Dbank's Time Series Object Model

When Dbank is installed in Windows 9x/NT, the setup program installs a dynamic link library called "readdb.dll" the Windows system area. This library defines a time series object (COM) that can be used in Microsoft Visual Basic, Visual Basic for Applications, Power Builder, Visual C++, and Active Server Pages to manage Dbank time series.

Much like one would define a standard integer in Microsoft Visual Basic, Dbank's time series object enables a programmer to define a time series object with predefined attributes, rules and procedures. These attributes and procedures allow a programmer to create Dbank data banks, revise existing time series in a data bank, extract fundamental properties of series saved in the databank, and manipulate a set of time series as a unit.

The purpose of this manual is to provide a brief, but comprehensive description of Dbank's time series object model. The manual assumes that your base language is Visual Basic (VB) or Visual Basic for Applications (VBA).

1. Accessing Dbank's Time Series Object From VB/VBA

Before you can access Dbank's time series object model, you need to (a) install Dbank; and (b) activate a reference to the dynamic link library that actually defines the object ("readdb.dll", typically saved in the Windows system area).

From VB, click on "Project", then "References" and select "Read and Write Dbank Time Series Files" by checking the appropriate check box.

From VBA (e.g., Microsoft Excel), click on "Tools", followed by "References" and select "Read and Write Dbank Time Series Files".

2. Defining a Time Series Object From Visual Basic/Visual Basic for Applications

Dbank's time series object is called *"dbTimeSeries"*. Much like any other objects in Visual Basic, VB/VBA's Dim statement is used to declare a new time series object. Thus:

```
Dim X as New dbTimeSeries
      'Defines X to be a Dbank time series object
```

The following syntax is also valid VB/VBA:

```
Dim Y as dbTimeSeries ' "new" keyword is absent
                      'Defines Y to be an object variable 'that
                      can reference a Dbank time-'series object
```

However, this statement does not actually create a time series object; here "Y" can only refer to an existing time series object. Normally, statements of this type are followed by:

```
Set Y = X
```

which essentially provides another name for the time series object X.

3. Defining a Time-Series Object from Visual Script/Active Server Pages

Dbank's dbTimeSeries object can be used within active server pages. The following VB script creates a dbTimeSeries object on the server itself:

```
dim objTsd
set objTsd = Server.CreateObject("DbankEngine.dbTimeSeries")
```

4. Defining the Number of Observations

When you first define a time series, it has only one observation, and the value of this observation is automatically set to Dbank's missing value code. Use the "Nobs" property (attribute) of a series to change the number of observations in the time series. The following code increases the number of observations of X to 100:

```
Dim X as New dbTimeSeries
X.Nobs = 100 'Change observation count from 1 to 100
```

5. Reading/Writing Dbank Time Series Variables

A fully qualified time series name in Dbank has the following form (optional components are indicated using {}):

{<ServerName>}Database{:}{[Group]}SeriesName

where:

<ServerName> = Name of an accessible Microsoft SQL Server, e.g., <ORION>
Database = Name of the SQL or Access database to store the time series
Group = time series group or container
SeriesName = time series name

If <ServerName> is omitted, Dbank user the Microsoft Access engine to store time series. For example,

```
"c:\Program Files\Dbank32\example[ace]a"
```

implies that the time series "a" can be found in the Microsoft Access database

```
"c:\Program Files\Dbank32\example.mdb"
```

in the group called "[ace]". As another example,

```
"<ORION>example[ace]a"
```

implies that the time series "a" can be located in the database "example" that exists on the Microsoft SQL Server called "ORION". Again, the time series belongs to the group called "[ace]".

The following VB/VBA statements read a Dbank time series and write it to new data bank on a Microsoft SQL Server called "TSSQL":

```
Dim X as New dbTimeSeries
If X.Read("c:\Program Files\Dbank32\example[ace]a") Then
            X.Save("<tsSQL>newdb[ace]a")
End If
```

The "Read" method retrieves a time series from a physical database (which means reading the actual numerical data as well as its attributes, e.g. its title) into "X" and returns "True" if the read was successful. In this example, the "Save" method writes the same series to a new data bank called "newdb" that exists on the SQL server "tsSQL".

Note that the "Save" method automatically creates new data banks. A new databank is automatically created before the actual save takes place.

Accessing values of a time series is straightforward using the "DataValue" property of a series. This property allows you to retrieve and set the value of a series at a particular date (or offset). For example, the following code creates a trend variable starting at 1 with 100 observations. The time series frequency is quarterly, and starts in 1961:1.

```
Dim X as New dbTimeSeries
X.Frequency = "Q"
X.Nobs = 100
X.StartDate = "1961:1"
For j = 1 to X.Nobs : X.DataValue(j) = j : Next j
X.Save("example[test]trend")

'Now display 1968:4
Msgbox X.DataValue("1968:4")
```

A usual feature of the DataValue() method is that you can retrieve values of the time series at different frequencies of the native frequency. For example,

```
Msgbox X.DataValue("1968A")
```

will display the "annual" value of X for 1968. Conversion will occur on the fly, using the conversion method attached to the X object.

This completes the basics of defining Dbank time series objects in Visual Basic/Visual Basic for Applications, and reading and writing time series variables to data banks. The remaining sections of this guide provide a reference to all the properties and methods supported by Dbank's time series object. It what follows, we assume that the time series object name is "ts".

Acf() Method [Double]

Returns the j^{th} autocorrelation coefficient of the series.

Syntax:

ts.**Acf**(j)

Example

```
'Display the first 10 autocorrelations
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
    For j=1 to 10
        MsgBox X.Acf(j)
   Next j
End If
```

Adf() Method [Boolean]

Returns the augmented Dickey-Fuller statistic of the time series. The caller can control the number of lags included in the fitted regression, as well as the order of the polynomial trend.

Syntax:

ts.**Adf**(Lags, TrendOrder)

The **Adf** property has these parts:

Part	***Type***	***Description***
[Lags=0]	Variant	Number of lags in the fitted regression.
[TrendOrder=-1]	Variant	Order of the polynomial included in the fitted regression.

Example

```
Dim X as New dbTimeSeries
Dim MaxLags, TrendOrder, j
MaxLags = 10
TrendOrder = 0 'Constant term only
If X.Read("example[ace]a") Then
    For j=1 to 10: MsgBox X.Adf(j, TrendOrder): Next j
End If
```

BackTrim() Method [dbTimeSeries]

Removes or deletes observations from the end of a series. This method returns a new time series object (dbTimeSeries).

Syntax:

ts.**BackTrim**(Amount)

The **BackTrim** property has these parts:

Part	***Type***	***Description***
[Amount]	Variant	Number of (recent) observations to strip from the series, specifically Obs(N-Amount+1:N), where N is the beginning number of observations.

If the argument is omitted and there is at least one non-missing value in the series, **BackTrim** removes all the trailing missing values found in the series.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      MsgBox X.Nobs 'Display the number of observations
      X.BackTrim(5) 'Remove 5 most recent observations
      MsgBox X.Nobs 'Display observation count
End If
```

Browse() Method [Boolean]

Invokes Dbank's time series browser form, which displays time series attributes in a Window. The Browse form is invoked in a non-modal fashion by default.

Syntax:

ts.**Browse**(*[Optional LoadAsModal=False]*)

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      If X.Browse Then
         MsgBox "Browse operation successful."
      End if 'Display X's attributes using Dbank's browser
End If
```

ClearData() Method [Variant]

Sets all the current observation in the series to the missing value. The observation count is not affected.

Syntax:

ts.**ClearData**

Example

```
Dim X as New dbTimeSeries
If X.Read(“example[ace]a”) Then
      X.ClearData
      MsgBox X.MissingCount=X.Nobs ‘Displays true
End If
```

ConversionMethod() Property [Enum, tsConversionMethod]

Sets the conversion method to be used by default when converting a series to any frequency that would result in a smaller number of observations (i.e., consolidation occurs).

Syntax:

ts.**ConversionMethod** = *ConversionMethodCode*

The following conversion methods are supported:

Code (Enum)	Consolidation/Conversion Method
tsAverage	Average of the observations
tsFirst	Use first observation
tsMidPoint	Mid-point observation
tsLast	Last Observation
tsSum	Sum of observations
tsNoMethod	None; equivalent to “no” consolidation
tsMaximum	Maximum Observation
tsMinimum	Minimum Observation
tsBeginning	First valid (non-missing) observation
tsEnding	Last valid (non-missing) observation
tsRange	Observation Range
tsCount	Number of observations in the consolidation set
tsVariance	Variance of the observations
tsStDeviation	Standard deviation of the observations
tsAverageIgnoreMissing	Average of the observations (ignoring missing values)
tsFirstIgnoreMissing	Use first observation (ignoring missing values)
tsMidPointIgnoreMissing	Mid-point observation (ignoring missing values)
tsLastIgnoreMissing	Last Observation (ignoring missing values)
tsSumIgnoreMissing	Sum of observations (ignoring missing values)
tsNoMethodIgnoreMissing	None; equivalent to “no” consolidation (ignoring missing values)
tsMaximumIgnoreMissing	Maximum Observation (ignoring missing values)
tsMinimumIgnoreMissing	Minimum Observation (ignoring missing values)
tsBeginningIgnoreMissing	First valid (non-missing) observation (ignoring missing values)
tsEndingIgnoreMissing	Last valid (non-missing) observation (ignoring missing values)
tsRangeIgnoreMissing	Observation Range (ignoring missing values)
tsCountIgnoreMissing	Number of non-missing observations in the consolidation set
tsVarianceIgnoreMissing	Variance of Observations (ignoring missing values)
tsStDeviationIgnoreMissing	Standard deviation of observations (ignore missing values)

Example

```
Dim X as New dbTimeSeries
If X.Read(“example[ace]a”) Then
      X.ConversionMethod = tsLast
      X.Save
End If
```

Convert() Method [dbTimeSeries]

The Convert() method alters the frequency of a series. It can either can reduce (i.e., consolidate) or expand (i.e., interpolate) the number of observations in a series. This method returns a dbTimeSeries object.

For example, if an "annual" series is converted to "quarterly", the Convert method can interpolate using any one of the following valid interpolation procedures:

Linear Interpolation
Cubic-Spline
Series Expansion
Repeat
Geometric Interpolation.

Dbank can position the original data at the beginning, middle, or end of a particular period before interpolation occurs.

Note that, by default, missing values will be ignored during consolidation. You can change this behavior by setting the optional "IgnoreMissing" parameter to True.

Syntax:

ts.**Convert**(TargetFrequency, *[ConversionMethod]*, *[InterMethod=tsSpline]*, *[IgnoreMissing=True]*)

The **Convert** method has these parts:

Part	***Type***	***Description***
TargetFrequency	String	Target frequency: "A"=Annual; "H"=Bi-Annual; "Q"=Quarterly; "M"=Monthly; "W"=Weekly; "F"=Financial (5-day); "S"=Financial (6-day); "D"=Financial (7-day)
[ConversionMethod]	Variant	Actual conversion method to use (Enum); If omitted, conversion method uses the "ConversionMethod" attribute of the series (if any).
[InterMethod]	Variant	Interpolation method to use if interpolation is required. InterMethod defaults to cubic spline interpolation if it is omitted from the argument list.
[IngnoreMissing]	Boolean	Determines whether missing values are to be ignored during consolidation. The default setting is True.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
     Msgbox X.Frequency 'Quarterly series
      X.Convert("A", tsAverage)
      'Move from quarterly to annual
      MsgBox X.Frequency 'Annual series
End If
```

ConvertName() Property [String]

Returns the full name of the default conversion method saved with the series. The result is a string reflecting the proper name of the default conversion method.

Syntax:

ts.ConvertName

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.ConvertName 'Displays "Average"
End If
```

CopyMe() Method [Boolean]

Creates a copy of an existing time series object. You can control whether the method also copies the data values associated with the time series using the optional "SkipDataCopy" parameter. CopyMe() returns "True" if successful.

Syntax:

ts.CopyMe(Target as dbTimeSeries**,** *[SkipDataCopy=False]***)**

Example

```
Dim X as New dbTimeSeries
Dim Y as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.CopyMe(Y)
End If
```

CopyMyData() Method [Boolean]

Copies the data values embedded in a time series to an array of double variables. The array of double values may optionally contain 3 dimensions. Make sure you set the optional parameter "HasMoreThanOneIndex" to "True" if you pass such an array to the procedure. The method will re-dimension the array if necessary so that it can contain all the data stored within the time series object. The CopyMyData() method is very useful for speeding up data manipulations that do not involve time arithmetic.

Syntax:

ts.CopyMyData(DataArray() as Double, Nobs, *[HasMoreThanOneIndex=False]***)**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
   ReDim DataValues(1 to X.Nobs)
   Call X.CopyMyData(DataValues(),X.Nobs)
   'DataValues() will now have a copy of all the values in X
End If
```

Correlate() Method [Double]

Returns the correlation coefficient between two series of the same frequency. You can optionally the covariance between two time series of the same frequency by setting the optional "ReturnCovariance" parameter to "True"

Syntax:

ts.**Correlate**(y as dbTimeSeries,*[ReturnCovariance=False]*)

Example

```
Dim X as New dbTimeSeries
Dim Y as New dbTimeSeries
Dim Ok as Boolean
Ok = X.Read("example[first.bp]bpeir")
Ok = Ok And Y.Read("example[first.bp]bpemsr")
Ok = (X.Frequency = Y.Frequency)
If Ok Then
    MsgBox X.Correlate(Y)'Display the correlation coefficient
    MsgBox X.Correlate(Y, True)
    'Display the covariance
```

Cost() Property [Long]

Sets (or returns) cost of updating a series from a remote site. This attribute is useful for primarily for data providers who use Dbank to distribute their data over the Internet. Note that this attribute is not normally used for standard time series work.

Syntax:

ts.**Cost**(*[Value as Long]]*

The **Cost** method has these parts:

Part	***Type***	***Description***
Value	Long	Cost in cents to update the series.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      X.Cost = 100 'Series costs $1 to update
      X.Save
End If
```

Covariance() Method [Double]

Returns the covariance between two series of the same frequency.

Syntax:

ts.**Covariance**(y as dbTimeSeries)

Example

```
Dim X as New dbTimeSeries
Dim Y as New dbTimeSeries
Dim Ok as Boolean
Ok = X.Read("example[first.bp]bpeir")
Ok = Ok And Y.Read("example[first.bp]bpemsr")
Ok = (X.Frequency = Y.Frequency)
If Ok Then
    MsgBox X.Covariance(Y)'Display the covariance
```

Created() Property [Double]

Returns the actual time that the series was added to the data bank (i.e., first saved to the data bank). The return value is a real value in Microsoft's date serial format. The integer part implies the year, month, and day that the series was created; the fractional part implies an hour and second.

Syntax:

ts.Created

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox Format(X.Created,"dddddd")
      'Display creation date using Windows long-date format
End If
```

CreatedBy() Property [String]

Returns the name of the account that added the time series to the databank.

Syntax:

ts.CreatedBy()

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.CreatedBy
End If
```

Databank() Property [String]

Sets or returns the fully qualified name of the data bank to which the series might be saved and/or updated.

Syntax:

ts.Databank [= value]

The **DataBank** property has these parts:

Part	***Type***	***Description***
[Value]	String	Data bank (or home) of the series.

Example

```
Dim X as New dbTimeSeries
If X.Read("c:\dbank32\example[first.bp]bpeir") Then
      MsgBox X.Databank 'Displays "c:\dbank32\example"
      X.DataBank = "c:\dbank32\newexample"
      MsgBox X.DataBank 'Displays "c:\dbank32\newexample"
      X.Save 'Saves X to c:\dbank32\newexample[first.bp]
End If
```

DataSource Property [String]

Sets or returns the data source of a series. The "data source" can be an arbitrary string that describes the actual source of the data.

Syntax:

ts.**DataSource** [= value]

The **DataSource** property has these parts:

Part	Type	Description
[Value]	String	Data source of the time series

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      X.DataSource = "DRI/Basic Economics"
      MsgBox X.DataSource 'Display the new data source
      X.Save 'Update the series
End If
```

DataValue Property [Double]

Sets or returns the value of a data point in a series. You must supply a valid time series index (integer or string) to successfully retrieve a particular data value from a time series object.

The DataValue() method can also be used to retrieve values of the series at frequencies other than the series actual frequency (as recorded in the time series object, i.e., returned by the "Frequency" method). For example, the DataValue() method can be used to retrieve annual value of a quarterly series. In this case, Dbank's time series object will perform an automatic conversion in memory and then return the appropriate annual value.

The conversion method attribute of the source series needs to be set for this feature to work properly.

Syntax

ts.**DataValue**(Index) [= value]

The **DataValue** method has these parts:

Part	***Type***	***Description***
Index	Variant	String or value defining the index of the observation
[Value]	Double	Observation value

Example 1

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      X.DataValue(10) = 0.5  'Change 10th observation to 0.5
      X.Save("example[ace]a_new")
End If
```

Example 2

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.DataValue("1975:1") 'Displays 6113
      MsgBox X.DataValue(25) 'Displays 6113
      X.DataValue("1975:1") = 6500
      MsgBox X.DataValue("1975:1") 'Displays 6500
      MsgBox X.DataValue(25) 'Displays 6500
End If
```

Remarks

- When a string is used to index an observation, it must be in the format understood by the series. This format basically depends on the frequency of the time series, and, in the case of weekly and financial data in particular, the date settings for Windows 9x/NT specified in Control Panel. The following table illustrates some of the string formats supported:

Table 2: Dbank Date Serial Formats

Frequency	Description	Index Format
"A", "Yn"	Annual or Higher	"yyyy"
"H"	Half-yearly	"yyyy:1" or "yyyy:2"
"Q"	Quarterly	"yyyy:p", p = 1, 2, 3, or 4
"R"	Every 4 months	"yyyy:p", p = 1,2,3
"O"	Every 2 months	"yyyy:p", p = 1,2,3,4,5, or 6
"M"	Monthly	"yyyy:p", $1 \le p \le 12$
"F"	Financial	"yy/mm/dd", depending on date setting in Windows Control Panel
"B"	Bank-Week	"yy/mm/dd", depending on date setting in Windows Control Panel
"S"	Financial (6-day)	"dd/mm/yy", depending on date setting in Windows Control Panel
"D"	Financial (7-day)	"mm/dd/yy", depending on date setting in Windows Control Panel

- When accessing financial data, be sure to be consistent with the current date settings in Windows control panel.

- You can specify a target frequency by adding the target frequency letter to the end of the date serial. Thus, MsgBox X.DataValue("1999A") will return the annual value of a time series in 1999 (using the internal conversion method to compute the annual value), *irrespective* of its actual frequency.

DateSerial() Property [String]

Returns the time-stamp of an observation in Dbank format, which depends on the series' frequency and your Window 9x/NT settings for dates (see Table 2, Page 20).

Syntax:

ts.**DateSerial**(Index)

The **DateSerial** method has these parts:

Part	***Type***	***Description***
Index	Long	Array index of an observation.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.DateSerial(25) 'Displays "1975:1"
      X.DataValue(X.DateSerial(25)) = 6500
      MsgBox X.DataValue("1975:1") 'Displays 6500
      X.Save
End If
```

Day Property [Long]

Returns a long integer representing the day of the month that an observation belongs to.

If the frequency of the series is less than weekly (i.e., yearly, annual, half-yearly, quarterly, or monthly), this function returns a integer that depends on the series' "ConversionMethod" attribute.

Syntax

ts.**Day**(Index)

The **Day** method has these parts:

Part	***Type***	***Description***
Index	Variant	String or value defining the index of the observation.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      X.ConversionMethod = tsAverage
      MsgBox X.Day(25) 'Returns 15
      X.ConversionMethod = tsFirst
      MsgBox X.Day(25) 'Returns 1
      X.ConversionMethod = tsFirst
      MsgBox X.Day(25) 'Returns last day of the month
      X.Save
End If
```

Remark:

When a string is used to index an observation, it must be in the format understood by the series. This format depends on the frequency of the series, and, for financial and weekly data in particular, the active dates format settings for Windows (see Regional Settings in Control Panel). See Page 20, Table 2 for more information.

Decimals() Property [Long]

Sets or returns the number of decimal points to show whenever a series is displayed in Dbank's viewer. The numerical values of the data are unaffected.

Syntax:

ts.**Decimals** [= value]

The **Decimals** method has these parts:

Part	***Type***	***Description***
Value	Long	Number of decimals

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      X.Decimals = 2
      X.Save 'Update the series
End If
```

Delete() Method [Boolean]

Removes a series from a data bank. Returns "True" if successful.

Syntax:

ts.Delete

Example

```
Dim X as New dbTimeSeries
X.Name = "example[first.bp]bpeir"
If X.Exists Then
      If X.Delete Then
            MsgBox Ucase$(X.Name) & " deleted successfully."
      End If
End If
```

DoornickHansen() Method [Double]

Returns the Doornick-Hansen statistic for the null hypothesis of normality of the time series. It possesses a Chi-Square distribution with 2 degrees of freedom under the null hypothesis of normality.

Syntax:

ts.**DoornickHansen**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.DoornickHanse
      X.Save
End If
```

Drop() Method

Drops (strips) all observations in a series *after* a given date/index.

Syntax:

ts.**Drop**(Index)

The **Drop** method has these parts:

Part	***Type***	***Description***
Index	Variant	Observation index.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      X.Drop("1975:1") 'Removes all data after 1975:1
      X.Save
End If
```

Remark:

When a string is used to index an observation, it must be in the format understood by the series. This format depends on the frequency of the series, and, for financial and weekly data in particular, the active dates format settings for Windows (see Regional Settings in Control Panel). See Page 20, Table 2 for more information.

Edit() Method [Boolean]

Invokes Dbank's time series editor, which allows you to edit the attributes and data stored within a time series object. Dbank's Edit form is invoked in a non-modal fashion by default.

Syntax:

ts.**Browse**(*[Optional LoadAsModal=False]*)

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      If X.Edit Then
         MsgBox "Edit operation successful."
      End if
End If
```

ExcelData() Method [Variant]

This method returns an array of variants containing the values embedded in the time series. The data are formatted so that they can be copied to Excel cell objects without further translation. In particular, the variant array will represent missing values as empty strings. Doing so causes the least havoc when transferring double values to Excel.

Syntax:

ts.ExcelData()

Example

```
Dim X as New dbTimeSeries
X.Name = "example[first.bp]bpeir"
If X.Exists Then
   Dim ExcelData as Variant
   ExcelData = X.ExcelData()
   MsgBox ExcelData(1) 'First value of X
End If
```

Exists() Property [Boolean]

Tests whether a time series exists in the named database. Returns “True” if successful; “False” otherwise.

Syntax:

ts.Exists

Example

```
Dim X as New dbTimeSeries
X.Name = “example[first.bp]bpeir”
If X.Exists Then
      If X.Delete Then
            MsgBox Ucase$(X.Name) & “ deleted successfully.”
      End If
End If
```

FindAllMyAliases() Method [Variant]

This method returns an array of variants containing the fully qualified names of all the series that point to the named time series. The “Count” parameter returns the number of alias variables found by the method.

Syntax:

ts.FindAllMyAliases(Count as Long**)**

Example

```
Dim X as New dbTimeSeries
X.Name = “example[first.bp]bpeir”
If X.Exists Then
   Dim Aliases as Variant, Count as Long
   Aliases = X.FindAllMyAliases(Count)
   If Count > 0 Then MsgBox Aliases(Count)
End If
```

FirstDay() Property [Long]

Sets (or returns) the first day of a time series. This property will affect the starting date of weekly and financial (5-day, 6-day, and 7-day) series. The frequency of the series should be set before using the FirstDay method.

Syntax:

ts.**FirstDay** [= value]

The **FirstDay** method has these parts:

Part	***Type***	***Description***
Value	Integer	First day of the month

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      X.FirstDay = 1: X.Save("example[new]bpeir")
End If
```

FirstPeriod() Property [Long]

Sets (or returns) the first period (month, quarter, half-year) of a series. This setting affects the starting date of a series. The frequency of the series must be set before using the **FirstPeriod** method.

Syntax:

ts.**FirstPeriod** [= value]

The **FirstPeriod** method has these parts:

Part	***Type***	***Description***
Value	Integer	First period of the year

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      X.FirstPeriod = 2: X.Save("example[new]bpeir")
End If
```

FirstQuartile Property [Double]

Returns the first-quartile of a series.

Syntax:

ts.**FirstQuartile**

Example

```
Dim X as New dbTimeSeries
If X.Read(“example[first.bp]bpeir”) Then
      MsgBox X.FirstQuartile ‘Display first quartile
End If
```

FirstValidIndex Property [Long]

Returns an integer that indicates the position or array index of the first valid (i.e., non-missing) observation in a series.

Syntax:

ts.**FirstValidIndex**

Example

```
Dim X as New dbTimeSeries
If X.Read(“example[first.bp]bpeir”) Then
      For j = X.FirstValidIndex to X.LastValidIndex
            If X.IsValidObs(j) Then
                  X.DataValue(j) = Log(X.DataValue(j))
                  ‘Convert series to logs
            End If
      Next j
      X.Save ‘Update the series
End If
```

FirstYear Property [Long]

Sets (or returns) the first year of a series. This setting affects the starting date of a series. The frequency of the series must be set before using the **FirstYear** method.

Syntax

ts.**FirstYear** [= value]

The **FirstYear** method has these parts:

Part	***Type***	***Description***
Value	Integer	First year

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      X.FirstYear = 1990
      X.Save("example[new]bpeir")
End If
```

FiscalOffset() Property [Long]

Sets or returns the "fiscal offset" of the time series. The fiscal offset is used to automatically lead/lag a time series the current number of places so that it returns the correct data value for a given fiscal (as distinct from calendar) date. For example, setting the fiscal offset to 3 for monthly data will lead the monthly data values by 3 places ***before*** returning the value associated with a given date (which in this case is often called the fiscal date).

Syntax:

ts.**FiscalOffset** [= NumberOfPeriods]

The **FiscalOffset** method has these parts:

Part	***Type***	***Description***
NumberOfPeriods	Long	Number of places to lead the series (specify a negative value for lag).

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
   X.FiscalOffset = 2: X.Save : End If 'Update the series
```

FiscalValue() Method [Double]

Returns the value of a data point in a series at a given fiscal (rather than calendar) date. You must supply a valid time series index (integer or string) to successfully retrieve a particular fiscal value from the time series object.

The **FiscalValue()** method can also be used to retrieve values of the series at frequencies other than the series actual frequency (as recorded in the time series object, i.e., returned by the "Frequency" method). For example, the **FiscalValue()** method can be used to retrieve annual fiscal value of a quarterly series. In this case, Dbank's time series object will perform an automatic conversion in memory and then return the appropriate annual value.

The fiscal offset of the source series needs to be set to a non-zero value for **FiscalOffset** to return values that are, for the same index, different from the **DataValue()** method. The default setting of **FiscalOffset** is zero.

FiscalValue() accepts the same arguments as the **DataValue()** method.

Footnote() Method [String]

Dbank supports observation footnotes. Observation footnotes allow you to add descriptive text to a particular observation. Any number of footnotes can be attached to a particular observation. However, each footnote must be given a unique name.

This method returns (or sets) the caption associated with a footnote at a particular date. You must supply a valid date index and the name of the footnote (typically a string) to retrieve the footnote entry.

Syntax:

ts.**Footnote**(DateIndex, FootNoteName) [= FootNoteEntry]

The **Footnote** method has these parts:

Part	***Type***	***Description***
DateIndex	Variant	String or value defining the index of the observation
FootNoteName	Variant	Footnote identifier.

Example

```
Dim X as New dbTimeSeries
If X.Read(“example[ace]a”) Then
   MsgBox X.FootNote(1,”First”) ‘Retrieve the footnote called “A”
   ‘Now update the footnote
   X.Footnote(1,”First”) = “This is my first footnote.”
End If
```

Frequency Property [String]

Sets or returns the frequency of a series. Dbank supports the following frequencies:

- Yearly ("Yn") [e.g., "Y2" means one data point every two years]
- Annual ("A")
- Half-yearly ("H")
- Every three months ("R")
- Every two months ("O")
- Monthly ("M")
- Quarterly ("Q")
- Weekly ("Wn"), where n is the starting day of the week (1 = Sunday)
- Bank-Week ("B"), in which observations occur 4 times per month, and on the 8, 15, 22, the last day of the month.
- 5-day Financial ("F")
- 6-day Financial ("S")
- 7-day Financial ("D")

Syntax:

ts.**Frequency** [= value]

The **Frequency** method has these parts:

Part	***Type***	***Description***
Value	String	Frequency code

Example

```
Dim X as New dbTimeSeries
X.Nobs = 100
X.Frequency = "Q"
X.StartDate = "1961:1"
For j = 1 to X.Nobs
      X.DataValue(j) = j^2
Next j
X.Save("example[test]trend_squared")
```

FrequencyCount Property [Long]

This method returns the frequency count of a series. This property is used only for series with yearly ("Y") frequency. It determines the number of years between observations. For example, the frequency count of annual data is one.

Syntax:

ts.**FrequencyCount** [= value]

The **FrequencyCount** method has these parts:

Part	***Type***	***Description***
Value	Long	Observation interval (must be positive)

Example

```
Dim X as New dbTimeSeries
X.Nobs = 100
X.StartDate = "1961"
X.Frequency = "Y3"  'Expect an observation every 3 years
For j = 1 to X.Nobs
      X.DataValue(j) = j^2
Next j
MsgBox X.FrequencyCount  'Displays "3"
```

FrontTrim() Property [dbTimeSeries]

Removes observations from the beginning of a series, and returns a dbTimeSeries object with the result.

Syntax:

ts.**FrontTrim**(Amount)

The **FrontTrim** property has these parts:

Part	***Type***	***Description***
[Amount]	Variant	Number of observations to strip from the start of the series. Can also be specified as a date, in which case the method removes all the observations before the indicated date.

If the argument is omitted, **FrontTrim** removes the leading missing values in the series.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      MsgBox X.Nobs 'Display the number of observations
      X.FrontTrim(5) 'Remove 5 starting observations
      MsgBox X.Nobs
      'Display observation count (95)
End If
```

FullName() Property [String]

Returns the fully qualified name of a series. Fully qualified names contain the data bank, group, and short name of the series in the following format:

databank[group{.sub_group{.sub_sub_group}}]short_name

Syntax:

ts.**FullName**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      MsgBox X.FullName 'Display full name of the series
      ' In this case, "example[ace]a"
End If
```

Remarks

- Series names and sub-group names can each be up to 64 characters long.
- Spaces are not permitted in series names, but allowed in group names.
- Periods (".") can be used in the series short name.
- There is no restriction on the depth of the data bank group tree. Simply separate each level of the tree by a period (".") . For example, "a.b.c" defines the group "c" which belongs to "b" (its parent). The group "b", in turn, belongs to "a".

GeometricMean() Property [Double]

Returns the geometric mean of a series.

Syntax:

ts.**Geometric Mean**

Example

```
Dim X as New dbTimeSeries
If X.Read(“example[first.bp]bpeir”) Then
      MsgBox X.GeometricMean ‘Display geometric mean
End If
```

GetIndex() Method [Long Integer]

Returns a positive integer indicating the position (or array index) of the observation in the series. The GetIndex() method accepts a variant/string argument that represents a valid date to Dbank and returns a long integer.

Syntax:

ts.**GetIndex**(DateSerial)

The **GetIndex** property has these parts:

Part	***Type***	***Description***
DateSerial	Variant	Dbank date string

Example

```
Dim X as New dbTimeSeries
If X.Read(“example[first.bp]bpeir”) Then
      MsgBox X.GetIndex(“1971:3”) ‘Displays 27
End If
```

GetTimeSeriesAttributes() Method [Boolean]

Populates a dbTimeSeries object with all of the attributes of a series without reading the observations. This function is provided to enhance the speed of certain routines that do not need access to the actual data saved with the series.

Syntax:

ts.**GetSeriesAttributes**(TimeSeriesName)

The **GetSeriesAttributes** method has these parts:

Part	***Type***	***Description***
TimeSeriesName	Variant	Fully qualified time series name

Example

```
Dim X as New dbTimeSeries
If X.GetTimeSeriesAttributes(“example[ace]a”) Then
      MsgBox X.GroupName ‘Displays “[ace]”
      MsgBox X.Frequency ‘Displays “N”
      MsgBox X.StartDate ‘Displays 1
      Msgbox X.Nobs      ‘Displays 100
End If
```

Remarks

- This function should be used with extreme caution. Because it does not read the observations of the time series, it is easy to accidentally set all the data in the series to missing in the actual data. The following code snippet will essentially destroy all the non missing data in the series:

```
Dim X as New dbTimeSeries
If X.GetTimeSeriesAttributes(“example[ace]a”) Then
      MsgBox X.Group ‘Display [ace]
      X.Clear
      X.Save ‘Clears the data (accidentally?).
      ‘Observation count remains 100 (but all are missing)
End If
```

Group Property [String]

Sets or returns the group name of the series. The format of the group name is:

"[group{.sub_group{.sub_group}}]"

For example, "a.b.c.d", or simply "a". Thus periods (".") separate levels in the tree.

Syntax:

ts.**GroupName** [= value]

The **GroupName** property has these parts:

Part	***Type***	***Description***
Value	Variant	String representing a group in the data bank.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      MsgBox X.GroupName 'Displays "[ace]"
      'Now change the group name to [test]
      X.GroupName = "[ace.first.second.not_practical]"
      MsgBox X.GroupName
      'Displays "[ace.first.second.not_practical]"
      X.Save
      '  Saves the series to
      ' "example[ace.first.second.not_practical]a"
End If
```

Remarks

- Group (and sub-group) names can each be up to 64 characters long.
- Spaces are not permitted in group names.
- There is no restriction on the depth of the data bank group tree. Simply separate each level of the tree by a period (".") . For example, "a.b.c" defines the group "c" which belongs to "b" (its parent). The group "b", in turn, belongs to "a".

HarmonincMean() Property [Double]

Returns the harmonic mean of a series.

Syntax:

ts.**HarmonicMean**

Example

```
Dim X as New dbTimeSeries
If X.Read(“example[first.bp]bpeir”) Then
      MsgBox X.HarmonicMean ‘Display harmonic mean
End If
```

HasFootNotes() Property [Boolean]

This property returns “True” if any of the values in the time series object have footnotes.

Syntax:

ts.**HasFootNotes**

Example

```
Dim X as New dbTimeSeries
If X.Read(“example[first.bp]bpeir”) Then
      If X.HasFootNotes Then
         MsgBox Ucase(X.FullName) & “ has some footnotes.”
      End If
End If
```

Histogram() Method [Boolean]

Returns a table of ordinates that a useful in creating an “optimal” histogram plot for a series. The routine returns the he optimal number of bins, the lower bound of the first bin, and the optimal increment.

Syntax:

ts.**Histogram**(Bins(), BinCount, LowerBound, Increment)

The **Histogram** method has these parts:

Part	***Type***	***Description***
Bins()	Long	Array of long integers (will be re-dimensioned) containing the count for each collection bin in the histogram.
BinCount	Long	The number of bins in Bins() (returned).
LowerBound	Double	Lower bound for the first bin in Bins() (returned).
Increment	Double	Increment for each bin (returned).

Example

```
ReDim Bins(1) as Long
Dim BinCount as Long
Dim LowerBound as Double
Dim Increment as Double
Dim X as new dbTimeSeries

If X.Read(“example[ace]a”) Then
      Call X.Histogram(Bins(), BinCount, LowerBound, Increment)
      Msgbox BinCount  ‘Displays 10
      MsgBox Bins(BinCount)  ‘Displays 2
      MsgBox LowerBound ‘Displays -2
      MsgBox Increment ‘Displays 0.5
End If
```

Index() Method [dbTimeSeries]

Generates a series index (sequence starting with 1 and increasing by 1 each period).
The length of the index variable is determined entirely by the argument to index, which must be positive.

Syntax

ts.**Index**(Length)

The **Index** method has these parts:

Part	***Type***	***Description***
Length	Long	Length of series (or number of observations)

Example

```
Dim X as New dbTimeSeries
X.Start = 1
X.Index(100)
X.Title = “Time Trend, starting with 1”
X.Save(“example[trend]index”)
```

InternetAddress() Property [Attribute]

Dbank can update series from a remote site. The **"InternetAddress"** property sets or returns the remote address of the update file (which must be in Dbank's FREE format; see the user guide for specifics). The remote address takes the following format:

Remote_Server_Name:Fully_Qualified_Name

For example, "dbank:\example\a.fre", where "dbank" represents an alias for the actual address of the server on your network or the internet.

Syntax:

ts.**InternetAddress** [= value]

The **InternetAddress** method has these parts:

Part	***Type***	***Description***
Value	String	Remote address of the update file.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      X.InternetAddress = "dbank:/example/a.fre"
      'Unix FTP server.
      X.Save
End If
```

Interpolate() Method [dbTimeSeries]

This method expands a series to a higher frequency and then replaces the resulting missing values in the series with estimated (or interpolated) values. Dbank supports eleven distinct interpolation methods. For each of these methods, you can control whether the known values of the series are positioned at the beginning, middle, or at the end of the corresponding period in the resulting series (which has a higher frequency). If you do not specify the interpolation method explicitly in the function call, Dbank uses a cubic spline to interpolate. Please refer to Table 3 for a complete list of all the interpolation method available in Dbank.

Syntax:

ts.**Interpolate**(TargetFrequency, *[InterpolationMethod=tsSpline]*)

The **Interpolate** method has these parts:

Part	***Type***	***Description***
TargetFrequency	String	Required (or target) frequency for the new series.
InterpolationMethod	Integer	Interpolation method to use. Defaults to tsSpline.

Example:

```
Dim X as New dbTimeSeries
Dim Y as dbTimeSeries
If X.Read(“example[ace]a”) Then
     Set Y = X.Interpolate(“D”) ‘Interpolate to daily frequency
      Y.Save(“example[ace]aD”)  ‘Save the result
End If
```

InterpolationMethod() Property [Enum: InterpolationMethods]

This method sets the default interpolation method to use when interpolation is required to perform a time series operation. Dbank supports eleven distinct interpolation methods. For each of these methods, you can control whether the known values of the series are positioned at the beginning, middle, or at the end of the corresponding period in the resulting series (which has a higher frequency). Please refer to Table 3 for a complete list of all the interpolation method available in Dbank. Note that the default interpolation method for all time series is cubic spline.

Syntax:

ts.**InterpolationMethod** [= IntMethod]

The **InterpolationMethod** method has these parts:

Part	***Type***	***Description***
IntMethod	Integer	Enum of type InterpolationMethods

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
     'Alter the default interpolation method.
      X.InterpolationMethod = tsRepeat
      X.Save
End If
```

IsAbsent() Property [Boolean]

Determines whether an observation index is outside the existing sample of the series.

Syntax:

ts.**IsAbsent**(DataIndex)

The **IsAbsent** property has these parts:

Part	***Type***	***Description***
DataIndex	Variant	String or value defining the index of the observation

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      MsgBox X.IsAbsent(0)
      'Displays True, because actual sample is from 1 to 100
     MsgBox X.IsAbsent(5) 'Displays False
End If
```

Remarks

- When a string is used to index an observation, it must be in the format understood by the time series. This format depends on the frequency of the time series, and, for weekly and financial data, the date settings specified Control Panel. See Table 2, Page 20, for specific details.

IsAConstant() Property [Boolean]

Returns "True" only if the series is a constant (i.e., all the observations have the same value).

Syntax:

ts.**IsAConstant**

Example

```
Dim X as New dbTimeSeries
X.Nobs = 100
For j = 1 to 100: X.DataValue(j) = 1: Next j
MsgBox X.IsAConstant 'Displays True
```

IsALink() Property [Boolean]

Returns “True” only if the series is an alias or pointer to another series.

Syntax:

ts.**IsALink()**

Example

```
Dim X as New dbTimeSeries
X.Nobs = 100
X.FullName = “example[ace]a”
If X.IsALink() Then
   MsgBox Ucase(X.FullName) & “ is a link variable”
End If
```

IsAScalar() Property [Boolean]

Returns “True” only if the series is a scalar variable. To be a scalar variable, the series must have a frequency of “N” (or none) and it must have only one observation. It must also have a start date of 1.

Syntax:

ts.**IsAScalar()**

Example

```
Dim X as New dbTimeSeries
X.FullName = “example[ace]a”
X.Read
If X.IsAScalar() Then
   MsgBox Ucase(X.FullName) & “ is a scalar”
End If
```

IsAMakeExpression() Property [Boolean]

Returns "True" only if the series is a make expression.

Syntax:

ts.**IsAMakeExpression()**

Example

```
Dim X as New dbTimeSeries
X.FullName = "example[ace]a"
If X.IsAMakeExpression() Then
   MsgBox Ucase(X.FullName) & " is a make expression"
   MsgBox "The Make expression is " & X.MakeString
End If
```

IsValidObs() Property [Boolean]

Tests whether an observation is valid (i.e., non-missing and within the current sample).

Syntax:

ts.**IsValidObs**(DataIndex)

The **IsValidObs** property has these parts:

Part	***Type***	***Description***
DataIndex	Variant	String or value defining the index of the observation

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      MsgBox X.IsValidObs(0)
        'Displays False, because sample is from 1 to 100
     MsgBox X.IsValidObs(5)
       'Displays True; actual value is 1.588419
End If
```

Remarks

- When a string is used to index an observation, it must be in the format understood by the series. This format depends on the frequency of the series, and, for weekly and financial data in particular, the date settings specified Control Panel. See Table 2, , for specific details.

JarqueBera() Property [Double]

Returns the Jarque-Bera test statistic, which tests for normality of a series.

Syntax:

ts.**JarqueBera**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.JarqueBera 'Display Jarque-Bera
End If
```

Key Property [Long]

Returns internal ID number of (which is automatically generated) of the series in the databank. This name is unique for a given data bank, and is available ONLY if the series already resides in a data bank. The time series key may be changed for the life-span of the dbTimeSeries object. It cannot be updated in the database.

Syntax

ts.**Key [= value]**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.Key 'Display internal key of the series
End If
```

Kurtosis Property() [Boolean]

Returns the kurtosis of a series.

Syntax:

ts.**Kurtosis**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.Kurtosis 'Display kurtosis
End If
```

Lag() Method [dbTimeSeries]

Shifts the start date of the series forward by a given number of periods.

Lag Method

Syntax:

ts.**Lag**(Amount)

The **Lag** method has these parts:

Part	***Type***	***Description***
[Amount=1]	Long	Number of periods to lag (which can be negative).

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      MsgBox X.Lag.StartDate
        'Displays 2, because sample was 1 to 100
     MsgBox X.Lag(-3).StartDate
       'Displays –1, because sample is was 2 to 101
End If
```

Remarks

- The number of observations in the series is not affected by the lag operation. Only the start date (and end date) of the series changes.

LastDay Property() [Long]

Returns the last day of a series. This property affects the starting date of weekly and financial (5-day, 6-day, and 7-day) series. The frequency of the series must be set before using the **LastDay** method.

Syntax:

ts.**LastDay**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      Msgbox X.LastDay 'Displays 15
End If
```

LastPeriod() Property [Long]

Returns the last period (month, quarter, half-year) of a series. The frequency of the series must be set before using the **LastPeriod** method.

Syntax:

ts.**LastPeriod**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.LastPeriod 'Displays 1 (1st quarter, 1991)
End If
```

LastValidIndex() Property [Long]

Returns an integer that indicates the position of the last non-missing (or valid) observation in a series.

Syntax:

ts.**LastValidIndex**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      For j = X.FirstValidIndex to X.LastValidIndex
            If X.IsValidObs(j) Then
                  X.DataValue(j) = Log(X.DataValue(j))
                  'Convert series to logs
            End If
      Next j
      X.Save 'Update the series
End If
```

LastYear() Property [Long]

Returns the last year of a series. The frequency of the series must be set before using the **LastYear** method.

Syntax:

ts.**LastYear**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.LastYear 'Displays 1991
End If
```

Lead() Method [Long]

Shifts the start date of the series backward by a given number of periods.

Syntax:

ts.**Lead**(Amount)

The **Lead** method has these parts:

Part	***Type***	***Description***
[Amount=1]	Long	Number of periods to lag (can be negative).

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      MsgBox X.Lead.StartDate
        'Displays 0, because sample was 1 to 100
     MsgBox X.Lead(3).StartDate
       'Displays 3, because sample was 0 to 99
End If
```

Remarks

- The number of observations in the series is not affected by the lead operation. Only the start date (and end date) of the series changes.

LinkName() Method [String]

Provided the series being referred to is a link variable, this method returns the fully qualified name of the series that the link variable points to. Note that link variables can also point to links. As such, the method accepts a "Recursive" parameter to allow you to determine the physical series that the link chain points to.

Syntax:

ts.**LinkName***([Recursive=True])*

The **LinkName** property has these parts:

Part	***Type***	***Description***
[Recursive]	Boolean	If "True", the method "walks" thru the link chain to return the fully qualified name of the last series in the chain (a proper time series).

Example

```
Dim X as New dbTimeSeries
X.Name = "example[ace]a"
If X.IsALink() Then
   MsgBox "Actual series: " & X.LinkName()
End If
```

LinkTo() Method [Boolean]

Creates a pointer (or new name/alias) to an existing series in a data bank. Pointers to pointers are allowed.

Syntax:

ts.**LinkTo** [= value]

The **LinkTo** property has these parts:

Part	***Type***	***Description***
[Value]	String	Name/alias for the series (can include a group specification).

Example

```
Dim X as New dbTimeSeries
X.Name = “example[ace]a”
If X.LinkTo(“example[newname]link_to_a_in_ace”) Then
   MsgBox “Link to [ace]a created successfully”
End If
```

LongName() Property [String]

Sets or returns the “long name” a series. The long name can be any string that might provide a more descriptive name for the series.

Syntax:

ts.**LongName** [= value]

The **LongName** property has these parts:

Part	***Type***	***Description***
[Value]	String	Descriptive name of the series.

Example

```
Dim X as New dbTimeSeries
If X.Read(“example[ace]a”) Then
      X.LongName = ”100 Random Numbers”
      MsgBox X.LongName ‘Display the long name
      X.Save ‘Update the series
End If
```

Magnitude() Property [Long]

Sets or returns the magnitude of a series (in terms of powers of 10). In the current version of Dbank, this property does not have any effect on the manner in which a series is processed internally. You must enter the actual value of the series when creating a series.

Syntax:

ts.**Magnitude** [= value]

The **Magnitude** property has these parts:

Part	***Type***	***Description***
[Value]	String	Descriptive name of the series.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      X.Magnitude = 3 Random Numbers"
      MsgBox X.Magnitude 'Display the long name
      X.Save 'Update the series
End If
```

MakeAttributes() Property [Long Integer]

Returns a long integer that reflects the current settings for computing the make equation attached to the series. This property can also be used to set these attributes. However, care must be taken to ensure that the proper values are used to set the attributes.

Syntax:

ts.**MakeAttributes** [= value]

The **MakeAttributes** property has these parts:

Part	***Type***	***Description***
[Value]	Long	Integer reflect the current make attribute settings.

Example

```
Dim X as New dbTimeSeries
Dim mA as Long
If X.Read("example[ace]a") Then
  mA = X.MakeAttributes
  If (mA And tsMakeRefreshORead) = tsMakeRefreshOnRead Then
   MsgBox "Series make equation will be updated on read."
  End If
End If
```

Attribute	***Value***	***Description***
tsMakeIgnoreMissingSeries	2	Series Make will not flag a missing series as an error. The equation will be computed assuming that the series is all 0's or all 1's depending on the operation performed (e.g., +, -).
tsMakeSpreadsheetMode	4	Series Make will convert any missing values encountered during a make operation to either 0 or 1 depending on the operation performed (e.g., +, -).
tsMakeMergeMode	16	Merge (i.e., do not overwrite) the result into the existing series.
tsMakeTrimMode	32	Trim missing values at the beginning or end of the series before saving the result.
tsMakeRefreshOnRead	64	ReMake the series before returning the result of the dbTimeSeries Read method. This feature ensures that the series reflects the impact of all edits (and is often called "dynamic" update).
tsMakeDefaultMode	0	None of the above.

MakeIgnoreMissingSeries() Property [Boolean]

Sets or returns the current setting for the "IgnoreMissingSeries" attribute of a series when it comes to computing any make equation attached to the particular series. If "True", Series Make will "ignore" all the missing series in a make expression by setting its value to 0 or 1 depending on the current make operation. As such, Series Make will not fail if it encounters a missing series in the make expression. The default setting for this property is "False".

Syntax:

ts.**MakeIgnoreMissingSeries** [= value]

The **MakeIgnoreMissingSeries** property has these parts:

Part	***Type***	***Description***
[Value]	Boolean	Controls the "IgnoreMissingSeries" attribute of a series.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      X.MakeIgnoreMissingSeries = True
      X.Save 'Update the series
End If
```

MakeMergeMode() Property [Boolean]

Sets or returns the current setting for the "MergeMode" attribute of a series when it comes to computing any make equation attached to the particular series. If "True", Series Make will merge its result into the existing values of the series. In merge mode, a missing value will not overwrite a non-missing value. Also, the series will be expanded to accommodate a greater number of observations if and when necessary. The default setting for this method is "False".

Syntax:

ts.**MakeMergeMode** [= value]

The **MakeMergeMode** property has these parts:

Part	***Type***	***Description***
[Value]	Boolean	Controls the "MergeMode" attribute of a series.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      X.MakeMergeMode = True
      X.Save 'Update the series
End If
```

MakeRefreshOnRead() Property [Boolean]

Sets or returns the current setting for the "RefreshOnRead" attribute of a series. If "True", the Read() method of dbTimeSeries (see below) will calculate any make expression attached to the series (in a recursive fashion) before returning the result of the Read operation to the caller. As a result, the series will always reflect the results of any interim updates to raw data in the system since the series was last read.

Syntax:

ts.**MakeRefreshOnRead**[= value]

The **MakeRefreshOnRead** property has these parts:

Part	***Type***	***Description***
[Value]	Boolean	Controls the "RefreshOnRead" attribute of a series.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      X.MakeRefreshOnRead = True
      X.Save 'Update the series
End If
```

MakeSpreadSheetMode() Property [Boolean]

Sets or returns the current setting for the "SpreadSheetMode" attribute of a series when it comes to computing any make equation attached to the particular series. If "True", Series Make will "ignore" all the missing values encountered in a make expression by setting the missing value to 0 or 1 depending on the current make operation. As such, Series Make will not propagate missing values during a make operation. The default setting for this property is "False".

Syntax:

ts.**MakeSpreadSheetMode** [= value]

The **MakeSpreadSheetMode** property has these parts:

Part	***Type***	***Description***
[Value]	Boolean	Controls the "SpreadSheetMode" attribute of a series.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      X.MakeSpreadSheetMode = True
      X.Save 'Update the series
End If
```

MakeString() Property

Sets or returns the algebraic expression used to compute (or derive) the series from other series or statistical functions supported by Dbank's "Series Make".

Syntax:

ts.**MakeString** [= value]

The **MakeString** property has these parts:

Part	***Type***	***Description***
[Value]	String	Any valid Dbank Make expression (see Dbank User Guide).

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      X.MakeString = "b+c+d+[ace]e" '(syntax is not checked)
      MsgBox X.MakeString 'Display the expression
      X.Save 'Update the series
End If
```

Remarks

- No computation is performed by this property. It changes the make expression without actually computing the result.
- The make expression is not parsed until it is computed using the ReMake method. Thus invalid expressions are allowed.

MakeTrimMode() Property [Boolean]

Sets or returns the current setting for the "TrimMode" attribute of a series when it comes to computing any make equation attached to the particular series. If "True", Series Make will remove any trailing or leading values of a series before returning the result to the caller. The default setting for this property is "False".

Syntax:

ts.**MakeTrimMode** [= value]

The **MakeTrimMode** property has these parts:

Part	***Type***	***Description***
[Value]	Boolean	Controls the "TrimMode" attribute of a series.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      X.MakeTrimMode = True
      X.Save 'Update the series
End If
```

Max Property() [Double]

Returns the maximum observation in a series.

Syntax:

ts.**Max**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.Max 'Display maximum
End If
```

Mean() Property [Double]

Returns the arithmetic mean (average) of a series.

Syntax:

ts.**Mean**

Example

```
Dim X as New dbTimeSeries
If X.Read(“example[first.bp]bpeir”) Then
      MsgBox X.Mean ‘Display mean
End If
```

MeanAbsoluteDeviation() Property [Double]

Returns the arithmetic mean (average) of the absolute deviations from the mean.

Syntax:

ts.**MeanAbsoluteDeviation**

Example

```
Dim X as New dbTimeSeries
If X.Read(“example[first.bp]bpeir”) Then
      MsgBox X.MeanAbsoluteDeviation ‘Display mean absolute
                              ‘deviation
End If
```

Median() Property [Double]

Returns the median of a series.

Syntax:

ts.**Median**

Example

```
Dim X as New dbTimeSeries
If X.Read(“example[first.bp]bpeir”) Then
      MsgBox X.Median ‘Display median
End If
```

MedianAbsDev Property [Double]

Returns the median absolute deviation of a series.

Syntax:

ts.**MedianAbsDev**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.MedianAbsDev
       'Display median absolute deviation
End If
```

Memo() Property [String]

Sets or returns the memo property of the series. This property can be any text of any length.

Syntax:

ts.**Memo** [= value]

The **Memo** property has these parts

Part	***Type***	***Description***
[Value]	String	Any valid string.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.Memo 'Display current memo item
     'Now change it…
      X.Memo = "Hello, World.  This is my first memo…"
      X.Save
End If
```

Merge() Method [dbTimeSeries]

Updates the series using another series of the same frequency. Optionally, missing values in the updating series can be ignored.

Syntax:

ts.**Merge**(y,*[IgnoreMissingValues]*)

The **Merge** property has these parts

Part	***Type***	***Description***
y	Object	Dbank time series object.
[*IgnoreMissingValues=True*]	Boolean	Switch that controls whether missing values in the "y" series are ignored.

Example

```
Dim X as New dbTimeSeries
Dim Y as New dbTimeSeries
Dim j as Long
Dim Ok as Boolean
Ok = X.Read("example[first.bp]bpeir")
Ok = Ok And Y.Read("example[first.bp]bpe")
 'Line up the observations
If Ok Then
      X.Merge(Y,False)
      X.Save("example[ace]merge")
End If
```

Min() Property [Double]

Returns the minimum value of a series.

Syntax

ts.**Min**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.Min 'Display minimum value
End If
```

MissingCount() Property [Long]

Returns the number of missing values in the series.

Syntax:

ts.**MissingCount**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.MissingCount  'Displays 0
      X.DataValue(1) = X.MissingValue
      MsgBox X.MissingCount  'Displays 1
End If
```

MissingValue() Property [Double]

Returns the actual value used to represent missing values. Dbank presently uses IEEE +INF (positive infinity).

Syntax:

ts.**MissingValue**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.MissingValue  'Displays +1#INF
      If X.DataValue(1) = X.MissingValue Then
         MsgBox "First observation is missing."
      End If
End If
```

MostRecentRevisionTime() Property [Double]

Returns the last time that the series was last revised. The return value is a double in Microsoft date serial format. The integer part implies the year, month, and day that the series was created; the decimal part implies an hour and second. This method is property is different from the "Revised" method in that it obtained this value by re-reading the appropriate field from the database in which the series resides. This method returns zero if the series does not exist in any database.

Syntax:

ts.**MostRecentRevisionTime**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.MostRecentRevisionTime
End If
```

Move() Method [Boolean]

Moves or renames an existing series in a database. It returns "True" if successful. Note that this method cannot be used to move a series to another database. You must use the Copy() and Delete() methods to achieve this.

Syntax:

Ts.**Move**(NewLocation)

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      If X.Move("[temp]g") Then MsgBox "Move successful"
End If
```

MoveToExcel() Method [Boolean]

Copies a series to an Excel spreadsheet (which needs to be open before the method is called).

Syntax:

ts.**MoveToExcel**(WorkSheet, StartingRowIndex, ColIndex, *[DateTitle]*)

The **MoveToExcel** method has these parts:

Part	***Type***	***Description***
WorkSheet	Object	Object containing an open Excel Worksheet
StartingRowIndex	Long	Starting row in the Excel spreadsheet
ColIndex	Long	Starting column in the Excel spreadsheet
[DateTitle]	String	Optional title for the column to contain the date strings. When absent, the date string column is suppressed.

Example

```
Dim xlApp As Object
Dim xlBook As Object
Dim xlSheet As Object
Dim xlsRange As Object
Set xlApp = CreateObject("Excel.Application")
Set xlBook = xlApp.Workbooks.Add
Set xlSheet = xlBook.WorkSheets(1)
Dim X as new dbTimeSeries
If X.Read(“example[ace]a”) Then
      X.MoveToExcel(xlSheet,1,1,”Date”)
End If
xlBook.SaveAs FileName$
xlApp.Quit
Set xlSheet = Nothing
Set xlBook = Nothing
Set xlApp = Nothing
```

MsDateValue() Method [Double]

Maps any Dbank date serial to an equivalent Microsoft date serial, given the series attributes.

Syntax:

ts.**MsDateValue**(DateIndex)

The **MsDateValue** method has these parts:

Part	***Type***	***Description***
[DataIndex]	Variant	String or value defining the index of the observation. Default value os 1.

Example

```
Dim X as new dbTimeSeries
If X.Read("example[first.bp]bpier") Then
      MsgBox Month(X.MsDateValue(5))
      'Returns 1 if X.ConversionMethod = tsFirst
      'Returns 2 if X.ConversionMethod = tsAverage
      'Returns 3 if X.ConversionMethod = tsLast
End If
```

Remarks

When a string is used to index an observation, it must be in the format understood by the series. This format depends on the frequency of the time series, and, for financial and weekly data, the active dates format settings for Windows (see Regional Settings in Control Panel). See , Table 2 for more information.

MyData() Method [Variant]

Returns a variant that is a pointer to a double array containing all the observations in the series.

Syntax:

ts.**MyData**

Example

```
Dim X as new dbTimeSeries
Dim Data as Variant
If X.Read(“example[ace]a”) Then
      Data = X.MyData
      For j = 1 to X.Nobs
            MsgBox Data(j)
            ‘This is much faster than DataValue Method
            ‘But also far less flexible
      Next j
End If
```

MyDateSerials() Method [Variant]

Returns a variant/pointer to a string array containing all the date serials of the series in Dbank’s date serial format.

Syntax:

ts.**MyDateSerials**

Example

```
Dim X as new dbTimeSeries
Dim DateSerials as Variant
If X.Read(“example[ace]a”) Then
      DateSerials = X.MyDateSerials
      For j = 1 to X.Nobs
            MsgBox DateSerials(j)
      Next j
End If
```

Name() Property [String]

Sets or returns the short name of the series. The series' "short name" is the name that appears after the group specification of the full name (i.e., all the characters to the right of "]").

Syntax:

ts.**Name**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.Name  'Displays bpeir
      X.Name = "NewName"
      MsgBox "New time series name is:" & X.Name
      X.Save
      ' Save the series to "example[first.bp]newname"
End If
```

Nobs() Property [Long]

Sets or returns the number of observations in a series.

Syntax

ts.**Nobs** [= value]

The **Nobs** property has these parts

Part	***Type***	***Description***
[Value]	Long	Any positive integer.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.Nobs  'Displays 89
      'Allocate another 11 spaces existing data
      'remains intact
      X.Nobs = 100
End If
```

ObsFootNoteText() Property

Sets or returns an observation footnote for any observation in the series. Dbank supports multiple footnote entries for each observation.

Syntax

ts**.ObsFootNoteText**(DateSerial, CodeName) [= value]

The **ObsFootNoteText** property has these parts

Part	***Type***	***Description***
DataIndex	Variant	String or value defining the index of the observation
[Value]	Variant	Footnote description of any length (string).

Example

```
Dim X as new dbTimeSeries
If X.Read(“example[first.bp]bpeir”) Then
      X.ObsFootNoteText(“1977:1”,”MyFirstFootNote”) _
      = “Observation is estimated (by linear interpolation).”
End If
```

ObsIndex() Method [Long]

Returns the observation index (long) of any value in a series. Returns –1 if the value cannot be located in the series. Note that it will return the index of the first observation that equals the value being searched for. You may start the search from the end of the series, and work backwards.

Syntax:

ts.**ObsIndex**(Value,*[StartFromTop])*

The **ObsIndex** method has these parts:

Part	***Type***	***Description***
Value	Double	Numerical value to search for.
[StartFromTop]	Boolean	Controls the starting point of the search.

Example

```
'Determine the index of the minimum value of the series

Dim X as new dbTimeSeries
If X.Read("example[ace]a") Then
   MsgBox "Minimum value occurs at " _
            & X.DateSerial(X.ObsIndex(X.Min))
End if
```

Offset() Method [Long]

Returns the offset needed to match up observations from different series (of the same frequency) in a rectangular array.

Syntax:

ts.**Offset**(y)

The **Offset** method has these parts:

Part	***Type***	***Description***
y	Object	Dbank time series object.

Example

```
Dim X as New dbTimeSeries
Dim Y as New dbTimeSeries
Dim j as Long
Dim o as Long
Dim Ok as Boolean
Ok = X.Read(“example[first.bp]bpeir”)
Ok = Ok And Y.Read(“example[first.bp]bpe”)
o = X.Offset(y)
‘Line up the observations
If Ok Then
   For j = 1 to X.Nobs
      If (j+o) > 0 Then
            MsgBox X.DataValue(j) & “ aligns with “ & MsgBox _
            Y.DataValue(j+o)
      End if
   Next j
End If
```

Op() Method [dbTimeSeries]

Allows you to do basic arithmetic operations on series with the same frequency. Please note that Dbank's dbTSMake and dbMake classes obviate the need for this method.

Syntax

ts.**Op**(y, Operator)

The **Op** method has these parts:

Part	***Type***	***Description***
y	Object	Dbank time series object.
Operator	Integer	Dbank time series operator code (Enum)

The following operators are supported:

Operator	CodeName & Value	Meaning
Add	tsAdd = 1	Ts+y
Subtract	tsSubtract = 2	Ts-y
Multiply	tsMultiply = 3	Ts*y
Divide	tsDivide = 4	Ts/y
Power	tsPower = 5	Ts^y
IntegerDivide	tsIntegerDivide = 6	
Mod	tsMod = 7	Ts Mod y
And	tsAnd = 8	Ts AND y
Xor	tsXor = 9	Ts Xor y
Equal	tsEqual = 10	Ts=y
Eqv	tsEqv = 11	Ts Eqv y
Greater	tsGreater = 12	Ts > y
Less	tsLess = 13	Ts < y
GreaterOrEqual	tsGreaterOrEqual = 14	Ts ≥y
LessOrEqual	tsLessOrEqual = 15	Ts ≤y
Or	tsOr = 16	Ts Or y
NotEqual	tsNotEqual = 17	Ts ≠y
Remainder	tsRemainder = 18	Ts \ y
Imp	tsImp = 19	Ts Imp y
DotProduct	tsDotProduct = 20	Ts ′ y
Max	tsMax = 21	Max(Ts,y)
Min	tsMin = 22	Min(Ts,y)

OriginalHeader() Property [String]

Sets or returns the original header attribute of the series. This attribute is for capturing the original header that came with the time series before (perhaps) it was converted to Dbank format.

Syntax:

ts.**OriginalHeader** [= value]

The **OriginalHeader** property has these parts

Part	***Type***	***Description***
[Value]	String	Any valid string.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.OriginalHeader 'Display original series header
     'Now change it…
      X.OriginalHeader = "New series descriptor."
      X.Save 'Update it
End If
```

OriginalStartDate() [Variant]

Dbank allows the caller to change the start date of a series. This method retrieves the original start date of the series attached to the series when it was saved to the database for the first time.

Syntax:

ts.**OriginalStartDate**

Example

```
Dim X as new dbTimeSeries
X.Nobs = 100
X.Frequency = "M"
X.Startdate = "1971:8"
X.Save("example[ace]start")
X.Read
X.Startdate = "1979:1"
MsgBox X.OriginalStartDate 'Returns date serial for 1971:8,
                           'not 1979:1 in particular
```

Period() Method [Long]

Returns the period (month, quarter, or half-year) that an observation belongs to.

Syntax:

ts.**Period**(*[DateSerial]*)

The **Period** property has these parts:

Part	***Type***	***Description***
DateSerial	Variant	Dbank date string. Default setting is one.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.Period("1971:3") 'Displays 3
      MsgBox X.Period(3) 'Displays 3
End If
```

Remarks

See Table 2 for more information on setting Dbank date serials.

Periodicity() Property [Long]

Returns an integer representing the periodicity of the series (if known). The periodicity value is dependent entirely on the frequency of the series.

Syntax:

ts.**Periodicity**

Example

```
Dim X as new dbTimeSeries
X.Frequency = "A"
MsgBox X.Periodicity  'Returns 1
X.Frequency = "Q"
MsgBox X.Periodicity 'Returns 4
X.Frequency = "H"
MsgBox X.Periodicity 'Returns 2
X.Frequency = "M"
MsgBox X.Periodicity 'Returns 12

'Function returns –1 for all other frequencies
```

Plot() Method [Boolean]

Invokes Dbank's series plotter, which allows you to plot series against time. Dbank's Plot form is invoked in a non-modal fashion by default.

Syntax:

ts.**Plot**(*[Optional LoadAsModal=False]*)

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      If X.Plot Then
         MsgBox "Plot operation successful."
      End if
End If
```

PopulationVar() Method [Double]

Returns the population variance of a series.

Syntax:

ts.**PopulationVar**(LagOrder)

The **PopulationVar** property has these parts:

Part	***Type***	***Description***
[LagOrder]	Long	Any positive integer.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.PopulationVar 'Display population variance
End If
```

Precision() Property [Enum, tsPrecision]

Sets or returns the precision with which the series (i.e., its observations) is saved in the data bank. Two settings are presently supported: (a) single (4-bytes) and (b) double (8-bytes). Only double precision format is support for SQL databases.

Syntax:

ts.**Precision** [= value]

The **Precision** property has these parts:

Part	***Type***	***Description***
[Value]	Long	**tsSingle** (0) or **tsDouble** (1)

Example

```
Dim X as new dbTimeSeries
If X.Read("example[ace]a") Then
      X.Precision = tsSingle
      X.Save
      X.Precision = tsDouble
      X.Save
End If
```

QStatistic() Method [Double]

Returns the Box-Ljung Q statistic of the series for a given number of lags.

Syntax:

ts.**QStatistic**(LagOrder)

The **QStatistic** property has these parts:

Part	***Type***	***Description***
LagOrder	Long	Any positive integer.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.Qstatistic(2) 'Display Qstatistic at 2 lags
End If
```

QuartileRange() Method [Double]

Returns the "quartile range" of a series (third quartile less first quartile).

Syntax:

ts.**QuartileRange**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.QuartileRange 'Display quartile range
End If
```

Range() Property [Double]

Returns the range of a series.

Syntax:

ts.**Range**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.Range 'Display range
End If
```

Read() Method [Boolean]

Reads series in a data bank to a time series object in memory. Both the data and the attributes of the series are read.

Syntax

ts.**Read**(FullTimeSeriesName,*[Quiet],[SuppressReMake])*

The **Read** method has these parts:

Part	***Type***	***Description***
FullTimeSeriesName	String	Fully qualified Dbank time series name.
[Quiet]	Boolean	Suppresses error messages when set to true.
[SuppressReMake]	Boolean	Suppresses remake on read when set to true.

Return Code

Returns "True" if successful, "False" otherwise.

ReMake() Method [Boolean]

This method recalculates the observations in a series according the setting of its make string (see series MakeString property).

Syntax:

ts.**ReMake**(*[SmartReMake], [DontTrim], [OverWriteTitle]*)

The **ReMake** method has these parts:

Part	Type	Description (if True)
[*SmartReMake=False*]	Boolean	Recompute the series only if right-hand side time series are more recent that series itself.
[*DontTrim=False*]	Boolean	Determines whether leading and trailing missing values are removed from the resulting series.
[*OverWriteTitle=False*]	Boolean	Determines whether the method overwrites the series title with the series'make expression.

Example

```
Dim X as new dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      X.SeriesMake = "holtwinters([first]bpeir,1997:1)"
      X.ReMake(True,False,False)
      X.Save("example[ace]holt_bpeir")
End If
```

RemoteTimeStamp() Property [Double]

Returns the last known time stamp of the remote series that is dedicated to updating the time series at the client. It is the responsibility of the remote data provider to create this time stamp at the server. Dbank obtains this time stamp when it retrieves the update field from the remote server, and saves it to the series saved at the client. Dbank uses this field to determine whether it is necessary to update the client. The return value is a real value in Microsoft's date serial format. The integer part implies the year, month, and day that the series was created; the fractional part implies an hour and second.

Syntax:

ts.RemoteTimeStamp

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox Format(X.RemoteTimeStamp,"dddddd")
      'Remote update date using Windows long-date format
End If
```

RemoteUpDateTimeStamp() Property [Double]

Returns the actual date and time that the series was updated from the remote server. Note that this can be different from the time stamp returned by the "**RemoteTimeStampMethod"**. The return value is a real value in Microsoft's date serial format. The integer part implies the year, month, and day that the series was created; the fractional part implies an hour and second.

Syntax:

ts.RemoteUpdateTimeStamp

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox Format(X.RemoteUpdateTimeStamp,"dddddd")
      'Remote update date using Windows long-date format
End If
```

RemoveScale() Property [Boolean]

This method removes the optimal scaling factor originally applied to the values in a series. It multiplies each value in the series by the optimal scaling factor returned by **"ScalingFactor"**. This method should only be used if the data has been already been scaled.

Syntax:

ts.RemoveScale

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
     X.ScaleData  'Scale the data value
      X.RemoveScale 'Remove the scale
      X.Save
      'Remote update date using Windows long-date format
End If
```

Rename() Method [Boolean]

Renames an existing series in a data bank. This method cannot be used to move a series to another data bank.

Syntax:

ts.**Rename**(NewSeriesName)

The **Rename** method has these parts:

Part	***Type***	***Description***
[*NewSeriesName*]	String	Fully qualified Dbank series name

Example

```
Dim X as new dbTimeSeries
X.FullName = "example[ace]a"
If X.Rename("example[ace]a_newname") Then
      MsgBox "Series renamed successfully!"
End if
```

ReplaceMissing() Method [dbTimeSeries]

Replaces all missing values in a series using any of the interpolation methods supported in Dbank.

Syntax:

ts.**ReplaceMissing**(Type, *[StartFromBottom=False]*)

The **ReplaceMissing** method has these parts:

Part	***Type***	***Description***
[Type=tsCubicSpline]	Integer	Valid Interpolation Method
[StartFromBottom]	Boolean	Start interpolation from the beginning of the vector.

Example

```
Dim X as new dbTimeSeries
If X.Read(“example[first.bp]bpeir”) Then
      X.DataValue(10) = X.MissingValue
      X.ReplaceMissing(tsLinear)
      X.Save(“example[first.bp]no_missing”)
End If
```

ReplaceMyData() Method [dbTimeSeries]

Rapidly replaces all the data values in a time series by moving the contents of a double array supplied by the user into the series object. It is used internally within Dbank to speed up certain numerical operations in Series Make, where reliance on DataValue() to update the individual values would result in a significant loss of performance. Note that you can control the starting position of the revised data by specifying an offset. If positive, Dbank will replace the internal data starting from the supplied offset. This method is typically used in conjunction with the CopyMyData() method.

Syntax:

ts.ReplaceMyData(DataArray() as Double, Nobs, *Offset=0]*,*[HasMoreThanOneIndex=False]***)**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
   ReDim DataValues(1 to X.Nobs)
   Call X.CopyMyData(DataValues(),X.Nobs)
   For j = 1 to X.Nobs : DataValue(j) = DataValues(j) /100
   Call X.ReplaceMyData(DataValues(),X.Nobs)
   'X has been scaled downwards by 100!
End If
```

Revised() Property [Double]

Returns the time that the series was last revised. The return value is a double in Microsoft date serial format. The integer part implies the year, month, and day that the series was created; the decimal part implies an hour and second.

Syntax:

ts.**Revised**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox Format(X.Revised,"dddddd")
                  'Display using Windows long-date format
End If
```

Save() Method [Boolean]

Save the current settings of the series to a data bank. The location is determined entirely by the full name of the series. However, specifying an optional argument in the Save method can override this behavior. The caller can also prevent the save method from accidently overwriting an existing series by setting the optional "AllowedToOverWrite" parameter to "False".

Syntax

ts.**Save**(FullTimeSeriesName,*[AllowedToOverWrite=True], [OnlySaveAttributes=False]*)

The **Save** method has these parts:

Part	***Type***	***Description***
[*FullTimeSeriesName*]	String	Fully qualified Dbank time series name
[*AllowedToOverWrite=True*]	Boolean	Determines whether one can overwrite existing series. Default setting is "True".
[OnlySaveAttributes=False]	Boolean	If "True", Saves only the non-data components (or attributes of a series).

ScaleData() Property [Boolean]

This method scales the values in a series using the optimal scaling factor. It divides each value in the series by the optimal scaling factor returned by the **"ScalingFactor"** method**,** which ensures that subsequent numerical operations loose the least significant digits.

Syntax:

ts.RemoveScale

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
     X.ScaleData  'Scale the data value
      X.RemoveScale 'Remove the scale
      X.Save
      'Remote update date using Windows long-date format
End If
```

ScaledMedianAbsoluteDev() Method [Double]

Returns the scaled median absolute deviation of a series.

Syntax:

ts.**ScaledMedianAbsoluteDev**

Example

```
Dim X as New dbTimeSeries
If X.Read(“example[first.bp]bpeir”) Then
      MsgBox X.ScaledMedianAbsoluteDev ‘Display it
End If
```

ScalingFactor() Method [Double]

Returns the optimal scaling factor for the series. The optimal scaling factor can be used to rescale the data so that subsequent numerical operations result in a minimum loss of significant digits.

Syntax:

ts.**ScalingFactor**

Example

```
Dim X as New dbTimeSeries
If X.Read(“example[first.bp]bpeir”) Then
      MsgBox X.ScalingFactor ‘Display it
End If
```

sConversionMethod() Property [String]

Sets the conversion method to be used by default when converting a series to any frequency that would result in a smaller number of observations (i.e., consolidation occurs). This property differs from the "ConversionMethod" property explained above in that the caller can specify the default conversion method using its actual name (a string) rather than an enum constant.

Syntax:

ts.**sConversionMethod** = *ConversionMethod*

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      X.sConversionMethod = "Last"
      X.Save
End If
```

Server() Property [String]

Sets or returns the name of the SQL server to which the series will be saved.

Syntax:

ts.**ServerName** = *[ServerName]*

Example

```
Dim X as New dbTimeSeries
If X.Read("<tsOrion>example[ace]a") Then
      MsgBox Ucase(X.ServerName) 'Returns "<tsOrion>"
      X.Save
End If
```

ShowAcf() Method [Boolean]

Invokes Dbank's series correlogram form, showing the auto and partial autocorrelations of the series. Dbank's Acf form is invoked in a non-modal fashion by default.

Syntax:

ts.**ShowAcf**(*[Optional LoadAsModal=False]*)

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      If X.ShowAcf Then
         MsgBox "Show Acf form operation successful."
      End if
End If
```

ShowErrorMessages() Property [Boolean]

Allows the caller to control whether the dbTimeSeries method displays error messages (i.e., loads a message box with a description of the error encountered) on the caller's screen.

Syntax:

ts.**ShowErrorMessages**(*Setting*)

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      If X.ShowErrorMessages Then
        MsgBox "Dbank will display all error messages in MsgBox."
      End if
End If
```

SignificantDigits() [Long]

Sets or returns the number of significant digits to display whenever a series is shown in Dbank's viewer. The numerical values of the data are unaffected.

Syntax:

ts.**SignificantDigits** [= value]

The **SignificantDigits** method has these parts:

Part	***Type***	***Description***
Value	Long	Number of significant digits

Example

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      X.SignificantDigits = 2
      X.Save("example[ace]a_s2")
End If
```

Skewness() Property [Double]

Returns the skewness coefficient of a series.

Syntax:

ts.**Skewness**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.Skewness 'Display skewness coefficient
End If
```

Smooth() Method [dbTimeSeries]

Fits a smooth line to the series using Friedman's super smoother. The original data is replaced with the smoothed values. Warning: smooth is not a centered moving average.

Syntax:

ts.**Smooth**

Example

```
Dim X as new dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      X.Smooth
      X.Save("example[first.bp]smooth")
End If
```

Sort() Method [dbTimeSeries]

Sorts the data in the series from highest to lowest. The original data is replaced with the sorted values.

Syntax:

ts.**Sort**

Example

```
Dim X as new dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      X.Sort
      X.Save("example[first.bp]sorted")
End If
```

StartDate() Property [Variant]

Sets or returns the starting date of the series. Method always returns a Dbank date serial (long).

Syntax:

ts.**StartDate** [= value]

The **StartDate** method has these parts:

Part	***Type***	***Description***
Value	Variant	Dbank date serial or string

Example

```
Dim X as new dbTimeSeries
X.Nobs = 100
X.Frequency = “M”
X.Startdate = “1971:8”
X.Save
```

StdDev() Property [Double]

Returns the sample standard deviation of a series.

Syntax:

ts.**StdDev**

Example

```
Dim X as New dbTimeSeries
If X.Read(“example[first.bp]bpeir”) Then
      MsgBox X.StdDev ‘Display standard deviation
End If
```

Sum() Property [Double]

Returns the sum of the values in a series.

Syntax:

ts.**Sum**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.Sum 'Display sum
End If
```

SummaryStatistics() Method [Boolean]

Invokes Dbank's summary statistics form, which provides a graphical view of approximately 15 summary statistics on the series. Dbank's SummaryStatistics form is invoked in a non-modal fashion by default.

Syntax:

ts.**SummaryStatistics**(*[Optional LoadAsModal=False]*)

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      If X.SummaryStatistics Then
         MsgBox "SummaryStatistics form invoked successfully."
      End if
End If
```

Take() Method [Boolean]

Drops (strips) all observations in a series *before* a given date/index.

Syntax:

ts.**Take**(Index)

The **Take** method has these parts:

Part	***Type***	***Description***
Index	Variant	Observation index.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      X.Take("1975:1") 'Removes all data before 1975:1
      X.Save
End If
```

Remarks

When a string is used to index an observation, it must be in the format understood by the time series. This format depends on the frequency of the series, and, for financial and weekly data in particular, the active date format settings for Windows (see Regional Settings in Control Panel). See , Table 2 for more information.

ThirdQuartile() Method [Double]

Returns the third quartile of the series.

Syntax:

ts.**ThirdQuartile**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.ThirdQuartile 'Display third quartile
End If
```

Title() Property [String]

Sets or returns a descriptive string that typically describes the contents of the series.

Syntax:

ts.**Title** [= value]

The **Title** property has these parts

Part	***Type***	***Description***
[Value]	String	Any valid string.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.Title 'Display current series descriptor
     'Now change it…
      X.Title = "New series descriptor."
      X.Save 'Update it
End If
```

TrendLine() Method [dbTimeSeries]

Fits a trend line to the series using Ordinary Least Squares. The original data is replaced with the fitted trend.

Syntax:

ts.**TrendLine**

Example

```
Dim X as new dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      X.TrendLine
      X.Save("example[first.bp]trend")
End If
```

Trim() Method [dbTimeSeries]

Trims (removes) observations from the beginning and end of a series.

Syntax:

ts.**Trim**(*[FrontAmount], [BackAmount]*)

The **Trim** method has these parts:

Part	***Type***	***Description***
[FrontAmount]	Variant	Amount to remove from the beginning of the series. *Removes leading missing values if omitted.*
[BackAmount]	Variant	Amount to remove from the end of the series. *Removes trailing missing values if omitted.*

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      X.Trim(5,10) 'Remove first 5 & last 10 observations
End If
```

UnitRootStatistics() Method [Boolean]

Invokes Dbank's unit root statistics form, which provides a graphical view of some standard unit root tests for time series data. Dbank's UnitRootStatistics form is invoked in a non-modal fashion by default.

Syntax:

ts.**UnitRootStatistics**(*[Optional LoadAsModal=False]*)

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      If X.UnitRootStatistics Then
         MsgBox "UnitRootStatistics form invoked successfully."
      End if
End If
```

Units() Property [String]

Sets or returns a descriptive, arbitrary string describing the units of measurement of the series.

Syntax:

ts.**Units** [= value]

The **Units** property has these parts

Part	***Type***	***Description***
[Value]	String	Any valid string.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.Units 'Display current units of measurement
     'Now change it…
      X.Memo = "Billions."
      X.Save 'Update it
End If
```

UpdatedBy() Property [String]

Returns the name of the account that last updated the time series to the databank. This field is set when the series is read from the database. As such, it will be inaccurate if another user updated the series since it was read into the dbTimeSeries object.

Syntax:

ts.UpdatedBy()

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.UpdatedBy
End If
```

ValidCount() Method [Long]

Returns the number of non-missing (or valid) observations in a series

Syntax:

ts.**ValidCount**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.ValidCount
       'Display number of valid observations
End If
```

Var() Method [Double]

Returns the sample variance of a series.

Syntax:

ts.**Var**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.Var 'Display sample variance
      MsgBox X.Var*(1-(1/X.Nobs))
       'Display estimate of pop. variance
End If
```

View() Method [Boolean]

Invokes Dbank's series' viewer, which provides a graphical view of a series and its main attributes. Dbank's View form is invoked in a non-modal fashion by default.

Syntax:

ts.**View**(*[Optional LoadAsModal=False]*)

```
Dim X as New dbTimeSeries
If X.Read("example[ace]a") Then
      If X.View Then
         MsgBox "UnitRootStatistics form invoked successfully."
      End if
End If
```

vonNuemannRatio() Method [Double]

Returns the von-Nuemann ratio statistic of the series.

Syntax:

ts.**vonNuemanRatio**

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.vonNuemannRatio 'Display von-Nuemann Ratio
End If
```

Year() Method [Long]

Returns the Julian year of an observation.

Syntax:

ts.**Year**(DateSerial)

The **Year** property has these parts:

Part	***Type***	***Description***
DateSerial	Variant	Dbank date serial.

Example

```
Dim X as New dbTimeSeries
If X.Read("example[first.bp]bpeir") Then
      MsgBox X.Year("1971:3") 'Displays 1971
      MsgBox X.Year(11) 'Displays 1971
End If
```

Zeros() Method [dbTimeSeries]

Generates a series with zeros for all observation values.

Syntax

ts.**Zeros**()

The **Zeros** method has these parts:

Part	***Type***	***Description***
Length	Long	Length of series (or number of observations)

Example

```
Dim X as New dbTimeSeries
X.Start = 1
X.Zeros(100)
X.Title = “Variable with 100 zeros”
X.Save(“example[ace]zeros”)
```