Dxy, ,Nx, Ny, Nxy, ,N(m)x, N(m)y, N(m)xy, ,Sx, Sy, Sxy, ,Cx, Cy, Cxy, ,Mx, My, Mxy, ,Rx, Ry, Rxy, ,ex, ey, or exy; where x indicates primary, y indicates beneficiary, and xy indicates joint life.
[3]	Participant age(s) (integer, long, or double scalar, list (i.e., vector), matrix of ages or blank). Blank indicates that you want to use the age settings in the named Administration Factor. Note that the age participant age settings in the library (see [0]) define the valid age range and Include Monthly ages defines whether fractional ages are allowed.
[4]	Participant sex (integer, long, or double scalar, list (i.e., vector), or matrix of sex codes: 0 = male; 1=female). If the Participant age is a scalar, then this must be a scalar. If the Participant age is a list, then this can be a scalar (i.e., use the same sex for all ages) or it must be a list with the same number of values as there are participant ages. If the Participant age is a matrix, then this can be a scalar (use the same sex for all ages) or it must be a matrix with the same number of rows and columns as there are for participant ages.
[5]	Beneficiary age(s) (integer, long, or double scalar, list (i.e., vector), matrix of ages, or blank when this parameter is not used). If the Participant's age is a scalar, then this can be a scalar, list or matrix. If the Participant's age is a list or a matrix, then this can be a scalar (use the same beneficiary age for all participant ages) or if it's a list/matrix, it must have the exact shape and number of values as the participant ages. Note that the beneficiary age settings in the library (see [0]) define the valid age range and Include Monthly ages defines whether fractional ages are allowed.
[6]	(not used) , the opposite of participant's sex will be assumed; this will be Beneficiary sex ; currently the only valid value is "", MV, vbNULL, vbEMPTY)
[7]	Interest rate(s) (scalar, list, or blank when you want to set the settings in the named Administration Factor) If this is a scalar, then it's a static rate (double). If it's a list, then the first element is a code (integer): 1=static rate, followed by a single rate (double); 2=variable forward rates, followed by an even number of duration (integer)/rate (double) pairs; 3=PBGC style rates, followed by 4 rates (doubles), the <i>immediate rate</i> , the <i>prior 7 years (K1)</i> , the <i>prior 8 years (K2)</i> , and the <i>all prior years (K3)</i> ; 4=segment-style rate, followed by 3 segment rates (doubles); 5=variable spot rates, followed by an even number of duration (integer)/rate (double) pairs.
[8]	Calculation year (integer or blank when you want to set the settings in the named Administration Factor) The calculation year is required for age by year of birth, dynamic and fully generational mortality tables.
[9]	not used (this will be the mortality assumptions; currently the only valid value is "", MV, vbNULL, vbEMPTY)

Return Value (Variant Array)	
[0]	Error code (long)
[1]	Error description (string) Notable Errors (also see Appendix A): 84 = Name: <i>xyz entry does not exist</i> 98 = parameter validation errors, e.g., Invalid Administration Factor name Administration Factor "xyz" has not been setup to handle annuity factors Invalid commutative function: <i>ddd</i> No payment forms are defined At least 2 payment forms must be defined to perform a Conversion Factor calculation. Requested payment form <i>pfname</i> is not defined in this Administration Factor Conversion factor requested but normal form has not been specified The primary's age must fall within the range <i>nn</i> to <i>mm</i> Monthly primary ages are NOT allowed by the Administration Factor library: <i>xyz</i>
[2]	Participant age(s) (integer, long, or double scalar, list (i.e., vector), or matrix of ages). Will be empty if participant age wasn't needed in the calculation.
[3]	Participant sex (integer, long, or double scalar, list (i.e., vector), or matrix of sex codes: 0 = male; 1=female). Will be empty if participant sex wasn't needed in the calculation.
[4]	Beneficiary age(s) (integer, long, or double scalar, list (i.e., vector), or matrix of ages). Will be empty if beneficiary age wasn't needed in the calculation.
[5]	Beneficiary sex (integer, long, or double scalar, list (i.e., vector), or matrix of sex codes: 0 = male; 1=female) Will be empty if beneficiary sex wasn't needed in the calculation.
[6]	Administration factors or commutation function results (double scalar, list or matrix)

Additional Methods

So far, this document has only discussed functions accessed by the PVal method. Additional methods are also available, as discussed below.

System Related

eng.ChangeWorkspace

Description (new in 3.19):

Change the version of ProVal utilized by the API for subsequent calls. For instance, useful to open a ProVal client utilizing the same version it was last opened by, without updating it to a newer version. This is illustrated in the C# code snippet below. The caller must know where the desired version of ProVal is installed. **NB!** After switching versions, API functions will act according to the selected ProVal version.

Argument

[0]	ProVal workspace name (e.g. "c:\WinTech\ProVal\PV317\ProVal.w3") (string)
------------	--

Return Value (None)

C# Code Snippet

```
RetVal = eng.PVCall("FILEOPEN", inputArray); // attempt to open client
errCode = (int)RetVal[0];
errDesc = (string)RetVal[1];
if (errCode == 35 || errCode == 37) // if client updated by older or newer version
{
    if (RetVal.Length == 3)
    {
        ver = (string)RetVal[2];
        clientVersion = Double.Parse(ver.Substring(0, 4));

        if (clientVersion == 3.18)
        {
            eng.ChangeWorkspace(@"C:\WinTech\ProVal\PV318\ProVal.w3");
            inputArray[0] = 1; // allow for client updates to this version
            RetVal = eng.PVCall("FILEOPEN", inputArray);
            errCode = (int)RetVal[0];
            errDesc = (string)RetVal[1];
        }
        else if (clientVersion == 3.17)
        {
            ...
        }
    }
}
```

Appendix A: Errors

Errors that apply to the ProVal API functions are listed below:

Error Codes and Descriptions	
-1	Invalid function name. Verify that the function name (passed as the first argument) in the PVal function is spelt correctly.
-2	ProVal API Licensing Requirements not met. Ensure that the API is adequately licensed to perform the function.
-3	Internal API Error. Contact ProVal Support for assistance.
-4	ProVal update available but not applied. Manually apply ProVal update to continue (refer to the ProVal Installation Guide readme.doc in the ProVal Installation Folder for details).
0	Success.
9	A path cannot be included with the database file name.
10	Incorrect number of arguments.
11	No client folder currently open.
12	Invalid ProVal database specified.
13	Invalid boolean argument (plus further message).
14	Invalid file name specification.
15	No database filename provided.
16	Append/Replace/Quit code must be A, R or Q.
17	Invalid code.
18	File already exists, quit without writing to it.
19	File is read-only or in use.
20	Invalid selection expression.
21	Error opening provided database file.
22	Invalid specification of field names.
23	Invalid library name.
24	File already exists. Enter a new name.
25	No records satisfy selection expression.

26	Database file cannot be modified because it is referenced.
31	Client Folder doesn't exist.
32	Client Folder is an invalid path.
33	Client Folder isn't a valid ProVal client folder.
34	Client Folder in use.
35	Client Folder needed updating, but UpdateFolderIfNecessary is False.
36	Unknown error opening client.
37	Client Folder has been updated by a newer version of ProVal.
38	Rights for File Open must be specified as MUSTWRITE, READWRITE or READONLY.
41	Invalid Record Layout specification.
42	CSV file does not exist.
43	Import validation error.
44	Data Import is not allowed into encrypted database files.
45	Invalid Merge/Update key field.
46	Duplicate keys found in the update file.
50	Valuation Sets have NOT been run in this project.
51	Asset & Funding Policy problems while validating Valuation Set.
52	Valuation Event problems while validating Valuation Set.
53	Error creating Valuation Set Exhibits.
54	Valuation Set does not exist.
55	Invalid valuation exhibit number.
56	Schedule of active data and/or distribution of inactive benefits not available.
57	Problems trying to write to the Access database file.
60	Valuation/Core Projection does not exist.
61	The Valuation cannot be run until you complete the following inputs.
62	Valuation/Core Projection has not been run.
63	Invalid date.
64	Valuation Assumptions/Projection Assumptions/Asset Funding Policy do not exist.
65	New name for Valuation Assumptions is already in use.

66	Valuation Assumptions are mismatched by funding/accounting type.
68	Row number for subtotals is invalid.
69	Row number for sensitivity is invalid.
70	Record layout does not exist.
75	Incorrect number of values, expecting "X" values for each Plan Constant
84	"Library name" entry does not exist
85	Incorrect number of items for library entry
86	New name for "Library name" is already in use
90	Asset data is not shaped correctly
91	Asset data has invalid first two rows
92	Asset data requires exactly one row of types X in column 2
93	Asset data has an incorrect outline string in column 3
94	Asset data has invalid values in column 4
95	Asset data has invalid values in column 5
99	Unknown error. Contact ProVal Support for assistance.
111	Valuation/Core Projection error (writes to message log).
121	Library Entry error (writes to message log).
201	Duplicate keys found in the database file that is being updated (can override).
211	Valuation/Core Projection has already been run (can override).
212	Requested Valuation/Core Projection name is already in use (can override).
213	Running this command could permanently change the Individual Results database (can override).
301	Problems with Import Schema (can override and writes to message log).
302	Error processing import file (can override and writes to message log).
303	Ambiguous coded field values that are similar to existing values (can override and writes to message log).
311	Running this command will erase the results of other objects (can override and writes to message log).