DataMax Software Group Inc.
1101 Investment Blvd.
El Dorado Hills, CA 95762
USA

RFgen Developers Reference Guide

All Editions
RFgen v5.2

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Visual Basic Scripts

Visual Basic for Applications (VBA) scripts allow developers to provide additional functionality to RFgen data collection applications by responding to events using Visual Basic programming statements (scripts). VBA Scripts allow the developer to enhance the capabilities offered by standard RFgen applications and other objects. In fact, developers may take total control over the client device by responding to field/system events, handling all data display functions, and even sending SQL statements through the RFgen database connection.

Programmable events include 'field' events (scripting is related to the current prompt) and application events (scripting will be executed whenever this event occurs; e.g. when an application is loaded). In addition, built-in VBA 'extensions' give the developer easy access to manipulate the client device, and quick access to the connected database.

Global User-Defined Subroutines and Functions

To use global subroutines and functions: Create a module file using the VBA Modules and put your 'global' subroutines and functions in that file.

On the Properties tab in the RFgen Application window, you will see a list of available VBA modules. Set each module to TRUE that you want that application to be able to access.

You will then be able to call any subroutines or functions from these modules in your RFgen Applications.

Note: There are 2 built in modules that are automatically referenced by RFgen Applications. These are:

`RFgen.bas` – The RFgen.bas is typically used to contain functions and procedures that need to be accessible from any transaction

`Win32.bas` – The Win32.bas is typically used to store global variables.

Using External ActiveX files in the VBA Environment

RFgen allows the incorporation of external ActiveX .EXE or .DLL modules by declaring an Object and using the CreateObject method.

Example:

```
Dim MyObject As Object
Set MyObject = CreateObject("name.cls")
' name is the name of the ActiveX module
' subsequent usage
MyObject.property
MyObject.method
```

The syntax is slightly different when using a DLL compiled using Visual Basic.

```
Dim MyDLL As Object
Set MyDLL = CreateObject("ourDDC.DDCcom")
```

Where: ourDDC is the executable name DDCcom is the public class name containing the functions you wish to call.

VBA Global Variables/Objects

Global variables and objects are available for use with RFgen, for storing and retrieving items related to a specific session.

The preferred method for using global variables is to declare the variables in the 'Win32.bas module, via the VBA Modules option. Any public variables declared in the Win32.bas module will be available to all RFgen applications and will maintain their values throughout the life of the RFgen session.

Examples:

Public sMyString As String

Public iMyInt As Integer

Public oMyObject As Object

VBA Declarations

The VBA 'General' object is where the developer may declare variables that are available to all events in the current application. User defined functions or subroutines may also be entered here.

Short Cut Keys

The following keyboard combinations will invoke the functions described below.

Middle click on tab

Closes tab.

Ctrl+F

This will open the search function when in the script view.

Ctrl+SHIFT+F

This will find a specified script on a form (application form).

The following keyboard combinations will provide the functions described below.

Ctrl+SHIFT+H

This will replace a specified script on a form (application form).

Ctrl + K, C

This will comment current line/highlighted section (app scripts and modules). This short cut key will also work in transaction macro scripts and in voice applications.

Ctrl + K, U

This will uncomment current line/highlighted section (app scripts and modules). This short cut key will also work in transaction macro scripts and in voice applications.

Ctrl+M+L

This will expand all functions in a script.

Ctrl+M+O

This will collapse all functions in a script.

Ctrl+M+M

This will expand the function the cursor is on.

Ctrl+ S

This will save the changes in the designer that has focus, or changes to the scripting page (apps and modules). This short cut key will also work in transaction macro scripts and in voice applications.

Application Designer ShortCuts

Hold down **ALT key while clicking on the page number** or a control will renumber that prompts tab sequence

Note: These short-cut key combinations are supported in RFgen 5.2.4.6 and higher.

Initialization Files

Initialization files are used to make changes to the default behavior of RFgen such as resetting data connections based on usage, indexing databases, preventing server-side dialog boxes from stopping the RFgen service and altering connectivity parameters for mobile devices.

The files are stored in the %AppData% / ProgramData / RFgen50, %AppData% / ProgramData / RFgen51, or %AppData% / ProgramData / RFgen52 folder. The parameters for your configuration settings belong under the [Environment] global setting.

Types are: Balloon.init, Clientini, ERPDialogsini, GPRSini, and RFgen.ini

RFgen.ini

The **RFgen.ini** initialization file provides configuration parameters for making changes to the default behavior of RFGen such as resetting data connections based on usage or indexing of databases. It can also be used to set transaction limits, and monitor JDE logs.

In all cases, the headings and settings are case sensitive, and the ini file must be created and modified using Notepad. The RFgen.ini file is stored in the RFgen directory to work. For example: C:\ProgramData\rfgen52

The RFgen.ini parameters are: TextHint (overrides Device Authorization), TextHint (Choose Camera), TextHint (JDE connectivity and JDE log.), TextHint(Batch client connection), TextHint, TextHint, TextHint (Checks for connection from a third-party load balancing server/applicance), TextHint (RFgen load balancing server wait times), TextHint (RFgen load balancing server logging) and TextHint (Shares RFgen client licenses), TextHint (emulate queing transactions), TextHint from Reports, TextHint, TextHint (Reset of Transactions), TextHint (AS400 index connections), TextHint (controls resets), and TextHint.

AutoAuthBATCH

The AutoAuthBATCH parameter in a RFgen.ini file enables a batch client to connect to the server regardless of the settings in the Mobile Development Studio > Devices > Authorized Devices screen or the Mobile Unity Platform Console > Devices screen.

Syntax:

```
[Environment]
AutoAuthBATCH=T
```

where True allows any batch client to connect; False will only connect if the batch client is authorized.

Camera

The Camera parameter in a RFgen.ini file will specify which camera RFgen should used if the Windows CE devices has multiple cameras.

Syntax:

```
[Environment]
Camera=n
```

where n is from 0 to the number of available cameras minus 1.

Check

The Check parameter in a RFgen.ini file will check for JDE connectivity before a business function is executed.

Syntax:

[JDE]
CheckX0010=0
where the option 0 turns the check off, and 1 is on.

Parameters for the X0010 call:

SystemCode=41
NextNumberingIndexNo=2
CompanyKey=00001
DocumentType=IA

CheckLog

The CheckLog parameter in a RFgen.ini file will monitor the JDE.log for designated strings within the JDE.log file's last 500 characters. It monitors for various test strings, and is used to reset child processes.

Syntax:

CheckLog=1

where the option 0 turns the log check off and 1 is on.

LogText1=RESETNOW
LogText2=JDB9900400

where Text1, Text2 ... will look for in JDE.log

If the LogText# is found, then it will stop and restart all RFgen child processes.

If one of the LogText parameters is found the server will:

1. Delete the log file. JDE creates a new file every time it starts.
2. Suspend all active client connections temporarily
3. Close all transaction manager client(s).
4. Close all data connections.
5. Start all data connections.
6. Start the transaction manager client(s).
7. Resume the active client connections.

ForceReset

The ForceReset parameter in a RFgen.ini file will reset the data connectors after the specified period of time has elapsed.

Syntax:

[Options]
ForceReset = nn

where nn is a positive number in minutes. The option 0 disables the reset function.

Indexviews

When RFgen indexes tables for a database connector, by default, it only includes tables. If a solution also requires table views use indexviews in the RFgen.ini file to set the index view.

Syntax:

[Environment]
indexviews = 1

where 1 enables this feature.

IsMobile

The IsMobile parameter in the RFgen.ini fie sets a queue on the server to pretend to be a mobile device queue so that if the program executes "Server.SyncApps" and points to another RFgen server, that queue will be moved to that other server instead of being processed on its current server.

Syntax:

[Environment]
IsMobile = name of the server

Example:

[Environment]
IsMobile = CostPoint

where "CostPoint" is the name of the other RFgen server

LBHealthChecker

These parameters enable the RFgen server to listen for a connection from a **third-party load balancing appliance/system** when you assign a LBHealthPort to the RFgen server and/or RFgen Mobile Development Studio in the "[Environment]" section of the RFgen.ini file.

The Load Balance Health Checker can also be configured to accept data and send data back to the third-party load balancing system. If the *LBHealthText* parameter is specified, then the RFgen server will send the text back to the requesting device before it closes the TCP connection and resumes listening for a connection.

A restart of server services may be needed to activate these changes.

For information on the RFgen Load Balancing Server ini settings, see LBS, LBSLog, or LBSRedirect.

Syntax:

[Environment]
LBHealthPort=21090

where ##### is the port that will listen for a connection from the third-party Load Balancer.

LBHealthText=Send this text back

where "Send this text back" is the text that is sent to the third-party Load Balancer.

LBHealth=1

where "1" enables the listening of the port; "0" disables it.

LBS

The LBS value (Load Balancing Server) in the RFgen.ini file controls how long an RFgen load balanced server will wait for replies from other RFgen servers. When a client attempts to connect a request is sent to all servers in the cluster. This setting controls how long the calling server will wait for replies.

Syntax:

[LBS]

ReplyTimeout=n milliseconds

where n is the number of milliseconds the server or calling server will wait for replies.

LBSLog

The LBSLog parameter in a RFgen.ini file enables the server to log load balancing information in a log file. This should be used with a Trace tool. For the option on creating the log file, see RFSVR.

Syntax:

[Environment]
LBSLog=1

where the option 0 is off; 1 is on.

LBSRedirect

The LoadBalanced Server LBSRedirect parameter in a RFgen.ini file allows servers to share client licenses but not client sessions.

Syntax

[Environment]
LBSRedirect=1

where the option 0 is off; 1 is on.

Example:

If you want to clients connected to a load balanced server to share a license but not the client session, use:

```
[Environment]
LBSRedirect = 0
```

MobileQueue

The MobileQueue setting allows an RFgen server to emulate an offline device in terms of queueing transactions. Using this setting in conjunction with Sevrer.SendQueue allows a server to queue transactions and then transfer the transactions to an alternate server using the Server.SendQueue language extension.

RFgen configuration settings are used if a server needs to push queued items from a local RFgen server to a different RFgen server for processing. Mobile devices also have scenarios where they need to push queued items to a RFgen server, and the "Server.SendQueue" transfers items to the RFgen server.

If you have multiple servers where one has the collected the data and queues, but the other server has the ERP connector, the "MobileQueue" command performs the same transfer.

Example
Server A receives the queued transactions from the handheld, but doesn't have the ERP connector to process them. Server B has the ERP connector. Server A does not need any queue configured at all for a queue name identified in the MobileQueue setting. Server A keeps the records internally.

Server B must have a queue configured and set to process the name mentioned in the MobileQueue setting. On a timed event, Server A performs a Server.SendQueue like a handheld would, and the contents of the internal queue will be sent to Server B.

This concept eliminates the need for Server A to have a data connector to Server B for the Server.MoveQueue command.

Note: This parameter is setup in the RFgen.ini file.

Syntax

```
[Environment]
MobileQueue=Queue1,Queue2,Queue3,…
```

PurgeAfterNBRDays

The Purge Data from Reports PurgeAfterNbrDays parameter in a RFgen.ini file will remove the performance data that is visible from Reports > Performance Monitoring in the Mobile Development Studio or Mobile Unity Platform Console.

Syntax

```
[Environment]
PurgeAfterNbrDays=nn
```

where nn is the number of days.

RFSVR###

The RFgen Server RFSVR parameter in a RFgen.ini file causes the servers to generate a log file.

This is currently limited to the LBSLog (load balanced information).

For details on start/stop of logging of information see LBSLog.

Syntax

```
[Trace]
RFSVR###=1
```

where 0 is off; 1 is on.

must correspond to the version of the RFgen server

Example

```
'For RFgen 5.20 this will generate a logfile:

[Trace]
RFSVR520=1
```

RefreshRate

RefreshRate controls how often statistical information is sent from the RFgen server to a connected Admin Dashboard.

Syntax:

```
[Environment]
RefreshRate=n seconds
```

ResetUDC

The ResetUDC parameter in a RFgen.ini file removes a connection from a pool, destroys it, and recreates the connection for the next user. This allows the system to release the memory for the process in the system.

Syntax:

```
[Options]
ResetUDCx = y
```

where x is set to the dataconnector and y is the value to the number of transactions that will be processed before the reset takes place.

Schema

In the case of some DB2 ODBC connections, some tables may not be visible to RFgen because of settings for the DB2 database. To make these tables visible, add an entry with the following format in the RFgen.ini file.

You may have as many schemas as needed. These entries are only examples.

Syntax

```
[AS400]     DB2 Data source name in RFgen

Schema1=S104WL5M.RB1     specific database / library index
Schema2=S104WL5M.QSYS    specific database / library index
```

TMTimeout

TMTimeout parameter in a RFgen.ini file allows a Transaction Macro to execute longer than the built-in, internal limit of 30-seconds. The built-in timer terminates any macro that runs longer than 30 seconds. Even though a macro is terminated, it stays in the queue. Should a long running macro get half done, terminate, and re-run, you could end up with a continuous loop. To prevent this problem, use the TMTout command to lengthen the timeout period.

You might also set the timeout to 120 seconds or higher if downloading a large volume of data from a server to a mobile device. After making this change the services should be restarted.

Syntax:

```
[Environment]
TMTimeout=nnn
```

Where nnn is the timeout in seconds

TranLimit

The **TranLimit** parameter in a RFgen.ini file enables the Transaction Manager to process a specified number of transactions in the queue and then reset the queue. (A reset destroys and restarts the client process in charge of processing the queue, and also recreates the data connector.) When this happens, the system releases the memory used by the process.

RFgen 5.1.1.18 added the ability to define multiple transaction limits instead of a single transaction limit in the RFgen.ini file. This enhancement was added so you can control the reset of individual transaction queues instead of having them all reset when just one limit is triggered.

To Set Limits - Specify a limit for all the queues with the command TranLimit=n by itself, or as a limit for individual queues by queue name.

Syntax:

```
[Environment]
TranLimit=n
```

Where n is the number of transactions that will be processed before the Transaction Management process as well as its data connections are shut down and then recreated.

Example:

```
TranLimit=300
```

If an RFgen queue accumulates 300 or more transactions, the Transaction Manager will shut down the data connector and recreate the connection.

To Reset Individual Queues - Specify by the unique name, the queue to be reset in a RFgen.ini file.

Syntax:	TranLimit = queuename1, queuename2, queuename3… Where queue1, queue2...queuex are the names for each individual queue and has its own section under the [Environment] setting to control the individual queues. Within the individual queue sections, place another TranLimit=n command where n is the number of iterations.

Example:	`[Environment]`
	`TranLimit=RFgenQueue1, TWDXX0100, TCCXX0200 [RFgenQueue1]`
	`TranLimit=100 [TWDXX0100] TranLimit=100 [TCCXX0200] TranLimit=200`

Balloon.Init

You can create a Balloon.Init file for the purposes of pre-setting the balloon's generic behavior (like a template) is consistent throughout the application by using values that are set here in the Balloon.Init file. This allows you to then customize just the message in a Balloon but keep its behavior (i.e. look and feel, animation etc.) the same.

The following parameters and example values should be listed under the [Environment] setting.

Example

```
[Environment]
Balloon.Text = "An error has occurred"
Balloon.Duration = 3
Balloon.Title = "Error!"
Balloon.ImageId = "ErrorIcon"
Balloon.CloseButton = True
Balloon.Animate = True
Balloon.WidthPercent = 0.6
Balloon.EndLocation = 200
Balloon.StartLocation = 0
```

Supported Versions: RFgen 5.2.5.1 and higher

Client.ini

The Client.ini file provides configuration parameters for the RFgen Windows Desktop client and can be used to change specific behaviors.

Location

This ini parameter is designed to allow the user to change the location of the client database. The client database stores the configuration of the Desktop Client.

The usual place the client.ini file is created is:

C:\Users*username*\AppData\Roaming\RFgen5 or RFgen51 or RFgen52

The Desktop Client reads the client.ini file for the rfgen.xdb, the database storing the configuration. If an entry exists it will use that database, if not the Desktop Client will look for the rfgen.xdb file in the default location. Here is an example of a database path entry:

```
[Environment]
Location=3,25,323,297
GUID=5907C90C-C337-4A57-8895-BE891D80CFD5
WindowState=1
Monitors=2
rfgendb=c:\programdata\rfgen51\rfgen.xdb
```

Batch Mode Stress Testing Flags

This feature is used to stress test a Desktop Client running in batch mode by implementing the "MultiBatch" and "IgnoreGUID" flags in the Client.ini file under the [Environment] section, in the AppData folder.

Syntax

[Environment]

MultiBatch=n This flag clones the existing batch mode database and renames it to that client's process ID.

IgnoreGUID=n This flag gives the client (Thin or Batch) a unique GUID, similar to how the /ignoreGUID cmd parameter works.

n is the expected number of clients to be launched, all using GUIDs and batch mode database files.

Example

```
[Environment]
MultiBatch=1    'The UserData.sqlite is cloned and renamed to UserData-{PID}.sqlite for one client.
IgnoreGUID=1    'A GUID is assigned to one client.
```

ERPDialogs.ini

The ERPDialogs.ini file provides the ability to automatically respond to dialog boxes created by software (such as JD Edwards) running on the RFgen (Communications) server.

Syntax:

Search, Title, Message, Reset, Close, Text

Search	1 = Search on Title, 2 = Search on Message
Title	Window Title
Message	Message Text (only if 'Search' = 2)
Reset	0 = Don't reset connection, 1-5 = Connection number to reset
Close	1 = Close Window, 2 = Click button
Text	Text on button to click (only if 'Close' = 2) (include "&" to represent underscore, e.g., "E&xit" for "Exit").

Example:

```
1, Database Password Entry, ,1,1,
1, Database Error,,1,1,
1, OneWorld Error,,1,1,
1, OneWorld,,1,1,
1, PeopleSoft Error,,1,1,
1, PeopleSoft,,1,1,
1, Remote Job,,1,1,
1, Database Password Entry, ,1,1,
1, Database Error,,1,1,
1, OneWorld Error,,1,1,
1, OneWorld,,1,1,
1, PeopleSoft Error,,1,1,
1, PeopleSoft,,1,1,
1, Remote Job,,1,1,
2, ,JDB9103 - Fetch failed, 1, 2, OK
2, ,to Activate the component and correct the problem, 1, 2, Switch To...
1, RFDB,, 1, 2, Retry
```

Even if 'Search' = 2, a window title must be included if one exists.

GPRS.ini

This ini file is designed to make the RFgen mobile client dial a connection so that data can be sent or retrieved to the server while in a disconnected state. This file is not intended to establish a connection for a thin client connection. If a General Packet Radio Service (GPRS) connection is desired for a thin client connection simply establish it first, then any VPN connections if required and then launch the RFgen thin client.

In a typical implementation Server.Connect and Server.Disconnect are used both at the beginning of the process, to retrieve validation data, and at the end to submit collected data. When the connect command is executed the mobile client checks the phonebook setting in the RFgenCFG.exe program and uses either WiFi or GPRS to make the connection. If GPRS is selected the ini file tells RFgen what settings to use. If RFgen needs to establish the VPN connection as well then those settings can be documented as well.

Note that the GPRS and VPN connections must first be setup in the CE environment and RFgen simply references and starts those connections.

Using any desired method, create and deploy a file called GPRS.ini to the install directory (by default: \Program Files\RFgenCE). It contains the following settings:

```
[GPRS]
Enabled=True
Name=GPRS
User=
Pwd=

[VPN]
Enabled=True
Name=My Work Network

User=
Pwd=
```

The **Enabled** parameter just tells RFgen if it should make that connection.

The **Name** parameter is the name used in the setup of the connection in the operating system's configuration.

The **User** and **Pwd** fields, if needed, are stored in the ini file as well.

Text Default Options

A default property allows a data value to be generated automatically. For example, if you had two warehouses, one in Boston and the other in Sacramento, and wanted the user to default to the Sacramento Warehouse without having to scan a barcode or enter the code for the Sacramento warehouse, the developer of the application can set the default value to "Sacramento" so that when the user clicked on the prompt for the Warehouse, "Sacramento" would display.

A blank value means that there is no default.

Placing a ';O' (semicolon and the letter O, not zero) after the default parameter is optional, and allows the default to be overwritten.

Note: If using the VBA scripts, the Got_Focus event for a prompt contains a variable called 'RSP' which may be set to the value of the data default and a variable called 'AllowChange' (for overriding the value) which represents the value of the ';O' expression; i.e., AllowChange = True, is the same as placing a ';O' after the specified default property.

Text Default

This will default the string value as entered. A string value can be any number, combination of letters, or special characters. Format is:

String value(;O)

Sacramento;O means default equals 'Sacramento', allow 'O'verwrite. Values contained in ( ) are optional.

System Date Default

This will default the current Windows system date in the format specified by 'exp1'. If no format is specified then the date will display using the format specified by the data item used for the prompt. Format is:

@DATE(;O)

@DATE;O means default equals current date displayed using MM/DD/YY format

System Time Default

This will default the current Windows system time using the format specified by the data item used for the prompt. Format is:

@TIME(;O)

@TIME(;O) means default equals current time

Translate Default - Displaying Data From Other Tables

This will extract data from another table; i.e., 'translate' the data. Format is:

@T,table,exp1,exp2,exp3,(exp4)(;O)

e.g.: @T,STATES,ID,3,NAM

Where:

Table is the name of a database 'translate table'.

exp1 is the column/field name for the translate table where indexed data is found.

exp2 is the current prompt number or column/field name where the key for the indexed table data (i.e., data to match on) can be found; value must be a number lower than the current category number.

exp3 is the column/field name for the translate table which contains the data to be retrieved.

exp4 an optional expression used only if exp2 is an 'unlinked' textbox field (i.e., not a database field for which RFgen can determine a data type); leave blank if exp2 data is numeric; let exp4 be a single quote " ' " if exp2 data is a string.

@T,STATES,ID,3,NAM retrieves a state name from a table called STATES. Here ' ID' is an indexed column/field name for the STATES table, and 'NAM' is the column/field name that contains the state name. Data from the current application, in prompt '3', provides the value of ID to use for the indexed search of the STATES table

Concatenation Default

This will concatenate any items specified, and default the resulting string. Valid items are entry categories or items enclosed by quotes. Format is:

@C,exp1,exp2,...,expN(;O)

@C,4,5,6,'***' means default equals prompt '4' value, joined with the prompt '5' value, joined with the prompt '6' value, joined with the text string '***'. Since no ';O' is present, the data will be entered and the application will continue at the next prompt

Character String Extraction Default

This will allow the default to be a portion of a previously entered input. Format is:

@SE,exp1,exp2,exp3(;O)

Where:

exp1 = entry category for the extraction.

exp2 = starting position of character string.

exp3 = length of string.

@SE,3,4,5;O means default equals data entered at prompt '3', starting at the '4'th character, and continuing for '5' characters

Calculation Defaults

This will allow the default to be the result of an on-line calculation. Calculations may be performed using previously entered entry categories. Calculations are performed from left to right, and only arithmetic data is allowed. Format is:

@=calculation(;O)(;precision 'n') n=0 to 4

Where:

+ = addition

- = subtraction

* = multiplication

/ = division

@=6+7*"1.5";;2 means default equals the result of adding data from prompt '6' and '7', then multiplying the result by '1.5', and displayed using '2' decimal points.

All calculations are rounded to the precision selected after each calculation. If no precision is specified, then '4' decimal precision is used

Image Name Retrieval

In the case image controls, the Defaults property is filled in with the name of the image resource that has been loaded into the prompt.

Last/Prior Entry Default

This will default the last/prior data value, entered in the previous use of this prompt. If the default value is empty then the AllowChange parameter will be ignored. Thus a default of "@Last" will stop and allow input on the first pass and skip the field on all subsequent passes. The intended behavior of "@Last" is to maintain the last entered value until the application is exited.

Format is:

@LAST(;O)

Lists (Listing Choices) Default

This default is used to list multiple specific choices, from which 1 item may be selected. Items selected appear in a 'List' on the device. Format is:

@list,heading,item,item,item,item,…

Where 'heading' is the column heading name to print.

@list,Colors,RED,WHITE,BLUE allows the user to select 1 of the 3 colors

Skip the Current Entry Default

This default is used to conditionally skip the current prompt. Format is:

@SKIP,exp1,exp2,exp3

Conditional Operators (exp1,exp2,exp3) are:

exp1 = the prompt number

exp2 = a conditional operator from 1 of the following:
= for an equal to comparison
for a not equal to comparison
> for greater than
< for less than
M for a matches comparison

nM for a not matches test
I: exp4 exists in exp2 (same as NC)
C: exp2 exists in exp4

NC: exp4 exists in exp2

exp3 = the prompt number or string value (in quotes) to match on.

@SKIP,2,=,'N' means to skip the current prompt if the data entered at prompt '2' equals 'N'.

Note in this example that exp1="2", exp2="=", and exp3="'N'"

Set Default

This default is used to validate data before inserting the specified default value. Format is:

@Set,exp1,exp2,exp3,exp4

Conditional Operators (exp1,exp2,exp3,exp4) are:

exp1 = the value that will become the default

exp2 = the parameter to be compared to exp4. This can be a literal, a prompt number, or the RSP variable.

exp3 = a conditional operator from 1 of the following:
= for an equal to comparison
for a not equal to comparison
> for greater than
< for less than
M for a matches comparison
nM for a not matches test
I: exp4 exists in exp2 (same as NC)
C: exp2 exists in exp4

NC: exp4 exists in exp2

exp4 = the second parameter to be compared to exp2. This parameter can be a literal or an edit such as NUM and ALPHA.

@Set,'Hello',2,=,'100260'

This sets the default to 'Hello' if prompt 2 is equal to the string value 100620.

@Set,'Hello','XYZ',#,ALPHA

This sets the default to 'Hello' if the string literal XYZ is not an ALPHA string. In this case, the default would never be used.

@Set,'Hello','3',C,'12345'

This sets the default to 'Hello' if the second parameter (3) is contained within the forth parameter (12345).

SQL Statement (List) Default

This default is used to develop multiple selection items from your database so that 1 may be selected. Format is:

@sql, statement

(where statement is a valid SQL select statement)

The heading that appears is the database column name.

@sql, select PONum from OpenPO would select the purchase orders numbers column, from open purchase orders table 'OpenPO'.

A List, when used, clears the existing screen and lists selected data items (vertically) on the display screen. The RFgen 'List' display feature is a 'full screen display'. The user then selects 1 of the items in the list. There are 3 examples of list creation which follow.

1. For predefined/known list box items use an '@list' default:

@list,heading,item.1,item.2,item.3,…,item.n

Here: '@list,' indicates to RFgen that the statement is a 'default' parameter 'heading' is a name for the list that prints at the top of the box when it appears on a device 'item.1, item.2,…' are a listing of selection items to appear in the list.

e.g. - @list,Colors:,RED,WHITE,BLUE means to display a list with a heading of 'Colors:', and 3 selection items RED, WHITE, and BLUE to appear in the list.

2. For selection items which come from a database table, use an '@sql' default:

An SQL statement as an application 'default property' (for the prompt that will use the list) may be used to select the necessary data from your database, as illustrated here:

@sql,select fieldname1 from tablename where fieldname2 ='%n'

Here: '@sql' indicates to RFgen that the statement is a 'default' parameter

'fieldname1' is the name of the table field/column to be displayed

'tablename' is the name of a database table which contains the fields/columns

'where fieldname2=' allows selection of certain data only

'%n' indicates data entered at prompt 'n' will be used for selection purposes.

@sql,select OrderNO from PObooks where PartNO ='%1' means to create a list box of order numbers from table 'PObooks' where the part number was entered at prompt number 1.

3. Alternatively, for database items, use a VBA script, e.g.:

```
Public Sub FieldName_GotFocus (Rsp As String, AllowChange As Boolean)

  Dim SQL As String

  SQL = "Select fieldname1 from tablename where
     fieldname2 = '%n'"
```

```
  Rsp = DB.MakeList(SQL)
  AllowChange = True
End Sub
```

Here, the RFgen VBA extension 'DB.MakeList' is used in a GotFocus event to make a list from the results of the specified SQL statement. The results are passed to a prompt default variable named 'Rsp'. A variable called AllowChange is set to True, meaning override allowed.

User Default

This default will put in the login name of the user. Format is:

@User

Edits Property Validation Options

Validations/edit checks are available to test if data meets a certain criteria (e.g., the data entered is numeric). A validation property is available for each prompt (i.e. TextBox or Memo controls that accept data entries) and validations are entered in the Properties window of graphical control. Multiple validations/edits are allowed. Select the Edits property, and select the drop down arrow. A multiple line input box will display. Enter one edit per line. *Note that as soon as an entry passes an edit, any subsequent edits are ignored*.

If an entry has no validation criteria, leave its edit property blank.

Text (Character String) Validation

This will test for an exact match between the data entered and the text edits required. A text edit may be entered either with or without quotes.

Format is:

'string' (in single quotes)

'CA' means the user must enter CA to pass this edit test

Pattern Match Validation

This will test the data entered to see if it matches the pattern specified. Format is:

```
xA or xN
```

For example, 2A means that the data entered must have the format 'AA' (i.e., match 2 alphabetic characters, for example: 'OR'). 4N means the data entered must be in the form 'NNNN' (i.e., 4 numbers; for example: '1234').

4N means the user must enter a valid 4-digit number.

Pattern Not Allowed Validation

This will test the data entered to assure that it does not match the pattern specified. Format is:

#PATTERN

>#4N means the user may enter anything but 4 numbers (i.e., not 4 numeric characters)

#Hello means the user may not enter Hello in the prompt.

Alpha Only Validation

This will test the data entered to assure that it only contains alpha characters (A-Z and a-z). Note: no spaces are allowed.

Format is:

ALPHA

Numeric Only Validation

This will test the data entered to assure that it only contains numeric characters (0-9), and the special characters '.' (period), '+' (plus), and '-' (dash). Note: no spaces are allowed.

Format is:

NUM

Integer Only Validation

This will test the data entered to assure that it only contains an integer number (no decimal). Note: no spaces are allowed.

Format is:

INT

Greater Than Validation

This will test the data entered to see if it is greater than a specified value.

Format is:

>number or >=number

>99 means that the user must enter a number greater than 99.

Less Than Validation

This will test the data entered to see if it is less than a specified value.

Format is:

><number or <=number

<99 means that the user must enter a number less than 99

Not Equal To Validation

This will test the data entered to assure that it does not equal a specified value. This is a string comparison only.

Format is:

#'stringvalue'

>#'99' means that the user must enter a string that does not equal '99'. Since prompts accept data as strings by default this validation would work with numbers while treating them like strings.

Data Range Validation

This will test the data entered to see if it is within a specified range. This is a numeric comparison only.

Format is:

RG/exp1,exp2

Where:

exp1 = starting number for range.

exp2 = ending number for range.

RG/1,100 means that the user must enter a number between 1 and 100, inclusive

Date Validation

This will test the data entered to see if it is a proper date.

Format is:

DATE

Entering MMDDYY, MMDDYYYY, MM-DD-YY, MM/DD/YY among others are all valid entries.

Index Validation

This will test the data entered to see if it can be found within the edit string.

Format is:

I:string

I:YN means data must be either 'Y' or 'N', or 'YN'.

Contains String Validation

This will test the data entered to see if it contains the specified edit string.

Format is:

C:string

>C:abc means data entered must contain the characters 'abc' somewhere within it

Not Condition Validation

This will test the data entered to see if it does not match a given condition.

Format is:

NC,exp1,exp2,exp3

Conditional Operators (exp1,exp2,exp3) are:

exp1 = the parameter to be compared to exp3. This can be a literal (in quotes, even if numeric, eg '3'), a prompt number, or the RSP variable.

exp2 = a conditional operator from 1 of the following:
= for an equal to comparison

for a not equal to comparison
> for greater than
< for less than
M for a matches comparison
nM for a not matches test
I: exp3 exists in exp1 (same as NC)
C: exp1 exists in exp3

NC: exp3 exists in exp1

exp3 = the second parameter to be compared to exp1. This parameter can be a literal or an edit such as NUM and ALPHA.

NC,2,=,'100260'

This checks prompt 2 to see if it is equal to the string value 100620. If it is NOT, the edit is satisfied.

NC,'XYZ',#,ALPHA

This compares if the string literal XYZ is not an ALPHA string. Because XYZ IS an alpha, this edit is satisfied.

>NC,'3',C,'12345'

This edit is NOT satisfied because the first parameter (3) is contained within the third parameter (12345).

Branch/Goto Validation

>Technically, this is not a validation. Allows the display to Goto a 'downstream' prompt, based upon the data 'RSP' entered.

Format is:

@GOTO,prompt#(,conditions)

@GOTO,5,RSP,=,"99" means to go to prompt 5 if the response at the current prompt equals "99". See the @SKIP default parameters for examples of 'conditions'

Strip (Data Entry) Validations

>Validates that 'str' is there, then strips data 'str' from the entry.

Format is:

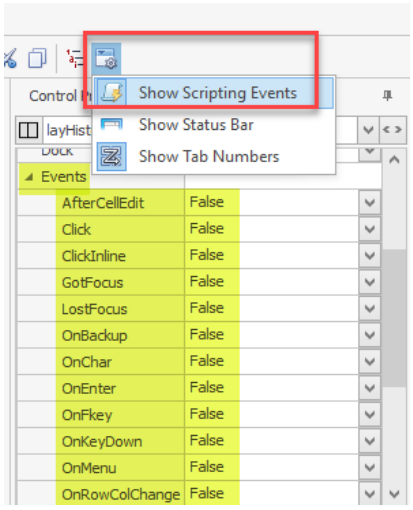
@STRIP,P,str to strip data prefix 'str' if it occurs

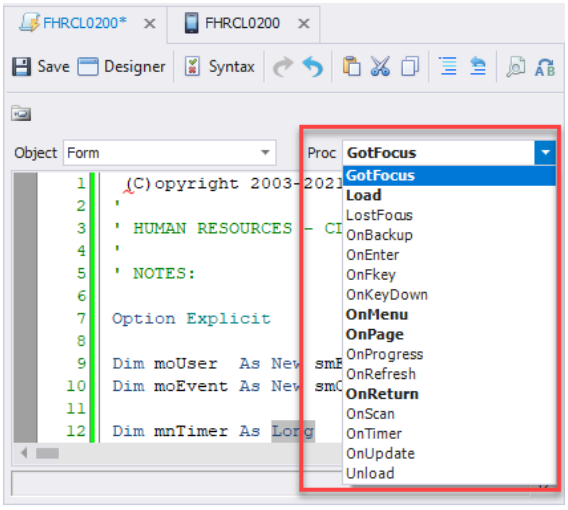
@STRIP,S,str to strip data suffix 'str' if it occurs

@STRIP,C,str to strip the first occurrence of the data 'str' if it is contained in the data.

@Strip,P,NBR: means to strip the character string 'NBR:' if it prefixes the data.

Events Property





You can access pre-scripted modules for managing user actions or device operations from:

* Application Designer: Options Menu > Show Scripting Events.

* Application Script: > [Select Object] > Select process from Proc drop down menu events. (Picture of events in Script designer.)

* RFgen Client Events - The are device events, they are not associated with graphical control object-s/prompts, and are source from the RFgen Scripting Modules. (Picture of RFgen Client Events).

The RFgen Client events from the RFgen.bas scripting module include: Initialize, InRange, OnConnect, OnDisconnect, OnLocale, OnMenu, OutOfRange, and Terminate.

The Application object-based/ script-based events are:

AfterCellEdit, Click, ClickInline, GotFocus, Load, LostFocus, OnBackup, OnChar, OnDisconnect, OnEnter, OnEscape, OnFkey, OnKeyDown, OnLocale, OnMenu, OnPage, OnProgress, OnReadData, OnRefresh, OnReturn, OnRowColChange, OnRowColClick, OnScan, OnSearch, OnSendComplete, OnSendProgress, OnTimer, OnUpdate, OnVocollect, and Unload.

Events that are global to the form are: GotFocus, Load, LostFocus, OnBackup, OnEnter, OnFKey, OnMenu, OnPage, OnProgress, OnRefresh, OnReturn, OnScan, OnTimer, OnUpdate, and Unload.

Events that are global to a page are: OnSwipe.

If you are working with an event that is associated with an object (also called a prompt), you can script a prompt to default to a data value (i.e. set a string to equal '1000') or data format (date/time format or trun-cate text strings).

For additional information on available parameters, see Setting Default Values in the RFgen Developers Reference Guide.

AfterCellEdit

The event occurs when the user makes a change to the contents of a cell in the DataGrid control. The event passes the row and column that was changed, the value it was changed to, and the Boolean option to cancel what the user did and set the cell back to its original value.

Group: Events

Applies to: Prompts (Graphical Mode Only)

Syntax: AfterCellEdit(Row, Col, Rsp, Cancel)

Row	(Long) is the row of the cell changed
Col	(Long) is the column of the cell changed
Rsp	(String) is the value entered by the user
Cancel	(Boolean) is set to True to discard the user's change

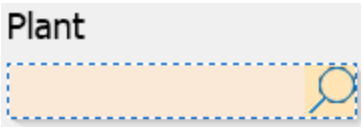
Click

The Click event occurs when the user clicks on a prompt or button with a mouse or pen. It is typically used to determine if a user has pressed a button on the screen.

Group: Events

Applies to: Applications, prompts

ClickInLine



The ClickInLine event occurs when the user taps on a button (i.e. the Search icon) that's inside a control which also accepts other actions like data entries. The ClickInLine event is typically used to determine if the button or icon was tapped.

Group: Events

Applies to: Prompts (Graphical Mode Only)

GotFocus

The GotFocus event occurs as soon as the user (either moving forward or backing up) reaches the data entry field. This event is typically used to generate a default value that the user can either accept or reject.

Group: Events

Applies to: Applications, Prompts

Syntax: GotFocus(Rsp, AllowChange)

Rsp (String) is the default value to display on the Client device.

AllowChange (Boolean) is set to True to allow the user to override the default, and is set to False to generate an OnEnter event bypassing user interaction at this field.

KeyPress

This event is only available for Screen Mapping (i.e. ASA 400 / green screen terminals).

Note: The majority of the screen mapping commands that handle the placement of text to and from a green screen / legacy system have been obsoleted in RFgen 5.2.

The KeyPress event occurs when the user presses and releases a key on the keyboard. This event is typically used to capture incoming keystrokes and perform some action based upon user input.

Group: Events

Applies to: Applications, Prompts

Syntax: KeyPress(KeyAscii)

KeyAscii (Integer) is the ASCII value of the character pressed on the client device.

Initialize

The Initialize event occurs when an RFgen client is initially loaded. It will occur only once, and is typically used to initialize variables, open additional database connections or create links to ActiveX objects.

This event is accessed from the **Solution Explorer > Scripting Modules -RFgen.bas**. Set the Object to "RFgen" and Proc to "Initialize".

Group: Events

Applies to: Applications (RFgen.bas)

Syntax: Initialize()

InRange

The OnInRange event occurs when a Mobile Client notices that there is an IP address available on the network adapter. This event does not execute when using the Roaming Client because the OnConnect will execute anyway.

This event is accessed from the Solution Explorer > Scripting Modules: RFgen.bas.

Set the Object to "RFgen" and Proc to "InRange".

Group: Events

Applies to: RFgen (RFgen.bas)

Syntax: OnInRange()

Load

The Load event occurs when the application is initially loaded. It will occur only once per form and is typically used to initialize variables.

Group: Events

Applies to: Applications (RFgen.bas)

Syntax: Load()

Lost Focus

The LostFocus event occurs when the prompt that had the focus is giving it up to another prompt. This would occur after the OnEnter event is finished and before the next prompt's GotFocus event is executed.

Group: Events

Applies to: Application, Prompts

Syntax: LostFocus()

OnBackup

The OnBackup event occurs when the prompt with the focus receives a command to go back to a previous prompt, as if the up arrow key was pressed.

Group: Events

Applies to: Application, Prompts

Syntax: OnBackup(Cancel)

Cancel (Boolean) is set to True if the backup movement should be stopped. Set it to False or do not change the default to allow the backup to continue.

OnConnect

The OnConnect event occurs when the Mobile Client in a roaming state and it automatically discovers it has connectivity to a network, and then makes a connection to the RFgen server. The event only executes after 10 seconds of continuous connectivity to the RFgen server.

To maintain data integrity it may be a good idea to include the Server.SyncApps and Server.SendQueue in this event.

This event is accessed from the **Solution Explorer > Scripting Modules:** RFgen.bas. Set the Object to "RFgen" and Proc to "OnConnect".

Group: Events

Applies to: RFgen (RFgen.bas)

Syntax: OnConnect()

OnCursor

The OnCursor event occurs whenever a cursor (arrow) key is pressed. This event is typically used to allow users to page through a list of data. The Up arrow is typically used to backup to the previous field (system default); however, to bypass system processing of this event, set Cursor = "" (null).

Group: Events

Applies to: Application, Prompts

Syntax: OnCursor(Cursor)

Cursor (String) is the character value of the key pressed. Possible values are ('U'p, 'D'own, 'R'ight, and 'L'eft).

Example:

```
Public Sub PartNo_OnCursor(Cursor As String)
 On Error Resume Next
 If Cursor = "U" Then
   'your logic
 End If
End Sub
```

OnDisconnect

The OnDisconnect event occurs when the Mobile Client in a roaming state and disconnects from the RFgen server because the client moved out range from the network. This event does not execute when the Server.Disconnect command is issued.

This event is accessed from the Solution Explorer > Scripting Modules: RFgen.bas. Set the Object to "RFgen" and Proc to "Initialize".

Group: Events

Applies to: RFgen (RFgen.bas)

Syntax: OnDisconnect()

OnEnter

The OnEnter event occurs when the user presses the Enter key on the Client device, or a default value specifies that no user input is allowed. This event is typically used to validate data entered, and/or adjust the display on the device. To reject the entry made by the user, set Cancel = True and RFgen will force the user to re-enter the field.

The behavior of the OnEnter event can be set to execute on prompts regardless of whether the prompt can receive focus by changing your server's environment settings or by using the InputState VBA Language extension.

For more information see the topic "Use Legacy OnEnter Event" under Configure Environment Settings, or review the code "InputState" in the Developers Reference Guide.

Group: Events

Applies to: Application, Prompts

Syntax: OnEnter(Rsp, Cancel, Message)

Rsp	(String) is the value entered/scanned by the user on the Client device.
Cancel	(Boolean) is set to False to accept the data entered and move to the next prompt, or is set to True to fail the edit check and force the user to re-enter data at the current field.
Message	(String) is a message to display on the Client device.

Supported Versions: 5.0, 5.1, 5.2 and all newer versions.

Notes: For more information see the topic "Use Legacy OnEnter Event" under Configure Environment Settings. "Prompt.InputState" for the OnEnter event was added in 5.2.2.0.

OnEscape

The OnEscape event occurs when the Escape key is pressed. This event is typically used to capture an incoming Escape keystroke and perform some action based upon user input.

Group: Events

Applies to: Application, Prompts

Syntax: OnEscape()

OnFkey

The OnFkey event occurs whenever a function key is pressed. In older versions of RFgen (5.1 and older), the Function keys F1-F4 triggered pre-defined system events. To bypass system processing of these events set Fkey = '0' (zero) and RFgen will ignore the event. In RFgen 5.2 and higher, the OnFKey event fires and then the key is evaluated for the configured functionality. If you perform some action via script to override the configured logic, RFgen needs a mechanism to stop processing at the event execution. The Handled variable was introduced to stop the processing. Set it to Handeld = True if you would like processing to stop.

Group: Events

Applies to: Application, Prompts

Syntax: OnFkey(KeyId, Handled)

KeyId (Long) is the value of the key pressed. Possible values range from (1-10).

Handled (Boolean) is used to stop processing.

Example:

```
Public Sub Form_OnFkey(KeyId As Long)
 On Error Resume Next
 If KeyId = 5 Then
' In this example we are using the F5
' key to execute logic.
' your logic
 End If
End Sub
```

OnLocale

The OnLocale event occurs after OnConnect (only when a client application makes a connection) and passes in the locale number based on the client device's location. In the case of the United States, number 1033 is returned. Based on this value you may set global values or default login forms (See App.ChangeLoginForm)

Group: Events

Applies to: Application, Prompts, and RFgen.bas

Syntax: OnLocale(nDeviceLocale)

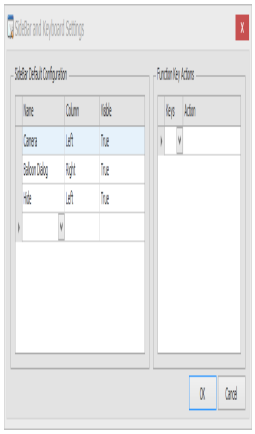
nDeviceLocale (Long) is the value of the client device's locale.

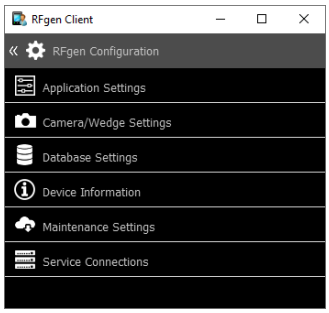
Example:

```
Public Sub RFgen_OnLocale(ByVal nDeviceLocale As Long)
```

```
On Error Resume Next
Select Case nDeviceLocale
 Case 1033
  App.ChangeLoginForm("RFLoginEnglish")
 Case Else
  App.ChangeLoginForm("RFLogin")
End Select
End Sub
```

OnMenu





The OnMenu event occurs when the user clicks on a menu item that is used outside of a mobile app (form). If one of the default actions is clicked, no code is required. For example, a button that brings up the RFgen Client Configuration menu.

This event is accessed from the Application Control Panel, Applications Script View, or the Solution Explorer > Scripting Modules: RFgen.bas. After RFgen.bas is selected, set the Object to "RFgen" and Proc to "OnMenu".

Group: Events

Applies to: RFgen (RFgen.bas), Application, and Prompt

Syntax: OnMenu (sMenuAction)

sMenuAction (String) is the Action name of the button clicked by the user.

Example:

```
Public Sub Form_OnMenu(MenuAction As String)
 On Error Resume Next
 MenuStrip.Show(False) ' hide the menu after a click
```

```
End Sub
```

OnPage

The OnPage event occurs when the user clicks on a page number icon/prompt that changes the focus to the page that was selected.

Group: Events

Applies to: Application Form Events

OnReadData

The OnReadData event occurs whenever data is successfully retrieved from a database by an RFgen application. This event is typically used to force an immediate repaint of the screen to display all linked prompts values to the user.

Group: Events

Applies to: Application

Syntax: OnReadData(TableName)

TableName (String) is the name of the table that was successfully queried.

Example:

```
Public Sub Form_OnReadData(TableName As String)
 On Error Resume Next
 Screen.Refresh ' Repaint the current screen
End Sub
```

OnRefresh

The OnRefresh event occurs when the screen is completely repainted (as when the user presses the F2 key). It is typically used to redisplay information to the user that was displayed via Screen.Print statements.

Group: Events

Applies to: Application Form event

Syntax: OnRefresh()

OnReturn

The OnReturn event occurs when the user returns from a called application that had its return flag set to True. It is typically used to process data from the called application.

Group: Events

Applies to: Application

Syntax: OnReturn(Name)

Name (String) is the name of the called application that is returning.

OnRowColChange

The OnRowColChange event occurs when the focus moves between a column or row in a control that supports them.

Group: Events

Applies to: Application, Prompts

Syntax:

```
OnRowColChange(Row, Col)
```

Row (Long) is the value of the row being selected.

Col (Long) is the value of the column being selected.

OnRowColClick

This event is triggered when a prompt supporting multiple columns and rows receives a click event. The cell (row and column) are captured in the event.

Group: Events

Applies to: Applications, Prompts

Syntax:

RowColClick(Row, Col)

Row (Long) the row on the control that was clicked.

Col (Long) the column on the control that was clicked.

OnScan

The OnScan event occurs when RFgen identifies a preamble string in the data stream coming from a client device. This event is typically used to validate that data was scanned, or to parse the data (PDF, RFID, etc.) into a more usable format. To reject the data scanned, set the ScanData = "" and RFgen will remove it from the data stream and prevent it from being entered into the current prompt.

Group: Events

Applies to: Application, Prompts

Syntax:

OnScan(ScanData)

ScanData (String) is the value scanned by the user on the client device.

OnSearch

The OnSearch event occurs only if a linked or unlinked text box has code in the OnSearch event and the user clicks on the generated search command button. This is only possible in graphical mode. Any VBA code can be placed in this event, even if it is not specifically related to searching a data source.

Group: Events

Applies to: Text prompts

Syntax:

OnSearch(Rsp, Cancel)

Rsp (String) The value in the textbox if any

Cancel (Boolean) Set to True to NOT run the OnEnter event and put the focus back on the prompt

OnTimer

The OnTimer event occurs at a user-defined interval when it is enabled. This event is used to perform some action based upon defined time interval. For example, how long it takes to complete a task.

Group: Events

Applies to: Application Form

Syntax: OnTimer()

Example:

First the timer interval is set and the timer is enabled.

```
Public Sub Form_Load()
 On Error Resume Next
 App.TimerInterval = 1000
 App.TimerEnabled = True
End Sub

'OnTimer event is given some logic.
Public Sub Form_OnTimer()
 On Error Resume Next
```

```
'your logic
End Sub
```

OnUpdate

The OnUpdate event occurs after the last prompt has been entered and just prior to RFgen updating the database. This event is typically used to validate that data was entered and process additional updates.

Group: Events

Applies to: Application

Syntax:

```
OnUpdate(Cancel)
```

Cancel (Boolean) is used to cancel the update.

OnVocollect

This event is only used with Vocollect data collection hardware. When the Vocollect device activates a Vocollect task on an application, a recordset is sent to RFgen for processing. If RFgen needs to communicate back to the user, RFgen passes to the Vocollect device another recordset.

Group: Events

Applies to: Applications where Vocollect hardware is used.

Syntax:

rsIn a DataRecord object that captures the collected data coming in from the Vocollect device

rsOut a DataRecord object built in the RFgen VBA code that is sent to the Vocollect device

Example:

```
OnVocollect(rsIn, rsOut)
Public Sub prTaskODRUpdateStatus_OnVocollect(ByVal rsIn As DataRecord, ByVal rsOut As DataRecord)
 On Error Resume Next
 '
 rsIn.Param("DateTime")
 rsIn.Param("SerialNumber")
 rsIn.Param("Operator")
 rsIn.Param("AssignmentId")
 rsIn.Param("LocationId")
```

```
  rsIn.Param("UpdateMode")
  rsIn.Param("UpdateStatus")
  '
  rsOut.Param("ErrCode") = 0
  rsOut.Param("Message") = ""
End Sub
```

OutOfRange

The OutOfRange event (previously called "OnOutOfRange event") occurs when a Mobile notices that the IP address is no longer available on the network adapter. This event does not execute when using the Roaming Client because the OnDisconnect will execute anyways.

This event is accessed from the Solution Explorer > Scripting Modules: RFgen.bas. Set the Object to "RFgen" and Proc to "OutOfRange".

Group: Events

Applies to: RFgen events (RFgen.bas)

Syntax: OutOfRange ()

Terminate

The Terminate event occurs when an RFgen client is terminating. It will occur only once, and is typically used to close manually opened database connections or release links to ActiveX objects.

This event is accessed from the Solution Explorer > Scripting Modules: RFgen.bas. Set the Object to "RFgen" and Proc to "Terminate".

Group: Events

Applies to: RFgen events (RFgen.bas)

Syntax: Terminate()

Unload

The Unload event occurs when the application is terminating. It will occur only once, and is typically used to release any system resources manually opened that are specific to this application.

Group: Events

Applies to: Application Form Events

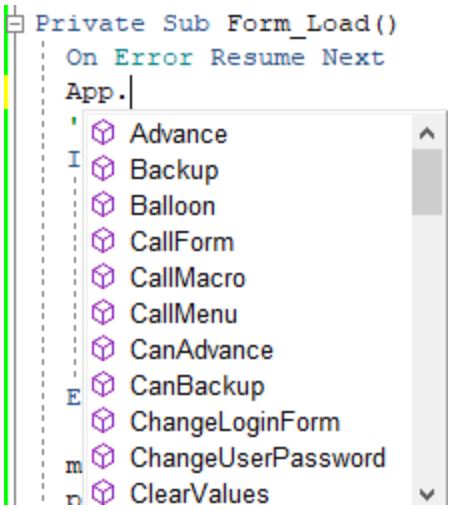
Syntax: Unload()

VBA Language Extensions

Visual Basic Assembly (VBA) Language Extensions are comprised of other categories of functions which are used to manage more aspects of how an application looks or behaves. For example, a VBA Language Extension could be used to "Call the Form" of a specific application, or control a client device data display, manipulate database records, communicate with other databases, control stored procedures, communicate with attached or Windows-based printers and much more.

Some extensions work with the majority of prompts (also called "controls") while others will only work with specific types of prompts.

RFgen provides a VBA Language Extension .Net style dropdown menu



that helps you complete a script at design time and view the VBA language extension options that are available.

Types of Extensions: Application-Based, Database Related, Database Storage Procedures, Data Record Objects, Device (Android, iOS and Win/CE), Device Object (Windows Mobile/CE), Dynamic Array, Embedded Procedure Object, Enterprise Resource Planning, JDE Processing Option, MQTT Object, Menu Strip, Parameter Object, Printer Extensions, Prompt-Specific, Screen Display, Screen Mapping, SearchList Object, Server-Based, "SMTP", Socket Object, Soft Input Panel, Stored Procedure, System Error, System Level, Transaction Management, and Web Object.

You can also refer to the User Guide or Help for detailed descriptions of the parameters associated with the various VBA extensions.

Application-Based Extensions

Application-based language extensions are a group of general commands that can be executed against an application or transaction macro. These types of commands are not specific to mobile devices, screen mapping or other specific functions.

The application based extensions are:

App.Backup, App.Balloon, App.CallForm, App.CallMacro, App.CallMenu, App.CanAdvance, App.ChangeLoginForm, App.ChangeUserPassword, App.ClearValues
App.ClientType, App.ConnectionAvailable, App.Display, App.ExecuteMenuSelection, App.ExitForm, App.ExitSession, App.GetString, App.GetValue, App.IpAddress, App.Locale, App.LogError, App.LogErrorEx, App.MacroName, App.MakeList, App.MenuName, App.MsgBox, App.FormName, App.PageNo, App.PageCount, App.PromptCount, App.PromptNo, App.Reload, App.SendChar, App.SendKey, App.SetDisplay, App.SetFocus, App.SetMenu, App.SetMenuCaption, App.SetPage, App.SetValue, App.Showform, App.ShowList, App.ShowWait, App.Signout, App.Sleep, App.Theme, App.TimerEnabled, App.TimerInterval, App.User, App.UserName, App.UserProperty, and App.UserRoles.

Balloon

This command will create a pop-up message on the screen that can last indefinitely or for some number of seconds.

As long as the App.Balloon call passes in either a valid Text, Title, or IconId, then it will be displayed with whatever combination of buttons requested. Original default behavior is kept the same when calling App.Balloon with just the original parameters filled in.

Group: Application-Based Extension

Syntax: App.Balloon(sText, nDuration)

sText (String) the message to display.

nDuration (Long) the number of seconds the message should display. Set to zero to clear the balloon or -1 for it to display indefinitely. If -1 is used, use a Balloon set to 0 duration to remove it.

Note: As of 5.2.5.x, "Balloon" was added as an object so it does not have to be used with the App object. The following parameters were added in 5.2.5.x

The format for calling this function with all optionals filled in will look like:

App.Balloon(sText, iDuration, sTitle, sImg, bClose, bAnimate, dWidthPercent, iEndLocation, iStartLocation)

Parameter	Description
Animate	bool value as to whether the balloon should animate moving across the screen. If EndLocation isn't set then it will move to the top of the form so that it is just below the form header, and if StartLocation is not set then it starts at the original centered location. Default value is false.
CloseButton	bool value as to whether to have a close button added to the balloon as an additional column. Default value is false
EndLocation	int value for the form-coordinate Y location that the balloon's animation will stop at. If animation is false then this has no effect and a default value of -1

Parameter	Description
	means that it will stop at the form's heading area
IconId	the name of the image to use as an icon from RFImages. This is optional and the default value of empty string makes the balloon draw with no image.
StartLocation	int value for the form-coordinate Y location of where the balloon should first be drawn at with the top of the dialog aligned to that location. A default value of -1 has it starting at the original location of where it is centered at.
Title	the message's title text, default value is empty and if left empty then the balloon's layout will only use one row. The title's text is set to bold style and +2 font size compared to the theme size
WidthPercent	double value restricted between 0.0 and 1.0 that is the percent of the form's width to lock the balloon's size to so that it then only autosizes the height of the balloon. The default value of 0.0 lets it autosize the width as well as the height like it originally did.

Example:

```
'This will cause a balloon message with a spinning icon to display "Transaction Pro-
cessing" for 5 sec then the icon stops spinning and the message disappears.
App.Balloon("Transaction Processing...", 5)
```

Example 2

```
App.Balloon("An error has occurred", 3, "Error!", "ErrorIcon", True, True, 0.6, 200, 0)
```

Version Supported: RFgen 5.0 and newer.
Note: RFgen 5.24 added these parameters: Animate, CloseButton, EndLocation, IconId, StartLocation, Title, and WidthPercent.

CanBackup

This function can be used to determine if the user can backup or not. For example, if the user is on a the first prompt on a form there isn't any other prompt to back up to, then the value will return false. If there is a prompt to backup to, then the return is True. This can also be used to determine if the user will be able to exit the form if there is or isn't another prompt.

Group: Application-Based Extension

Syntax:

App.CanBackup( )

Example:

```
Private Sub TextBox1_GotFocus(ByRef Rsp As String, ByRef AllowChange As Boolean)
```

```
On Error Resume Next
    Dim bool As Boolean
      bool = App.CanAdvance
        App.MsgBox "Can.Advance returns "& bool
        bool = App.CanBackup
      App.MsgBox "Can.Backup returns "& bool
    End Sub
```

Versions Supported: RFgen 5.2.4.2 and newer.

CallForm

This command will transfer the user to another application. There is an option to return from the called application to the previous one, but if another CallForm command was issued to go to a third application first, then as each application is exited, the user will be brought back to each of the previous ones until finally coming back to the original application. There is no limit to the number of "sub" applications that can be called.

The CallForm will fire at the end of the current event that the code is at. The CallForm function can only be called once per an event. If the ReturnFromCall is set to True, then on return, the form will fire the Form_ OnReturn event. If you need to go to a different form as in-line code, then use the App.ShowForm function instead.

Group: Application-based Extensions

Syntax: App.CallForm(ByVal FormName As String, [ReturnFromCall As Boolean = False], [SaveRSP As Boolean = False], [InitScreen As Boolean = False], [FormOptions])

FormName (String) The app form to be called. Select it from the pop-up list of forms when you enter "App.CallForm" and press the Tab key.

ReturnFromCall (Boolean) Optional. True returns to the current application after collecting data in the called application. False remains in the called application. The default is False.

SaveRSP (Boolean) Optional. True retains the value in the current prompt's text field. The default is False.

InitScreen (Boolean) Optional. True deletes the screens memory object prior to going to the application.

FormOptions Optional. Is the whitespace delimited list.

Example:

```
Private Sub Form_OnFkey(ByVal KeyId As Long, ByRef Handled As Boolean) On Error Resume Next
  ' Each CallFrom will take the user to the expected Form after the Exit Sub
    Select Case KeyId
```

```
    Case 1
        ' Simple example, leaving the Calender From will return the user to this app App.CallForm("Calendar", True)
    Case 2
        ' Simple Example, leaving the ErrorPage Form will return user to their menu App.CallForm("ErrorPage")
    Case 3
        ' Complex example, will take user to the ProcessingPage app with the ActionCode and Format options
    ' Will return user to this form when they are done in ProcessingPage app App.CallForm("ProcessPage", True,
                    True, , "-ActionCode=0560 -Format=mmDD-YYYY")
  End Select
End Sub
```

Versions Supported: RFgen 4.0 and newer.

Notes: See also App.ShowForm. The syntax may be different in order versions.

CallMacro

This function is used to call any transaction macro and pass in the required passing parameters. It returns True if the macro was successfully completed. These macros can be created from the Solutions Explorer > Transactions Tree. See also App.MacroName.

Group: Application-Based Extension

Syntax: enValue = App.CallMacro(sMacroName, bQueueOffline, [vParams])

enValue	(enMacroResult) – Values are: MacroFailed MacroNotProcessed MacroQueued MacroSucceeded
sMacroName	(String) – This is the name of the macro to be called.
bQueueOffline	(Boolean) – This determines whether the macro can be queued for later processing if the host is not currently available.
vParams	(Variant) – Optional: A series of passing parameters as required by the selected macro. Note: these parameters are the fields you defined when you wrote the macro.

Example:

```
Dim enValue As enMacroResult
enValue = App.CallMacro("SaveData", True, "Sam", "2")
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

CallMenu

This command will transfer the user to another menu. It is typically used to log a user into the server when bypassing the Logon process. If the application making the menu call was deep within the menu structure, the server will assume that the called menu is the root menu. Going back up 1 level from a called menu will take you to the Login screen if one exists. (Note: the new menu will be called only after the Visual Basic Event has been exited.)

Group: Application-Based Extensions

Syntax: App.CallMenu(sName)

sName (String) is the name of the menu to call as defined by the Menus tree.

Example:

```
App.CallMenu("Sample") ' Calls the "Sample" menu.
App.CallMenu("HR,PO") ' Calls the just these specific menus.
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

CanAdvance

This function can be used to determine if the user proceed to the next prompt by checking if there is another prompt to move to, or if the user is at the last prompt. For example, if the user is on a the last prompt on a form there isn't any other prompt to move forward to, then the value will return false. If there is a prompt to move forward to, then the return is True.

Group: Application-Based Extension

Syntax:

App.CanAdvance( )

Example:

```
Private Sub TextBox1_GotFocus(ByRef Rsp As String, ByRef AllowChange As Boolean)
On Error Resume Next
    Dim bool As Boolean
      bool = App.CanAdvance
        App.MsgBox "Can.Advance returns "& bool
        bool = App.CanBackup
      App.MsgBox "Can.Backup returns "& bool
    End Sub
```

Versions Supported: RFgen 5.2.4.2 and newer.

CanBackup

This function can be used to determine if the user can backup or not. For example, if the user is on a the first prompt on a form there isn't any other prompt to back up to, then the value will return false. If there is a prompt to backup to, then the return is True. This can also be used to determine if the user will be able to exit the form if there is or isn't another prompt.

Group: Application-Based Extension

Syntax:

App.CanBackup( )

Example:

```
Private Sub TextBox1_GotFocus(ByRef Rsp As String, ByRef AllowChange As Boolean)
On Error Resume Next
    Dim bool As Boolean
      bool = App.CanAdvance
        App.MsgBox "Can.Advance returns "& bool
        bool = App.CanBackup
      App.MsgBox "Can.Backup returns "& bool
    End Sub
```

Versions Supported: RFgen 5.2.4.2 and newer.

ChangeLoginForm

This function will set the default login application for the duration of the client's session. The purpose is to allow several different login applications (possibly different languages) to exist at the same time and have the client's device specify which login application should be used. In the RFgen.bas module use the OnLocale event to get the client's location and then use App.ChangeLoginForm in that event to display the proper application for the user.

Syntax: App.ChangeLoginForm (sName)

sName (String) is the default login application name for the connected Client device.

Example:

```
App.ChangeLoginForm("RFLoginSpanish")
App.ChangeLoginForm("RFLoginForkLift")
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

ChangeUserPassword

This command will change the user's password to a new value if the old password is correctly provided. This is a permanent change stored in the system.

Group: Application-Based Extension

Syntax: [bValue =] App.ChangeUserPassword(sOldPwd, sNewPwd)

bValue (Boolean) Optional – is True or False based on the success of the command.

sOldPwd (String) is the user's current password

sNewPwd (String) is the new password

Example:

```
App.ChangeUserPassword("Mike123", "Mike456")
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

ClearValues

This command will erase the values of all fields contained in the specified application. It is used to reset the values for applications that must be called more than once.

Group: Application-Based Extension

Syntax: App.ClearValues([vName])

vName (Variant) Optional – is the name of the application's prompt to erase.

Example:

```
App.ClearValues("Cycle") ' Clears the values on the application titled "Cycle".
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

ClientType

This function will return the type of client connecting in to the server.

Group: Application-Based Extension

Syntax: sType = App.ClientType

sType (String) "TE", "GUI", "MOBILE", "XML", "TMQUEUE", or "TMEVENT"

ConnAvailable

This function will return a Boolean (True / False) response that indicates whether or not the specified data connection is currently working.

Group: Application-Based Extension

Syntax: bValue = App.ConnAvailable(vSource)

bValue (Boolean) equals True if the server is currently connected to the specified source; equals False if it is not available

vSource (Variant) is the string representation or the numeric representation of the data connection

Example:

```
Dim bCheck As Boolean
bCheck = App.ConnAvailable(1)
bCheck = App.ConnAvailable("RFSample")
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

Display

This property returns the name of the active display and is also used to change the active display immediately to a different display at runtime. The App.SetDisplay is similar but it does NOT return the name of the active display.

Group: Application-Based Extension

Syntax: App.Display=French

Example:

```
Private Sub Form_Load()
  On Error Resume Next
  App.SetDisplay("Standard1")  'This is the default display.
    sName=App.Display   'Returns the name of the active display
    TextBox1.Text=sName   'Displays the name of the active display which is Standard1
  End Sub
    Dim sName As String
    Dim sOtherDisplayName As String
```

```
Private Sub BtnDisplayName_Click()    'First click changes the display; Second shows the name of changed dis-
                                      play
  On Error Resume Next
    App.Display="NonStandard"    ' Sets display to the NonStandard Display (which has a yellow background)
      If App.Display = "NonStandard" Then
      App.MsgBox(sOtherDisplayName)
    TextBox1.Text=sOtherDisplayName
    sOtherDisplayName=App.Display
  Else
 App.MsgBox("Failure")
 End If
End Sub
```

Version Supported: RFgen 5.2.3.0 and newer.

ExecuteMenuSelection

This command is used to select a menu item based on the index number or name of the item in the menu. This command could be used to automate menu selections.

Group: Application Based Extensions

Syntax: App.ExecuteMenuSelection(vSelection)

vSelection (Variant) either the menu index or display text of the menu prompt's item to be selected and executed.

Example:

```
App.ExecuteMenuSelection("Inventory Transfer")
App.ExecuteMenuSelection(1)
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

ExitForm

This command exits the current application. The VBA event where this is called must exit before this statement is carried out. Therefore it is always best to use an Exit Sub statement immediately after this command.

Group: Application-Based Extension

Syntax: App.ExitForm

Example:

```
Private Sub btnExit_Click()
On Error Resume Next
App.ExitForm
End Sub
```

See also: App.Signout.

ExitSession

This command exits the current device session completely, the same as entering 'Q' at the original login screen. The VBA event where this is called must exit before this statement is carried out. Therefore it is always best to use and Exit Sub statement immediately after this command.

Group: Application-Based Extension

Syntax: App.ExitSession

See also: App.Signout.

FormName

This property contains the name of the application and will return the application's name to any process that calls for it as a string. This property is read-only.

Group: Application-Based Extension

Syntax: App.FormName

Example:

'This example script displays the name of the application in a label when the button is clicked.

```
Dim ApplicationName As String
 Private Sub btnDisplayAppName_Click()
    ApplicationName = App.FormName
  lblApplicationName.Caption = App.FormName
 End Sub
```

History App.Name changed to App.FormName in RFgen release 5.2.4.3.

GetString

This function will return the Translation Text from the Text Resource (the files containing the TextIds, strings, and translated values for each string) that was saved as part of the application configuration. There are two ways this command can be used. Either enter the Text ID and optionally the default text value, or just enter the Text Resource value in place of the Text ID with no optional default text parameter. If no default is specified, App.GetString will use the TextId as the default.

Group: Application-Based Extension

Syntax: sValue = App.GetString(TextID, [vDefaultText])

sValue (String) is the text string extracted from the Text Resource based on the locale of the client.

TextID (Variant) is the Text ID value acting as a key to the Text Resource. Alternatively it can be the Text Resource value (an alternate key) to the Text Resource.

vDefaultText (Variant) Optional – is the text to be used if the Text ID cannot be located in Text Resources.

Example:

Dim sText As String

If the keys for a text resource were:

Text Id: 25

Text Resource: Hello

and the translation grid contained:

Locale: English

String Value: Hello There!

sText = App.GetString(25)

sText = Hello There!

sText = App.GetString(25, "Hello")

sText = Hello There!

sText = App.GetString("Hello")

sText = Hello There!

sText = App.GetString(99, "Hello")

sText = Hello - Because the ID could not be found the default text was used

sText = App.GetString(99)

sText = <nothing> - Because the ID could not be found and there was no default text

sText = App.GetString("Hi")

sText = Hi - Because the ID could not be found but it was of string type so it was considered the default text

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

GetValue

This function will get the current value of a field in the current recordset or the value of a specific prompt.

Group: Application-Based Extension

Syntax: vValue = App.GetValue(vField, [vFormName])

vValue (Variant) is the value extracted. (Note: this will always be a string value regardless of the actual field's data type.)

vField (Variant) to extract the value from a prompt, or is the name of a field in the current recordset.

vFormName (Variant) Optional – is the name of the application's recordset to extract data from. This is typically used to extract data from a calling application's recordset.

Example:

```
App.GetValue(2) ' Gets value from prompt #2
App.GetValue("PartNo") ' Gets the value from "PartNo"
App.GetValue(2,"Cycle") ' Gets the value from the second prompt on the "Cycle" application
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

IpAddress

This function will return the IP Address of the current Client device.

Group: Application-Based Extension

Syntax: sValue = App.IpAddress()

sValue (String) is the IP Address of the connected Client device.

Example:

```
Dim sAddress As String
```

```
sAddress = App.IpAddress
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

Locale

This function can set or return the number or string associated with the current locale of the connecting device. This command accepts a number Windows Language Code Identifier (LCID) and sets the locale within the client session. Examples would be 1033 for United States and 1041 for Japan. A web search for 'locale codes' should provide a list of LCIDs.

Group: Application-Based Extension

Syntax: nValue = App.Locale

Alternate: App.Locale = nValue

nValue (Long) is the number associated with the current locale

nValue (String) is the text ID associated with the current locale

Example

```
Dim nLocale As Long

nLocale = App.Locale App.Locale = 1033

nLocale = App.Locale
App.Locale = "en-US"
```

Version Supported: RFgen 4.0, 4.1, 5.0 and newer.

LogError

This extension writes an entry into the General Error Log file. The purpose for the percent signs is if you use this command in a global capacity and want to pass in parameters. Each percent sign is simply replaced with the next item in the Params list.

Group: Application-Based Extension

Syntax App.LogError(sProcedure, sErrDesc, [sParams])

sProcedure (String) is typically the application name

sErrDesc (String) is the ErrDesc entry

sParams (string array) Optional – these are parameters that will be substituted into the ErrDesc where a percent sign is used.

Example>

```
App.LogError("Application1", "Invalid Item")

Time..: 3/27/2006 1:37:55 PM
```

Process: Application1

ErrDesc: Invalid Item

App.LogError("Application1", "Invalid Item by user %", App.User)

Time..: 3/27/2006 1:37:55 PM

Process: Application1

ErrDesc: Invalid Item by user Sam

App.LogError("Application1", "Invalid Item % % % ", "1", "2", "3")

Time..: 3/27/2006 1:37:55 PM

Process: Application1

ErrDesc: Invalid Item 1 2 3

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

LogErrorEx

This extension is the same as LogError but lets the user change the category value of the log entry.

Group: Application-Based Extension

Syntax: App.LogError(sProcedure, sErrDesc, Category, [sParams])

sProcedure (String) is typically the application name

sErrDesc (String) is the ErrDesc entry

Category (enErrorCategory) is the list of message types such as Error, Information, System, Trace and Warning

sParams (string array) Optional – these are parameters that will be substituted into the ErrDesc where a percent sign is used

Example:

App.LogErrorEx("App1", "Invalid Item", catInfo)

Time..: 3/27/2014 1:37:55 PM

Process: App1

ErrDesc: Invalid Item

Category: Information

Version Supported: RFgen 5.0, 5.1, 5.2 and newer.

MacroName

This is a read only property that returns the active, macro name under execution in a transaction. See also CallMacro.

Group: Application-Based Extensions

Syntax: App.MacroName

(String) The name of the Macro being executed.

Example:

```
App.MacroName("Sample")
```

Versions Supported: RFgen 5.2.4.2 and newer.

MakeList

This function builds a scrolling list of items that may then be presented to the user for selection using App.ShowList.

Group: Application-Based Extension

Syntax: App.MakeList(sListData, vValue, vDisplay, [vHeading])

sListData (String) is the variable containing the list being created.

vValue (Variant) is the value that is returned when the user selects a specific list item.

vDisplay (Variant) is what the user will see for a specific list item. (Non-numeric values must be in quotes.)

vHeading (Variant) Optional – is the heading for the list. (Note: it only needs to be assigned once).

Example:

```
Dim sRsp As String
Dim sList As String
App.MakeList sList, 2000, "2000 ABC Company", "Select order"
App.MakeList sList, 2001, "2001 Widgets R Us"
App.MakeList sList, 2002, "2002 GoodFellows Inc."
sRsp = App.ShowList(sList)
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

MenuName

This function returns the current menu name but cannot create or set the menu name.

Group: Application-Based Extension

Syntax: App.MenuName ( )

Syntax: App.SetMenu(sMenuName)

sMenuName (String) is the default menu name for the connected Client device.

Example:

```
Dim sCurrentMenuName as String
    sCurrentMenuName = App.MenuName
App.MsgBox(sCurrentMenuName)
```

Related Topics: App.SetMenu

Versions Supported: RFgen 5.2.4.3 and newer.

MsgBox

This function displays a message box to the user. The construction of the message box depends on: a. Your text string entry; b) Which message box type was selected (i.e. information only versus ones with buttons); c) Which button values you select and whether the buttons are customized; d) If you added customized text for the caption of the buttons; and e) If you added a caption to your message box in addition to the message. When a user selects a button, you can use the return to control the next behavior of your strip.

If you have a message box script that defines customized buttons, and you only want a string value returned when a user selects one of the customized buttons at runtime, you can force RFgen to only return the string value of the customized button.

To force this change, set the **Configuration > Environment Settings > System Options > Use Legacy Message Box** to **True**.

If however, you want an integer value after the user selects a message box button (a customized button or VBA provided button), leave the **Use Legacy Message box** value equal to **False**.

Group: Application-Based Extension

Syntax: vValue = App.MsgBox(sMessage, [enType], [iDefault], [vButtons], [vCaption])

The 'value', 'type', 'default', and 'buttons' variables match those of the Visual Basic MsgBox function; e.g.:

vValue (Variant) will contain an integer or string indicating which selection was made.
vbOK =OK button
vbCancel =Cancel button
vbAbort=Abort button
vbRetry =Retry button
vbIgnore = Ignore button
vbYes =Yes button
vbNo =No button

sMessage (String) the message text to be displayed on the device

enType (enMsgBoxTypes) Optional – the expected responses:
vbOkOnly = OK
vbOkCancel = OK, Cancel
vbAbortRetryIgnore = Abort, Retry, Ignore
vbYesNoCancel = Yes, No, Cancel
vbYesNo = Yes, No
vbRetryCancel = Retry, Cancel
vbCritical – displays the Critical icon
vbCustom – displays the Custom icon
vbExclamation – displays the Exclamation icon
vbInformation – displays the Information icon
vbQuestion – displays the Question icon

iDefault (Integer) Optional – an optional message box default:
1 = First Button
2 = Second Button
3 = Third Button

vButtons (Variant) Optional – captions for the custom buttons;
"[A] [B]" will set "A' as the caption for the
first button, "B" for the second.

vCaption (Variant) Optional – the text that appears as the title of the message box

Example:

```
Dim vValue As Variant
vValue = App.MsgBox("Ok?", vbCritical+vbRetryCancel)
' Combine the Type parameter and sequence number to create the icon and the button type desired.
vValue = App.MsgBox("Continue with transaction?", vbYesNo, 1)
Dim vValue as Variant
vValue = App.MsgBox("Continue with transaction?", vbYesNo, 1, "[Good] [Bad]", "Program Stopped")

If App.MsgBox("OK?", vbYesNo) = vbNo Then Exit Sub
```

The result in Value will be a string containing the label of the button pressed only if custom buttons are used. Otherwise the button number is returned. Defining unique buttons returns the name of the selected button.

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

PageCount

This function will return the number of pages that exist in an application. If no pages are used, then it will return "0".

Group: Application-Based Extension

Syntax: vValue = App.PageCount

vValue (Variant) the number of pages in the current application

Example:

```
Dim iVal As Integer
iVal = App.PageCount
```

Version Supported: RFgen 5.1 and newer.

PageNo

App.PageNo returns the page number (the value assigned to the "(PageNo)" property) of the active page. If the form does not have any pages, it returns "0".

Group: Application-Based Extension

Syntax: AppPgNbr = App.PageNo

app.PageNo as Long

Example:

```
If App.PageNo = 1 Then App.ExitForm Else Form.Backup
```

Version Supported: RFgen 5.1 and newer.

PromptCount

This function will return the number of prompts that exist in an application. It is typically used when looping through the prompts to set properties like Visible.

Group: Application-Based Extension

Syntax: vValue = App.PromptCount

vValue (Variant) the number of prompts in the current application

Example:

```
Dim iVal As Integer
iVal = App.PromptCount
```

PromptNo

This VBA extension returns the prompt number of the prompt that has focus. Prompt numbers are the sequence ids assigned to an object that can receive focus. This value is read-only.

Group: Application-Based Extension

Syntax: Nbr = App.PromptNo

Nbr (Integer) the number of the current prompt

Example:

```
Dim PromptNbr As Long
PromptNbr = App.PromptNo
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

Reload

This command acts like an internal CallForm request, and will fully reload the existing form. If the RFMenu is the active form, App.Reload will reset the menu control and reload the current menu.

Group: Application-Based Extension

Syntax: App.Reload( )

Example:

```
'This simply reloads the form. There are no parameters
App.Reload
```

Versions Supported: All.

Note: In 5.2.4.2 and newer, this resets the menu control and reloads the current menu when in the RFMenu.

SendChar

This function sends an ASCII character directly to the control on the screen as if it came from the keyboard. The server does not intercept the character and process it. It is similar to the VB 'SendKeys' command.

Group: Application-Based Extension

Syntax: App.SendKey(nKeyASCII)

nKeyASCII (Long) the ASCII value of the character

Example:

```
App.SendKey(43) ' this would send the * character
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

SendKey

This function sends the hexadecimal value that is equivalent to the physical keyboard or virtual-key code to the server as if it came from the physical or virtual keyboard. The server will process it if it has special meaning. For example, sending an F key will get processed in the Form_OnFkey event if it is programmed to do so. It is similar to the VB 'SendKeys' command.

Note: To assign a function to a Fkey see the topic Configuring Sidebar Menus and Function Keys.

Group: Application-Based Extension

Syntax: App.SendKey(vkKeyCode)

vkKey Code Constants	Description of Key / Function
vkClear	Clear key
vkRefresh	Refresh the screen/data key
vkExit	Exit the application key
vkTab	Tab (to next prompt) key
vkBackTab	Tab Backwards (to prior prompt) key
vkDelete	Delete key
vkInsert	Insert key
vkUp	Up arrow key
vkDown	Down arrow

	key
vkRight	Right arrow key
vkLeft	Left arrow key
vkPageUp	PAGEUP key
vkPageDown	PAGEDOWN key
vkHome	Home Key
vkEnd	End Key

vkF1	vkF13
vkF2	vkF14
vkF3	vkF15
vkF4	vkF16
vkF5	vkF17
vkF6	vkF18
vkF7	vkF19
vkF8	vkF20
vkF9	vkF21
vkF10	vkF22
vkF11	vkF23
vkF12	vkF24

Example:

```
App.SendKey(vkClear) ' this sends the Clear key
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

SetDisplay

This function sets additional (alternate) display names for an application. This provides a display "container" for variations to the applications such as when the app runs in a different languageor is used with special prompt captions (i.e. Extra large text if displaying on a forklift). The prompts may have their font sizes changed to be seen more easily on large displays from a distance. The underlying code references the prompts by their Field ID or number so the many displays of the application only need one code base.

See also App.Display.

Group: Application-Based Extension

Syntax: App.SetDisplay (sDisplay)

sDisplay (String) the name established for the alternative application display.

Note: In RFgen 5.2, the ability to modifyor set the display's width and height programatically was removed. The original purpose of modifying a display's dimensions was for telnet systems which are not supported in 5.2. The client size is now determined by the RFgen client.

Example:

For example, if you had these displays created:

- Default
- Spanish
- Forklift

Then this command will bring up the last display:

```
App.SetDisplay("Forklift")
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

SetFocus

This command is used to reposition the focus to a specific prompt or the TabNo (tab number) assigned to a specific prompt.

App.SetFocus should be the last statement for an event since subsequent statements will be ignored.

If the optional SaveRSP parameter is used the data entered for the current prompt will be saved in the current record, prior to the App.SetFocus. The SaveRSP flag does not apply to the Rsp variable in the OnEnter event but will only save the text in the textbox itself. To alter the textbox value in code use the RFPrompt (1).Text property and give it a value.

The ValidateRSP flag will check the text value against the Edits property and also execute the OnEnter event. This makes the most sense when the SetFocus method is used in an event other than the OnEnter event. If the SetFocus method is used in the OnEnter event the ValidateRSP flag will not process the edits again or run through the OnEnter event a second time. Be sure not to cause one SetFocus command to execute another SetFocus command as this may lead to unintended results.

Group: Application-Based Extension

Syntax: [bValue =] App.SetFocus (vFieldId, [bSaveRSP], [bValidateRSP])

bValue (Boolean) Optional – Returns True or False depending on the success of the command

vFieldId (Variant) is the prompt's Field Id or number receiving the focus.

bSaveRSP (Boolean) Optional – set to True to save any changes to the current prompts value before switching focus to the new prompt.

bValidateRSP (Boolean) Optional – set to True to call the edit checks and to re-run the OnEnter event.

Example:

```
App.SetFocus(5) ' Cursor will go to prompt #5

App.SetFocus("PartNo") ' Focus goes to PartNo prompt

App.SetFocus("PartNo", True, True) ' Cursor will go to "PartNo", save current Text property and execute edits and OnEnter event

'Moves the cursor to the next prompt that is two tab numbers higher than current prompt.

App.SetFocus(RFPrompt(App.PromptNo).TabNo + 2)
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

SetMenu

This function will set the default menu for the current Client device. This is typically used to set an initial menu for an undefined user (i.e., you are bypassing the normal login process.)

The optional parameter ClearStack replaces the existing menu stack with the one specified if its set to True. (True is the default.) False will add the specified menu to the top current menu stack and place the other menus behind it.

App.SetMenu also reinitializes the menu immediately if RFMenu is the active form.

Group: Application-Based Extensions

Syntax: App.SetMenu(sMenuName, [bClearStack])

sMenuName	(String) is the default menu name for the connected Client device.
bClearStack	(Boolean) Optional. True will replace the existing menu stack with this menu. False will add it to the top of the current menu stack.

Example:

```
App.SetMenu("Sample", True)

'Will replace the current menu and set the new default menu to "Sample".
```

Note: RFgen 5.2.4.2 added parameter "ClearStack".

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

SetMenuCaption

This function allows you to override the default menu name. For example, if you are on the main menu you could display the user's name instead.

Group: Application-Based Extensions

Syntax: App.SetMenuCaption( )

Example:

```
'Personalize the menu caption to the logged in username in the RFMenu app
    Dim sShowMyUserNameHere As String
      sShowMyUserNameHere = App.UserName
    App.SetMenuCaption (sShowMyUserNameHere)
```

Versions Supported: RFgen 5.2.4.2 and newer.

SetValue

This command will set the value of a field in the current recordset or of a specific prompt.

Group: Application-Based Extension

Syntax: App.SetValue(vField, vValue, [vName])

vField (Variant) to set the value or a prompt, or is the name of a field in the current recordset.

vValue (Variant) is the value to insert into the current recordset. (Note: all values are treated as strings until passed to the database.)

vName (Variant) Optional – is the name of the application's recordset to update. This is typically used to insert data into a calling application's recordset.

Example:

```
App.SetValue(1,"FX1") ' Puts "FX1" in prompt #1.
App.SetValue("PartNo","FX1","Cycle")
```

Puts "FX1" in "PartNo" prompt on "Cycle" application.

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

ShowForm

This will immediately call the requested form and hold script execution on that line until the user exits the displayed form. App.ShowFormis intended to display things like a Search form, Error form, Detailed information, etc. where you want to pause current form execution, display / capture input from the other form, then return and continue processing. In other words, App.ShowForm lets you call another application without having to complete the current application's execution where as in the App.CallForm language extension, the current app execution must complete before calling another app form.

When you use ShowForm, your script that executes the command to show the form must originate from the root (original) form. If the requested form displayed and completed its OnUpdate event, then App.ShowForm returns True. If it doesn't then it returns False.

You can also use the Options parameter to load default values. (Options is like a GetValue function.)

Group: Application-Based Extensions

Syntax App.ShowForm [name], [options]

FormName (String) The name of the other application form to be displayed.

FormOptions (This value is not used at this time.

Example

```
Option Explicit
Dim bool As Boolean

Private Sub Button1_Click()  On Error Resume Next
bool = App.ShowForm("ShowedFormModified")
App.MsgBox("App.ShowForm returns " & bool)
End Sub

Private Sub Form_OnUpdate(ByRef Cancel As Boolean)  On Error Resume Next
App.MsgBox("App.ShowForm returns " & bool)
End Sub
Option Explicit

Private Sub Form_OnUpdate(ByRef Cancel As Boolean)  On Error Resume Next
App.ExitForm
End Sub
```

Version Supported: RFgen 5.2.4.0 and newer.

ShowList

This function causes a scrolling list to be displayed on the Client device and allows the user to make a selection. (Maximum entries: 32,767.)

Group: Application-Based Extension

Syntax: vValue = App.ShowList(sList, [bForceDisplay])

vValue (Variant) is the value that is returned when the user selects a specific list item.

sList (String) is the list to be returned for display (i.e., set RSP = Value to display the list on the Client device.)

bForceDisplay (Boolean) Optional – Use True or False to specify sending this command immediately rather than at the end of the current event. The default is False.

Example:

See App.MakeList and DB.MakeList

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

Sleep

This command will cause the Client device to sleep for the specified number of seconds. (Note: any activity by the user will terminate the sleep mode.) The Visual Basic Wait command may be a viable alternative.

Group: Application-Based Extension

Syntax: App.Sleep(nSeconds)

nSeconds (Long) is the number of seconds to suspend program operation. This is usually used to flash a message to the user, and then to erase it. (Note: The new timer feature is recommended for time related programming.)

Example:

```
App.Sleep(1)
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

ShowWait

This function will display a spinning icon (an animated image).

Group: Application-Based Extensions

Syntax: App.ShowWait( )

Show (Boolean) True enables this feature; False disables it.

Example:

```
'When the button is clicked a spinning icon will appear to indicate the system is processing a request.
Private Sub btnFOk_Click()
    On Error Resume Next
      App.ShowWait(True)
    End Sub
```

Versions Supported: RFgen 5.2.4.2 and newer.

Signout

This command will cause the client to sign out to the login form.

Group: Application-Based Extensions

Syntax: App.Signout( )

Example:

```
Private Sub Button1_Click()
App.SignOut
'The RFgen Login form displays.
End Sub
```

Versions Supported: RFgen 5.2.4.2 and newer.

SQLNum

Group: Application-Based Extension

Example:

```
Dim sValue As String
sValue = SQLNum("123.456,78") ' returns "123,456.78"
```

Theme

This command provides a quick way to change the active theme in code. It only affects the local emulator, and doesn't change anything on the device.

Group: Application-Based Extension

Syntax: App.Theme

Version Supported: RFgen 5.1 and newer.

TimerEnabled

This command will enable / disable the 'Form_OnTimer' event within the client session. It is a good idea to always disable the application level timer event when exiting the application. Use the Form_Unload event or just before any App.CallForm command.

Group: Application-Based Extension

Note: see the Mobile Development Studio sample 'FieldService' VBA script for an example.

Syntax: App.TimerEnabled = bValue

Alternate: bValue = App.TimerEnabled

bValue (Boolean) is the state of the timer event. Set to 'True' to enable timer functions.

Example:

See TimerInterval.

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

TimerInterval

This command sets the interval for the Form_OnTimer event.

Group: Application-Based Extension

Syntax: App.TimerInterval = nValue

Alternate: nValue = App.TimerInterval

nValue (Long) is the number of milliseconds between timer events; i.e., for 10 seconds, set TimerInterval = 10000.

Example:

```
App.TimerEnabled = True
App.TimerInterval = 1000
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

Update

This function will invoke RFgen's normal last-prompt update cycle which will call the Form_OnUpdate event and, if successful, clear the form and reset the focus to the first input prompt. It will return a Boolean value if the update processing was successful.

Group: Application-Based Extension

Syntax: App.Update

User

This function can set or return the user of the current Client device. The default login screen sets this user. This id will then be used by any process that needs to know the logged-in user.

See also: App.UserName.

Group: Application-Based Extension

Syntax: sUser = App.User()

sUser (String) is the current user of the connected Client device.

Example:

```
Dim sUser As String
sUser = App.User
App.User = "SAM"
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

UserName

This function sets and returns the active username even if the user is not managed by RFgen. For example, if SSO, Adm or WebServices was used to log in the user, if the return package provides the username, then this property will get the information from the package and set it up in the Admin Console (RFgen Mobile Enterrpise Dashboard) so the active username can display in the Dashboard. If no user record exists, this property will also create a new user record for the active user id specified via App.User.

Group: Application-Based Extension

Syntax: App.UserName ( )

Example:

'This example was based on the factory-provided RFLogin but the real intent of ApplUserName is for SSO logins.

```
Private Sub Button2_Click()
On Error Resume Next
  Dim sWhoIsUser As String
    sWhoIsUser = App.UserName
  TextBox2.Text = sWhoIsUser
End Sub
```

Versions Supported: RFgen 5.2.4.2 and newer.

UserProperty

This Application property can set or return the value of the specified property on the currently logged in user's profile. This command references the App.User function to get the correct user and then looks up the

parameter specified. This function also allows you to set properties for users even if they are not managed by RFgen.

If no user record exists, it will create a new user record for the active user id specified via App.User. To create the new user record, the active user must first be selected before executing the App.User command. See also App.UserName.

Group: Application-Based Extension

Syntax: vValue = App.UserProperty(vProperty)

Alternate: App.UserProperty(vProperty) = vValue

vValue (Variant) is the value stored under the property's value on the Users profile.

vProperty (Variant) is the property added to the current user's profile

Example:

```
Dim vUserAge As Variant
vUserAge = App.UserProperty("Age")
App.UserProperty("Language") = "English"
```

Notes: Modified in RFgen 5.2.4.2 to set properties on users not managed by RFgen.

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

UserRoles

This function allows the user to access to an application on a menu where the application and menu roles are defined and the application form's Role property is defined.

App.UserRoles can also be used to dynamically update a user's record when and ONLY when the active user has been selected. (An active user is a user that shows up in the RFgen User Management Console.)

You can use this script to add user roles to multiple users. For information on setting up the areas for restricting user access to an application, see To Limit User Access.

Group: Application-Based Extension

Syntax: = App.UserRoles()

sAdmin (String) is the role name assigned to the application.

Example:

```
App.UserRole = "Admin"
```

Version Supported: RFgen 5.1, 5.2 and newer.

Database Related Extensions

Database related commands send and receive data to and from ODBC data connections. When operating in a mobile environment, the SQL statements are directed to the local database on the mobile device. The list of parameters are:

DB.BeginTrans, DB.CommitTrans, DB.Count, DB.Execute, DB.Extract, DB.LogOff, DB.LogOn, DB.MakeList, DB.OpenResultset, DB.RedirectDataSource, DB.RollbackTrans, DB.SaveBitmap,and DB.UseDataSource

BeginTrans

This function begins a new transaction. The server will track any changes made to the database until either a DB.CommitTrans or DB.RollbackTrans are executed. You cannot have a second DB.BeginTrans for the same data source before committing or rolling back the first one. On a mobile device, there is only 1 data connection and it must be committed or rolled back before another DB.BeginTrans is used.

Group: Database Related Extensions

Syntax:

DB.BeginTrans

[bValue =] DB.BeginTrans(vSource)

bValue (Boolean) Optional – indicates success or failure. If this command should fail because the connection is not available, the remainder of the code should not be executed. There may be unreliable results.

vSource (Variant) data source name (DSN) or the data source number

Example:

```
DB.BeginTrans("RFSample")

DB.BeginTrans(1)
```

CommitTrans

This function saves any changes to the database that were started after the DB.BeginTrans command was executed and ends the current transaction.

Group: Database Related Extensions

Syntax:

[bValue =] DB.CommitTrans(vSource)

bValue (Boolean) Optional – indicates success or failure

vSource (Variant) data source name (DSN) or the data source number

Example:

```
DB.CommitTrans("RFSample")
DB.CommitTrans(1)
```

Count

This function will return the number of rows contained in a specified rows item returned from the DB.Execute function or any Dynamic Array.

Group: Database Related Extensions

Syntax: vNbr = DB.Count(sRows)

vNbr (Variant) is the number of rows contained in the Records item.

sRows (String) is the string representation of a static recordset generated by a SQL statement using the DB.Execute function.

Example:

```
Dim sSQL As String
Dim sCols As String
Dim sRows As String
Dim iNbr As Integer
sSQL = "select * from ItemMaster"
DB.Execute(sSQL, sCols, sRows)
iNbr = DB.Count(sRows)
```

Execute

This function will execute a pass-through SQL statement on the host through the connection maintained by the server. Any results from the statement will be returned in a text-delimited object. (Note: this means that the item is a static view of the database and cannot be updated.)

Group: Database Related Extensions

Syntax:

[vErrNo =] DB.Execute(sSQL, [vColumns], [vRows])

vErrNo (Variant) Optional – is a return value; a value of '0' (zero numeric) means the SQL statement processed normally.

sSQL (String) is the SQL statement to be sent to the database. As this is a pass through statement, its syntax must be understood by the database, as no pre-processing will occur.

vColumns (Variant) Optional – is a string representation of the columns returned by the SQL statement. This is optional because Insert, Update, and other statements do not return data.

vRows (Variant) Optional – is a string representation of the static rows of an SQL statement. This is optional because Insert, Update, and other statements do not return data.

Example:

```
Dim sSQL As String
Dim sCols As String
Dim sRows As String
sSQL = "select * from Inventory"
DB.Execute(sSQL, sCols, sRows)
```

In the case of an insert or an update, the sCols and sRows variables would not be necessary because no recordset is returned.

Extract

This function will extract from a recordset the one value at the specified column and row. A specific column and row intersect at a single value.

Group: Database Related Extensions

Syntax:

vValue = DB.Extract(sColumns, sRows, iRecNo, vFieldNo)

vValue (Variant) is the value extracted. (Note: this will always be a string value regardless of the actual field's data type.)

sColumns (String) is the string variable used to hold the list of columns in the DB.Execute command.

sRows (String) is the string variable used to hold the rows of data in the DB.Execute command.

iRecNo (Integer) is the row number within the retrieved recordset to extract data from.

vFieldNo (Variant) is the column number (numeric), or column name (String) within the retrieved recordset to extract data from.

Example:

```
Dim sSQL As String
```

```
Dim sCols As String
Dim sRows As String
Dim sPart As String
sSQL = "select * from ItemMaster"
DB.Execute(sSQL, sCols, sRows)
sPart = DB.Extract(sCols, sRows, 1, "PartNo")
```

LogOff

This function is used to logoff a database connection. Note: this function does not need to be called by the user. The server will call it automatically when shutting down the session. The first database connection is the default Source value.

Group: Database Related Extensions

Syntax:

[bValue =] db.LogOff (vSource)

bValue (Boolean) Optional – A True/False notification that the command processed successfully

vSource (Variant) data source name (DSN) or the data source number

Example:

```
Dim bValue As Boolean
bValue = db.LogOff(1)
db.LogOff("SQL")
```

LogOn

This is used to logon the database connection and specify a user / password sequence for a database connection. Use of this feature is optional.

Note: the user does not always require this function as the server calls it automatically when a session starts.

Group: Database Related Extensions

Syntax:

[bValue =] db.LogOn ([vSource], [vUserId], [vUserPwd])

bValue (Boolean) Optional – A True/False notification that the command processed successfully

vSource (Variant) data source name (DSN) or the data source number

vUserId (Variant) Optional – The login name to be used to connect to the database system

vUserPwd (Variant) Optional – The login password to be used to connect to the database system

Example:

```
Dim bValue As Boolean
bValue = db.LogOn("SQLServer", "SQLUser", "SQLPass1")
bValue = db.LogOn(1, "SQLUser", "SQLPass1", "CLIENT=800|TYPE=3")
bValue = db.LogOn("Access", "AccessUser", "AccessPass1")
```

MakeList

This function executes a pass-through SQL 'select' statement against the database and converts the results into a scrolling list of items when used with App.ShowList.

Group: Database Related Extensions

Syntax:

sMyList = DB.MakeList(sSQL, [bRtnAllCols], [bNormalize], [bScale, nMaxRows)

sMyList (String) is the list to be returned for display (i.e., set RSP = Value to display the list on the Client device.)

sSQL (String) is the SQL 'SELECT' statement to be sent to the database.

bRtnAllCols (Boolean) Optional – when set to True will return all the columns as the potential key, not just the first column. Default is False.

bNormalize (Boolean) Optional – when set to True will trim the spaces from the data so that it will display consistently. Default is False.

bScale (Boolean) Optional – formats the numeric values in the specified column if the database does not store decimals. This option will position a decimal at a position from the right side. A comma will be used for large numbers. This field is locale specific and will use the appropriate characters for each region. Default is True.

nMaxRows (Long) Optional – limits how many rows will be allowed in the list. Default is 500.

Example:

Inside the OnEnter event where Rsp is declared as a String:

```
Dim sSQL As String
Dim sMyList As String
sSQL = "select PartNo from Inventory"
sMyList = DB.MakeList(sSQL, True, True, True, 1000)
Rsp = App.ShowList(sMyList)
Or
```

```
Rsp = App.ShowList(DB.MakeList(sSQL, True, True))
```

OpenResultset

This function will execute a pass-through SQL statement on the host through the connection maintained by the server. Any results from the statement will be returned in a RDO Resultset object. (Note: this item defaults to a static view of the database and cannot be updated. Also, under the configuration of the database, the 'Connect Using…' setting must match the declaration of the recordset; ADO versus RDO.)

Group: Database Related Extensions

Syntax:

oRs = DB.OpenResultset(sSQL, [vCursorType], [vLockType])

oRs (rdoResultset Object or adoRecordset Object) is a snapshot type resultset generated from an SQL statement.

sSQL (String) is the SQL statement to be sent to the database. This is a pass through statement; its syntax must be understood by the database, as no pre-processing will occur.

vCursorType (Variant) Optional – indicates the type of cursor used by the recordset object.

vLockType (Variant) Optional – indicates the type of lock placed on records during editing.

ADO Cursor Types

0 Provides a static copy of the records (you can't see additions, changes or deletions by other users). You can only move forward through the recordset. Forward-only is the ADO default cursor type.

1 Existing records at time of creation are updateable. You can't see additions or deletions. All types of movement are enabled.

2 Dynamic requires more overhead, because updates are immediate and all types of movement are enabled. The dynamic cursor isn't currently supported by the Microsoft Jet OLE DB Provider, and therefore defaults to a keyset cursor when adOpenDynamic is applied to a Jet database.

3 Provides a static copy of the records (you can't see additions, changes or deletions by other users), but all types of movement are enabled.

ADO Lock Types

1 This value indicates read-only records where the data cannot be altered.

2 This value indicates pessimistic locking, record by record. The provider does what is necessary to ensure successful editing of the records, usually by locking records at the data source immediately after editing.

This lock type is supported by the Microsoft® OLE DB Provider for AS/400 and VSAM and the Microsoft OLE DB Provider for DB2. However, the OLE DB Provider for AS/400 and VSAM internally maps this lock type to adLockBatchOptimistic.

3 This value indicates optimistic locking, record by record. The provider uses optimistic locking, locking records only when the Update method is called.

This lock type is not supported by the OLE DB Provider for DB2.

4 This value indicates optimistic batch updates and is required for batch update mode.

This option is not supported by the OLE DB Provider for DB2.

Example:

Using rdoResultset, the RDO language extensions must be enabled.

```
Dim oRs As rdoResultset
Dim sSQL As String
Dim sDesc As Variant
sSQL = "select * from ItemMaster where PartNo = '100620'"
Set oRs = DB.OpenResultset(sSQL)
sDesc = oRs!Description
```

RedirectDataSource

This command will disable the server's auto processing of SQL statements to determine their correct source and will instead route all SQL traffic that is intended for the specified ODBC data source to always go to the other specified data source. To clear a redirection simply set the source and destination to the same data connection.

Group: Database Related Extensions

Syntax:

bValue = DB.RedirectDataSource(vFromSource, vToSource)

bValue (Boolean) indicates success or failure

vFromSource (Variant) is the name or number of the data source to redirect.

vToSource. (Variant) is the name or number of the data source to receive the other database's SQL statements.

Example:

```
Dim bValue as Boolean
bValue = DB.RedirectDataSource("CAPlant", "NVPlant")
```

RollbackTrans

This function cancels any changes made during the current transaction and ends the transaction.

Group: Database Related Extensions

Syntax:

[bValue =] DB.RollbackTrans(vSource)

bValue (Boolean) Optional – indicates success or failure

vSource (Variant) data source name (DSN) or the data source number

Example:

```
DB.RollbackTrans("RFSample")
DB.RollbackTrans(1)
```

SaveBitmap

This command will save the byte array of a bitmap picture directly to a database. The record in the database must already exist.

Note: An alternative way to save is SigToImg.

Group: Database Related Extensions

Syntax:

[bValue =] DB.SaveBitmap(sTable, sField, bData, [vWhere])

bValue (Boolean) Optional – indicates success or failure

sTable (String) is the name of the table in the database

sField (String) is the field name within the table

bData (Byte) is the byte array containing the image

vWhere (Variant) Optional – a SQL parameter that updates or inserts the image to a specific record

Example:

```
Dim bVal as Boolean
Dim bImage() as Byte
Dim nSize As Long

Open "C:\MyPic.bmp" For Binary Access Read As #1
```

```
  nSize = LOF(1)
  ReDim bImage(nSize - 1)
  Get #1, , bImage
Close #1
'
bVal = DB.SaveBitmap("Images", "Image", bImage, "PartNo='100620'")
```

UseDataSource

This command will disable the server's auto processing of SQL statements to determine their correct source and will instead route all SQL traffic to the specified data source. Specifying zero will re-enable the server's auto processing and let it determine which data connection should receive the SQL statement based on which connection was used to download the table structures.

Group: Database Related Extensions

Syntax:

[bValue =] DB.UseDataSource(vSource)

bValue (Boolean) Optional – indicates success or failure of the language extension

vSource (Variant) is the name or number of the data source to which all future SQL traffic will be routed.

Example:

```
DB.UseDataSource("RFSample")
DB.UseDataSource(0)
```

Database Stored Procedure Object

This object is *obsolete* and has been replaced with the Embedded Procedure structure.

The StoredProc Object is used to execute a stored procedure. It features the following set of properties and methods with which you can manipulate the object and its content.

The VBA extensions under this category are:

StoredProc.CommandTimeout, StoredProc.Data, StoredProc.DataSource, and StoredProc.Execute

CommandText

Sets or returns a value containing a provider command such as a stored procedure call.

Syntax: StoredProc.CommandText = sValue

Alternate: sValue = StoredProc.CommandText

sValue (String) stored procedure name

Example:

See Execute

CommandTimeout

Indicates how long to wait while executing a command before terminating the attempt and generating an error.

Group: Database Stored Procedure Object

Syntax:

StoredProc.CommandTimeout = nValue

Alternate: nValue = StoredProc.CommandTimeout

nValue (Long) time in seconds, default is 30

Example:

See Execute

CommandType

Specifies the type of command prior to execution to optimize performance.

Syntax: StoredProc.CommandType = enValue

Alternate: nValue = StoredProc.CommandType

nValue (Long) numeric value for the command type

enValue (enCommandTypes) Sets one of the following values:

Constant	Description
dbCmdText	Evaluates CommandText as a textural definition of a command.
dbCmdStoredProc	Evaluates CommandText as a stored procedure that will return records.
dbCmdExecuteNoRecords	Evaluates CommandText as a stored procedure that will return no records.
dbCmdUnknown	The type of command in the CommandText property is not known.
dbExecuteNoRecords	The stored procedure does not return records.

dbManualDeclare	This will execute the stored procedure directly without checking the syntax.
dbOpenNoRecords	For connection to Sybase.

Example:

See Execute

CreateParameter

Creates a new Param Object with the specified properties.

Syntax: CreateParameter (nIndex, [enDatatype], [enDirection], [nSize], [vValue])

nIndex (Long) Represents the parameter's index

enDatatype (enDataTypes) Optional – the parameter's data type

enDirection (enDirections) Optional – the parameter's direction

nSize (Long) Optional – the parameter's length

vValue (Variant) Optional – the parameter's value

Example:

See Execute

Database Stored Procedure Object

This object is *obsolete* and has been replaced with the Embedded Procedure structure.

The StoredProc Object is used to execute a stored procedure. It features the following set of properties and methods with which you can manipulate the object and its content.

The VBA extensions under this category are:

StoredProc.CommandTimeout, StoredProc.Data, StoredProc.DataSource, and StoredProc.Execute

DataSource

This property indicates which data source to connect.

Group: Database Stored Procedure Object

Syntax:

StoredProc.DataSource = sValue

Alternate: sValue = StoredProc.DataSource

sValue (String) name of the data source

Example:

See Execute

Dict

A string representation of the columns returned by the SQL statement.

Syntax: sValue = StoredProc.Dict

sValue (String) representation of the columns returned

Example:

See Execute

Execute

Executes the stored procedure in the CommandText Property.

Group: Database Stored Procedure Object

Syntax:

[bOK =]StoredProc.Execute

bOK (Boolean) True or False based on the success of calling the stored procedure regardless of the outcome

Example:

```
Dim vDict As Variant
Dim vData As Variant
Dim spObj As StoredProc
Set spObj = New StoredProc
spObj.DataSource = "Oracle"
spObj.CommandText = "byroyalty"
spObj.CommandType = dbCmdStoredProc
spObj.Param(1).DataType = dbInteger
spObj.Param(1).Direction = dbParamInput
spObj.Param(1).Value = 100
```

```
spObj.Execute
vDict = spObj.Dict
vData = spObj.Data
```

DataRecord Object

This object gives the user information about a stored recordset populated by other language extensions.

Group: DataRecord Object

To use this object, start with a declaration similar to:

```
Dim oData as New DataRecord
```

The objects in this group are:

oData.AddNew, oData.Clear, oData.IsEOF, oData.MoveFirst, oData.MoveLast, oData.MoveNext

oData.MovePrevious, oData.MoveTo, oData.Param, oData.ParamCount, oData.ParamName, oData.RowCount, and oData.SchemaId

AddNew

This method creates an additional entry in the DataRecord. As an example, when the TM.GetItems method populates a DataRecord from a list of items in a queue, this method is used internally to grow the DataRecord until the list is complete. If this object is being used for other purposes the AddNew method may be used to grow the DataRecord as needed.

Group: DataRecord Object

Syntax: oData.AddNew

Clear

This method clears the object of any previously loaded information

Group: DataRecord Object

Syntax: oData.Clear

IsEOF

This method (End-Of-File) returns a True if the current pointer in the recordset is beyond the last entry or if there are no entries contained in the object.

Group: DataRecord Object

Syntax: [bOK =] oData.IsEOF

bOK (Boolean) Optional – returns the True or False

MoveFirst

This selects (moves the pointer to) the first entry in the object's list of values.

Group: DataRecord Object

Syntax: [bOK =] oData.MoveFirst

bOK (Boolean) Optional – returns True or False for the success of the command

MoveLast

This selects (moves the pointer to) the last entry in the object's list of values.

Group: DataRecord Object

Syntax: [bOK =] oData.MoveLast

bOK (Boolean) Optional – returns True or False for the success of the command

MoveNext

This selects (moves the pointer to) the next entry in the object's list of values.

Group: DataRecord Object

Syntax: [bOK =] oData.MoveNext

bOK (Boolean) Optional – returns True or False for the success of the command

MovePrevious

This selects (moves the pointer to) the previous entry in the object's list of values.

Group: DataRecord Object

Syntax: [bOK =] oData.MovePrevious

bOK (Boolean) Optional – returns True or False for the success of the command

MoveTo

This selects (moves the pointer to) a specific entry in the recordset.

Group: DataRecord Object

Syntax: [bOK =] oData.MoveTo(nRow)

bOK (Boolean) Optional – returns True or False for the commands success

nRow (Long) the row number to move the object's pointer

Param

This property returns the values stored in the named columns of the DataRecord.

Group: DataRecord Object

Syntax: Param(vParamID, [vRowNo]) = vValue

Alternate: vValue = Param(vParamID, [vRowNo])

vValue (Variant) the stored value given a parameter ID

vParamID (Variant) then name of a column in the DataRecord

vRowNo (Variant) Optional – if the DataRecord is a table containing multiple rows of data, this parameter will move the data pointer to the specified row before retrieving the data.

Example:

```
Dim vData As Variant

oData.Param("ErrMsg") = "Wrong value."
vData = oData.Param("DeviceNo")
vData = oData.Param("FormId")
vData = oData.Param("IPAddress")
vData = oData.Param("ExecDate")
```

ParamCount

For a given row that is a table, this function will contain the number of columns returned.

Group: DataRecord Object

Syntax: vValue = oData.ParamCount

vValue (Variant) contains the count of the columns in the DataRecord

Example:

For example, a table named Inventory has 2 fields, Part and Quantity, and has 200 records.

```
Dim iValue As Integer

iValue = oData.ParamCount    ' will return 2.
```

ParamName

This property is a collection of parameter IDs. In the case of the TM.GetItems use of the DataRecord object the parameter IDs are:

(1) QueueName	(5) Source	(9) UserId	(13) ExecTime
(2) SeqNo	(6) Name	(10) DeviceNo	(14) Status
(3) TranDate	(7) Record	(11) IPAddress	(15) ErrMsg
(4) TranTime	(8) FormId	(12) ExecDate	

If the DataRecord object is used for another purpose the parameter names would be different. See the ParamCount property to get a count of the parameter IDs.

Group: DataRecord Object

Syntax: sValue = oData.ParamName([nIndex])

sValue (String) the name of the parameter at the specified index

nIndex (Long) Optional – the index of the parameter to be evaluated

Example:

When using the TM.GetItems method, to retrieve the name of the macro for the first row in the DataRecord:

```
Dim oData As New DataRecord
Dim sMacroName As String
TM.GetItems(tmCompleted, Date, App.User, oData)
oData.MoveFirst
sMacorName = oData.Param(6)
sMacorName = oData.Param("Name")
```

RowCount

For a given record set, this function will get the number of rows returned in the object.

Group: DataRecord Object

Syntax: vValue = oData.RowCount

vValue (Variant) the number of rows in the DataRecord

Example:

When using the TM.GetItems method, the number of queue entries returned would be returned.

```
Dim oData As New DataRecord
```

Dim iValue As Integer

TM.GetItems(tmCompleted, Date, App.User, oData)

iValue = oData.RowCount

SchemaId

This property returns the Task ID of a Vocollect task that is currently loaded in the DataRecord.

Group: DataRecord Object

Syntax: sValue = oData.SchemaId

sValue (String) the task Id of a Vocollect task

Example:

Dim sValue As String

sValue = oData.SchemaId

Device Extensions - Android, iOS, and WinCE

These commands are used specifically on a mobile device installations. They provide methods for synchronizing with the server, sending and receiving data, queuing transactions on the host, etc. For queuing commands, the server's Transaction Manager and the client's Transaction Manager must be enabled.

The commands available for the device.[command] extension for (all platforms) are:

Device.ClickAndSkipPrompts, Device.ClickCoordinates, Device.DeleteFile, Device.Dir

Device.DispSleep, Device.EnableGPS, Device.FileExits, Device.ForceLocal

Device.GetGPSInfo, Device.GetTimeInfo, Device.GoOffline, Device.GoOnline,

Device.GUID, Device.HasUI, Device.Id, Device.IsOffline, Device.IsOnline,

Device.IsQueue, Device.OffsetLngLat, Device.Platform,

Device.PlaySoundFile, Device.PlotInit, Device.PlotStart, Device.ReadFile,

Device.ScanEnable, Device.ScanTrigger, Device.SendBlueTooth, Device.SendCommPort,

Device.SendUSB, Device.SetCameraOptions, Device.SetUSB, Device.Shell, Device.ShowConfig,

Device.TakePicture, Device.Vibrate, and Device.WriteFile

There are also additional extensions supported only for Windows CE devices when calling the Windows device as an object through COM.

Note:

For events that are related to device actions, see RFgen.bas under Events Property.

ClickAndSkipPrompts

This method will turn on or off the user's ability to click into prompts in a random sequence. The line "Device.ClickAndSkipPrompts(False)" will allow the user to click in the very next prompt or any prompt that already contains data. All other prompts will not receive focus if they are clicked. Using Device.Click-AndSkipPrompts(True) will set the client back to its default behavior.

Group: Device Extensions

Syntax: Device.ClickAndSkipPrompts(bEnabled)

bEnabled (Boolean) True or False turns this feature on or off.

Example:

```
Device.ClickAndSkipPrompts(True)
```

Dir

This function lists files and folders in a given directory.

Group: Device Extensions

Syntax: bOK = Device.Dir(vFiles, vDirectories)

bOK	(Boolean) Returns True if the call was successful; False if the call failed.
vFiles	(Variant) A string to be populated with a list of files in the folder. This list will be delimited with Chr(1)
vDirectories	(Variant) A string to be populated with a list of directories in the folder. This list will be delimited with Chr(1).

Example:

```
Private Sub Button4_Click()
    On Error Resume Next
      Dim bOk As Boolean
      Dim sFiles As String
      Dim sDirs As String
        bOk = Device.Dir("C:\temp\", sFiles, sDirs)
          If bOk Then
            App.MsgBox("Success")
```

```
        Else
          App.MsgBox("Failure")
        End If
End Sub
```

Versions Supported: RFgen 5.2.3.0 and newer.

DispSleep

This command prevents the Android's screen from going to sleep. This is a useful feature for situations when the Android device time resets to a different time than what is currently configured when it goes to sleep.

Group: Device Extensions

Syntax: Device.DispSleep (bEnabled)

bEnabled (Boolean). True prevents the Android device from sleeping. False allows the Android device to sleep.

Example:

```
Device.DispSleep(True)
```

Supported Versions: RFgen 5.2.4 and higher.

DeleteFile

This command will delete a specified file on the mobile device. String data or binary data can be read.

Group: Device Extensions

Syntax: [bOK =] Device.DeleteFile(sPath)

bOK (Boolean) Optional – the success or failure of the command.

sPath (String) the path of the file to be deleted.

Example:

```
Private Sub ButtonDelete_Click()
    On Error Resume Next
      Dim bOk As Boolean
        bOk = Device.DeleteFile("C:\temp\test.txt")
          If bOk Then
          App.MsgBox("Success")
        Else
      App.MsgBox("Failure")
    End If
```

```
End Sub
```

Versions Supported: RFgen 5.2.3.0 and newer.

EnableGPS

This property will activate or deactivate the GPS receiver in the mobile device. Use Device.GetGPSInfo to retrieve the real-time GPS data.

Group: Device Extension

Syntax: Device.EnableGPS(bValue)

bValue (Boolean) set to True or False to activate or deactivate the GPS receiver

Example:

```
Device.EnableGPS(True)
```

FileExists

This command checks is a specified file exists on a device.

Group: Device Extensions

Syntax: [bOK =] Device.FileExists(sFileName)

bOK (Boolean) Optional – the success or failure of the command.

sFileName (String) specifies where on the mobile device the file should be found

vData (Variant) contains the data read from the file. String or binary data is allowed.

Example:

```
' For this sample code, first copy a file to the device.
    Dim szBuf() As Byte
      ReadBinaryFile("c:\\temp\\my-test-file.txt", szBuf)
    Device.WriteFile("%PUBLIC%/my-dir/my-test-file.txt", szBuf, False)
  ' Check for the existence of the file
    If Device.FileExists("%PUBLIC%/my-dir/my-test-file.txt") Then
      App.MsgBox("Test file exists")
    Else
      App.MsgBox("Test file is missing")
    End If
```

```
' Check for a non-existent file
  If Device.FileExists("%PUBLIC%/my-dir/this-file-does-not-exist") Then
    App.MsgBox("Failed to confirm nonexistent file")
  Else
    App.MsgBox("Nonexistent file confirmed")
  End If
```

Supported Versions: RFgen 5.2.4.8 and higher.

ForceLocal

This property will tell the Mobile Client in a Roaming state that it should only look to either the server or local database for all data retrieval and updates. For example, if Mobile Client in a Roaming state was connected to the server and the ForceLocal was set to False, when the user executes a TM.QueueMacro it would actually be queued on the server not the local device. DB.Execute would look to the server automatically to retrieve the latest data rather than the locally stored data on the device.

Group: Device Extensions

Syntax: Device.ForceLocal = bValue

bValue (Boolean) set to True or False to force data access to either the server or local database

Example:

```
Device.ForceLocal = True
```

GetGPSInfo

This command will retrieve GPS data from an attached GPS receiver on the mobile device. If no receiver is available or no signal is received the output will be zeros. This command would need to be called repeatedly to update the screen if the device is in motion. Use the Form_OnTimer event to continuously populate variables.

Group: Device Extensions

Note: The operating system GPS APIs must exist on the device. To configure the GPS settings on the mobile device locate the GPS configuration, set the Program COM port to 'none' and set the Hardware COM port to the proper number based on the manufacturer's documentation.

Syntax: [bOK =] Device.GetGPSInfo([vLongitude], [vLatitude], [vHeading], [vAltitude], [vSpeed])

OK (Boolean) Optional – the success or failure of the command.

Longitude (Variant) Optional – returns the longitude as a float value

Latitude (Variant) Optional – returns the latitude as a float value

Heading (Variant) Optional – returns the heading as a value (0-360)

Altitude (Variant) Optional – returns the altitude in meters

Speed (Variant) Optional – returns the speed in knots (nautical miles)

Example:

```
Dim bOK as Boolean
Dim vLong as Variant
Dim vLat as Variant
Dim vHead as Variant
Dim vAlt as Variant
Dim vSpeed as Variant
bOK = Device.GetGPSInfo (vLong, vLat, vHead, vAlt, vSpeed)
bOK = Device.GetGPSInfo (, , , vAlt)

For converting meters to US feet use the following conversion:
vAlt = vAlt * 3.28083989501312

For converting knots to US MPH use:
vSpeed = vSpeed * 1.15077945

For converting knots to KPH use:
vSpeed = vSpeed * 1.852

For converting the heading into a direction use:
Select Case vHead
 Case Is < 22.5:  sText = "N"
 Case Is < 67.5:  sText = "NE"
 Case Is < 112.5: sText = "E"
 Case Is < 157.5: sText = "SE"
 Case Is < 202.5: sText = "S"
 Case Is < 247.5: sText = "SW"
 Case Is < 292.5: sText = "W"
 Case Is < 337.5: sText = "NW"
 Case Else: sText = "N"
End Select
```

GetTimeInfo

This command will retrieve the Offset value which is the difference of the device time and UTC in minutes. (The offset is NOT the difference between the device time and server time in UTC.)

You can call the function from an event such as a button click.

DeviceTime can use the normal date formatting functions, but by default it will be formatted like "11/1/2017 8:52:12 AM".

Group: Device Extensions

Syntax: [bOK =] Device.GetTimeInfo([vDeviceTime],[MinutesFromUTC])

bOK (Boolean) – returns a True if the mobile device successfully obtains the offset value. False if it fails.

vDateTime (Varient) – Returns the value in minutes.

Example:

This is an example of a pop up message box showing the device's time and UTC offset:

```
Dim DeviceTime As Date
Dim MinutesFromUTC As Integer
Device.GetTimeInfo(DeviceTime, MinutesFromUTC)
App.MsgBox("Device Time = " & DeviceTime & ", which is " & MinutesFromUTC & " minutes from UTC")
```

GoOffline

This command returns a True / False regarding the attempt to close the socket connecting the mobile client to the server. This will make the device go from THIN client mode to a MOBILE disconnected state.

Group: Device Extensions

Syntax: [bOK =] Device.GoOffline([vUser], [vMenu], [vForm])

bOK (Boolean) Optional – returns a True if the mobile device successfully closes the socket to the server. Since the thin client is shut down with this command, evaluating the response variable will never be reached unless there is a failure.

vUser (Variant) Optional – If a user is specified then the device will set the current user and load the menu associated with the user bypassing the login screen.

vMenu (Variant) Optional – Specifying a specific menu requires that the optional User parameter be filled in. The User parameter will still load the default menu for that user but then the Menu parameter will load, in addition. This means that if the user backs out of the specified menu the server will display the user's default menu.

vForm (Variant) Optional – If the form is specified then the form will be loaded after the specified menu or the default menu. The Form depends on the UserID being filled in.

Example:

```
Dim bOK as Boolean
bOK = Device.GoOffline
bOK = Device.GoOffline("Sam")
bOK = Device.GoOffline("Sam", ,"IMASTER")
bOK = Device.GoOffline("Sam", "BasicApps", "IMASTER")
```

GoOnline

This command returns a True / False regarding the status of opening a socket connection with the server. This will make a disconnected MOBILE client an online THIN client to the server. If the user is going online for the first time the server will present the login screen. If the user goes online with the configured client inactivity timeout value they would see the session as they left it when going mobile. This command will not work for Mobile clients with a startup mode of Disconnected.

Group: Device Extensions

Syntax: [bOK =] Device.GoOnline([vServer], [nPort])

bOK (Boolean) Optional – return a True if the mobile device is successfully connected to the server. Since the mobile client is shut down with this command, evaluating the response variable will never be reached unless there is a failure.

vServer (Variant) Optional – parameter to specify which server name or IP address should be used. If this is not specified, the registry settings will be used.

nPort (Long) Optional – parameter to specify which server port number should be used. If this is not specified, the registry settings will be used.

Example:

```
Dim bOK as Boolean
bOK = Device.GoOnline("192.168.123.45", 21098)
```

Id

This returns True if the client has a user interface and False if it doesn't. For example, if the customer has a Transaction Management service running in their environment, there is no user interface for the client. Or if the customer is running Vocollect, a voice-controlled system, there is not a graphical user interface for the client; Or if the customer is running a services console, there is no user interface for the client.

See also: Device.GUID.

Group: Device Extensions

Syntax: Device.HasUI

Example:

```
Device.HasUI=True
```

Versions Supported: RFgen 5.2.4.2 and newer.

Id

This returns the name of a given device for Android, iOS, CE, and Windows Desktop clients. If testing with no client connection, nothing is returned, and an error is generated and sent to the Event Log.

See also: Device.GUID.

Group: Device Extensions

Syntax: Device.ID

Example:

```
Dim sMyDevice as String
sMyDevice = Device.ID
```

IsQueue

This returns True if the client session is in a Timed Event, or in a Transaction Queue.

See also: Device.GUID.

Group: Device Extensions

Syntax: Device.IsQueue

Example:

```
Public Function Transaction(ByRef tt As Variant) As Boolean
On Error Resume Next 'Transaction = True
If Device.IsQueue=True Then
App.Advance
End Function
```

IsOffline

This command returns a True / False regarding the status of an open socket connection with the server. This command is typically used to evaluate which state the client is in and then set variables that would vary

between the server and the mobile device such as file path information.

Group: Device Extensions

Syntax: bOK = Device.IsOffline

bOK (Boolean) return a True if the mobile device is currently not connected to the server.

Example:

```
Dim bOK as Boolean
bOK = Device.IsOffline
```

Id

This returns True if the client has a user interface and False if it doesn't. For example, if the customer has a Transaction Management service running in their environment, there is no user interface for the client. Or if the customer is running Vocollect, a voice-controlled system, there is not a graphical user interface for the client; Or if the customer is running a services console, there is no user interface for the client.

See also: Device.GUID.

Group: Device Extensions

Syntax: Device.HasUI

Example:

```
Device.HasUI=True
```

Versions Supported: RFgen 5.2.4.2 and newer.

GetGPSInfo

This command will retrieve GPS data from an attached GPS receiver on the mobile device. If no receiver is available or no signal is received the output will be zeros. This command would need to be called repeatedly to update the screen if the device is in motion. Use the Form_OnTimer event to continuously populate variables.

Group: Device Extensions

Note: The operating system GPS APIs must exist on the device. To configure the GPS settings on the mobile device locate the GPS configuration, set the Program COM port to 'none' and set the Hardware COM port to the proper number based on the manufacturer's documentation.

Syntax: [bOK =] Device.GetGPSInfo([vLongitude], [vLatitude], [vHeading], [vAltitude], [vSpeed])

OK (Boolean) Optional – the success or failure of the command.

Longitude (Variant) Optional – returns the longitude as a float value

Latitude (Variant) Optional – returns the latitude as a float value

Heading (Variant) Optional – returns the heading as a value (0-360)

Altitude (Variant) Optional – returns the altitude in meters

Speed (Variant) Optional – returns the speed in knots (nautical miles)

Example:

```
Dim bOK as Boolean
Dim vLong as Variant
Dim vLat as Variant
Dim vHead as Variant
Dim vAlt as Variant
Dim vSpeed as Variant
bOK = Device.GetGPSInfo (vLong, vLat, vHead, vAlt, vSpeed)
bOK = Device.GetGPSInfo (, , , vAlt)

For converting meters to US feet use the following conversion:
vAlt = vAlt * 3.28083989501312

For converting knots to US MPH use:
vSpeed = vSpeed * 1.15077945

For converting knots to KPH use:
vSpeed = vSpeed * 1.852

For converting the heading into a direction use:
Select Case vHead
 Case Is < 22.5:  sText = "N"
 Case Is < 67.5:  sText = "NE"
 Case Is < 112.5: sText = "E"
 Case Is < 157.5: sText = "SE"
 Case Is < 202.5: sText = "S"
 Case Is < 247.5: sText = "SW"
 Case Is < 292.5: sText = "W"
 Case Is < 337.5: sText = "NW"
 Case Else: sText = "N"
```

```
End Select
```

Id

This returns the graphical user interface identifier (GUID) for the attached device.

See also: Device.GUID

Group: Device Extensions

Syntax: Device.ID

Example:

```
Dim sMyDeviceGUID as String
    sMyDeviceGUID = Device.GUID
```

Versions Supported: RFgen 5.2.4.2 and newer.

IsOffline

This command returns a True / False regarding the status of an open socket connection with the server. This command is typically used to evaluate which state the client is in and then set variables that would vary between the server and the mobile device such as file path information.

Group: Device Extensions

Syntax: bOK = Device.IsOffline

bOK (Boolean) return a True if the mobile device is currently not connected to the server.

Example:

```
Dim bOK as Boolean
bOK = Device.IsOffline
```

IsOnline

This command returns a True / False regarding the status of an open socket connection with the server. This command is typically used to evaluate which state the client is in and then set variables that would vary between the server and the mobile device such as file path information.

Group: Device Extensions

Syntax: bOK = Device.IsOnline

bOK (Boolean) return a True if the mobile device is connected to the server.

Example:

```
Dim bOK as Boolean
bOK = Device.IsOnline
```

OffsetLngLat

This command is used to help set the GPS latitude and longitude. The latitude and longitude passed in will be used internally to calculate the offset needed to adjust your current GPS position to be at the provided latitude and longitude. The reason an offset is needed is so that RFgen can mathematically scale the positioning of the reference point on a map image you provided.

Group: Device Extensions

Syntax Device.OffsetLngLat(dNewLng, dNewLat)

Parameter	Description
dNewLng	The longitude of the device's current location.
dNewLat	The latitude of the device's current location.

Example>

```
'If showing the location of RFgen in El Dorado Hills
Device.OffsetLngLat(-121.05474, 38.62096)
```

Versions Supported: …RFgen 5.2.4.6 and newer.

Platform

This command returns an enumeration indicating the platform of the connected device.

Group: Device Extensions

Syntax: enValue = Device.Platform

enValue (Enumeration)
0 = Device_None (Application Testing)
1 = Device_WinCE (CE / Mobile Device)
2 = Device_Desktop (Windows Desktop Client)
3 = Device_Android (Android Device)
4 = Device_iOS (Apple Device)
5 = Device_Vocollect (Vocollect Talkman)
6 = Device_TN (Standard Telnet Client)
7 = Device_SOA (SOA Service Connection)

Example:

```
Dim vValue as Variant
```

```
vValue = Device.Platform
```

PlaySoundFile

This command plays any mobile-supported sound file by specifying the path to the file where the file exists. Note: For RFgen 5.1, use Device.PlaySound.

- For Windows CE, the path is the location of the Windows CE device.
- For Android and iOS, only .wav files are supported, and a copy of the source file MUST BE PLACED on the RFgen server under `\ProgramData\RFgen51` or \ProgramData\RFgen52.

When the sound is played at runtime, the .wav file is first copied from the server to the client when the client connects to the server.

You can also use this command to vibrate an Android or iOS device. For details see Vibrate.

Group: Device Extensions

Syntax:

Device.PlaySoundFileFile(sPath), Device.PlaySoundFile("Number, Vibrate")

sPath (String) the full path where the file is stored on the RFgen server

.wav The sound wav file to be played. The duration is the duration of the .wav file.

Example:

```
This example is applicable for Android or iOS.
Device.PlaySoundFile("\ProgramData\RFgen52\ERROR.WAV")
This example is applicable for Windows CE:
Device.PlaySound("\Program Files\RFgen51\ERROR.WAV")
```

PlotInit

This command initializes the plot points on a map so that an end user can see a start and destination points on an image such as a map.

Group: Device Extensions

Syntax Device.PlotInit(BSTR MapSource, double Geo1Lat, double Geo1Lng, int Geo1X, int Geo1Y, double Geo2Lat, double Geo2Lng, int Geo2X, int Geo2Y, int MapEdgePixels, BSTR MapPrompt, BSTR userPin, enAlign UserAlign, BSTR TargetPin, enAlign TargetAlign)

Parameter	Description
Geo1Lat, Geo1Lng	The latitude and longitude of the first reference point on the map image.This is used to calibrate the longitude and latitude geo coordinates to X and Y pixel coordinates of the map image.

Parameter	Description
Geo1X, Geo1Y	The X and Y pixel values of where 1st reference point is on the map image.
Geo2Lat, Geo2Lng	The latitude and longitude of the 2nd reference point on the map image.
Geo2X, Geo2Y	The X and Y pixel values of where the 2nd reference point is on the map image.
MapEdgePixels	How many pixels to add as padding so that when the user's pin icon is close to the edge there is this many pixels spacing the pin from the edge. A value of 0 means that as the user moves to the edge of the zoomed-in map, they are able to be right against the edge of the map with nothing past them being displayed. A value of 15 means that there will be 15px of extra map space displayed past the user up until they go off the entire full map.
MapPrompt	The name of the image control being used for this map image.
MapSource	The image name of the map being loaded into the image control on the form. This accepts both the name of an Image from RFImages as well as an absolute path to an image locally such as "%APPDATA%\MapImages\Map.png".
TargetAlign	The alignment property for where the user pin should remain in respect to the map area.
TargetPin	The name of the image control being used as the pin for the target's location.
UserAlign	The alignment property of the target pin.
UserPin	The name of the image control being used as the pin for the user's location.

Example Under construction.

Versions Supported: …RFgen 5.2.4.6 and newer.

PlotStart

This command refreshes the plot points on a map so that an end user can see his or her start and destination points when the user's location changes relative to the destination. This command is designed to work with Device.PlotInit.

Group: Device Extensions

Syntax Device.PlotStart(RefreshRate, optional TargetLat, double Geo1Lng, optional TargetLng)

Parameter	Description
Refresh Rate	(Integer) Sets in milliseconds the redrawing of the map image and repositioning of the user and target pins. The time is based on the form's internal clock/timer.
TargetLat	(Optional, Double) - The value of the latitude of the target.
TargetLng	(Optional, Double) - The value of the longitude of the target.

Example

'After PlotInit is called, the user can then start the timed plotting by calling:

Device.PlotStart(int RefreshRate, double TargetLat, double TargetLng)

Versions Supported: …RFgen 5.2.4.6 and newer.

ReadFile

This command will read data from a file on the mobile device. String data or binary data can be read.

Group: Device Extensions

Syntax: [bOK =] Device.ReadFile(sFileName, vData)

bOK (Boolean) Optional – the success or failure of the command.

sFileName (String) specifies where on the mobile device the file should be found

vData (Variant) contains the data read from the file. String or binary data is allowed.

Example:

```
Dim bOK As Boolean
Dim vData As Variant
Dim sFile As String
sFile = "\Program Files\MyFile.txt"
bOK = Device.ReadFile(sFile, vData)
```

ScanEnable

This command enables or disables the user's ability to scan a barcode. This command applies only for a sub-set of Intermec and Motorola handheld devices. One example might be to disable the scanner just before displaying a message box so that additional scans do not accidently acknowledge the message box and the user is unaware there was an error causing future scans to fail as well.

The ScanEnable function also enables barcode scanning using the device's camera.

Group: Device Extensions

Syntax: Device.ScanEnable(bScanOn)

bScanOn (Boolean) when set to True enables the laser scanner

Example:

Device.ScanEnable(True)

ScanTrigger

This command turns on the laser scanner for a subset of Intermec and Zebra (Motorola) handheld devices for Window CE.

On Android and iOS devices, this extension will also scan barcodes if the device uses a camera which supports scanning of barcodes.

See also the device extension "ScanEnable".

Android or iOS devices.

Group: Device Extensions

Syntax: Device.ScanTrigger(nTimeout)

nTimeout (Long) is the timeout period in milliseconds. The default is 10,000 milliseconds (10 seconds).

Example:

```
Device.ScanTrigger(10000)
```

SendBlueTooth

This command sends data (packets) through the mobile device BlueTooth port. Before you use this command, setup the port that will be used to send/receive data via BlueTooth. See the Device.SetBlueTooth for more details.

Group: Device Extensions

Syntax: [bOK] = Device.SendBlueTooth(vPacket, vPort)

bOK (Boolean) Optional return value showing success or failure of received packet

vPacket (Variant) is the text stream to be sent to the serial port.

vPort (Variant) is the name of the port

Note: To obtain the name of the devices (the vPort value), you will need to manually pair the two BlueTooth devices first. Refer to your device manufacturer documentation on how to enable BlueTooth and pair a device.

This command is currently only supported on Android devices.

Example:

Sending a Tab character to a device that expects this format.

```
Device.SendBlueTooth("&T",3)
```

SendCommPort

Sends data to the device serial port. This command replaces the commands Device.PrinterOn and Device.PrinterOff that are still supported but are now obsolete. The server will automatically turn

transparent print mode on or off as required.

Group: Device Extensions

Syntax: [bOK] = Device.SendCommPort(vPacket, iPort)

bOK (Boolean) Optional return value showing success or failure

vPacket (Variant) is the text stream to be sent to the serial port.

iPort (Integer) when set to a number other than zero it will route the data to this COM port, if already configured. When set to zero, the default, it will send to the first COM port configured with the Device.SetCommPort.

Example:

Sending a Tab character to a device that expects this format.

```
Device.SendCommPort("&T",3)
```

SendUSB

The Device.SendUSB command sends raw data (data packets) through the device's USB port that was enabled using the Device.SetUSB command. These commands can be used for sending print commands to a label printer, and are only supported on Android devices.

Group: Device Extensions

Syntax: bOK=Device.SendUSB(vPacket, vDevice)

bOK (Boolean) Optional - returns True if all bytes in the datapacket were received; False if it failed.

vPacket (Variant) is the text stream to be sent to the serial port.

vDevice (Variant) is the name of the device / printer.

Example:

```
On Error Resume Next
Dim sZPL as String
Device.SetUSB(True, "PC43t")      'This enables the USB connection between the client device and peripheral (i.e. printer).
sZPL = "^XA^PRB^FS^PFO^FS^BY3,2.0^FS"      'Lable format in Zebra programming language
sZPL = sZPL & "^FT100,120^B3N,N,102,N,N^FD1234-5678^FS"

Device.SendUSB(sZPL, "PC34t")      'This sends the Zebra print command over USB to the connected peripheral (printer).
```

SetBlueTooth

The Device.SetBlueTooth command and the Device.SendBlueTooth command enables the RFgen client to send datapackets to a paired peripheral devices using BlueTooth technology. (For example, you can set up your barcode scanner's BlueTooth radio to send string data to a label printer's BlueTooth radio.) The Device.SetBlueTooth procedure is not used for sending transactions to RFgen.

To use this command you will need to know the device's name as that represents the device port. Device names can be obtained by first pairing the devices and viewing the names used in the pairing process. Refer to your device hardware manufacturer's documentation on how to enable BlueTooth and pair it with another device.

See also Device.SendBlueTooth.

Group: Device Extensions

Syntax: Device.SetBlueTooth

bVal (Boolean) True enables the BlueTooth technology on the device

vPort (Variant) is the port number on the device that will be used to send data to a peripheral device.

SetCameraOption

This command changes some default settings for supported cameras. Since different cameras may use different settings, values for some IDs cannot be known beforehand. For example, you may want to disable the flash button before scanning on iOS, or keep it on.

Group: Device Extensions

Syntax: Device.SetCameraOption(enOption, vValue)

enOption (enCameraOptions) this ID contains either ImageBrightness, ImageContrast, or ImageFlash

vValue (Variant) the camera's expected value for the named ID

Example:

```
Device.SetCameraOption(ImageFlash, False)
```

SetCommPort

This command will enable or disable the serial port on a device.

Group: ceObject

Syntax: Device.SetCommPort(bEnabled, iPort, nBaudRate, iByteSize, enStopBits, enParity, enFlowControl, nPollingInterval, [bIsUnicode], [bAddPrefix], [nPacketSize])

bEnabled: (Boolean) True / False, enable or disable the serial port

iPort (Integer) 1,2,3,etc - the port number to activate

nBaudRate (Long) 9600, 19200, etc.

iByteSize (Integer) 7, 8, etc.

enStopBits (enStopBits) Select the appropriate stopbits from the drop-down list (1,1.5 or 2)

enParity (enParity) Select the appropriate Parity from the drop-down list (none, even, odd, etc.)

enFlowControl (enFlowControl) Select the appropriate FlowControl from the drop-down list (CTS, DSR, XONXOFF, etc.)

nPollingInterval (Long) The device will poll the serial port and look for data coming in using this timing interval in milliseconds.

bIsUnicode (Boolean) Optional – alerts the server that the attached COM device sends and receives in UNICODE. The default is False.

bAddPrefix (Boolean) Optional – when set to true, all data returned in the scan event coming from a COM port will have the COM port as a prefix

nPacketSize (Long) Optional – when set to a positive number, the data will only be returned in strings of this size. RFgen will cache the data until the packet size has been read.

Example:

```
Device.SetCommPort(True, 1, 9600, 8, 1, NONE, NONE, 1000, False, True, 8)
```

SetUSB

The Device.SetUSB command enables the USB port on the device so it can send raw data (data packets) to a USB connected device / printer using the Device.SendUSB command.

The Device.SetUSB and Device.Send USB commands are currently only supported on Android devices.

When the client connects to the printer, a pop-up requesting permission to access the printer will display. Once the authorization is granted, the user is not prompted again. Permissions are required only once per a session.

Group: Device Extensions

Syntax: bOK=Device.SetUSB (bEnabled, vDevice)

bEnable (Boolean) - True enables the USB technology on the device; False disables the connection.

vDevice (Variant) is the name of the peripheral device (i.e. printer) connected to the USB.

Example

```
On Error Resume Next
Dim sZPL as String
Device.SetUSB(True, "PC43t")      'This enables the USB connection between the client device and peripheral (i.e. printer).

sZPL = "^XA^PRB^FS^PFO^FS^BY3,2.0^FS"      'Lable format in Zebra programming language
sZPL = sZPL & "^FT100,120^B3N,N,102,N,N^FD1234-5678^FS"

Device.SendUSB(sZPL, "PC34t")      'This sends the Zebra print command over USB to the connected peripheral (printer).
```

Versions Supported: RFgen 5.2.2.1 and newer.

Devices Supported: Android

Shell

This function gives the user the ability to launch a program that is supported by the device's shell in the scripting environment. For example, it can be used to execute a program that opens a webpage in a specific browser.

Group: Device Extensions

Syntax: Device.Shell (ByVal File as String, ByVal Options As enDeviceShow)

File as String	This first parameter contains the name of the file or shell program stored on the device. Use closing quotes around the file name or command.
Options As	
enDeviceShow	This second parameter controls the window display for the program called in the first parameter "File as String".

Device_SHOWMAXMIZED = Activates the window and displays it as a maximized window.

Device_SHOWMINIMIZED = Activates the window and displays it as a minimized window.

Example:

```
Device.Shell("https://www.rfgen.com",Device_SHOWMINIMIZED)
Device.Shell("https://www.rfgen.com",Device_SHOWMAXIMIZED)
```

Device.ShowConfig

The Device.ShowConfig command .

This command will cause the device's user interface configuration menu to display.

Group: Device Extensions

Syntax: Device.ShowConfig ( )

* There are no parameters for this command.

Example:

```
Private Sub Button1_Click()
  On Error Resume Next
    Device.ShowConfig
  End Sub
```

Version Supported: RFgen 5.2.4.8 and newer.

Notes: In prior versions of RFgen, if a user on a client wanted to view their device configuration, the device configuration menu was accessed via an system icon which would bring up the Configuration menu. In RFgen 5.2.4.8 and higher, this command can be triggered via a click event or you can use this command for the F1 key.

TakePicture

This command will retrieve an image from an Android, iOS, or Window CE camera, and store the format of the image (i.e. png, jpeg, or bmp) as follows:
1) Take the picture; 2) link the captured image to the prompt by its name or the prompt's array value; and 3) Set the timeout value on the client for the period where the image is sent to the server before the image is dropped back into the prompt on the client.

Group: Device Extensions

Syntax: Device.TakePicture(Image() as Byte, [ByVal Timeout As Long=30], [ByVal Prompt]) As Boolean

bOK (Boolean) Optional – The success or failure of the command.

bImage (Byte Array) - Stores the image file using the prompt's name or the array value.

Timeout (Long) - The timeout value for Thin client waiting for a response from the server. The default is 30 seconds.

Prompt (Prompt)- The name of the prompt or the prompt's value that provides the name or value for the image file.

Example:

```
Dim bOK As Boolean
Dim bImage() As Byte
imgPic.Image.Bitmap = bImage
Device.TakePicture (bImage, 30, "imgPicture_pg1") 'where imgPicture_pg1 is the image control name
```

Vibrate

This command option must be used with the Device.PlaySound command to vibrate an Android or iOS device. Not all devices support this option on Android or iOS devices. It is not supported on Windows CE devices.

Group: Device Extensions

Syntax: Device.PlaySound("Number, Vibrate")

Number How long (duration) of a vibration in seconds. Defaut is 1 second.

Vibrate This command will vibrate the device.

Example

```
Device.PlaySound("1,Vibrate")
```

WebLogin

The Device.WebLogin language extension initiates a SSO session for Android clients in Thin Client mode, running RFgen 5.2.3.3 (or higher), and initiates a SSO session for iOS and Windows Desktop in Thin Client mode, running RFgen 5.2.4.1.

Batch mode is not yet supported.

As the name implies, the login session is initiated via a web session from the client to the target URL of the third-party SSO solution provider.

The web client will await a URL redirect to the provider Redirect URL. When the Redirect URL is reached, the web client will immediately close the entire URL and URL parameters will be returned in the ReturnParams parameter. If the web client does not reach the Redirect URL, the web client will close.

Group: Device Extensions

Syntax: Device.WebLogin(ReturnParams as String, RedirectURL as String, Timeout as Interger) As Boolean

Return - (Boolean) The return value of the base extension is True on no error, False on error. If returning True for success, the web client reached the Redirect URL and closed.

ReturnParams - (String) If successful, this value will hold the Redirect URL and any additional URL parameters returned from the web service.

URL - (String) Login URL of the Single Sign On (SSO) service provider/server.

RedirectURL - (String) The SSO provider's base URL address on SSO Login redirects. The web client will await a web service redirect to this URL to trigger storing URL parameters and existing the client.

Timeout (Miliseconds) - The default timeout value is 90,000 milliseconds while waiting to reach the Redirect URL.

Example:

```
Dim bSuccess as Boolean
Dim sReturnParameters As String
bSuccess = Device.WebLogin(sReturnParameters, "https://ssoprovider.login/","http://localhost:12345/",90000)
```

Versions Supported: RFgen 5.2.3.2 and newer.

Devices Supported: Android. 5.2.4.1; iOS and Windows Desktop/Ce

WriteFile

This command will write data to a file on the mobile device. String data or binary data can be written.

Group: Device Extensions

Syntax: [bOK =] Device.WriteFile(sPath, vData, bOverwrite)

bOK (Boolean) Optional – the success or failure of the command.

sPath (String) specifies where on the mobile device the file should be placed

vData (Variant) contains the data to be written to the file. If it is string data it will be saved in Unicode format otherwise it will be left as is. A byte array is also allowed.

bOverwrite (Boolean) set to True, this command will overwrite an existing file, False will return a failure because the file already existed

Example:

```
Dim bOK As Boolean
Dim sData As String
Dim sPath As String
sPath = "\Program Files\MyFile.txt"
sData = "Sample Text"
bOK = Device.WriteFile(sPath, sData, True)
```

WriteFileTimeout

This command is used to set the timeout period when writing to a file that is large.

Group: Device Extensions

Syntax: [bOK =] Device.WriteFileTimeout(sFileName)

bOK (Boolean) Optional – the success or failure of the command.

sFileName (String) specifies where on the mobile device the file should be found

vData (Variant) contains the data read from the file. String or binary data is allowed.

Example:>

```
'Create a large (at least 50 MB) test file on your server. For example: "c:\temp\large-test-file"
' Add a form that runs code such as the following:
    ' Set long timeout
        Device.WriteFileTimeout = 60
    ' Copy file to the device.
        App.Balloon("Sending (long timeout)...", -1)
          Dim szBuf() As Byte
            ReadBinaryFile("c:\\temp\\large-test-file", szBuf)
            Device.WriteFile("%PUBLIC%/my-dir/large-test-file", szBuf, False)
    ' Check for the existence of the file
        If Device.FileExists("%PUBLIC%/my-dir/large-test-file") Then
          App.MsgBox("Test file exists")
        Else
          App.MsgBox("Test file is missing")
        End If
```

Supported Version: 5.2.4.8

cdObjects

The mobile environment also supports the ability to load and execute COM objects on the Windows CE device.

COM objects are loaded and run by executing the language extensions ceObject.Create, ceObject.Execute, and ceObject.Release.

ceObject is an object you must declare as a variable first and then use that variable.

The objects must be derived from the IDispatch interface and must be compatible with VBA scripting environments.

Additional variables/objects for Windows CE are:

DeviceObject, Name, ReturnValue, and SetCommPort.

Note: Microsoft Support for Windows CE devices was sun setted in 2020.

DeviceObject

This object is declared as a variable type and used to execute COM objects from the CE device. It is not part of the Device object directly but is a subset of CE functionality specifically for calling COM objects.

Group: ceObject

Examples:

The following examples would all start with:

```
Dim [WithEvents] oMyObj As DeviceObject
```

If the object supports events include the WithEvents statement in the Dim statement. If WithEvents is declared as part of the Dim statement, this event would also be available from the script window's drop-down:

```
Private Sub oMyObj_OnEvent(ByVal EventName As String, ByRef rsData As DataRecord)
End Sub
```

Create

This command loads a COM object stored on the CE device. Calling Create multiple times on the same program id will increment the reference count. Create will only register for events, if specified, on the first call to Create.

The SysErr.Number object should be inspected to determine if Create was successful.

Group: ce Object

Syntax: [bOK =] oMyObj.Create

bOK (Boolean) Optional – the success or failure of the command.

Example: *(See Execute)*

Execute

This command executes the COM object stored on the Windows CE device. Execute will first determine if the COM object has been loaded. If the object has not been loaded then Execute will attempt to load the object without events. If the object could be loaded successfully, the method executes. The object will remain loaded after this method completes execution.

The SysErr.Number object should be inspected to determine if Execute was successful.

Group: ce Object

Syntax: [bOK =] oMyObj.Execute(sMethod, [vParams])

bOK (Boolean) Optional – the success or failure of the command.

sMethod (String) Method is a text string that represents the method that you wish to execute.

vParams (Variant) Optional – parameter array where you specify the parameters to the method. The parameters must be specified in the appropriate order that the method expects.

Example:

```
Dim bOK as Boolean
Dim oMyObj As DeviceObject
Dim vErr As Variant
Dim sParam1 As String
Dim sParam2 As String

Set oMyObj = New DeviceObject
oMyObj.Name = "DeviceObjectTest.ioInterface"
oMyObj.Create
sParam1 = "Value1"
sParam2 = "Value2"
bOK = oMyObj.Execute("Method1", sParam1, sParam2)
vErr = oMyObj.LastError
If vErr <> "" Then
 App.MsgBox "COM failure: " & CStr(vErr)
Else
 App.MsgBox "COM object returned: " & oMyObj.ReturnValue
End If
oMyObj.Release
```

LastError

This property returns the last error generated from the Execute method.

Group: ceObject

Syntax: vValue = oMyObj.LastError

vValue (Variant) is the error number returned from the COM object

Example: *(See Execute)*

Name

This property sets the program ID of the object that you wish to load on a Windows CE device. The object must have been successfully registered on the device for the Create function to succeed.

Group: ceObject

Syntax: oMyObj.Name = sProgID

Alternate: sProgID = oMyObj.Name

sProgID (String) a string indicating the program ID of the object that you wish to execute.

Example: (See Execute)

Release

This command releases the COM object on a Window CE device. Calling Release multiple times on the same program ID will decrement the reference count. Release will free the object when the reference count reaches 0.

The SysErr.Number object should be inspected to determine if Release was successful.

Group: ceObject

Syntax: [bOK =] oMyObj.Release

bOK (Boolean) Optional – the success or failure of the command.

Example: (See Execute)

ReturnValue

This property returns any value being passed back from the COM object on a Windows CE object.

Group: ceObject

Syntax: [vValue =] oMyObj.ReturnValue

vValue (Variant) Optional – the value returned by the COM object.

Example *(See Execute)*

Dynamic Array Extensions

This product supports a special kind of string variable called a 'dynamic array'. The word 'dynamic' means that this type of variable is designed to allow stored data to grow or shrink in size. In addition, the stored

data is easily accessible and changeable. Dynamic arrays allow volumes of data to be organized, stored, referenced, counted, and manipulated by use of just 1 single string variable. Access to the data is instantaneous via the Dynamic Array VBA extensions that follow.

Dynamic Array Structure

Dynamic Arrays are special string variables that use low end ANSI characters (1, 2, and 3) as 'delimiters' to separate data into Fields using Chr(1), Subfields using Chr(2), and Sub-subfields using Chr(3). These delimiters were chosen because data being entered or scanned are never expected to contain them. To use Dynamic Arrays in your programming, you do not need to reference these characters, the server simply uses them internally to manage stored data.

Why Use Dynamic Arrays?

1. There are no limits to the amount of data that may be stored.

2. The alternative is to declare and use subscripted variables, the number of which may be insufficient .

3. The server has been optimized to access and manipulate this type of data instantaneously.

For illustrative purposes, the 'Field' and 'Subfield' delimiters are represented in the examples below as follows:

Dynamic Array Field delimiter (Chr(1)) as a '&'

Dynamic Array Subfield delimiter (Chr(2)) as a '#'

Dynamic Array Sub-Subfield delimiter (Chr(3)) as a '@'

These functions work on fields, subfields, or sub-subfields, and are available for use with the Dynamic Array Extension:

DCount (Delimiter Count), Del (deletes), Ext (extracts), FixLeft (pads from left), Fix Right (pads from right), Ins (insert value)

LeftField (search from left then extract), Locate (find index), LocateAdd (add value if it doesn't exist), LocateDel (removes a value), LocateField(searches by string or delimiter)Rep (replaces), RField (searches from right then extracts), SortAlpha (sorts alphabetically), and SortNum (sorts numerically)

DCount

The DCount function (Delimiter Count) may be used to determine the number of occurrences of a specified delimiter within a string, or a string variable. The command will also add 1 to the count to actually represent the number of values or fields stored within the string variable. This makes the DCount command really return the number of available fields in the record so that extra logic is not necessary when determining the length of a loop structure used to manipulate the contents of the string.

Group: Dynamic Array Extensions

Syntax: vNbr = DCount(vVAR, vDELIM)

vNbr (Variant) is a number of delimiters found in the variable plus 1.

vVAR (Variant) is the string value to be evaluated

vDELIM (Variant) is a specified delimiter character, or multi-character string to count; e.g., to count dynamic array delimiters use Chr(1), Chr(2), Chr(3).

Example:

```
Dim nNbr As Long
Dim sNames As String
sNames = "John&Mike&Albert"
nNbr = DCount(sNames,Chr(1)) ' Nbr = 3
sNames = "&Mike&Albert"
nNbr = DCount(sNames,Chr(1)) ' Nbr = 3
sNames = "&Mike&"
nNbr = DCount(sNames,Chr(1)) ' Nbr = 3
```

Note that +1 has been added to the actual count of Chr(1) (represented here with a character '&') count of 2. Also note that just because the fields are empty, they are valid slots where data may be inserted.

Del

The Delete function deletes a field, subfield, or sub-subfield from a dynamic array.

Group: Dynamic Array Extensions

Syntax: vREC = Del(vREC, vFLD, [vSUB], [vSSUB])

vREC (Variant) is the dynamic array variable name

vFLD (Variant) is the array field number to search

vSUB (Variant) Optional – is the array subfield number

vSSUB (Variant) Optional – is the array sub-subfield number

Example:

```
Dim sNames As String
sNames = "John&Mike&Albert"  ' Three records of names
sNames = Del(sNames, 3)     'sNames becomes "John&Mike"
sNames = "John&Mike&Albert"  ' Three records of names
sNames = Del(sNames, 2)     'sNames becomes "John&Albert"
```

This next example shows each record with 3 values; Name, Age and Language.

```
sNames = "John#31#English&Mike#34#Spanish&Albert#40#English"
```

```
sNames = Del(sNames, 1)
  'sNames becomes "Mike#34#Spanish&Albert#40#English"
sNames = "John#31#English&Mike#34#Spanish&Albert#40#English"
sNames = Del(sNames, 2)
  'sNames becomes "John#31#English&Albert#40#English"
```

You can also delete portions of a record.

```
sNames = "John#31#English&Mike#34#Spanish&Albert#40#English"
sNames = Del(sNames, 2, 1)
 'sNames becomes "John#31#English&34#Spanish&Albert#40#English"
sNames = "John#31#English&Mike#34#Spanish&Albert#40#English"
sNames = Del(sNames, 2, 2)
'sNames is now "John#31#English&Mike#Spanish&Albert#40#English"

sNames = "John#31#English&Mike#34#Spanish&Albert#40#English"
sNames = Del(sNames, 2, 3)
  'sNames becomes "John#31#English&Mike#34&Albert#40#English"
```

Using the Sub-subfield concept, maybe the language has sub categories like Canadian English and United Kingdom English.

```
sNames = "John#31#Canadian English@UnitedKingdom English&Mike...
sNames = Del(sNames, 1, 3, 1)
  'sNames becomes "John#31# UnitedKingdom English&Mike...
```

Here everything between the Chr(1) delimiters is considered the first field. The 2 versions of English are a part of the third sub-field. So reading the DEL statement: "Using the first field, look at the third sub-field (all the languages) and delete the first sub-subfield.

A Dynamic Array is like a database stored as a single string. Everything between the Chr(1) delimiters are fields referenced by the first number. If there are any Chr(2) delimiters, each piece makes up the complete record. If there are any Chr(3) delimiters, they make up the complete subfield.

Ext

The Extract function returns a field, subfield, or sub-subfield value from a dynamic array.

Group: Dynamic Array Extensions

Syntax: vVAL = Ext(vREC, vFLD, [vSUB], [vSSUB])

vVAL (Variant) is the string value extracted from the array.

vREC (Variant) is the dynamic array variable name

vFLD (Variant) is the array field number to search

vSUB (Variant) Optional – is the array subfield number

vSSUB (Variant) Optional – is the array sub-subfield number

Example:

```
Dim sNames As String
Dim sValue As String
sNames = "John&Mike&Albert" ' Three records of names
sValue = Ext(sNames, 2)    'sValue becomes "Mike"
sNames = "John&Mike&Albert" ' Three records of names
sValue = Ext(sNames, 3)    'sValue becomes "Albert"
```

This next example shows each record with 3 values; Name, Age and Language.

```
sNames = "John#31#English&Mike#34#Spanish&Albert#40#English"
sValue = Ext(sNames, 2, 3)  'sValue becomes "Spanish"
```

Here Mike#34#Spanish is the second record and the third subfield value was retrieved.

Using the Sub-subfield concept, maybe the language has sub categories like Canadian English and United Kingdom English.

```
sNames = "John#31#Canadian English@UnitedKingdom English&Mike...
sValue = Ext(sNames, 1, 3, 1)   'sValue  becomes "Canadian English"
```

There are times when the Extract function can be very helpful. For example, if you create a comma-delimited list of items and must parse through it, writing code to manipulate the string and keep track of the pointer can be extensive. Instead use a statement like:

```
sList = Replace(sList, ",", Chr(1))
```

This replaces the comma with the Chr(1) character. Then in the loop you only need to EXT(sList, i) to get each entry out. Use the DCount command to get the length of the list.

FixLeft

This function pads a raw string to the left with a specified character until the specified length is reached. If the optional 3rd parameter is not specified, spaces are used. (This is not specifically a Dynamic Array command but is commonly used to manipulate them.)

Group: Dynamic Array Extensions

Syntax: vValue = FixLeft(vStringData, vChars, [vPadChar])

vValue (Variant) is the padded string value

vStringData (Variant) is the string containing the raw data

vChars (Variant) is the total size of the resulting string

vPadChar (Variant) Optional – is the pad character to use. If none is specified, spaces are used

Example:

```
Rsp = FixLeft(Rsp, 8, "A")
```

If Rsp was "123" then Rsp becomes "123AAAAA" because it puts the character 'A' as many times as necessary after the Rsp value until the length of Rsp is 8, left justifying the StringData value.

FixRight

This function pads a raw string to the right with a specified character until the specified length is reached. If the optional 3rd parameter is not specified, spaces are used. (This is not specifically a Dynamic Array command but is commonly used to manipulate them.)

Group: Dynamic Array Extensions

Syntax: vValue = FixRight(vStringData, vChars, [vPadChar])

vValue (Variant) is the padded string value

vStringData (Variant) is the string containing the raw data

vChars (Variant) is the total size of the resulting string

vPadChar (Variant) Optional – is the pad character to use. If none is specified, spaces are used

Example:

```
Rsp = FixRight(Rsp, 8, "A")
```

If Rsp was "123" then Rsp becomes "AAAAA123" because it puts the character 'A' as many times as necessary before the Rsp value until the length of Rsp is 8, left justifying the StringData value.

Ins

The Insert function inserts a value into a field, subfield, or sub-subfield of a dynamic array.

Group: Dynamic Array Extensions

Syntax: vREC = Ins(vREC, vFLD, [vSUB], [vSSUB], vSTR)

vREC (Variant) is the dynamic array string variable name

vFLD (Variant) is the array field number to search

vSUB (Variant) Optional – is the array subfield number

vSSUB (Variant) Optional – is the array sub-subfield number

vSTR (Variant) is data or variable name, which will be inserted at the dynamic array location (vFLD, vSUB, vSSUB).

Example:

```
Dim sNames As String

sNames = Ins(sNames, 2, 0, 0, "John")

Inserts string 'John' into dynamic array sNames. If sNames was originally null, then after the insert, field2 contains
'John'; i.e., sNames = '&John'. Here, field1 is null.

Note: Suppressing the zeros [i.e., sNames = INS(sNames, 2, "John")] gives the same result.


sNames = Ins(sNames, 2,"Mike")
```

If data already exists for field2 (John), then an additional insert would move field2 data to field3; i.e., sNames = '&Mike&John'. Here, field1 is null, field2 = 'Mike', and field3 = 'John'.

Using the subfields the following string can be created one element at a time using 9 Insert statements:

```
"John#31#English&Mike#34#Spanish&Albert#40#English"

sNames = Ins(sNames, 1, "John")

sNames = Ins(sNames, 1, 2, "31") ' First record, second field

sNames = Ins(sNames, 1, 3, "English") ' First record, third field


sNames = Ins(sNames, 2, "Mike")

sNames = Ins(sNames, 2, 2, "34") ' Second record, second field

sNames = Ins(sNames, 2, 3, "Spanish") 'Second record, third field

sNames = Ins(sNames, 3, "Albert")

sNames = Ins(sNames, 3, 2, "40") ' Third record, second field

sNames = Ins(sNames, 3, 3, "English") 'Third record, third field
```

If you used insert statements and the sub-subfield values you can get this:

```
"John#31#Canadian English@UnitedKingdom English&Mike"


by

sNames = Ins(sNames, 1, "John")

sNames = Ins(sNames, 1, 2, "31") ' First record, second field
```

sNames = Ins(sNames, 1, 3, 1, "Canadian English") ` First record, third field, first sub-field

```
sNames = Ins(sNames, 1, 3, 2, "UnitedKingdom English") ' First record, third field, second sub-field

sNames = Ins(sNames, 2, "Mike")
```

LField

The LField function searches a string from the left to extract a sub-string by using a specified delimiter character.

Group: Dynamic Array Extensions

Syntax: vVAL= LField(vStringData, vDelimiter, vStartField, [vNumberFields])

vVAL (Variant) is the resulting sub-string value located in StringData

vStringData (Variant) is a string or string variable to be searched

vDelimiter (Variant) is a specified delimiter character

vStartField (Variant) Optional – is the Delimiter to start from when searching the StringData

vNumberFields (Variant) Optional – specifies the number of sub-fields to retrieve.

Example:

If VAR = "111|222|333", then:

VAL = LField(VAR, "|", 1) returns VAL = '111'

VAL = LField(VAR, "|", 3) returns VAL = '333'

Note: If StartField = 1, then the LField function will return a sub-string which starts at the beginning of VAR, up to the first occurrence of Delimiter.

Locate

The Locate function may be used to find the index of a field, subfield, or sub-subfield within a dynamic array.

Group: Dynamic Array Extensions

Syntax: vNbr = Locate(vVAL, vStringData, [vFLD], [vSUB], [vOption])

vNbr (Variant) is an integer that indicates the location of VAL, or 0 (zero) if VAL is not located.

vVAL (Variant) is the string value to be located

vStringData (Variant) is the dynamic array variable that will be searched

vFLD (Variant) Optional – is the array field number to search

vSUB (Variant) Optional – is the array subfield number to search

vOption (Variant) Optional – the column within the row to search

Example:

The '&' character is used like the Chr(1) delimiter to separate the different rows. The '#' character is used like the Chr(2) delimiter to separate different values within a record, In this case, the name and the age and the language. The '@' symbol is used like Chr(3) to delimit between a sub-field. In this case, the difference between 2 languages.

```
Dim Nbr As Integer
Dim sNames As String
sNames = "John#31#Canadian English@UnitedKingdom English&Mike"
Nbr = Locate("Mike", sNames)  ' Nbr = 2
Nbr = Locate("John", sNames)  ' Nbr = 0 since there is depth to record 1
Nbr = Locate("John", sNames, 1) ' Nbr = 1
Nbr = Locate("31", sNames, 1)   ' Nbr = 2 since 1st record was specified
Nbr = Locate("UnitedKingdom English", sNames, 1, 3)
```

Nbr = 2 since 1st record and 3rd subfield was specified

If you do not know the row number where your data is then you can specify 0 in place of the Field and Sub values to search the entire array.

```
Nbr = Locate("31", sNames, 0, 0, "V2")  ' Nbr = 1
```

Assuming you know the layout of the array (Name, Age, Language) then you can locate the "31" anywhere in the array and specifically look at column 2 for the match. This will return which row is the first with the data.

If your columns are Chr(3)-delimited and the rows are still Chr(1)-delimited then use an "S" to find the row:

```
sNames = "Names" & "John@Mike@Steve@Rob" & "Addresses"
Nbr = Locate("Steve", sNames, 0, 0, "S3")  ' Nbr = 2
```

You still need to know the format of the array so, in this case, Steve was found in the known third position of the unknown second row.

LocateAdd

The LocateAdd function will add a value to the dynamic array only if the value being added does not already exist. If the value does exist but has associated subfields the new value will still be added to the end of the list. This function does not interact with SUB and SSUB fields and therefore treats a whole record with subfields as 1 string during the compare.

Group: Dynamic Array Extensions

Syntax: LocateAdd(vVAL, vInfo, [vDLM])

vVAL (Variant) is the value to be located or added to the Dynamic Array.

vInfo (Variant) is the dynamic array variable that will be searched.

vDLM (Variant) Optional – is the delimiter to use when searching.

Example:

```
Dim sNames As String
    sNames = "John#31#Canadian English@UnitedKingdom English&Mike"
      LocateAdd("Mike", sNames) ' Since Mike already exists nothing is added
      LocateAdd("John", sNames) ' Since John is associated with subfields in the first record, it will again be added
                to the end of the string:
    "John#31#Canadian English@UnitedKingdom English&Mike&John"
LocateAdd("Albert", sNames, "X")
```

This adds "Albert" to the end of the list and uses the letter 'X' to separate the last entry and Albert.

LocateDel

The LocateDel function will remove a value from the dynamic array and report the location in the list where it was. If the value does not exist, 0 is returned. This function does not interact with SUB and SSUB fields and therefore treats a whole record with subfields as 1 string during the compare.

Group: Dynamic Array Extensions

Syntax: vNbr = LocateDel(vVAL, vInfo, [vDLM]).

vNbr (Variant) the position in the dynamic array where the VAL was found and deleted.

vVAL (Variant) the value to be located and deleted from the Dynamic Array.

vInfo (Variant) is the dynamic array variable that will be searched.

vDLM (Variant) Optional – is the delimiter to use when searching.

Example:

```
Dim Nbr As Integer
Dim sNames As String
 sNames = "John#31#Canadian English@UnitedKingdom English&Mike"
    Nbr = LocateDel("Mike", sNames) ' Nbr = 2, the second record
    Nbr = LocateDel("31", sNames, Chr(2))
    Nbr = 2, since the age is a subfield specifying the Chr(2) delimiter finds "31" in the second position of the first
              record.
```

LocateField

The LocateField function may be used to find the index of a field, subfield, or sub-subfield within a dynamic array.

For example, if your string is: "Tom;32;England*Jane;29;Spain"

If you wanted to locate "32" with delim ";" the function will return 2 (the second field).

If you wanted for locate "England*Jane" with delim ";" the function will return 3 because they are considered 1 unit.

Note that the semicolon is ascii 59 and the asterisk is ascii 42.

The way the function operates is by searching the string on the designated character until it reaches an ascii character that has a lower value.

Group: DynamicArrayExt

Syntax: LocateField(ByVal Find as Variant, ByVal String as Variant, [Byval Delim]) as Variant

Find	(Variant) will return the index position of the sought after string/field.
String	(Variant) is the string value to be located.
[Delim]	(Variant) is the delimiter variable to be located.

Example:

```
Public dynArray As String
   Dim i As Variant
      Public Property Get delim(sign) As String
         Dim v As Variant
            v = Array(" ", "#", "@", "&")
         delim = v(sign)
      End Property
  Private Sub btnLoc1_Click()
  On Error Resume Next
      If TextBox1.Text = "" Then
            App.Balloon("Must have Field to find in Textbox", 1)
         Exit Sub
      End If
```

```
i = LocateField(TextBox1.Text, dynArray)
   App.MsgBox(TextBox1.Text & " is at: " & i)
End Sub
```

Rep

The Replace function replaces a field, subfield, or sub-subfield in a dynamic array.

Group Dynamic Array Group

Syntax StringData = vRep(vStringData, vFLD, [vSUB], [vSSUB], vSTR)

vStringData (Variant) is the dynamic array string variable name

vFLD (Variant) is the array field number to search.

vSUB (Variant) Optional – is the array subfield number.

vSSUB (Variant) Optional – is the array sub-subfield number.

vSTR (Variant) is data or variable name, which will be inserted at the dynamic array location (vFLD, vSUB, vSSUB).

Example

```
Given : "John#31#Canadian English@UnitedKingdom English&Mike"
StringData = Rep(StringData, 2, "Albert")  ' removes Mike and returns:
"John#31#Canadian English@UnitedKingdom English&Albert"
StringData = Rep(StringData, 1, "Fred")  ' removes whole first record and returns: "Fred&Albert"
Using subfields, given:
 "John#31#Canadian English@UnitedKingdom English&Mike"
StringData = Rep(StringData, 1, 2,  "80")  ' using the first record and the second subfield, 31 is replaced with 80 giving:
"John#80#Canadian English@UnitedKingdom English&Mike"
StringData = Rep(StringData, 1, 3, 2, "US English")  ' using the first record, the third subfield and the second sub subfield, United Kingdom English is replaces with US English giving:
"John#80#Canadian English@US English&Mike"
```

RField

The RField function searches a string from the right to extract a sub-string by using a specified delimiter character.

Group: Dynamic Array Extensions

Syntax: vVAL= RField(vStringData, vDelimiter, vStartField, [vNumberFields])

vVAL (Variant) is the resulting sub-string value located in StringData

vStringData (Variant) is a string or string variable to be searched

vDelimiter (Variant) is a specified delimiter character

vStartField (Variant) is the Delimiter to start from when searching the StringData

vNumberFields (Variant) Optional – specifies the number of sub-fields to retrieve.

Example:

If VAR = "111|222|333", then:

VAL = RField(VAR, "|", 1) returns VAL = '333'

VAL = RField(VAR, "|", 3) returns VAL = '111'

Note: If StartField = 1, then the RField function will return a sub-string which starts at the end of VAR, up to the first occurrence of Delimiter.

Embedded Procedure Object

The following section describes:

- **Embedded Procedures** – ERP and other stored procedures, macros, etc. that allow functions to be executed without using a front-end interface.
- **Properties and Methods** – The properties of the function are the values sent and received from the backend system and the methods are commands used to manipulate the function such as 'execute' or 'clear' that operate on the object itself.

Embedded Procedures

An Embedded Procedure refers to embedding VBA code that performs a previously created task. The user may generate code for:

- Business Functions - typically used with ERP systems.
- Business Objects - an object variable used by Microsoft Axapta.
- Database Table - used to generate Insert SQL statements.
- Stored Procedures - found in ODBC compliant databases.
- Transaction Macros - to be queued or executed.
- Vocollect Recordsets - defined Vocollect resources.
- Web Service Objects - based on Web Service Connector.

Embedded Procedures are intended to be an automated approach to writing VBA code that specifies parameters and then executes the procedure. Although you can write the code yourself, RFgen has a code generator that makes this process much easier.

Methods and Properties

The methods and properties available in the Embedded Procedures group are:

emProc.Clear, emProc.ColumnCount, emProc.DataSource, emProc.DebugLog, emProc.DisableParam, emProc.Execute,

emProc.ExecuteMethod, emProc.LogMode, emProc.Name, emProc.Param, emProc.ParamCount, emProc.ParamEx, emProc.ParamName>

emProc.Queue, emProc.QueueName, emProc.QueueOffline, emProc.QueueSeqNo, emProc.RowCount, and emProc.SetKeyField.

Clear

After the programmer or the Execute method has altered parameters, this command clears all parameters so it may be used again. This only effects the emProc.Param( ) values.

Group: Embedded Procedure Object

Syntax: emProc.Clear

ColumnCount

For a given table, this function will contain the number of parameters or columns as they appear when down-loaded.

Group: Embedded Procedure Object

Syntax: vValue = emProc.ColumnCount(vParamId)

vValue (Variant) contains the number of columns or parameters in the specified table.

vParamId (Variant) the name of the table usually referred to by name and in double quotes.

Example:

Dim nValue As Long

nValue = emProc.ColumnCount("RETURN")

ColumnName

For a given table or function, this function will display the name of a parameter when referenced by its index value. If there are no tables use the property ParamName.

Group: Embedded Procedure Object

Syntax: sValue = emProc.ColumnName(vTable, [nIndex])

sValue (String) contains the name of the parameter

vTable (Variant) the table or function name that contains parameters

nIndex (Long) Optional – the number of the parameter for which the name is retrieved. Left off and the complete list will be returned.

Example:

If the embedded procedure had these parameters declared:

emProc.Param("PLANT", "SIGN") = "I"

emProc.Param("PLANT", "OPTION") = "EQ"

emProc.Param("PLANT", "PLANT_LOW") = "3000"

then emProc.ColumnName("PLANT", 3) would return "PLANT_LOW"

DataSource

This property indicates the name of the data source to connect to. This is generally setup in the HOSTS file assigning an IP address to a name.

Group: Embedded Procedure Object

Syntax: emProc.DataSource = sValue

sValue (String) name of the data source

Example:

emProc.DataSource = "RFSample"

DebugLog

This property contains the path to the log file. The log file will contain all function calls by appending to this file.

Group: Embedded Procedure Object

Syntax: emProc.DebugLog = sValue

Alternate: sValue = emProc.DebugLog

sValue (String) the path to a text file

Example:

```
emProc.DebugLog = "\Program Files\Status.txt"
```

DisableParam

Only used for SAP connectivity, this method allows the user to request that SAP not send data back in tables that will not be used. If a BAPI returns 5 tables of data but only 1 will be used, the other 4 can be turned off to increase the speed SAP returns with results.

Group: Embedded Procedure Object

Syntax: emProc.DisableParam(vParamID, bDisabled)

vParamID (Variant) the name of the table that is not necessary.

bDisabled (Boolean) set to True if the table should be ignored.

Execute

After all values are correctly set, this command sends the values to the ERP system for evaluation. The function being executed is contained in the emProc.Name property. The properties of the procedure are then filled in with the results.

Syntax: [bValue =] emProc.Execute

bValue (Boolean) Optional – contains a True / False based upon its success.

Example:

```
SysteErr.Clear
emProc.Execute
```

ExecuteMethod

For Microsoft ERP systems, the user may specify a specific method to be executed. After all values are correctly set, this command sends the values to the ERP system for evaluation. The function being executed is contained in the emProc.Name property. The properties of the procedure are then filled in with the results.

Group: Embedded Procedure Object

Syntax: [bValue =] emProc.ExecuteMethod(sName)

bValue (Boolean) Optional – contains a True / False based upon its success.

sName (String) the name of the method to execute

LogMode

This property can be set to log nothing, everything or just errors as it relates to executing business functions, stored procedures or macros. All messages are written to the error log.

Group: Embedded Procedure Object

Syntax: emProc.LogMode = enValue

Alternate: enValue = emProc.LogMode

enValue (enLogMode) an enumeration containing 3 possible values:

LogNever – nothing is written to the log
LogAlways – all data, successes and failures are logged
LogFailure – only embedded procedure failures are logged

Example:

emProc.LogMode = LogAlways

Name

This property contains the function name to be executed.

Group: Embedded Procedure Object

Syntax: emProc.Name = sValue

Alternate: sValue = emProc.Name

sValue (String) the name of the business function

Example:

```
emProc.Name = "BAPI_GOODSMVT_CREATE"
```

Param

This property controls all the business function's parameter values.

Syntax: emProc.Param(vParamId, [vColName], [nRowNo]) = vValue

Alternate: vValue = emProc.Param(vParamId, [vColName], [nRowNo])

vValue (Variant) the result of the parameter's value or the value being placed in the parameter

vParamId (Variant) the name of the table usually referred to by name and in double quotes.

vColName (Variant) Optional – the name of the column within a table parameter

nRowNo (Long) Optional – the row number within a table parameter

Example:

```
emProc.Param("YEAR") = "2013"
 - or -
Dim vValue As Variant
vValue = emProc.Param("RESULTS", "ID", 1)
```

ParamCount

For a given business function, this function will contain the number of parameters as they appear when downloaded. Each table will count as 1 parameter.

Group: Embedded Procedure Object

Syntax: vValue = emProc.ParamCount

vValue (Variant) contains the number of parameters for the specified business function.

Example:

Dim nValue as Long

nValue = emProc.ParamCount

ParamEx

For SAP systems only, this property controls all the business function's parameter values and allows for nested parameters, like a table parameter that contains another table.

Group: Embedded Procedure Object

Syntax:

emProc.ParamEx(vParamId, vColName, [nRowNo]) = enValue

Alternate: enValue = emProc.ParamEx(vParamId, vColName, [nRowNo])

enValue (EmbeddedParam) the result of the parameter's value or the value being placed in the parameter

vParamId (Variant) the name of the table usually referred to by name and in double quotes.

vColName (Variant) the name of the column within a table parameter

nRowNo (Long) Optional – the row number within a table parameter

Example:

```
Dim sError As Variant

sError = emProc.ParamEx("RETURN", "MESSAGE", 1)
```

In the case of nested parameters, specify the parameter ID and the column that contains the nested table and then use "dot" notation to extend the statement.

```
Dim vValue As Variant

vValue = emProc.ParamEx("ParamId",  "Col1").Param("SubParam", "SubCol")
```

This notation can be used indefinitely to set or obtain data from structures within other structures. The properties available after the ParamEx property are:

```
emParam.ParamEx("ParamID", "Col").ColumnCount
```

emParam.ParamEx("ParamID", "Col").ColumnName

emParam.ParamEx("ParamID", "Col").Param

emParam.ParamEx("ParamID", "Col").ParamEx

emParam.ParamEx("ParamID", "Col").RowCount

ParamName

For a given function, this function will display the name of a parameter when referenced by its index value. If there are tables that contain parameters, use the property ColumnName.

Group: Embedded Procedure Object

Syntax: sValue = emProc.ParamName([nIndex])

sValue (String) contains the name of the parameter

nIndex (Long) Optional – the number of the parameter for which the name is retrieved. Left off a full list is returned.

Queue

This function returns a Boolean value depending on if the transaction could be queued. This is in place of executing the business function. The Transaction Manager will add this business function to the queue and execute it when the host is available and when it gets to the top of the queue. (Also see QueueSeqNo)

Group: Embedded Procedure Object

Syntax: bValue = emProc.Queue

bValue (Boolean) contains a True if the transaction was successfully queued.

Example:

Dim bValue as Boolean

bValue = emProc.Queue

QueueName

Multiple queues are allowed in the transaction manager process (as configured in Configure \ System Options \ Service Options.) This property will return the name of the queue currently in use. It can also be used to set which queue processes the transaction.

Group: Embedded Procedure Object

Syntax: sValue = emProc.QueueName

Alternate: emProc.QueueName = sValue

sValue (String) contains the name of the queue

Example:

Dim sValue as String

sValue = emProc.QueueName

- or -

emProc.QueueName = "AltQueue"

QueueOffline

This property sets or returns whether the embedded procedure is set to queue if the host is offline.

Group: Embedded Procedure Object

Syntax: bValue = emProc.QueueOffline

Alternate: emProc.QueueOffline = bValue

bValue (Boolean) contains a True or False depending on if the transaction can be or should be queued.

Example:

Dim bValue as Boolean

bValue = emProc.QueueOffline

- or -

emProc.QueueOffline = True

QueueSeqNo

This property returns the sequence number given to the queued function using the emProc.Queue method.

Group: Embedded Procedure Object

Syntax: nValue = emProc.QueueSeqNo

nValue (Long) contains the sequence number generated by the Transaction Manager when the function was queued.

Example:

Dim nValue As Long

emProc.Queue

nValue = emProc.QueueSeqNo

RowCount

For a given table, this function will contain the number of rows returned from the function given the passed parameters.

Group: Embedded Procedure Object

Syntax: vValue = emProc.RowCount(vParamId)

vValue (Variant) contains the number of rows in the specified table.

vParamId (Variant) the name of the table, usually referred to by name and in double quotes.

Example:

```
For i = 1 To emProc.RowCount("ET_BAPIRET")
   If emProc.Param("ET_BAPIRET", "TYPE", i) = "E" Then
      tmErrorText = GetErrDesc(emProc.Param("ET_BAPIRET", "NUMBER", i), _
         emProc.Param("ET_BAPIRET", "ID", i), _
         emProc.Param("ET_BAPIRET", "MESSAGE_V1", i), _
         emProc.Param("ET_BAPIRET", "MESSAGE_V2", i), _
         emProc.Param("ET_BAPIRET", "MESSAGE_V3", i), _
         emProc.Param("ET_BAPIRET", "MESSAGE_V4", i))
      GoTo ExitMe
   End If
Next
```

SetKeyFields

For a given table, this function will contain the list of keys to be used in the internal Where clause when updating a table.

Group: Embedded Procedure Object

Syntax: [bValue] = emProc.SetKeyFields(ParamArray)

bValue (Boolean) Optional – returns the success or failure of the command

ParamArray (param array of Variants) the name of each key for the table in quotes and separated by a comma

Example:

Dim emCUSTOMERS As New EmbeddedProc

emCUSTOMERS.Name = "CUSTOMERS"

emCUSTOMERS.DataSource = "RFSample"

emCUSTOMERS.Param("CUSTID") = 1001

emCUSTOMERS.Param("HOSTID") = "A1"

emCUSTOMERS.Param("NAME") = "Microsoft"

emCUSTOMERS.SetKeyFields("CUSTID", "HOSTID")

emCUSTOMERS.ExecuteMethod("update")

Enterprise Resource Planning Extensions

Commands relating to ERP capabilities are called from the ERP object.

The commands available for the Enterprise Resource Planning Extension are:

ERP.BeginTrans, ERP.CommitTrans, ERP.LogOff, ERP.LogOn, ERP.MakeList, ERP.ReadData, ERP.RollbackTrans, ERP.SetHardRelease, and ERP.SetSession.

BeginTrans

This command is used to retrieve an ERP connection from the managed pool and keep it for an unspecified amount of time. It would typically be used to execute a sequence of ERP commands against a pooled connection. Note: If the connection is not pooled, or will call only a single business function, this command is not needed. The first ERP data connection is the default Source value.

Group: Enterprise Resource Planning Extensions

Syntax: [bValue =] ERP.BeginTrans([vSource])

bValue (Boolean) Optional – A True/False notification that the command processed successfully

vSource (Variant) Optional – data source name (DSN) or the data source number

Example:

Dim bValue As Boolean

bValue = ERP.BeginTrans(1)

ERP.BeginTrans("SAP")

CommitTrans

This command is used to release an ERP connection back to the managed pool once the process is finished with it. Note: You must always use this function paired with ERP.BeginTrans, otherwise you will deplete the

pool of connections and prevent other users from having ERP access. The first ERP data connection is the default Source value.

Group: Enterprise Resource Planning Extensions

For an SAP system, this command will also execute BAPI_TRANSACTION_COMMIT.

Syntax: [bValue =] ERP.CommitTrans([vSource])

bValue (Boolean) Optional – A True/False notification that the command processed successfully

vSource (Variant) Optional – data source name (DSN) or the data source number

Example:

```
Dim bValue As Boolean
bValue = ERP.CommitTrans(1)
ERP.CommitTrans("SAP")
```

LogOff

This function is used to logoff a non-pooled ERP connection. Note: this function does not need to be called by the user. The server will call it automatically when shutting down the session. The first ERP data connection is the default Source value.

Group: Enterprise Resource Planning Extensions

Syntax: [bValue =] ERP.LogOff (vSource)

bValue (Boolean) Optional – A True/False notification that the command processed successfully

vSource (Variant) data source name (DSN) or the data source number

Example:

```
Dim bValue As Boolean
bValue = ERP.LogOff(1)
ERP.LogOff("SAP")
```

LogOn

This is used to logon the ERP connection and specify a user / password sequence for a non-pooled ERP connection (optional).

Note: the user does not always require this function as the server calls it automatically when a session starts.

Group: Enterprise Resource Planning Extensions

Syntax: [bValue =] ERP.LogOn (vSource, [vUserId], [vUserPwd], [vOptions])

bValue (Boolean) Optional – A True/False notification that the command processed successfully

vSource (Variant) data source name (DSN) or the data source number

vUserId (Variant) Optional – The login name to be used to connect to the ERP system

vUserPwd (Variant) Optional – The login password to be used to connect to the ERP system

vOptions (Variant) Optional – SAP only additional parameters that can be added to the logon string. Separate multiple parameters with the pipe (|) symbol.

CLIENT	- SAP Client
LANG	- Logon language
SYSNR	- SAP System number
ASHOST	- SAP application server
MSHOST	- SAP message server
GWHOST	- Gateway host
GWSERV	- Gateway service
R3NAME	- R/3 name
GROUP	- Group of SAP application servers
TPHOST	- Host of the external server program
TYPE	- Type of remote host (2 = R/2, 3 = R/3, E = External)
TRACE	- Enable RFC trace (1=on, 0 = off)
CODEPAGE	- Initial code page in SAP notation
LCHECK	- Enable logon check at open time (1=on, 0 = off)
QOSDISABLE	- Disable Quality of Service check (1=disable, 0 = enable)
GRT_DATA	- Additional data for GUI
USE_GUIHOST	- Redirect remote GUI to this Host
USE_GUISERV	- Redirect remote GUI to this Service
USE_GUIPROGID	- Server program id that will start the remote GUI
SNC_MODE	- Enable Secure Network Communications (1=on, 0 = off)
SNC_PARTNERNAME	- SNC partner (p:CN=R3, O=XYZ-INC, C=EN)
SNC_QOP	- SNC Level of security (1-9)
SNC_MYNAME	- SNC Name - overrides default SNC partner
SNC_LIB	- SNC service library path

DEST	- R/2 destination
SYSTEM	- Name of the system as used in the system landscape definition
LOGONMETHOD	- Authentication method used to log on to the destination (UIDPW, SAPLOGONTICKET, X509CERT)

Example:

```
Dim bValue As Boolean
bValue = ERP.LogOn("SAP", "User345", "Pass345")
bValue = ERP.LogOn(1, "User345", "Pass345", "CLIENT=800|TYPE=3")
bValue = ERP.LogOn("JDE", "JDEUser", "JDEPass1")
```

MakeList

This function executes a pass-through SQL 'select' statement against the ERP system's database and converts the results into a scrolling list. You may use App.ShowList to display the list to the user. The first ERP data connection is assumed as the source.

Group: Enterprise Resource Planning Extensions

Syntax: sMyList = ERP.MakeList(sSQL, [bRtnAllCols], [bNormalize], [bScale, nMaxRows)

sMyList (String) is the list to be returned for display (i.e., set RSP = MyList to display the list on the Client device.)

sSQL (String) is the SQL 'SELECT' statement to be sent to the database.

bRtnAllCols (Boolean) Optional – when set to True will return all the columns as the potential key, not just the first column. Default is False.

bNormalize (Boolean) Optional – when set to True will trim the spaces from the data so that it will display consistently. Default is False.

bScale (Boolean) Optional – formats the numeric values in the specified column if the database does not store decimals. This option will position a decimal at a position from the right side. A comma will be used for large numbers. This field is locale specific and will use the appropriate characters for each region. Default is True.

nMaxRows (Long) Optional – limits how many rows will be allowed in the list. Default is 500.

Example:

```
Dim sSQL As String
Dim sMyList As String
sSQL = "select PartNo from ItemMaster"
sMyList = ERP.MakeList(sSQL, True, True, True, 1000)
Rsp = App.ShowList(sMyList)
```

```
Or
Rsp = App.ShowList(ERP.MakeList(sSQL, True, True))
```

ReadData

This command executes a read-only SQL statement against the ERP system. Note: In the case of SAP, the business function RFC_READ_TABLE is used and it is not an officially released BAPI and has significant limitations. Therefore it may not work with all versions of SAP. If this is the case, please contact support for a solution.

Group: Enterprise Resource Planning Extensions

Syntax: [bValue =] ERP.ReadData(sSQL, sCols, sRows)

bValue (Boolean) Optional – A True/False notification that the command processed successfully

sSQL (String) the SQL statement to be executed on the ERP table

sCols (String) a variable containing the columns of the resulting data

sRows (String) a variable containing the rows of resulting data

SAP Limitations – since the RFC_READ_TABLE function is used, SQL statements must specify at least 1 field. Using an asterisk (*) or a function will not be accepted. Examples are "select * ", "select count(*) ", select SUM(QTY) …", etc. Only a comma-delimited list of field names is acceptable. Finally you may only specify 1 table in the SQL statement, no joins.

For example, if you would like to retrieve the Material numbers from a table, such as the Material Documents table within SAP, specify the following:

Example:

```
Dim bValue As Boolean
bValue = ERP.ReadData("select MATNR from MSEG", sCols, sRows)
```

RollbackTrans

This command is used to undo executed functions against an ERP connection assuming the ERP system is capable of undoing those functions. The first ERP data connection is the default Source value.

For an SAP system, this command will also execute BAPI_TRANSACTION_ROLLBACK.

Group: Enterprise Resource Planning Extensions

Syntax: [bValue =] ERP.RollbackTrans ([vSource])

bValue (Boolean) Optional – A True/False notification that the command processed successfully

vSource (Variant) Optional – data source name (DSN) or the data source number

Example:

```
Dim bValue As Boolean
bValue = ERP.RollbackTrans(1)
ERP.RollbackTrans("SAP")
```

SetHardRelease

This function (exclusively for JDE connections) is to be called at the beginning of any transaction macro that will be affected by the resource leak in JD Edwards. It sets an internal flag that will release the data pointers when an ERP.CommitTrans or ERP.RollbackTrans is called. At this point the internal flag is toggled off.

Group: Enterprise Resource Planning Extensions

Syntax: ERP.SetHardRelease

Example:

```
ERP.SetHardRelease
```

SetSession

If there is more than one ERP connection and a transaction may need them independently but at the same time, ERP.SetSession will direct embedded business functions to the specified session, ignoring the defaults. The 'DataSource' method will overwrite this function if it is included in the VBA code.

Group: Enterprise Resource Planning Extensions

Syntax: [bValue =] ERP.SetSession (vSource)

bValue (Boolean) Optional – A True/False notification that the command processed successfully

vSource (Variant) data source name (DSN) or the data source number. Setting this value to 0 will re-enable automatic determination of which data connection should be used based on which connection was used to download a particular business function being called.

Example:

```
Dim bValue As Boolean
bValue = ERP.SetSession(1)
ERP.SetSession("SAP")
```

Form Extensions

Form language extensions are a group of general commands that control the properties or behavior of a form or child controls on a form.

The extensions available with the form are:

Form.Advance, Form.Align, Form.Backcolor, Form.Backcolorex, Form.Backstyle, Form.backup, Form.BorderStyle, Form.Caption, Form.Checked, Form.Children, Form.Defaults, Form.Edits, Form.ErrMsg, Form.FieldID, Form.Font, Form.ForeColor, Form.format, Form.Height, Form.IconSet, Form.Image, Form.Index, Form.InputState, Form.KeyBoardMode, Form.Label, Form.Layout, Form.Left, Form.List, Form.Map, Form.MaskInput, Form.MonitorKeys, Form.PayNo, Form.Parent, Form.Required, Form.ScrollBars, Form.SelLength, Form.SetFocus, Form.ShowKeyBoard, Form.State, Form.Tab, Form.TabNo, Form.Tag, Form.Text, Form.Top, Form.Type, Form.Visible, and Form.Width.

Form.IconSet

This command is used to quickly change the icons displayed on a form. For example, if you have users for whom you want to change which icons are available at a specific moment in time, this extension allows you to change them as needed.

Group: Prompt-Specific Extensions

Syntax: Form.IconSet (Index, IconType, IconName, IconAction, IconVisible)

Index	(Long) the index of the existing icon The icon's index is typically in Themes > Application > SystemIcons > Manage Icons Collection > Icon ID
IconAction	The new icon action that will replace the action assigned to the existing icon -- select from Intellisense.
IconName	The name of the icon to be changed on a form. Select the icon switch to from Intellisense.
IconType	The icon that will replace the existing icon -- select from the drop down list in Intellisense.
IconVisible	(Optional) Sets whether the icon will be visible or not at runtime.

Example:

```
Form.IconSet(1,iconArrowFilled, "", "Backup")
```

Versions Supported: RFgen 5.2.4.2 and newer.

Form.PageNo

This command returns the page number of the active page (the page that is displaying at runtime.)

Group: Prompt-Specific Extensions

Syntax:

vPage = Form.PageNo

vPage (Variant) is the page number.

Example:

```
If MenuAction = "SysBack" Then
  MenuAction = ""
 'Get the page number of the active page and
 'make it the Case number
  Select Case Form.PageNo
   Case 1: App.ExitForm
   Case 2: Call BackFromPage2
   Case 3: Call BackFromPage3
  End Select
End If
Public Sub BackFromPage2
 '1stLine is on page 1
  lstLine.SetFocus
End Sub
Public Sub BackFromPage3
 'Txt2Mat is on page 2
 Txt2Mat.SetFocus
End Sub
```

JDE Processing Option

This function is used with systems in the Oracle JD Edwards ERP, and requires the downloaded JDE processing options to the form where you plan to use this object.

The parameters available for use with the Oracle JD Edwards Processing Options Extension are:

Count, JDEProcOpt, ParamName, ProgamId, SetProcOptionFields, TemplateId, Value, and Version.

JDEProcOpt

Use this object to retrieve and interact with processing options for specified versions of the JDE application. (Requires the JDE Processing Option be downloaded to your form.)

The properties of the function are values sent and received from the JDE Processing Options database schema.

Count

The number of processing option values in the JDEProcOpt object.

Group: JDE Processing Option

Syntax: Object value, Object property = JDEcount

Example:

```
Dim oJDE    As New JDEProcOpt
Dim iCnt    As Integer
oJDE.ProgramId = "P4113"
oJDE.Version     = "ZJDE0001"
iCnt = oJDE.Count
This will return total count 19
```

P4113.ZJDE0001

###	Parameter Name	Page	Seq No	Description
1	Documenttype			[0.0] Documenttype
2	DefaultFROMPrimLocation		1	[0.1] DefaultFROMPrimLocation
3	DefaultTOPrimLocation		2	[0.2] DefaultTOPrimLocation
4	ProtectCosts	2		[2.0] ProtectCosts
5	SummaryMode	2	1	[2.1] SummaryMode
6	AllowHeldLots	2	2	[2.2] AllowHeldLots
7	AllowOverQtyAvailable	2	4	[2.4] AllowOverQtyAvailable
8	JournalEntriesVersion	1	1	[1.1] JournalEntriesVersion
9	ItemLedgerVersion	1	2	[1.2] ItemLedgerVersion
10	OutInteroperabilityType	3		[3.0] OutInteroperabilityType
11	AgreementAssignProcess	4		[4.0]AgreementAssignProcess
12	PODefaultLotStatus	2	5	[2.5] PODefaultLotStatus
13	LotGroup	2	3	[2.3] LotGroup
14	LPNGenerationMethod	5		[5.0] LPNGenerationMethod
15	BuildStructure	5	1	[5.1] BuildStructure
16	LicensePlateWindow	5	2	[5.2] LicensePlateWindow
17	SearchAndSelect		3	[0.3] SearchAndSelect
18	PCW10Version	1	3	[1.3] PCW10Version
19	ProductionNoConsumption	2	6	[2.6]ProductionNoConsumption

ParamName

The name of a processing option at a specified index.

Group: JDE Processing Option

Syntax: Object value, Object property = JDEparamname

Example:

Dim sDOCTYPE As String

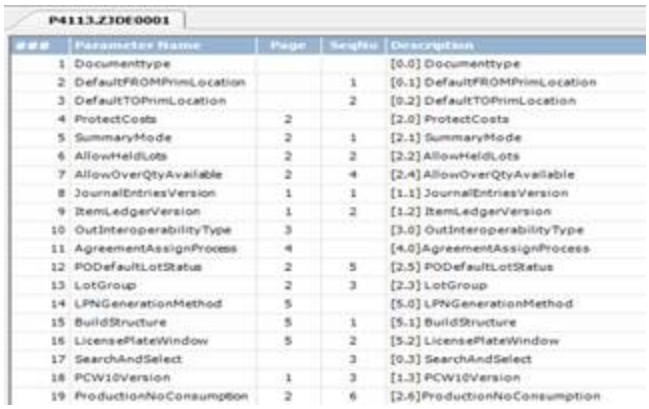
Dim oJDE As New JDEProcOpt

oJDE.ProgramId = "P4113"

oJDE.Version = "ZJDE0001"

sDOCTYPE = oJDE.ParamName(1)

The result of sDOCTYPE will be "Documenttype"



P4113.ZJDE0001

###	Parameter Name	Page	Seqno	Description
1	Documenttype			[0.0] Documenttype
2	DefaultFROMPrimLocation		1	[0.1] DefaultFROMPrimLocation
3	DefaultTOPrimLocation		2	[0.2] DefaultTOPrimLocation
4	ProtectCosts	2		[2.0] ProtectCosts
5	SummaryMode	2	1	[2.1] SummaryMode
6	AllowHeldLots	2	2	[2.2] AllowHeldLots
7	AllowOverQtyAvailable	2	4	[2.4] AllowOverQtyAvailable
8	JournalEntriesVersion	1	1	[1.1] JournalEntriesVersion
9	ItemLedgerVersion	1	2	[1.2] ItemLedgerVersion
10	OutInteroperabilityType	3		[3.0] OutInteroperabilityType
11	AgreementAssignProcess	4		[4.0]AgreementAssignProcess
12	PODefaultLotStatus	2	5	[2.5] PODefaultLotStatus
13	LotGroup	2	3	[2.3] LotGroup
14	LPNGenerationMethod	5		[5.0] LPNGenerationMethod
15	BuildStructure	5	1	[5.1] BuildStructure
16	LicensePlateWindow	5	2	[5.2] LicensePlateWindow
17	SearchAndSelect		3	[0.3] SearchAndSelect
18	PCW10Version	1	3	[1.3] PCW10Version
19	ProductionNoConsumption	2	6	[2.6]ProductionNoConsumption

ProgramId

The programID of the JDE processing option.

Group: JDE Processing Option

Syntax:

Object value, Object property = JDEprogramID

Example:

```
Private moProcOPT As New JDEProcOpt
' Get Proc.Opt. Version from Menu
  msPgm    = App.GetValue("Pgm")
  msVersion = App.GetValue("Vers")
  moProcOPT.ProgramId = "P4113"
  moProcOPT.Version   = "ZJDE0001"
  msDOCTYPE = moProcOPT.Value("Documenttype")
' get DocType from Proc.Opt.
  msDOCTYPE = GetProcOpt(msPgm,msVersion,"1;1",sHeader)
```

TemplateId

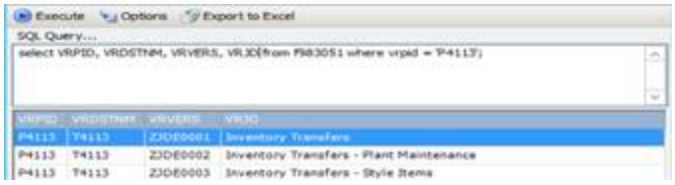
The ID of the template used for the Processing Object.

Group: JDE Processing Option

Syntax: Object value, Object property = JDEtemplateID

Example:

```
Dim sTemplateId     As String
Dim oJDE            As New JDEProcOpt
oJDE.ProgramID = "P4113"
oJDE.Version   = "ZJDE0001"
sTemplateId = oJDE.TemplateId
The result of sTemplateId is 'T4113'
```



Value

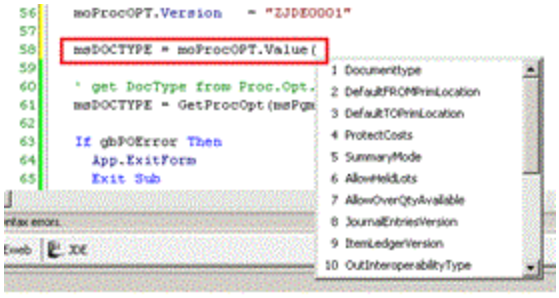
This property returns the value specified in the process option table.

Group: JDE Processing Option

Syntax:

Object value, Object property = ProcOPT.value( )

Note: To view Value property parameters, hit CTRL+Space after you open the parenthesis of a value property. Alternatively, you can right-click in the scripting window, and select "embed code" which gives you an option to generate some of this code.



Example:

```
Private moProcOPT    As New JDEProcOpt
' Get Proc.Opt. Version from Menu
  msPgm     = App.GetValue("Pgm")
  moVersion = App.GetValue("Vers")
  moProcOPT.ProgramId = "P4113"
  moProcOPT.Version   = "ZJDE0001"
  msDOCTYPE = moProcOPT.Value("Documenttype")
' get DocType from Proc.Opt.
  msDOCTYPE = GetProcOpt(msPgm,msVersion,"1;1",sHeader)
```

Version

Obtains the version of the object specified.

Group: JDE Processing Option

Syntax: Object value, Object property = JDEversion

Example:

```
  ' Get Proc.Opt. Version from Menu
    msPgm  = App.GetValue("Pgm")
    msVersion = App.GetValue("Vers")
    moProcOPT.ProgramId = "P4113"
    moProcOPT.Version   = "ZJDE0001"
    msDOCTYPE = moProcOPT.Value("Documenttype")

  ' get DocType from Proc.Opt.
    msDOCTYPE = GetProcOpt(msPgm,msVersion,"1;1",sHeader)
```

List

This property contains a collection of properties, methods, and objects for creating and stylizing a list. The list collection can be generated via the AddItem method or with the DB.MakeList or the App.MakeList or Prompt.List.LoadCells functions and then assigned to the list box, combo box, or menu via this property.

These properties and methods only apply to prompts that can contain a list such as the ListBox, MenuList and Combobox, DataGrid, TreeView and PanelList.

Group: Prompt-Specific Extensions

Syntax: RFPrompt.List.[The property or method selected from Intellisense List).

Examples:

```
'The following uses the DB.MakeList function in the ListBox_GotFocus event to populate the Lsit property of teh ListBox control.

Dim sMyList As String

Dim sSQL As String

sSQL = "select PartNo from Inventory"

sMyList = DB.MakeList(sSQL)

RFPrompt (4).List = sMyList
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

List

This property contains a collection of properties, methods, and objects for creating and stylizing a list. The list collection can be generated via the AddItem method or with the DB.MakeList or the App.MakeList or Prompt.List.LoadCells functions and then assigned to the list box, combo box, or menu via this property.

These properties and methods only apply to prompts that can contain a list such as the ListBox, MenuList and Combobox, DataGrid, TreeView and PanelList.

Group: Prompt-Specific Extensions

Syntax: RFPrompt.List.[The property or method selected from Intellisense List).

Examples:

```
'The following uses the DB.MakeList function in the ListBox_GotFocus event to populate the Lsit property of teh ListBox control.

Dim sMyList As String

Dim sSQL As String

sSQL = "select PartNo from Inventory"

sMyList = DB.MakeList(sSQL)

RFPrompt (4).List = sMyList
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

List.ActiveRow

This returns a pointer to the currently selected row, or returns nothing if no row was selected. It is simply a fast path to what the user clicked on.

[Control].List. ActiveRow is different than [Control].List.Index in that there is an integer representing the active element in the list (-1 = unselected, 0-nn for elements). You would use [Control].List.Index like [Con-

trol].List.Row([Control].List.Index).Cell(1).Value. But with the active row object, you would use it like [Control].List.ActiveRow.Cell(1).Value.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.ActiveRow

ActiveRow (vbaRow) The place where the cursor is to be returned.

Example

```
Dim i As Integer
Dim sRowData As String
Dim iCol As Integer
    For i = 1 To iCol
        sRowData = sRowData & CStr(RFPrompt(sRFPrompt).List.ActiveRow.Cell(i).Value) & vbCrLf
    Next
```

List.AddColSet

This method is used to add a set of columns in an empty ListBox control, or to replace an existing set of columns in a ListBox. This method is also used to add a new panel to a PanelList control in the event you want the arrangement of data to be different from the previous panel.

Group: Prompt-Specific Extensions

Syntax:

Prompt.List.AddColSet( )

Example:

```
'Builds your first set of columns
  ListBox1.List.AddColSet(1)
'Makes the columns easier to see
  ListBox1.List.ColSet(1).LineStyle = LineBoth
'Adds a column and designates it as the first in column in the set (group) of columns.
   ListBox1.List.ColSet(1).AppendColumn
'Adds a caption to the column heading to the first column in a set of columns
  ListBox1.List.ColSet(1).Column(1).Caption = "Col1"
```

List.AddItem

This method is used to build a list of values in a Search List or a List control and return a row object. This can be used for a single or multiple columns if desired.

Group: Prompt-Specific Extensions

Syntax: [oVBARow] = List.AddItem(SelValue,,DisplayColumn())

vSelValue (Variant) Selects the values from a referenced variable, where the first SelValue is used to define the number of columns that are to be created in the list, and the second SelValue defines the number of rows that are to be created and displayed in the list.

DisplayColumns () Will display the data that was added to the Search List

vParamArray Is a keyword. Creates an array from the values provided by the two arguments SelValue.

vbaRows Optional. Returns the value for the vba row object.

Example:

The following uses the AddItem method in the OnEnter event in one prompt to populate the list in another prompt.

```
For i = 1 To iCnt
sSrcBin = DB.Extract(sCols, sRows, i, "VLPLA")
sOrder = DB.Extract(sCols, sRows, i, "WHO")
If sSrcBin <> "" Then
If Locate(sSrcBin, sRows1) = 0 And Locate(sOrder, sDups) = 0 Then
oList.AddItem(sOrder, RemoveZeros(sOrder))
LocateAdd(sOrder, sDups)
End If
End If
Next
```

List.AddItemEx

This alternate method is used to manually add items to a MenuList control set to a graphical mode. You must specify the value to be returned should the user choose the selection. Image Resource name applies a graphic to the list entry. The Options parameter is used if the MenuList control is acting as a menu where the user can select forms to go to and each one requires additional passed information. See the Options column in the Menus / Roles List. The Display Columns array is the data to be displayed across multiple columns if desired.

Group: Prompt-Specific Extensions

Syntax:

[vIndex =] PromptID.List.AddItemEx(vValue, vImage, vOptions, vDispCols)

vIndex (Variant) Optional – the row index where the new entry was appended

vValue (Variant) the item to add to the list

vImage (Variant) the image associated with the list entry

vOptions (Variant) the option for the menu item

vDispCols (Variant) the data to be displayed. This is in the format: "display1", "display2", "display3". The MenuList control will determine the widest value in each column and format the columns so they align properly regardless of font.

Example:

The following uses the AddItemEx method in the OnEnter event in one prompt to populate the list in another prompt.

```
Public Sub txtBox_OnEnter(Rsp As String, Cancel As Boolean, ErrMsg As String)
  On Error Resume Next
  lstCars.List.AddItemEx("1","imgCheap", "", "VW Bug","Chevy Colt","Dodge Dart")
  lstCars.List.AddItemEx("2","imgMiddle", "", "Prius","Camry","Tacoma")
  lstCars.List.AddItemEx("3","imgExpensive", "", "Corvette","Tesla","Lamborghini")
End Sub
```

List.BeginUpdate

This extension will begin the process of preventing the list from drawing its content while the client is waiting for the server to retrieve all the changes. The client will continue to wait until a prompt.List.EndUpdate command is issued.

There are a variety of scenarios where this command can be used, but one of the most common is with the processing of large lists of data. The submission of partial data packets to the client can trigger time-consuming screen updates which can be a toll on the client . This command will help with performance by having the server do the updates, then, when the command "prompt.List.EndUpdates" is executed, the list can revert to its normal drawing state.

Remember to use the prompt.List.EndUpdate script to restore the list to its drawing state or else you'll have a blank screen.

Group: Prompt-Specific Extensions

Syntax: Prompt.List.BeginUpdate()

Example:

```
Private Sub Form_Load()
On Error Resume Next
'This refreshes the breadcrumbs at the top of a form with a user's prompt selections.
```

```
        Form.Caption = App.GetString(Link.msTitle)
        txtFilter.Visible = False
        listItems.List.BeginUpdate
        listItems.List.LoadCells(Link.msList, Link.mbRowValue, Link.miColumns)
        listItems.List.EndUpdate
        Link.mbCancel = True
    End Sub
>
```

Versions Supported: RFgen 5.2.4.2 and newer.

List.Cell

This method is used to read or change values or properties of a specific cell within a control that contains multiple rows and columns.

Group: Prompt-Specific Extensions

Syntax:

PromptID.List.Cell(Row, Col).<method or property>

Row (Long) specifies the row number in the grid.

Col (Long) specifies the column number in the grid.

Example:

```
RFPrompt("lstCars").List.Cell(2, 2).Value = "1969"
RFPrompt(2).List.Cell(2, 2).Bold = True
```

List.Cell(x,y).BackColor1

This method is used to read or change the primary background color of a specific cell within a control that contains multiple rows and columns.

Group: Prompt-Specific Extensions

Syntax:

PromptID.List.Cell(x,y).BackColor1 = vValue

Alternate: lValue = PromptID.List.Cell(x,y).BackColor1

lValue (Long) is the color.

vValue (Variant) sets the color.

Example:

```
lstCars.List.Cell(2, 2).BackColor1 = RGB(255,255,0) 'yellow
lstCars.List.Cell(2, 2).BackColor1 = &HFF0000      'blue
lstCars.List.Cell(2, 2).BackColor1 = QBColor(5)    'magenta
RFPrompt("lstCars").List.Cell(2, 2).BackColor1 = vbWhite
RFPrompt(1).List.Cell(2, 2).BackColor1 = vbWhite
```

List.Cell(x,y).BackColor2

This method is used to read or change the secondary background color of a specific cell within a control that contains multiple rows and columns. It is used to produce gradients from one color to another.

Group: Prompt-Specific Extensions

Syntax:

PromptID.List.Cell(x,y).BackColor2 = vValue

Alternate: lValue = PromptID.List.Cell(x,y).BackColor2

lValue (Long) is the color.

vValue (Variant) sets the color.

Example:

```
lstCars.List.Cell(2, 2).BackGradient = GradientVertical
lstCars.List.Cell(2, 2).BackColor2 = RGB(255,255,0) 'yellow
lstCars.List.Cell(2, 2).BackColor2 = &HFF0000      'blue
lstCars.List.Cell(2, 2).BackColor2 = QBColor(5)    'magenta
RFPrompt("lstCars").List.Cell(2, 2).BackColor2 = vbWhite
RFPrompt(1).List.Cell(2, 2).BackColor2 = vbWhite
```

List.Cell(x,y).BackGradient

This property uses either just the BackColor(1) for solid backgrounds or both BackColor properties to create gradients in one of several directions for a specific cell within a control that contains multiple rows and columns.

Group: Prompt-Specific Extensions

Syntax:

PromptID.List.Cell(x,y).BackGradient = enValue

enValue (enGradients) enumeration that contains GradientDefault, GradientDiagonalLeft, GradientDiagonalRight, GradientHorizontal, GradientNone, GradientVertical and GradientVerticalSplit

Example:

```
lstCars.List.Cell(2, 2).BackColor1 = RGB(0,0,255)
lstCars.List.Cell(2, 2).BackColor2 = vbWhite
lstCars.List.Cell(2, 2).BackGradient = GradientVertical
lstCars.List.Cell(2, 2).BackGradient = GradientVertical
lstCars.List.Cell(2, 2).BackGradient = GradientVertical
```

List.Cell(x,y).Bold

This property accesses the prompt's data field bold option for a specific cell within a control that contains multiple rows and columns.

Group: Prompt-Specific Extensions

Syntax:

PromptID.List.Cell(x,y).Bold = bValue

Alternate: bValue = PromptID.List.Cell(x,y).Bold

bValue (Boolean) is True or False for bold or not bold.

Example:

```
lstCars.List.Cell(2, 2).Bold = True
RFPrompt("lstCars").List.Cell(2, 2).Bold = True
RFPrompt(2).List.Cell(2, 2).Bold = True
```

List.Cell(x,y).Expanded

This property expands or collapses all child nodes of the row specified for a Tree control. The column number is typically 1.

Group: Prompt-Specific Extensions

Syntax:

PromptID.List.Cell(x,y).Expanded = bValue

Alternate: bValue = PromptID.List.Cell(x,y).Expanded

bValue (Boolean) is True or False for expanding or collapsing all child nodes of the specified row

Example:

```
lstCars.List.Cell(2, 1).Expanded = True
lstCars.List.Cell(2, 1).Expanded = False
```

List.Cell(x,y).ForeColor

This method is used to read or change the fore color of a specific cell within a control that contains multiple rows and columns.

Group: Prompt-Specific Extensions

Syntax:

PromptID.List.Cell(x,y).ForeColor = vValue

Alternate: lValue = PromptID.List.Cell(x,y).ForeColor

lValue (Long) is the color.

vValue (Variant) sets the color.

Example:

```
lstCars.List.Cell(2, 2).ForeColor = RGB(255,255,0) 'yellow
lstCars.List.Cell(2, 2).ForeColor = &HFF0000 'blue
lstCars.List.Cell(2, 2).ForeColor = QBColor(5) 'magenta
RFPrompt("lstCars").List.Cell(2, 2).ForeColor = vbWhite
RFPrompt(1).List.Cell(2, 2).ForeColor = vbWhite
```

List.Cell(x,y).Indent

This property gets or sets the level of the specified node within the tree. This property is only used for the Tree control.

Group: Prompt-Specific Extensions

Syntax:

PromptID.List.Cell(x,y).Indent = nValue

Alternate: nValue = PromptID.List.Cell(x,y).Indent

nValue (Long) is the node level within the tree

Example:

```
lstCars.List.Cell(5, 1).Indent = 4
```

List.Cell(x,y).Italic

This property accesses the prompt's data field italic option for a specific cell within a control that contains multiple rows and columns.

Group: Prompt-Specific Extensions

Syntax:

PromptID.List.Cell(x,y).Italic = bValue

Alternate: bValue = PromptID.List.Cell(x,y).Italic

bValue (Boolean) is True or False for italic or not italic.

Example:

```
lstCars.List.Cell(2, 2).Italic = True
RFPrompt("lstCars").List.Cell(2, 2).Italic = True
RFPrompt(2).List.Cell(2, 2).Italic = True
```

List.Cell(x,y).Underline

This property accesses the prompt's data field underline option for a specific cell within a control that contains multiple rows and columns.

Group: Prompt-Specific Extensions

Syntax:

PromptID.List.Cell(x,y).Underline = bValue

Alternate: bValue = PromptID.List.Cell(x,y).Underline

bValue (Boolean) is True or False for underline or not underline.

Example:

```
lstCars.List.Cell(2, 2).Underline = True
RFPrompt("lstCars").List.Cell(2, 2).Underline = True
RFPrompt(2).List.Cell(2, 2).Underline = True
```

List.Cell(x,y).Value

This property accesses the prompt's data field value for a specific cell within a control that contains multiple rows and columns.

Group: Prompt-Specific Extensions

Syntax:

PromptID.List.Cell(x,y).Value = vValue

Alternate: vValue = PromptID.List.Cell(x,y).Value

vValue (Variant) is the value within the cell.

Example:

```
lstCars.List.Cell(2, 2).Value = "1"
RFPrompt("lstCars").List.Cell(2, 2).Value = "1"
vValue = RFPrompt(2).List.Cell(2, 2).Value
```

List.Clear

This method is used to delete all columns and column sets from a List Box, Combo Box or Menu List. Note that there is no function to restore the columns or column sets to their original designer values after they have been cleared (deleted).

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Clear

Clear Will delete columns and column sets.

bClearColumns (Boolean) Optional – True will delete all columns and column sets; False will also delete all columns and column sets.

Example:

```
lstCars.List.Clear(False)
RFPrompt("lstCars").List.Clear
RFPrompt(2).List.Clear
```

List.Column(x)

This method is used to read or change properties of a specific column within a control that contains columns.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(Col).<method or property>

Column(x) (Long) specifies the column number in the grid

Example:

```
lstCars.List.Column(2).Width = 10
```

```
RFPrompt(2).List.Column(2).Width = 10
```

List.Column(x).Align

This property aligns the text or object of the column within a control that contains columns. For text types use the Text enumerations. For Image or Checkbox types use the general enumerations. The options are: BottomCenter, BottomLeft, BottomRight, CenterCenter, CenterLeft, CenterRight, TextCenter, TextLeft, TextRight., TopCenter, TopLeft, TopRight

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).Align = enValue

enValue (enColAlign) an enumeration that contains BottomCenter, BottomLeft, BottomRight, CenterCenter, CenterLeft, CenterRight, TextCenter, TextLeft, TextRight, TopCenter, TopLeft, TopRight

Example:

```
lstCars.List.Column(2).Align = TextCenter
RFPrompt("lstCars").List.Column(2).Align = TextCenter
```

List.Column(x).Autosize

This property will size the column based on the widest value in that column.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).Autosize = bValue

bValue (Boolean) set to True or False to autosize the width of the column.

Example:

```
lstCars.List.Column(2).Autosize = True
RFPrompt("lstCars").Autosize = True
RFPrompt(2).Autosize = True
```

List.Column(x).BackColor1

This method is used to read or change the primary background color of the whole column within a control that contains multiple rows and columns.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).BackColor1 = vValue

Alternate: lValue = PromptID.List.Column(x).BackColor1

lValue (Long) is the color.

vValue (Variant) sets the color.

Examples:

lstCars.List.Column(2).BackColor1 = RGB(255,255,0) 'yellow

lstCars.List.Column(2).BackColor1 = &HFF0000 'blue

lstCars.List.Column(2).BackColor1 = QBColor(5) 'magenta

RFPrompt("lstCars").List.Column(2).BackColor1 = vbWhite

RFPrompt(1).List.Column(2).BackColor1 = vbWhite

List.Column(x).BackColor2

This method is used to read or change the secondary background color of the whole column within a control that contains multiple rows and columns. It is used to produce gradients from one color to another.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).BackColor2 = vValue

Alternate: lValue = PromptID.List.Column(x).BackColor2

lValue (Long) is the color.

vValue (Variant) sets the color.

Example:

```
lstCars.List.Column(2).BackGradient = GradientVertical
lstCars.List.Column(2).BackColor2 = RGB(255,255,0) 'yellow
lstCars.List.Column(2).BackColor2 = &HFF0000 'blue
lstCars.List.Column(2).BackColor2 = QBColor(5) 'magenta
RFPrompt("lstCars").List.Column(2).BackColor2 = vbWhite
RFPrompt(1).List.Column(2).BackColor2 = vbWhite
```

List.Column(x).BackGradient

This property uses either just the BackColor(1) for solid backgrounds or both BackColor properties to create gradients in one of several directions for the whole column within a control that contains multiple columns.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).BackGradient = enValue

enValue (enGradients) enumeration that contains GradientDefault, GradientDiagonalLeft, GradientDiagonalRight, GradientHorizontal, GradientNone, GradientVertical and GradientVerticalSplit

Example:

```
lstCars.List.Column(2).BackColor1 = RGB(0,0,255)
lstCars.List.Column(2).BackColor2 = vbWhite
lstCars.List.Column(2).BackGradient = GradientVertical
RFPrompt("lstCars").List.Column(2).BackGradient = GradientVertical
RFPrompt(1).List.Column(2).BackGradient = GradientVertical
```

List.Column(x).Bold

This property accesses the column's bold option for the whole column within a control that contains columns.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).Bold = bValue

Alternate: bValue = PromptID.List.Column(x).Bold

bValue (Boolean) is True or False for bold or not bold.

Example:

```
lstCars.List.Column(2).Bold = True
RFPrompt("lstCars").List.Column(2).Bold = True
RFPrompt(2).List.Column(2).Bold = True
```

List.Column(x).Caption

This property accesses the caption associated with the specified column.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).Caption = vValue

Alternate: sValue = PromptID.List.Column(x).Caption

sValue (String) is the title of the column.

vValue (Variant) sets the title of the column.

Example:

```
lstCars.List.Column(2).Caption = "Model"
```

```
RFPrompt("lstCars").List.Column(2).Caption = "Model"
RFPrompt(1).List.Column(2).Caption = "Model"
```

List.Column(x).DisplayOnly

This property accesses the display only property associated with the specified column. Setting it to True makes all the cells in that column unchangeable.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).DisplayOnly = bValue

Alternate: bValue = PromptID.List.Column(x).DisplayOnly

bValue (Boolean) is the display only state for the prompt.

Example:

```
lstCars.List.Column(2).DisplayOnly = True
RFPrompt("txtPart").List.Column(2).DisplayOnly = False
RFPrompt(2).List.Column(2).DisplayOnly = False
```

List.Column(x).FontSize

This property accesses the font size parameter associated with a specified column in a coltrol that supports columns like a Listbox or Combobox.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).FontSize = lValue

Alternate: vValue = PromptID. List.Column(x).FontSize

vValue (Variant) is the font size.

lValue (Long) sets the font size.

Example:

```
lstCars.List.Column(2).FontSize = 15
```

List.Column(x).ForeColor

This property accesses the column's fore color property associated with the specified column.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).ForeColor = lValue

Alternate: vValue = PromptID. List.Column(x).ForeColor

vValue (Variant) is the color.

lValue (Long) sets the color.

Example:

```
lstCars.List.Column(2).ForeColor = RGB(255,255,0) 'yellow
lstCars.List.Column(2).ForeColor = &HFF0000      'blue
lstCars.List.Column(2).ForeColor = QBColor(5)    'magenta
RFPrompt("lstCars").List.Column(2).ForeColor = vbWhite
RFPrompt(2).List.Column(2).ForeColor = vbWhite
```

List.Column(x).Format

This property affects the format of the whole specified column. It is only an extension of the Format VBA command.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).Format = sValue

Alternate: sValue = PromptID.List.Column(x).Format

sValue (String) is the format mask to use when displaying data for the prompt.

Examples:

```
lstCars.List.Column(2).Format = "hh:mm"
RFPrompt("lstCars").List.Column(2).Format = "hh:mm"

c - General Date
dddddd - Long Date
ddddd - Short Date
ttttt - Long Time
hh:mm AMPM - Medium Time
hh:mm - Short Time
$#,##0.00 or ($#,##0.00) - Currency 0.00 - Fixed
#,##0.00 - Standard 0.00% - Percent 0.00E+00 - Scientific
Yes/No - Return "No" if zero, else return "Yes"
True/False - Return "True" if zero, else return "False"
On/Off - Return "On" if zero, else return "Off"
```

For further examples get help on the VB FORMAT command.

List.Column(x).ImageHeight

This property sets the images in this column to a specific height with prompts that support column.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).ImageHeight = lValue

Alternate: lValue = PromptID.List.Column(x).ImageHeight

lValue (Long) is the pixel height size for images in this column

Example:

```
lstCars.List.Column(1).ImageHeight = 10
```

List.Column(x).ImageWidth

This property sets the images in this column to a specific width with prompts that support column.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).ImageWidth = lValue

Alternate: lValue = PromptID.List.Column(x).ImageWidth

lValue (Long) is the pixel width size for images in this column

Example:

```
lstCars.List.Column(1).ImageWidth = 10
```

List.Column(x).Italic

This property accesses the column's italic option for the whole column within a control that contains columns.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).Italic = bValue

Alternate: bValue = PromptID.List.Column(x).Italic

bValue (Boolean) is True or False for italic or not italic.

Examples:

lstCars.List.Column(2).Italic = True

RFPrompt("lstCars").List.Column(2).Italic = True

RFPrompt(2).List.Column(2).Italic = True

List.Column(x).MarginBottom

This property pads the bottom of all the cells in a specified column. It also contributes to the overall height of the entire row.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).MarginBottom = lValue

Alternate: lValue = PromptID.List.Column(x).MarginBottom

lValue (Long) the padding in pixels between the contents of a cell and the bottom of the cell.

Example:

```
lstCars.List.Column(2).MarginBottom = 5
RFPrompt("lstCars").List.Column(2).MarginBottom = 5
RFPrompt(2).List.Column(2).MarginBottom = 5
```

List.Column(x).MarginLeft

This property pads the left of all the cells in a specified column.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).MarginLeft = lValue

Alternate: lValue = PromptID.List.Column(x).MarginLeft

lValue (Long) the padding in pixels between the contents of a cell and the left of the cell.

Examples:

lstCars.List.Column(2).MarginLeft = 5

RFPrompt("lstCars").List.Column(2).MarginLeft = 5

RFPrompt(2).List.Column(2).MarginLeft = 5

List.Column(x).MarginRight

This property pads the right of all the cells in a specified column.

Syntax: PromptID.List.Column(x).MarginRight = lValue

Alternate: lValue = PromptID.List.Column(x).MarginRight

lValue (Long) the padding in pixels between the contents of a cell and the right of the cell.

Examples:

```
lstCars.List.Column(2).MarginRight = 5
```

```
RFPrompt("lstCars").List.Column(2).MarginRight = 5
RFPrompt(2).List.Column(2).MarginRight = 5
```

List.Column(x).MarginTop

This property pads the top of all the cells in a specified column. It also contributes to the overall height of the entire row.

Syntax: PromptID.List.Column(x).MarginTop = lValue

Alternate: lValue = PromptID.List.Column(x).MarginTop

lValue (Long) the padding in pixels between the contents of a cell and the top of the cell.

Examples:

```
lstCars.List.Column(2).MarginTop = 5
RFPrompt("lstCars").List.Column(2).MarginTop = 5
RFPrompt(2).List.Column(2).MarginTop = 5
```

List.Column(x).ScaleDecimals

This property formats the numeric values in the specified column if the database does not store decimals. This option will position a decimal at a position from the right side. A comma will be used for large numbers. This field is locale specific and will use the appropriate characters for each region.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).ScaleDecimals = vValue

Alternate: vValue = PromptID.List.Column(x).ScaleDecimals

vValue (Variant) is the decimal position.

Examples:

```
lstCars.List.Column(1).ScaleDecimals = 2    'This should how 30.00 if the value was 3000.
RFPrompt("lstCars").List.Column(1).ScaleDecimals = 2
RFPrompt(8).List.Column(1).ScaleDecimals = 2
```

List.Column(x).Style

This property gets or sets the type of column. The values are Text, Image, and Check box.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).Style = enStyle

Alternate: enStyle = PromptID.List.Column(x).Style

enStyle (enColumnStyle) Contains ColumnStyleCheck, ColumnStyleImage, and ColumnStyleText

Examples:

lstCars.List.Column(1).Style = ColumnStyleCheck

RFPrompt("lstCars").List.Column(1).Style = ColumnStyleText

RFPrompt(8).List.Column(1).Style = ColumnStyleImage

List.Column(x).TrimSpaces

This property formats the values in the specified column by deleting the leading and trailing spaces in the data.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).TrimSpaces = bValue

Alternate: bValue = PromptID.List.Column(x).TrimSpaces

bValue (Boolean) set to True to trim all space from the data.

Examples:

lstCars.List.Column(1).TrimSpaces = True

RFPrompt("lstCars").List.Column(1). TrimSpaces = True

RFPrompt(8).List.Column(1). TrimSpaces = True

List.Column(x).Underline

This property accesses the column's underline option for the whole column within a control that contains columns.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).Underline = bValue

Alternate: bValue = PromptID.List.Column(x).Underline

bValue (Boolean) is True or False for underline or not underline.

Examples:

lstCars.List.Column(2).Underline = True

RFPrompt("lstCars").List.Column(2).Underline = True

RFPrompt(2).List.Column(2).Underline = True

List.Column(x).Visible

This property makes visible or invisible the whole column within a control that contains columns.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).Visible = bValue

Alternate: bValue = PromptID.List.Column(x).Visible

bValue (Boolean) is True or False for column visibility.

Examples:

lstCars.List.Column(2).Visible = True

RFPrompt("lstCars").List.Column(2).Visible = True

RFPrompt(2).List.Column(2).Visible = True

List.Column(x).Width

This property sets or returns the width of the column within a control that contains columns.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Column(x).Width = vValue

Alternate: vValue = PromptID.List.Column(x).Visible

vValue (Variant) sets or gets the width of the specified column.

Examples:

lstCars.List.Column(2).Width = 25

RFPrompt("lstCars").List.Column(2).Width = 25

RFPrompt(2).List.Column(2).Width = 25

List.Columns

This property returns the number of columns in the control.

Group: Prompt-Specific Extensions

Syntax: vValue = PromptID.List.Columns

vValue (Variant) gets the number of columns in the control.

Example:

```
Dim iCnt As Integer
iCnt = lstCars.List.Columns
iCnt = RFPrompt("lstCars").List.Columns
iCnt = RFPrompt(2).List.Columns
```

List.Count

This function returns the number of items in the list collection.

Group: Prompt-Specific Extensions

Syntax: vValue = PromptID.List.Count

vValue (Variant) the number of items in the list

Example:

```
Dim nValue as Long
nValue = lstCars.List.Count
nValue = RFPrompt("lstCars").List.Count
nValue = RFPrompt(2).List.Count
```

List.Data

The list collection can be generated via the DB.MakeList, App.MakeList or ERP.MakeList functions and then assigned to the Listbox, Combobox or MenuList via this property.

Group: Prompt-Specific Extensions

Syntax: PromptID.List = vMyList

Alternate: sMyList = PromptID.List

vMyList (Variant) the list generated with the MakeList functions.

sMyList (String) the list contained in the prompt

Example:

The following uses the DB.MakeList function in the ListBox_GotFocus event to populate the List property of the list box.

```
Dim sMyList As String
Dim sSQL As String
sSQL = "select PartNo from Inventory"
```

sMyList = DB.MakeList(sSQL)

lstParts.List.Data = sMyList

or

lstParts.List.Data = DB.MakeList(sSQL)

List.EndUpdate

The command will restore a list to its normal drawing state, and should be used after calling List.BeginUpdate extension. For more information, see prompt.List.BeginUpdate.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.EndUpdate( )

Example:

```
Private Sub Form_Load()
On Error Resume Next
'This refreshes the breadcrumbs at the top of a form with a user's prompt selections.
   Form.Caption = App.GetString(Link.msTitle)
   txtFilter.Visible = False
   listItems.List.BeginUpdate
   listItems.List.LoadCells(Link.msList, Link.mbRowValue, Link.miColumns)
   listItems.List.EndUpdate
   Link.mbCancel = True
End Sub
```

Versions Supported: RFgen 5.2.4.2 and newer.

List.Index

This property returns or sets the current list index property.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Index = nValue

Alternate: vValue = PromptID.List.Index

nValue (Long) sets the item index in the list

vValue (Variant) gets the item index in the list

Example:

```
Dim nValue As Long
```

```
nValue = Listbox1.List.Index
nValue = RFPrompt("Listbox1").List.Index
RFPrompt(2).List.Index = 5
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

List.InsertItem

This method is used to manually insert items to a list control rather than simply adding new items to the list. You must specify the index in the list as the insertion point, the value to be returned should the user choose the selection and the Display Columns array of the data to be displayed, across multiple columns if desired. See the List.AddItem also.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.InsertItem(nIndex, vValue, vDispCols)

nIndex	(Long) the insertion point in the list
vValue	(Variant) the returned value of the item to add to the list
vDispCols	(Variant) the data to be displayed. This is in the format: "display1", "display2", "display3". The MenuList control will determine the widest value in each column and format the columns so they align properly regardless of font.

Example:

The following uses the AddItem and InsertItem method in the OnEnter event in one prompt to populate the list in another prompt.

```
Public Sub txtBox_OnEnter(Rsp As String, Cancel As Boolean, ErrMsg As String)
 On Error Resume Next
 lstCars.List.AddItem("2001","2001 ABC Motor Co.")
 lstCars.List.AddItem("2002","2002 Widgets R Us")
 lstCars.List.InsertItem(1, "2000","2000 Goodfellow Inc.")
End Sub
```

This places the "2000 Goodfellow Inc." entry first in the list.

List.InsertItemEx

This alternate method is used to manually insert items to a MenuList control rather than simply adding new items to the list. You must specify the index in the list as the insertion point, the value to be returned should the user choose the selection, the Image Resource for the graphic, the Options parameter if required and

the Display Columns array of the data to be displayed, across multiple columns if desired. See the List.AddItemEx also.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.InsertItemEx(nIndex, vValue, vImage, vOptions, vDispCols)

nIndex	(Long) the insertion point in the list
vValue	(Variant) the item to add to the list
vImage	(Variant) the image associated with the list entry
vOptions	(Variant) the option for the menu item
vDispCols	(Variant) the data to be displayed. This is in the format: "display1", "display2", "display3". The MenuList control will determine the widest value in each column and format the columns so they align properly regardless of font.

Example:

The following uses the AddItemEx and InsertItemEx method in the OnEnter event in one prompt to populate the list in another prompt.

```
Public Sub txtBox_OnEnter(Rsp As String, Cancel As Boolean, ErrMsg As String)
 On Error Resume Next
 lstCars.List.AddItemEx("2","imgCheap", "", "VW Bug","Chevy Colt","Dodge Dart")
 lstCars.List.AddItemEx("3","imgMiddle", "", "Prius","Camry","Tacoma")
 lstCars.List.InsertItemEx(1, "1","imgExpensive", "", "Corvette","Tesla","Lamborghini")
End Sub
```

This places the expensive list first in the list.

List.Node(nodeId)

This is used to access the row object on a Treeview control. The list index returns the nodeId (which acts like an absolute index into the tree structure). All row and cell properties resolve this but the Row property is the index into that nodes children. For example, a tree with 10 base rows and 10 children per row, will have 100 nodes. List.Row will support access to the 10 base rows, whereas List.Node will access all 100 nodes.

Group: Prompt-Specific Extensions

Syntax: List.Node(NodeId)

NodeID	(Long) The identification of the active row

Example:

```
Private Sub Button1_Click()
```

```
On Error Resume Next
  App.Balloon(TreeView1.List.Node(1).Cell(1).Value,1)
End Sub
```

List.LoadCells

This method is an alternate method for loading data in a list control and will dynamically format the lines of data (cells) in RFgen to match the source's line of data. You can also specify which part of the source's cell data will the last of the loaded cells. In a formatted RFgen list line, it looks like this:

```
Value | Cell 1 | Cell 2 | ...
```

where each row is separated by an AM and each value or cell is separated by a VM.

Note: List.LoadCells should not be used with image-based list controls or with Search lists.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.LoadCells(CellData,HasValue,LastCell)

CellData (Varient) Is the data to be loaded into the list control.

HasValue (Boolean) Is an optional parameter specifying whether the cell data includes the row value. The default is False. If set to False, then the first cell specified will be duplicated and used as the row value. If set to True, then the row value will be used.

LastCell Is an optional parameter specifying whether to restrict the list data to a number of cells in the data provided. For example if you have a list containing 4 cell values per row returned from a SQL or Web Service call but you only want to load 2 of them, set this value to 2 (assuming one based arrays).

Example:

```
Private Sub Form_Load()
On Error Resume Next
'This refreshes the breadcrumbs at the top of a form with a user's prompt selections.
  Form.Caption = App.GetString(Link.msTitle)
  txtFilter.Visible = False
  listItems.List.BeginUpdate
  listItems.List.LoadCells(Link.msList, Link.mbRowValue, Link.miColumns)
  listItems.List.EndUpdate
  Link.mbCancel = True
End Sub
```

Versions Supported: RFgen 5.2.4.2 and newer.

List.PageDown

This method scrolls the contents of a list object prompt down 1 page. A page is defined as the number of visible rows.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.PageDown

Example: *See List.PageUp*

List.PageUp

This method scrolls the contents of a list object prompt up 1 page. A page is defined as the number of visible rows.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.PageUp

Example:

This uses a Form_OnFkey event to activate the Listbox Page methods.

```
Public Sub Form_OnFkey(Fkey As Long)
  On Error Resume Next
  If Fkey = 5 Then
    lstParts.List.PageUp
  ElseIf Fkey = 6 Then
    lstParts.List.PageDown
  End If
End Sub
```

List.ScrollDown

This method scrolls the contents of a list object prompt down 1 row.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.ScrollDown

Example:

See List.ScrollUp

List.ScrollUp

This method scrolls the contents of a list object prompt up 1 row.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.ScrollUp

Example:

This uses a Form_OnFkey event to activate the List box Scroll methods.

```
Public Sub Form_OnFkey(Fkey As Long)
 On Error Resume Next
 If Fkey = 5 Then
  lstParts.List.ScrollUp
 ElseIf Fkey = 6 Then
   lstParts.List.ScrollDown
 End If
End Sub
```

List.SetColumn

This method is used to format the displayed columns. First specify which column is to be formatted, what the caption of that column is, how wide to make the column and how to justify the content. In addition to the contents being formatted, the caption of the control can also be overwritten with the column names if the ListHeading property is set to True. Then the caption will be spaced exactly the same as the columns.

Group: Prompt-Specific Extensions

Syntax: PromptID. List.SetColumn(nCol, sHeading, vDefWidth, [eValue], [bTrimSpaces], [nScaleDecimals])

nCol (Long) the column number to be affected by the formatting

sHeading (String) the title that will appear across the top of the column

vDefWidth (Variant) the width of the column in characters. If -1 is used the column will dynamically size to the widest value in the column.

eValue (enColAlign) Optional – how to align the column; either BottomCenter, BottomLeft, BottomRight, CenterCenter, CenterLeft, CenterRight, TextCenter, TextLeft, TextRight., TopCenter, TopLeft, TopRight

bTrimSpaces (Boolean) Optional – formats the values in the specified column by deleting the leading and trailing spaces in the data.

nScaleDecimals (Long) Optional – formats the numeric values in the specified column if the database does not store decimals. This option will position a decimal at a position from the right side. A comma will be used for large numbers. This field is locale specific and will use the appropriate characters for each region.

Example:

```
oMyList.SQL = "select * from PLANTS"
```

```
oMyList.SetColumn 1, "ID", 5, TextLeft, True
oMyList.SetColumn 2, "NAME", 21, TextLeft
sName = oMyList.ShowList
```

List.Sorted

This property indicates whether the items of the list box are to be sorted alphabetically. Numbers are sorted alphabetically, not numerically. Therefore 1, 10, 100, 2, 3, 4 are numbers sorted alphabetically.

Group: Prompt-Specific Extensions

Syntax: PromptID. List.Sorted = bValue

Alternate: bValue = PromptID. List.Sorted

bValue (Boolean) set to True to sort the items in ascending order

Example:

```
Dim bValue As Boolean
lstParts.List.Sorted = True
RFPrompt("lstParts").List.Sorted = True
bValue = RFPrompt(2).List.Sorted
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

List.RemoveColSet

This method is used to manually remove a ColSet from a list object prompt.

Group: Prompt-Specific Extensions

List.RemoveItem

This method is used to manually remove items from a list object prompt. If the third entry is removed all entries further down are shifted up 1 place.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.RemoveItem (vLocation)

vLocation (Variant) the location to remove the item from the list

Example:

```
lstParts.List.RemoveItem(3) ' Removes the third item
RFPrompt("lstParts").List.RemoveItem(3)
```

```
RFPrompt(2).List.RemoveItem(3)
```

List.Row.Index

This property returns the current list row index value. It is a read only property.

Group: Prompt-Specific Extensions

Syntax: PromptID.List.Row.Index

vValue (Variant) returns the row index value

Examples:

Listbox1.List.Row(i).Index

List.RowSelector

This method activates or deactivates row highlighting during row selection for list box controls

Group: Prompt-Specific Extensions

Syntax: PromptID.List.RowSelector (bValue)

bValue(Boolean) is the display state for the row selector

Example:

```
lstParts.List.RowSelector = False 'Turns off Row Highlighting
```

List.SetDefaultColSet

This method sets the default as either a one-or zero-based collection. For example, if you have a group of columns, you can set the default to identify the first column as Col 0 (zero-based collection) and the next one, Col 1 and the last one as Col 3.

Group: Prompt-Specific Extensions

List.Value(x)

This property returns the value in the specified row for the control. A row can have a value if it was placed there via an AddItem or InsertItem method or the ListData property of the control.

Group: Prompt-Specific Extensions

Syntax: vValue = PromptID. List.Value(Row)

Row (Long) is the row number in the list

vValue (Variant) is given the value assigned to the control's row

Examples:

Dim vValue As Variant

vValue = lstParts.List.Value(1)

vValue = RFPrompt("lstParts").List.Value(1)

vValue = RFPrompt(2).List.Value(1)

MQTT Object

The Message Queueing Telemetry Transport (MQTT) object is used to publish messages and/or receive messages (subscribe) via communication conducted between the client and the RFgen server, and the MQTT broker and RFgen server.

The MQTT language extensions are organized by Properties and Methods.

MQTT Object Properties

The properties for the Socket object are: ClientId, Error, ServerAddress, and Timeout.

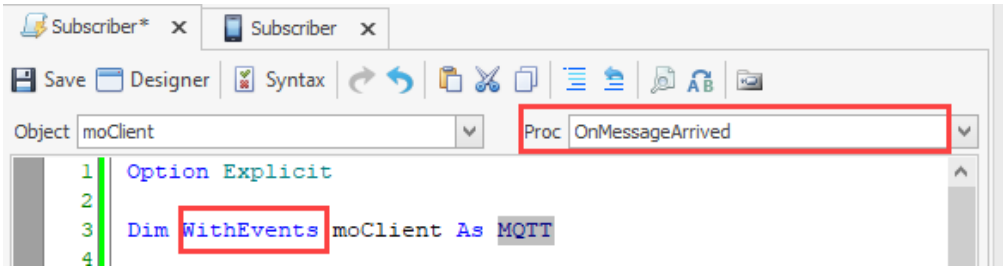
MQTT Object Methods

The methods for the MQTT socket object are: Connect, Disconnect, GetLastMessage, Publish, SetLWT, SetPassword, SetUsername, SSL_SetCalPath, SSL_SetKeyStore, SSL_SetPrivateKey , SetPrivateKeyPassword, and Subscribe.

MQTT Events

There are no events unless you declare "WithEvents" with the MQTT object. Then the OnMessageArrived event is available. For example:

```
Dim WithEvents moClient as new MQTT
```



Versions Supported: RFgen 5.1.1.35 and 5.2.3 and newer.

ClientId

The ClientId sets the string name of the MQTT object.

This is a read-only property and is unavailable at design time. The value returned is a string.

Group: MQTT Object

Syntax: sValue = oMQTT.ClientId

ClientId (String) the name of the client

Example:

See the example under MQTT.Connect

Versions Supported: RFgen 5.2.3.0 and newer.

Connect

This method initiates a connection between the MQTT broker (MQTT server) and client, and sets the timeout value for the client. If the connection is established, this returns a true value. If no connection was made, a false value is returned. A connection will disconnect automatically if the timeout period is exceeded.

Group: MQTT Object

Syntax: oMQTT.Connect(ServerAddress, ClientId, Timeout) as Boolean

ServerAddress (String) the tcp IP address or name of the MQTT broker

ClientId (String) the name of the client

Timeout (Long) the time in seconds a client waits before the connection is closed

bOK (Boolean) True means there is already a connection; False means there isn't a connection.

Example:

```
Dim WithEvents moClient As New MQTT
    Private Sub Form_Load()
      On Error Resume Next
        App.Balloon("Connecting...", -1)
        If Not moClient.Connect("tcp://localhost:1883", "RFgen_Publisher_Client_001", 4000)
            Then
              App.MsgBox("Error: " & moClient.Error)
            Exit Sub
        End If
      App.Balloon("", 0)
    End Sub
```

Versions Supported: RFgen 5.1.3 and newer.

Disconnect

This method will disconnect the client from the broker, and return a true or false value if the disconnect succeeds or fails. The purpose of this method is to provide the developer a way to handle a disconnect (i.e. the user exits the app). If it was disconnected, this returns a true value; if it remains connected, this returns a false value. Note that client connections are dropped automatically if the MQTT broker does not sense the client connection is alive.

Group: MQTT Object

Syntax: oMQTT.Disconnect as Boolean

Example:

```
Dim WithEvents moClient As New MQTT
    Private Sub Form_Load()
      On Error Resume Next
        App.Balloon("Connecting...", -1)
        If Not moClient.Connect("tcp://localhost:1883", "RFgen_Publisher_Client_001", 4000)
            Then
            moClient.Disconnect
              App.ExitForm
            Exit Sub
        End If
      App.Balloon("", 0)
    End Sub
```

Versions Supported: RFgen 5.1.1.35 and 5.2.3 and newer.

Error

This property returns the text of the last error.

This is a read-only property and is unavailable at design time. The value returned is a string.

Group: MQTT Object

Syntax: sValue = oMQTT.Error

Error (String) text of the last error

Example:

See the example under MQTT.Connect

Versions Supported: RFgen 5.1.1.35 and 5.2.3 and newer.

OnMessageArrived

The Message Queueing Telemetry Transport (MQTT) object event occurs when a message has arrived at its destination. It is used to process incoming data.

Note: To display this event in the Dev Studio Scripting View, you must include "WithEvents" in your declaration of the MQTT object variable. For example:

"Dim WithEvents moClient as new MQTT"

Group: MQTT Object

Type: Event

Syntax: Private Sub oMQTT_OnMessageArrived(ByVal Topic As String, ByVal Message As String)

Example:

```
'Private Sub moClient_OnMessageArrived(ByVal Topic As String, ByVal Message As String)
' On Error Resume Next
' App.MsgBox("Incoming Topic: " & Topic & "\n" & Message, vbOKOnly)
'End Sub
    Private Sub moClient_OnMessageArrived(ByVal Topic As String, ByVal Message As String)
      On Error Resume Next
    End Sub
```

Versions Supported: RFgen 5.2.3.0 and newer.

Publish

This MQTT Publish function publishes the message by topic and payload via the MQTT broker and returns a true or false if the action succeeded. Publishing requires the topic (subject), payload (content/message), quality of service (QOS), and retain parameters.

The QoS is used to agree on the level of guaranteed message delivery between the sender and receiver. This is important since MQTT manages the re-transmission of messages and will guarantees delivery (even when the underlying transport is not reliable), QoS makes communication in unreliable networks a lot easier. The MQTT has three QoS levels.

The Retain parameter stores the last published message in the broker and sends the last good message to new subscribers when they come online and subscribe to a new topic. By receiving the retained message as

soon as the new subscriber comes online, the subscriber will not be in the dark wondering if their subscription is working or not. This also eliminates the amount of time a subscriber waits for the next updated topic.

Group: MQTT Object

Type: Function

Syntax: oMQTT.Publish(Topic, Payload, QOS, Retain) As Boolean

Publish (Boolean) returns true if publish succeeded, false if it failed

Topic (String) The topic (subject) of the message

Payload (Long) the time in seconds a client waits before the connection is closed

QoS (Long) QOS 0: At most once (deliver and forget); QOS 1: At lease once; QOS 2: Exactly once

Retain (Boolean) True sets the message to be retained. False will not retain the message.

Example:

```
Dim WithEvents moClient As New MQTT
    Private Sub btnPublish_Click()
      On Error Resume Next
        Dim sIndex As String
        sIndex = Split(cboQOS.Text, " ")(0)
            If Not moClient.Publish(txtTopic.Text, txtPayload.Text, CLng(sIndex), chkRetain.Checked) Then
                App.MsgBox("Error: " & moClient.Error)
            Exit Sub
          Else
        App.MsgBox("Published to " & txtTopic.Text)
      End If
    End Sub
```

Versions Supported: RFgen 5.1.1.35 and 5.2.3 and newer.

ServerAddress

The ServerAddress property returns the MQTT server (broker) address in the form of string such as 11.0.0.127.

This property is a read-only and is unavailable at design time.

Group: MQTT Object

Syntax: sValue = oMQTT.ServerAddress

sValue (String)The IP address or name of the server/MQTT broker

Example:

See the example under MQTT.Connect

Versions Supported: RFgen 5.1.1.35 and 5.2.3 and newer.

SetLWT

The set Last Will and Testament (SetLWT) method is used to store a message in the MQTT broker which is published from the broker to connected clients in the event a client disconnects ungracefully. For example if a client's wi-fi connection drops unexpectedly, or the client's goes dead, the MQTT broker will detect the client is no longer connected and broadcast the dead client's Last Will and Testament to all connected clients.

Group: MQTT Object

Syntax: oMQTT.SetLWT(Topic, Payload)

Topic (String) The subject of the message/information that a client subscribes to

Payload (String) The text (message) that is associated with the topic

Example:

```
Dim WithEvents moClient As MQTT
moClient.SetLWT("LWT Topic", "This client is no longer on line.")
```

Versions Supported: RFgen 5.1.1.35 and 5.2.3 and newer.

SetPassword

The username, password, and SSL options are used for client authentication with the MQTT broker. These should be set before calling the oMQTT.Connect function.

Group: MQTT Object

Type: Command

Syntax: oMQTT.SetPassword(Password)

Password (String) The text string the user enters for a password

Example:

```
Private Sub Form_Load()
```

```
    On Error Resume Next
    App.Balloon("Connecting...", -1)
      Set moClient = New MQTT
          If Not moClient.Connect("tcp://localhost:1883", "RFgen_Subscriber_Client_001", 4000) Then
            App.MsgBox("Error: " & moClient.Error)
            Exit Sub
          End If
          ' New connect will disconnect old client
          If Not moClient.Connect("tcp://localhost:1883", "RFgen_Subscriber_Client_002", 4000) Then
            App.MsgBox("There was an error initializing client")
            Exit Sub
          End If
      moClient.SetLWT("LWT Topic", "Here's a payload")
      moClient.SetPassword("09287Tzzx")
      moClient.SetUsername("Sam")
      moClient.SSL_SetPrivateKey("/uhhh")
    App.Balloon("", 0)
  End Sub
```

Versions Supported: RFgen 5.1.1.35 and 5.2.3 and newer.

SetUserName

The username, password, and SSL options are used for client authentication with the MQTT broker. These should be set before calling the oMQTT.Connect function.

Group: MQTT Object

Type: Command

Syntax: oMQTT.SetUserName(Username)

Username (String) The text string that authenticates the username with the MQTT broker

Example:

```
  Private Sub Form_Load()
    On Error Resume Next
    App.Balloon("Connecting...", -1)
      Set moClient = New MQTT
          If Not moClient.Connect("tcp://localhost:1883", "RFgen_Subscriber_Client_001", 4000) Then
```

```
            App.MsgBox("Error: " & moClient.Error)
            Exit Sub
        End If
        ' New connect will disconnect old client
        If Not moClient.Connect("tcp://localhost:1883", "RFgen_Subscriber_Client_002", 4000) Then
            App.MsgBox("There was an error initializing client")
            Exit Sub
        End If
    moClient.SetLWT("LWT Topic", "Here's a payload")
    moClient.SetPassword("09287Tzzx")
    moClient.SetUsername("Sam")
    moClient.SSL_SetPrivateKey("/uhhh")
  App.Balloon("", 0)
End Sub
```

Versions Supported: RFgen 5.1.1.35 and 5.2.3 and newer.

SSL_SetCaPath

The username, password, and SSL options are used for client authentication with the MQTT broker. These should be set before calling the oMQTT.Connect function.

Group: MQTT Object

Type: Command

Syntax: oMQTT.SSL_SetCaPath(Path)

Path (String) The file path to the SSL certificate authority system

Example:

Versions Supported: RFgen 5.1.1.35 and 5.2.3 and newer.

SSL_SetKeyStore

The username, password, and SSL options are used for client authentication with the MQTT broker. These should be set before calling the oMQTT.Connect function.

Group: MQTT Object

Type: Command

Syntax: oMQTT.SSL_SetKeyStore(Path)

Path (String) The path to the SSL Key Storage location

Example:

To be supplied.

Versions Supported: RFgen 5.1.1.35 and 5.2.3 and newer.

SetSSL_SetPrivateKey

The username, password, and SSL options are used for client authentication with the MQTT broker. These should be set before calling the oMQTT.Connect function.

Type: Command

Syntax: oMQTT.SetPrivateKey(Path)

Path (String) The file path to a SSL file on the MQTT broker/server where the private key is set.

Example:

To be supplied.

Versions Supported: RFgen 5.1.1.35 and 5.2.3 and newer.

SSL_SetPrivateKeyPassword

The username, password, and SSL options are used for client authentication with the MQTT broker. These should be set before calling the oMQTT.Connect function.

Group: MQTT Object

Type: Command

Syntax: oMQTT.SSL_SetPrivateKeyPassword(Password)

Password (String) The password to a SSL file on the server

Example:

To be supplied.

Versions Supported: RFgen 5.1.1.35 and 5.2.3 and newer.

Subscribe

The MQTT Subscribe function enables a client to subscribe to a message publisher by topic and QoS level. The capture of events for the specific client can also be turned on through the EnableEvents parameter if desired for troubleshooting purposes.

Note: Use the oMQTT.Connection function to connect the client.

Group: MQTT Object

Type: Function

Syntax:

oMQTT.Subscribe(Topic, Payload, QOS, Retain) As Boolean

Subscribe	(Boolean) returns true if subscription succeeded, false if it failed
Topic	(String) The topic (subject) of the message
QoS	(Long) QOS 0: At most once (deliver and forget); QOS 1: At lease once ; QOS 2: Exactly once
EnableEvent	(Boolean) True sets the message to be retained. False will not retain the message.

Example:

```
Dim WithEvents moClient As New MQTT
    Private Sub btnPublish_Click()
      On Error Resume Next
        Dim sIndex As String
        sIndex = Split(cboQOS.Text, " ")(0)
            If Not moClient.Publish(txtTopic.Text, txtPayload.Text, CLng(sIndex), chkRetain.Checked) Then
                App.MsgBox("Error: " & moClient.Error)
            Exit Sub
          Else
        App.MsgBox("Published to " & txtTopic.Text)
      End If
    End Sub
```

Versions Supported: RFgen 5.1.1.35 and 5.2.3 and newer.

Timeout

This property sets how long (in seconds) the client will wait for a response from the MQTT broker before the client connection is dropped.

Group: MQTT Object

Syntax: sValue = oMQTT.Timeout

sValue	(Long)The time in seconds

Example:

See the example under MQTT.Connect

Versions Supported: RFgen 5.1.1.35 and 5.2.3 and newer.

MenuStrip (SideBar) Extensions

The menu strip (also called the **SideBar**) is a panel of commands that appears on the top, sides, or bottom of a screen. It can be launched when the user taps an icon from the form header or a button on the body of the form or turned off by the user.

The items that display in the Menu Strip, now called the SideBar, are configured from the Mobile Development Studio **Configuration > SideBar and Key Settings > SideBar and Keyboard Settings** screen.

Where the SideBar is docked and which icon/image is used to launch the SideBar is set in **Mobile Themes > SideBar**. You can also set the SideBar launch icon on an specific application's form. For details on creating the SideBar and setting the properties that make it visible, refer to the topic "To Add a SideBar" in the online help or RFgen User's Guide.

The MenuStrip extension can be used to script the behaviors of the menu items on the SideBar. For example, if a user needs to bring up a form separate from the application they are running, you can script this action with the MenuStrip.AppendItem language extension.

The commands available for the MenuStrip Extensions are:

MenuStrip.AddIcon, MenuStrip.AppendItem, MenuStrip.Clear, MenuStrip.Count, MenuStrip.DelIcon, MenuStrip.Enable, MenuStrip.Item,

MenuStrip.Refresh, MenuStrip.RemoveItem, MenuStrip.Reset, MenuStrip.SetItem, MenuStrip.Show, and MenuStrip.Visible

Supported Versions: See individual commands for supported versions.

AppendItem

The buttons on a Menu Strip/SideBar buttons are typically configured from the Dev Studio **Configuration > SideBar and Keyboard Settings > System Menu Configuration** table.

The VBA MenuStrip.AppendItem appends other menu actions/buttons to the SideBar.

Group: MenuStrip Extensions

Syntax: MenuStrip.AppendItem (Action, Display, ImageName)

Action (enSideBar) is the enumeration describing the built-in command that is executed when the user taps the SideBar menu button. The possible values and descriptions for each enumeration is listed below under "enSideBar".

vDisplay (Variant) is the label for the button. The string can also be the TextID from Text Resources.

vImageName (Variant) is the icon/image used as the button. Use the id of the image listed under Images.

[Name] Optional. If the sbAction is CallForm, use this parameter to list the name of the application that the user is to be transferred to.

[Option] Optional. If the sbAction is CallMenu, use this parameter to pass the Form_Load event from the prompt that has focus.

enSideBar

- sbBack – Executes the OnBackup event, which typically moves the prompt to the previous prompt/page.
- sbCallEvent – Executes the OnMenu( ) event from where the prompt has focus. For example, TextBox1_OnMenu(). The Form_OnMenu() always executes as well.
- sbCallForm – Executes the RFgen built-in CallForm application. It transfers the user to another application. This action requires you include the **Name** of the application the user will be transferred to.
- sbCallMenu – Executes the RFgen built-in CallMenu application. It transfers the user to another menu. This action requires the **Option** parameter.
- sbClear – Clears the data entered for the current prompt.
- sbConfig – Launches the RFgen built-in device configuration screen.
- sbExit – Exists the current process.
- sbKeyboard – Shows or hides the soft input panel (SIP) also called a soft keyboard.
- sbNone – Toggles the display of the SideBar depending on the values set for System Icon Action property in the Application Theme, and for the AutoHide property in the SideBar Theme.
- sbScan – Accesses the scan function on the device.
- sbSearch – Launches the OnSearch event from the prompt that has focus.
- sbSideBar – Shows or hides the sidebar.
- sbSignOut – Launches the RFgen Login application so the user can sign out.
- sbSubmit – Enters the data appearing for the current prompt.
- sbTerminate – Executes the terminate event which is typically used to manually close an open database connection, session connection, or release links to ActiveX objects.

Example

```
'This is an example of a Submit action that is appended to the menu strip (sidebar).

MenuStrip.AppendItem (sbSubmit, "Submit", "DownArrow")

'This adds a button to the menu strip that performs the Submit action, displays the word 'Submit' and shows a down arrow icon on the button.
```

Example

```
'This is an example of a CallForm action that is added to the menu strip (sidebar).
'The initial items are the menu strip are first cleared and then the buttons labeled "Process Reports" is displayed on the menu strip.
MenuStrip.Clear
MenuStrip.AppendItem(sbCallForm,"CallFormProcess","32.jpg","Process Reports", "-sbCall_Plant=WHS254 -sbCall_LOCN=SacramentoCA")
----------
`In ProcessReports form:
    If App.GetValue("sbCall_Plant") = "WHS254" And App.GetValue("sbCall_LOCN") = "SacramentoCA"
      Then
    App.Msgbox("CallForm correctly called the form and returned proper load options")
  End If
```

Supported Version RFgen 5.0 and newer.

Clear

This command will delete all the buttons on the menu strip.

Group: Menu Strip Extensions

Syntax: MenuStrip.Clear

This command will delete all the buttons on the menu strip.

Example:

```
MenuStrip.Clear
```

Count

This command will return the number of buttons that are currently on the menu strip.

Group: Menu Strip Extensions

Syntax: MenuStrip.Count

Example:

```
Dim iCnt As Integer
iCnt = MenuStrip.Count
```

Supported Versions: RFgen 5.2 and newer.

Enable

This command will enable or disable a button on the menu strip by referencing its Display value. The code should check if the value is an INT and if so, treat it as an index. Otherwise, it should look up the parameter by name.

Group: Menu Strip Extensions

Syntax: MenuStrip.Enable(vDisplay, [bEnable])

vDisplay (Variant) is the button's string name (sDisplay) or index number.

bEnable (Boolean) Set to True or False to enable or disable the button.

Example:

```
Menustrip.Enable("Search",false);
Menustrip.Enable(1,true);
```

Supported Versions: RFgen 5.1 and newer.

Item

This command will return the Action value of the button referenced by its index value.

Group: Menu Strip Extensions

Syntax: MenuStrip.Item(nIndex)

nIndex (Long) is the index value of the requested button

Example:

```
Dim sName As String
sName = MenuStrip.Item(4) ' the Exit button
```

If the forth button is the Exit button, "Exit" is returned.

Supported Version RFgen 5.0 and newer.

Refresh

This command will redraw the buttons on the Menustrip in case the buttons have been changed in code.

Group: Menu Strip Extensions

Syntax: MenuStrip.Refresh

RemoveItem

This command will remove a button from the menu strip by referencing its index. Removing and item by its name would be more self-documenting and is preferred but this command is useful when iterating through a loop.

Group: Menu Strip Extensions

Syntax: MenuStrip.RemoveItem(nIndex)

nIndex (Long) is the index value of the requested button

Example:

```
MenuStrip.RemoveItem(4) ' the Exit button
```

Supported Versions: RFgen 5.0 and newer.

RemoveItemByName

This command will remove a button from the menu strip by referencing its Action value.

Group: Menu Strip Extensions

Syntax: MenuStrip.RemoveItemByName(sAction)

sAction (String) is the Action name of the requested button

Example:

```
MenuStrip.RemoveItemByName("Exit")
```

Supported Versions: RFgen 5.0 and newer.

Reset

This command will return the menu strip to the default settings configured in the Configuration / Environment Properties screen.

Group: Menu Strip Extensions

Syntax: MenuStrip.Reset

Example:

MenuStrip.Reset

SetItem

This command will change an existing button to have different properties. For example, turn the Submit button into an Exit button.

Group: Menu Strip Extensions

Syntax: MenuStrip.SetItem(nIndex, sAction, sDisplay, sImageName)

nIndex (Long) is the index value of the requested button

sAction (String) is the ID of the button and also can execute one of the built-in commands. These are the internal command that can be used:

- Submit – means to enter the data appearing for the current prompt
- Refresh – refreshes the display screen
- Clear – clears the data entered for the current prompt
- Exit – exits the current process (same as F4 normally does if configured to do so).
- Search – executes the OnSearch event if one exists for the prompt

sDisplay (String) is the text to be displayed on the button.

sImageName (String) is the Image Resource ID to be displayed on the button.

Example:

MenuStrip.SetItem(4, "Submit", "Submit", "DownArrow")

Supported Versions: RFgen 5.0 and newer.

Show

This command will show or hide the menu strip.

Group: Menu Strip Extensions

Syntax:

MenuStrip.Show(bShow)

bShow (Boolean) Set to True of False to show or hide the menu strip

Example:

```
MenuStrip.Show(False) ' hides the menu strip
MenuStrip.Show(True)  ' shows the menu strip
```

Supported Versions: RFgen 5.0 and newer.

Param Object

The Param object represents a parameter or argument associated with a StoredProc object based on a parameterized stored Procedure. The Param object represents in/out arguments and the return values of stored

procedures.

Group: Stored Procedure Object

Syntax: spValue = StoredProc.Param(nIndex).oProperty

spValue (StoredParam) data value of the parameter

nIndex (Long) index of the parameter

oProperty (object) the available objects are listed below

Example:

The stored procedure object types available are:

Count, Datatype, Direction, NumericScale, Object, Precision, Prepared, Results, Size, and Value.

Param( ).Datatype

This property indicates the data type of the Param Object.

Group: Stored Procedure Object

Syntax: StoredProc.Param(nIndex).Datatype = enValue

Alternate: nValue = StoredProc.Param(nIndex).Datatype

nIndex (Long) index of the parameter

nValue (Long) the numeric value of an enDataTypes value

enValue (enDataTypes) sets one of the following values:

Constant	Description
dbBigInt	An 8-byte signed integer.
dbBinary	A binary value.
dbBoolean	A Boolean value.
dbBSTR	A null-terminated char str (Unicode).
dbChar	A String value.
dbCurrency	A currency value. Currency is a fixed-point number with 4 digits to the right of the decimal point. It is stored in an 8-byte signed integer scaled by 10,000.
dbDate	A Date value. A date is stored as a Double, the whole part of which is the number of days since December 30, 1899, and the fractional part of which is the fraction of a day.
dbDBDate	A date value (yyyymmdd).

dbDBTime A time value (hhmmss).

dbDBTimeStamp A date-time stamp (yyyymmddhhmmss plus a fraction in billionths).

dbDecimal An exact numeric value with a fixed precision and scale.

dbDouble A double-precision floating point value.

dbEmptyNo value was specified.

dbError A 32-bit error code.

dbGUID A globally unique identifier (GUID).

dbIDispatch A pointer to an IDispatch interface on an OLE object.

dbInteger A 4-byte signed integer.

dbIUnknown A pointer to an unknown interface on an OLE object.

dbLongVarBinary A long binary value.

dbLongVarChar A long String value.

dbLongVarWChar A long null-terminated string value.

dbNumeric An exact numeric value with a fixed precision and scale.

dbSingle A single-precision floating point value.

dbSmallInt A 2-byte signed integer.

dbTinyInt A 1-byte signed integer.

dbUnsignedBigInt An 8-byte unsigned integer (DBType_UI8).

dbUnsignedInt A 4-byte unsigned integer.

dbUnsignedSmallInt A 2-byte unsigned integer.

dbUnsignedTinyInt A 1-byte unsigned integer.

dbUserDefine A user-defined variable.

dbVarBinary A binary value.

dbVarChar A string value.

dbVariant An Automation Variant.

dbVarWChar A null-terminated Unicode chr string.

dbWChar A null-terminated Unicode chr string.

Param( ).Direction

This property indicates whether the Parameter represents an input parameter, an output parameter, or both, or if the parameter is the return value from a stored procedure.

Group: Stored Procedure Object

Syntax: StoredProc.Param(nIndex).Direction = enValue

Alternate: nValue = StoredProc.Param(nIndex).Direction

nIndex (Long) index of the parameter

nValue (Long) the numeric value of an enDirections value

enValue (enDirections) sets or returns one of the following values:

Constant	Description
dbParamInput	Default, indicates an input parameter
dbParamOutput	Indicates an output parameter
dbParamInputOutput	Indicates both an input and output parameter
dbParamReturnValue	Indicates a return value.

Param( ).NumericScale

This property indicates the scale of the numeric value by specifying the number of decimal places to which the number will be resolved. Not all database types support this parameter. If it is not supported, the ODBC driver will ignore this setting. The default is 0.

Group: Stored Procedure Object

Syntax: StoredProc.Param(nIndex).NumericScale = nValue

Alternate: nValue = StoredProc.Param(nIndex).NumericScale

nIndex (Long) index of the parameter

nValue (Long) the number of places to resolve the parameter's value

Param( ).Precision

This property indicates the degree of precision for numeric values by specifying the maximum number of digits used for a parameter. Not all database types support this parameter. If it is not supported, the ODBC driver will ignore this setting. The default is 0.

Group: Stored Procedure Object

Syntax: StoredProc.Param(nIndex).Precision = nValue

Alternate: nValue = StoredProc.Param(nIndex).Precision

nIndex (Long) index of the parameter

nValue (Long) the maximum number of integers that will make up the size of the value

Param( ).Size

This property indicates the maximum size, in bytes or characters of the Param Object. Use the size property to determine the maximum size for values written to or read from the value property of the Param Object.

Group: Stored Procedure Object

Syntax: StoredProc.Param(nIndex).Size = nValue

Alternate: nValue = StoredProc.Param(nIndex).Size

nIndex (Long) index of the parameter

nValue (Long) size in bytes or characters

Param( ).Value

This property indicates the value assigned to the Param object. Use the Value property to set or return parameter values with Param Objects.

Group: Stored Procedure Object

Syntax: StoredProc.Param(nIndex).Value = vValue

Alternate: vValue = StoredProc.Param(nIndex)

vValue (Variant) value of the dbParam

nIndex (Long) index of the parameter

ParamCount

This function indicates the number of parameters associated with the current stored procedure object. A parameter's index may be zero-based so the count may appear one less than expected.

Group: Stored Procedure Object

Syntax: vValue = StoredProc.ParamCount

vValue (Variant) the number of parameters associated with this stored procedure.

Prepared

This property indicates whether or not to save a compiled version of a command before execution. The prepared property is used to have the provider save a prepared (or compiled) version of the query specified in the CommandText property before the object's first execution. If the property is False, the provider will execute the object directly without creating a compiled version.

Group: Stored Procedure Object

Syntax: StoredProc.Prepared = bValue

Alternate: Value = StoredProc.Prepared

bValue (Boolean) set to True to save a compiled version before execution

Results

This property represents the entire set of records from a base table or the results of an executed command.

Group: Stored Procedure Object

Syntax: oValue = StoredProc.Results

oValue (rdoResultset, adoRecordset) recordset object containing any rows returned by the stored procedure.

Printer Extensions

Printer commands specifically interact with creating and delivering data to tethered or IP addressable printers.

The extensions available are:

Activate, Copies, EndDoc, FontBold, FontItalic, FontName, FontSize, FontStrickThru, FontUnderline, GetName, NewPage, Orientation, PageWidth, Print, PrintQuality, PrintRaw and SubmitPrintJob.

Supported Versions: RFgen 4.0 and newer.

Activate

This command will redirect output to the specified Windows printer. Specifying the printer by name should be exactly the same as it appears in the Windows Control Panel under the Printers section. Make sure this is the first command used and reused after any EndDoc commands.

Group: Printer Extensions

Syntax: Printer.Activate(vName)

vName (Variant) is the printer name in the 'Printers Window'. Click Start/ Settings/Printers. Note: You may use the printer number if desired.

Example:

```
Printer.Activate("HPDeskJet 697C")
```

Supported Versions: RFgen 4.0 and newer.

Copies

This property accesses the copy count parameter associated with a specific Windows printer. If it is not specified, the default is 1 copy.

Group: Printer Extensions

Syntax: Printer.Copies = nValue

nValue (Long) is the number of copies to print.

Example:

```
Printer.Copies = 10
```

Note: there is a special condition with the Copies, Orientation and PrintQuality commands. Since the server manages the beginning and ending of a print job, the VBA scripts must not use one of these commands after the Printer.Print command. These 3 commands will start a new print job and all settings defined up to this point will be lost.

This is valid:

```
Printer.Copies = 10
Printer.Orientation = PrintHorizontal
Printer.Print "Sample Text"
Printer.EndDoc
```

This is not because the Orientation line will lose the 2 lines above it.

```
Printer.Copies = 10
Printer.Print "Sample Text"
Printer.Orientation = PrintHorizontal
Printer.EndDoc
```

Supported Versions: RFgen 4.0 and newer.

EndDoc

This command will close the print job on the selected Windows Printer and start the printing process.

Group: Printer Extensions

Syntax: Printer.EndDoc

Example:

```
Printer.EndDoc
```

Note: All printer settings such as font, copies, orientation and others are reset to defaults after this command. They must be re-issued to take effect on the next print job.

Supported Versions: RFgen 5.2 and newer.

FontBold

This property accesses the font bold parameter associated with a specific Windows printer.

Group: Printer Extensions

Syntax: Printer.FontBold = bValue

bValue (Boolean) determines whether text is printed using the bold font option.

Example:

```
Printer.FontBold = True
```

Supported Versions: RFgen 4.0and newer.

FontItalic

This property accesses the font italic parameter associated with a specific Windows printer.

Group: Printer Extensions

Syntax: Printer.FontItalic = bValue

bValue (Boolean) determines whether text is printed using the italic font option.

Example:

```
Printer.FontItalic = True
```

Supported Versions: RFgen 4.0 and newer.

FontName

This property accesses the font name parameter associated with a specific Windows printer.

Group: Prompt-Specific Extensions

Syntax:

Printer.FontName = sValue

sValue (String) specifies the type of font to use.

Example:

```
Printer.FontName = "Arial"
```

FontSize

This property accesses the font size parameter associated with a specific Windows printer.

Group: Printer Extensions

Syntax: Printer.FontSize = nValue

nValue (Long) is the font size to use.

Example:

```
Printer.FontSize = 12
```

Supported Versions: RFgen 4.0 and newer.

FontStrikeThru

This property accesses the font strike thru parameter associated with a specific Windows printer.

Group: Printer Extensions

Syntax: Printer.FontStrikethru = bValue

bValue (Boolean) determines whether text is printed using the strike thru font option.

Example:

```
Printer.FontStrikeThru = True
```

Supported Versions: RFgen 4.0 and newer.

FontUnderline

This property accesses the font underline parameter associated with a specific Windows printer.

Group: Printer Extensions

Syntax: Printer.FontUnderline = bValue

bValue (Boolean) determines whether text is printed using the underline font option.

Example:

```
Printer.FontUnderline = True
```

Supported Versions: RFgen 4.0 and newer.

GetName

This function returns the name for the specified Windows printer number. You may use this command within a loop to get a list of names for the user to pick from. See Printer.Activate for setting a printer.

Group: Printer Extensions

Syntax: sName = Printer.GetName(nIndex)

sName (String) is the Windows name of the requested printer.

nIndex (Long) is the Windows printer number.

Example:

```
Dim sName As String
sName = Printer.GetName(1)
```

Supported Versions: RFgen 4.0 and newer.

NewPage

This command will send a page eject character to the selected Windows Printer.

Group: Printer Extensions

Syntax: Printer.NewPage

Example:

```
Printer.NewPage
```

Supported Versions: RFgen 5.2 and newer.

Orientation

This function accesses the orientation parameter associated with a specific Windows printer.

Group: Printer Extensions

Syntax: Printer.Orientation = enValue

enValue (enPrintOrientation) is the text orientation to use.
PrintHorizontal (Portrait)
PrintVertical (Landscape)

Example:

```
Printer.Orientation = PrintHorizontal
```

Note: there is a special condition with the Copies, Orientation and PrintQuality commands. Since the server manages the beginning and ending of a print job, the VBA scripts must not use one of these commands after the Printer.Print command. These 3 commands will start a new print job and all settings defined up to this point will be lost.

This is valid:

```
Printer.Copies = 10
```

```
Printer.Orientation = PrintHorizontal
Printer.Print "Sample Text"
Printer.EndDoc

This is not because the Orientation line will lose the 2 lines above it.

Printer.Copies = 10
Printer.Print "Sample Text"
Printer.Orientation = PrintHorizontal
Printer.EndDoc
```

Supported Versions: RFgen 4.0 and newer.

PageWidth

This function sets the printer print width to a specific value. The value is reset to 0 after an EndDoc command is issued.

Group: Printer Extensions

Syntax: Printer.PageWidth = nValue

nValue (Long) is the printer width.

Example:

```
Printer.PageWidth = 80
Printer.PageWidth = 132
```

Supported Versions: RFgen 4.0 and newer.

Print

This command will print the text on the selected Windows printer. Each Print statement will be on a new line.

Group: Printer Extensions

Syntax: Printer.Print(sText)

sText (String) is the text stream that is to be sent to the network printer.

Example:

```
Printer.Print("20lb Super Green Lawn Food")
```

Supported Versions: RFgen 4.0 and newer.

PrintQuality

This property accesses the print quality parameter associated with a specific Windows printer.

Group: Printer Extensions

Syntax: Printer.PrintQuality = enValue

Alternate: enValue = Printer.PrintQuality

enValue (enPrintQuality) is the print quality desired.
DraftQuality
HighQuality
LowQuality
MediumQuality

Example:

Printer.PrintQuality = MediumQuality

Note: there is a special condition with the Copies, Orientation and PrintQuality commands. Since the server manages the beginning and ending of a print job, the VBA scripts must not use one of these commands after the Printer.Print command. These 3 commands will start a new print job and all settings defined up to this point will be lost.

This is valid:

```
Printer.Copies = 10
Printer.Orientation = PrintHorizontal
Printer.Print "Sample Text"
Printer.EndDoc
```

This is not because the Orientation line will lose the 2 lines above it.

```
Printer.Copies = 10
Printer.Print "Sample Text"
Printer.Orientation = PrintHorizontal
Printer.EndDoc
```

Supported Versions: RFgen 4.0 and newer.

PrintRaw

This command will send a byte stream to the selected printer. A typical example would be sending escape sequences.

Group: Printer Extensions

Syntax: Printer.PrintRaw(vData)

vData (Variant) is the byte stream or string that is to be sent to the printer. If the passed value is an actual byte array it will be sent to the printer unmodified. If it is not a byte array the server will convert it to an SBCS string and send that.

Example:

```
Dim sData As String
sData = "Print Me"
Printer.PrintRaw(sData)
Converts "Print Me" to Unicode and sends it
```

Supported Versions: RFgen 4.0 and newer.

SubmitPrintJob

This function is used to submit a print job through a network printer the RFgen server is connected.

Group: Printer Extensions

Syntax: bValue = Printer.SubmitPrintJob(Template, Printer, Tcp Port, pParams)

bValue (Boolean) Optional – Returns True or False depending on the success of the sub-mission.

Template (String) The name of the template that contains the formatting of the print job. (Reserved for future use.)

Printer (String) is the network name/IP of the device/peripheral that will be printing the job.

TcpPort (Long) is the TCP port number of the printer.

Params (Variant) The first parameter is the printer-specific code to print a barcode label. The subsequent parameters are reserved for future use.

Example:

```
Dim sOutput As String
sOutput = "<STX><ESC>P<ETX>"
sOutput &= "<STX>E2;F2<ETX>"
sOutput &= "<STX>H0;o200,10;c2;f3;k5;d0,20<ETX>"  Item
sOutput &= "<STX>H1;o200,400;c2;f3;k5;d0,20<ETX>"  'Rev
sOutput &= "<STX>B2;o170,10;c0;h50;w2;f3;k6;d0,28;<ETX>" 'Item Barcode
sOutput &= "<STX>I2;o110,200;c2;f3;k5;d0,28<ETX>"  'Item Interpretive
sOutput &= "<STX>R<ETX>"
sOutput &= "<STX><ESC>E2<CAN><ETX>"
```

```
sOutput &= "<STX>" & "Item:"  & "<CR><ETX>"
sOutput &= "<STX>" & "Rev: A" & "<CR><ETX>"
sOutput &= "<STX>" & "73-0000032" & "<CR><ETX>"
sOutput = sOutput & "<STX><RS>" & "2" & "<ETB><ETX>"

Printer.SubmitPrintJob("", "158.100.100.100", 9100, sOutput)
```

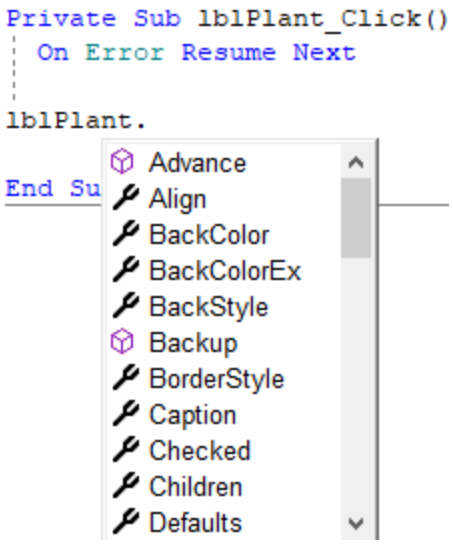
Supported Versions: RFgen 5.1 and newer.

Prompt-Specific Extensions

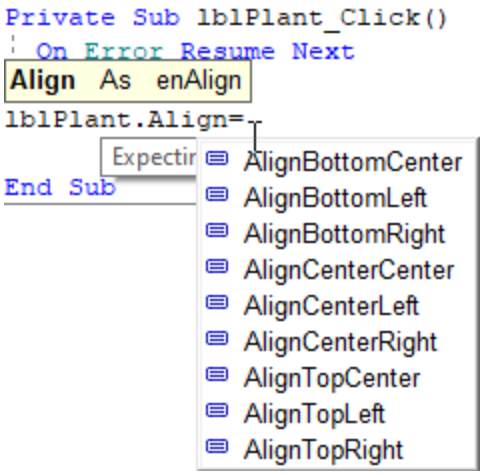
These properties and methods are available at run time by using the name of the prompt or the RFPrompt wrapper function with specifying either the prompt name or prompt number. Be careful using the prompt number since adding or deleting prompts over time will create bugs in the script. Prompt numbers are best used when looping through the prompts to set a property like Visible = False. At design time, these properties can be found on the Fields Properties tab.

The list of extensions available to each prompt is vast to list here. In order to view which extensions are available, in the script view, enter the prompt id followed by a dot.

For example, if you added a Label control to your form, and the lable is *lblPlant*, in your script, enter "lblPlant" and select a procedures such as the Click event. The following Intellisense display will show something similar to this below:



Depending on the item selected, you can then continue to add the next extension. For example, if you selected "Align" you can use the Intellisense to select the value that specifies the alignment.



Note that some extensions are only available for specific types of prompts.

Align

This property aligns the text of a label or the text in the field of a prompt either left, center, right or vertically. The vertical option only applies to labels.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Align = enValue

enValue (enAlign) an enumeration that contains AlignCenter, AlignLeft, AlignRight and AlignVertical.

Example:

```
txtBox.Align = AlignCenter
RFPrompt("txtBox").Align = AlignCenter
RFPrompt(2).Align = AlignCenter
```

AutoSize

This property is only used on the Label, Image, and MenuList controls. The options automatically size the control based on the content.
Fill - Fills the screen from its starting location on down.
Horizontal - Spans the content based on the width of the screen.
None - Disallows auto sizing.
Vertical - Stretches the content to match the vertical height of the control.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Autosize = enValue

enValue (enSizeModes) an enumeration that contains AutosizeContent, AutosizeFill, AutosizeHorizontal, AutosizeNone and AutosizeVertical

Example:

```
Image1.Autosize = AutosizeContent
RFPrompt("Image1").Autosize = AutosizeContent
RFPrompt(2).Autosize = AutosizeContent
```

BackColor

This property accesses the prompt's data field background color and is the primary background color. BackColorEx is only used to create gradients.

Group: Prompt-Specific Extensions

Syntax:

PromptID.BackColor = vValue

Alternate: lValue = PromptID.BackColor1

nValue (Long) is the color.

vValue (Variant) sets the color.

Example:

```
txtPart.BackColor = RGB(255,255,0) 'yellow
txtPart.BackColor = &HFF0000      'blue
txtPart.BackColor = QBColor(5)    'magenta
RFPrompt("txtPart").BackColor = vbWhite
RFPrompt(1).BackColor = vbWhite
```

Version Supported: RFgen 5.2 and newer. Replaced BackColor1.

BackColorEx

This property accesses the prompt's data field and is used to produce gradients from the first color to this color.

Group: Prompt-Specific Extensions

Syntax:

PromptID.BackColorEx = vValue

Alternate: lValue = PromptID.BackColorEx

nValue (Long) is the color.

vValue (Variant) sets the color.

Example:

```
txtPart.BackColor = vbWhite
txtPart.BackColorEx = vbRed
txtPart.BackGradient = GradientDiagonalRight
RFPrompt("txtPart").BackColor = vbWhite
RFPrompt("txtPart").BackColorEx = vbRed
RFPrompt("txtPart").BackGradient = GradientDiagonalRight
RFPrompt(2).BackColor = vbWhite
RFPrompt(2).BackColorEx = vbRed
RFPrompt(2).BackGradient = GradientDiagonalRight
```

Version Supported: RFgen 5.2 and newer. Replaced BackColor2.

BackGradient

This property uses either just the BackColor(1) for solid backgrounds or both BackColor properties to create gradients in one of several directions in graphical mode.

Group: Prompt-Specific Extensions

Syntax:

PromptID.BackGradient = enValue

enValue (enGradients) enumeration that contains GradientDefault, GradientDiagonalLeft, GradientDiagonalRight, GradientHorizontal, GradientNone, GradientVertical and GradientVerticalSplit

Group: Prompt-Specific Extensions

Example:

```
txtPart.BackColor1 = RGB(0,0,255)
txtPart.BackColor2 = vbWhite
txtPart.BackGradient = GradientVertical
RFPrompt("txtPart").BackGradient = GradientVertical
RFPrompt(2).BackGradient = GradientVertical
```

Version Supported: RFgen 5.0. Replaced by BackStyle in RFgen 5.1.

BorderStyle

This property affects the border associated with a specific prompt. The properties for the various borders are set from the Application Designer > Control Properties > [name of the control] > BorderStyle, which is typically set to "(Default)".

Group: Prompt-Specific Extensions

Syntax: PromptID.BorderStyle = enBoarderStyle

Alternate: enValue = PromptID.BorderStyle

enValue (enBorderStyles) is the border style used to render the control object. The options are BoarderDefault, BoarderFlat, BorderNone, BorderRaised, BorderSunken, BorderThick, and BorderUnderline.

Example:

```
txtPart.BorderStyle = BorderNone
RFPrompt("txtPart").BorderStyle = BorderNone
RFPrompt(1).BorderStyle = BorderNone
```

Version Supported: RFgen 5.0 and newer.

Caption

This property accesses the caption associated with a specific prompt.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Caption = vValue

Alternate: sValue = PromptID.Caption

sValue (String) is the caption of the prompt.

vValue (Variant) is the caption to display for the prompt.

Example:

```
txtPart.Caption = "Part Number"
RFPrompt("txtPart").Caption = "Part Number"
RFPrompt(1).Caption = "Part Number"
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

Checked

This property accesses the status of a checkbox object associated with a specific prompt.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Checked = bValue

Alternate: bValue = PromptID.Checked

bValue (Boolean) is the checked state of the prompt.

Example:

```
chkPrint.Checked = True
bValue = chkPrint.Checked
RFPrompt("chkPrint").Checked = False
RFPrompt(2).Checked = False
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

Children

Group: Prompt-Specific Extensions

This property is used for objects with parent-child relationships. If the child and parent have a property in common but with different values (i.e. background color, forecolor, fontsize etc.) the child's property value trumps the parent property value. There are four methods that are used with the Children property: Children.Append, Children.Count, Children.Item, and Children.Remove.

Children.Append

Use this method to clone objects contained by a parent control (Layout, Panel or TabControl). Children.Append takes a single parameter which is the control that you wish to clone and appends it to a VBA Prompt Array.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Children.Append(ByVal pCtl As vbaPrompt, Optional Top As Long, Optional Left As Long)

ByVal pCtl is the prompt you want copied.

ByVal (Long) Top and Left is the location.

Example:

In the Cloning App below, the panel control (pnlChild) is cloned and will be appended to the end of the child controls of pChildren. The new controls will have the same parent that the pChildren control came from.

```
Dim pChildren As vbaPromptArray
Set pChildren = pnlParent.Children
pChildren.Append(pnlChild)
```

Version Supported: RFgen 5.1 and newer.

Children.Count

This method will return the number of child prompts that exist in the child collection. Its generally used when looping through the prompts to set properties.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Children.Count

nCount = Children.Count

Example:

```
Dim nCount As Integer
Dim pParent As vbaPrompt
nCount = pParent.Children.Count
```

Version Supported: RFgen 5.1 and newer.

Children.Item

This method is used to access the array elements of the vbaPromptArray collection.

Group: Prompt-Specific Extensions

Syntax: PromptID.Children.Item(i)

ByVal (Variant) sets the item identifier for the child prompt.

i The index ID of a cloned control in a vbaprompt array.

Note: The variable "i" can also be used to reference the name of a prompt when given in string format.

Example:

In this example, pnlParent is the Panel Control.

```
Dim pChildren as vbaPromptArray
```

```
Set pChildren = pnlParent.Children
pChildren.Item(3).Caption = "Enter"
pChildren.Item("TextBox1").Caption = "Enter"
```

Version Supported: RFgen 5.1 and newer.

Children.Remove

This method is used to remove a child object from the vbaPromptArray collection.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Children.Remove(i)

ByVal (Variant) Index is the identifier

ByVal (Boolean) True if its successfully removed; False if it fails.

Example:

```
Dim pChildren As vbaPromptArray
Set pChildren = pnlParent.Children
pChildren.Remove(3)
```

Version Supported: RFgen 5.1 and newer.

Cloning

Group: Prompt-Specific Extensions

See "Children.Append"

To clone child objects, see Children.Append method.

For information on how to access a cloned object, refer to "**How to access cloned objects**."

Defaults

This property accesses the default property associated with a specific prompt.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Defaults = vValue

Alternate: sValue = PromptID.Defaults

sValue (String) is the default expression for the prompt.

vValue (Variant) sets the default expression for the prompt.

Example:

```
txtYesNo.Defaults = "N"
RFPrompt("txtYesNo").Defaults = "N;O" ' with override
RFPrompt(2).Defaults = "N;O" ' with override
The letter 'N' will be the default.
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

DisplayOnly

This property accesses the display only property associated with a specific prompt. Setting it to True makes a prompt into a display only field, while setting it to False allows users to access and change data at the prompt.

Group: Prompt-Specific Extensions

Syntax:

PromptID.DisplayOnly = bValue

Alternate: bValue = PromptID.DisplayOnly

bValue (Boolean) is the display only state for the prompt.

Example:

```
txtPart.DisplayOnly = True
RFPrompt("txtPart").DisplayOnly = False
RFPrompt(2).DisplayOnly = False
```

Version Supported: RFgen 4.0, 4.1, 5.0, and 5.1. Removed in 5.2.

Edits

This property accesses the Edits property associated with a specific prompt.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Edits = vValue

Alternate: sValue = PromptID.Edits

sValue (String) is the edit conditions for the prompt.

vValue (Variant) sets the edit conditions for the prompt.

Example:

```
txtPart.Edits = "2N" ' only accept 2 numbers
RFPrompt("txtPart").Edits = "2N"
RFPrompt(2).Edits = "2N"
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

ErrMsg

This property accesses the error message property associated with a specific prompt.

Group: Prompt-Specific Extensions

Syntax:

PromptID.ErrMsg = vValue

Alternate: sValue = PromptID.ErrMsg

sValue (String) is the error message to display for the prompt.

vValue (Variant) sets the error message to display for the prompt.

Example:

```
txtPart.Edits = "2N" ' only accept 2 numbers
txtPart.ErrMsg = "Must be 2 numbers."
RFPrompt("txtPart").ErrMsg = "Must be 2 numbers."
RFPrompt(2).ErrMsg = "Must be 2 numbers."
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

FieldId

This property returns the field ID of a specific prompt. If the prompt is linked to a database field, that field name is returned.

Group: Prompt-Specific Extensions

Syntax:

sValue = PromptID.FieldId

sValue (String) is the field id/name for the prompt.

Example:

```
Dim sValue As String
sValue = txtPart.FieldId
sValue = RFPrompt("txtPart").FieldId
sValue = RFPrompt(2).FieldId
```

Font.Bold

This property accesses the prompt's data field bold option.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Font.Bold = bValue

Alternate: bValue = PromptID.Font.Bold

bValue (Boolean) is True or False for bold or not bold.

Example:

```
txtPart.Font.Bold = True
RFPrompt("txtPart").Font.Bold = True
RFPrompt(2).Font.Bold = True
```

Font.Italic

This property accesses the prompt's data field italic option.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Font.Italic = bValue

Alternate: bValue = PromptID.Font.Italic

bValue (Boolean) is True or False for italics or no italics.

Example:

```
txtPart.Font.Italic = True
RFPrompt("txtPart").Font.Italic = True
RFPrompt(2).Font.Italic = True
```

Font.Size

This property accesses the prompt's data field text size.

Group: Prompt-Specific Extensions

Syntax:

PromptID. Font.Size = nValue

Alternate: vValue = PromptID.Font.Size

vValue (Variant) is the font size.

nValue (Long) sets the font size. Default is set in the theme.

Example:

```
txtPart.Font.Size = 16
RFPrompt("txtPart").Font.Size = 16
RFPrompt(2).Font.Size = 16
```

Font.Underline

This property accesses the prompt's data field underline option.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Font.Underline = bValue

Alternate: bValue = PromptID.Font.Underline

bValue (Boolean) True or False for underline or no underline.

Example:

```
txtPart.Font.Underline = True
RFPrompt("txtPart").Font.Underline = True
RFPrompt(2).Font.Underline = True
```

ForeColor

This property sets the color for the font (text). The

Group: Prompt-Specific Extensions

Syntax:

PromptID.ForeColor = nValue

Alternate: vValue = PromptID.ForeColor

vValue (Variant) is the color.

nValue (Long) sets the color.

Example:

```
txtPart.ForeColor = RGB(255,255,0) 'yellow
txtPart.ForeColor = &HFF0000      'blue
txtPart.ForeColor = QBColor(5)    'magenta
RFPrompt("txtPart").ForeColor = vbWhite
RFPrompt(2).ForeColor = vbWhite
```

Version Supported: RFgen 5.0 and newer.

Form.PageNo

This command returns the page number of the active page (the page that is displaying at runtime.)

Group: Prompt-Specific Extensions

Syntax:

vPage = Form.PageNo

vPage (Variant) is the page number.

Example:

```
If MenuAction = "SysBack" Then
   MenuAction = ""
  'Get the page number of the active page and
  'make it the Case number
   Select Case Form.PageNo
    Case 1: App.ExitForm
    Case 2: Call BackFromPage2
    Case 3: Call BackFromPage3
   End Select
End If
Public Sub BackFromPage2
 '1stLine is on page 1
   lstLine.SetFocus
End Sub
```

```
Public Sub BackFromPage3
'Txt2Mat is on page 2
 Txt2Mat.SetFocus
End Sub
```

Format

This property accesses the format of a specific prompt. It is only an extension of the Format VBA command.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Format = vValue

Alternate: sValue = PromptID.Format

sValue (String) is the format mask to use when displaying data for the prompt.

vValue (Variant) sets the format mask to use when displaying data for the prompt.

Example:

```
txtTime.Format = "hh:mm"
RFPrompt("txtTime").Format = "hh:mm"
c - General Date
dddddd - Long Date
ddddd - Short Date
ttttt - Long Time
hh:mm AMPM   - Medium Time , 12-hr format
hh:mm - Short, 12-hr format
HH:mm - Short, 24-hr format
$#,##0.00 or ($#,##0.00) - Currency 0.00 - Fixed
#,##0.00 - 'Standard 0.00% - Percent 0.00E+00 - Scientific
Yes/No - Return "No" if zero, else return "Yes"
True/False - Return "True" if zero, else return "False"
On/Off - Return "On" if zero, else return "Off"
```

For further examples get help on the VB FORMAT command.

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

Height

This property accesses the Height property for a prompt object. This value can be changed at run time if there is a need to change prompt sizes on the screen. Depending on the client environment the default values may be in either pixels or characters.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Height = nValue

Alternate: vValue = PromptID.Height

vValue (Variant) is the display length for a prompt object.

nValue (Long) sets the display length for a prompt object.

Example:

```
Dim vValue As Variant
vValue = txtPart.Height
txtPart.Height = 10
vValue = RFPrompt("txtPart").Height
RFPrompt("txtPart").Height = 10
vValue = RFPrompt(2).Height
RFPrompt(2).Height = 10
```

Version Supported: RFgen 5.0 and newer.

Form.IconSet

This command is used to quickly change the icons displayed on a form. For example, if you have users for whom you want to change which icons are available at a specific moment in time, this extension allows you to change them as needed.

Group: Prompt-Specific Extensions

Syntax: Form.IconSet (Index, IconType, IconName, IconAction, IconVisible)

Index (Long) the index of the existing icon

The icon's index is typically in Themes > Application > SystemIcons > Manage Icons Collection > Icon ID

IconAction The new icon action that will replace the action assigned to the existing icon -- select from Intellisense.

IconName The name of the icon to be changed on a form. Select the icon switch to from Intellisense.

IconType The icon that will replace the existing icon -- select from the drop down list in Intellisense.

IconVisible (Optional) Sets whether the icon will be visible or not at runtime.

Example:

```
Form.IconSet(1,iconArrowFilled, "", "Backup")
```

Versions Supported: RFgen 5.2.4.2 and newer.

Image.Bitmap

This property accesses the image in the signature capture object and allows the user to write it to a file. This property can also read from a file and place the BMP formatted image in an Image control. This function will support Base64 images. Be sure to have pressed Enter on the signature box before trying to read the picture from it. After the OnEnter event finishes the prompt will have a value.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Image.Bitmap = bImage

Alternate: bImage = PromptID.Image.Bitmap

bImage (Variant) is the byte array containing the image.

Example:

reading from a file:

```
Dim bImage() As Byte
Open "C:\sig.bmp" For Binary Access Read As #1
 ReDim bImage(LOF(1))
 Get #1,,bImage
Close #1
Image1.Image.Bitmap = bImage
```

Example:

writing to a file:

```
Dim bImage() As Byte
bImage = sigSignature.Image.Bitmap
```

```
Open "C:\sig.bmp" For Binary Access Write As #1
 Put #1,,bImage
Close #1
```

Image.DisplayMode

This function places an image into the control and has options to tile, stretch, not alter or position the image in a number of ways. The options are: BottomCenter, BottomLeft, BottomRight, CenterCenter, CenterLeft, CenterRight, TopCenter, TopLeft, TopRight, Default, Stretched, Disabled and Tile.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Image.DisplayMode(enVal)

Alternate: enVal = PromptID.Image.DisplayMode

enVal (enImageAlign) is the alignment style for the image

Example:

```
btnPrint.Image.DisplayMode = ImageAlignCenterCenter
RFPrompt("btnPrint").Image.DisplayMode = ImageAlignTopLeft
RFPrompt(2).Image.DisplayMode = ImageAlignBottomRight
```

Image.GetBitmap

This function retrieves a BMP object stored within the database and displays it for an image prompt.

Group: Prompt-Specific Extensions

Syntax: [bValue =] PromptID.Image.GetBitmap(SQL)

SQL (String) is the ODBC call to retrieve the OLE object stored in the database.

bValue (Boolean) Optional – returns True or False for the success of the command

Example:

```
Dim sSQL As String
sSQL = "Select Image from Inv where PartNo = '100'"
imgPartImage.Image.GetBitmap(sSQL)
RFPrompt("imgPartImage").Image.GetBitmap(sSQL)
RFPrompt(2).Image.GetBitmap(sSQL)
```

Image.LoadResource

This function retrieves an image object stored as an Images resource in the Solution Explorer > Images tree and displays it as a prompt.

Group: Prompt-Specific Extensions

Syntax:

[bValue =] PromptID.Image.LoadResource(sName)

sName (String) is the Image ID used to retrieve the Images resource object stored in the master database.

bValue (Boolean) Optional – returns True or False for the success of the command

Example:

```
imgPartImage.Image.LoadResource("stop")
RFPrompt("imgPartImage").Image.LoadResource("stop")
RFPrompt(2).Image.LoadResource("stop")
```

Image.Path

This property retrieves a graphic file and displays it in an image prompt. When using thin client mode the supported image types are BMP, DIB, GIF and JPG. When running in mobile mode, only BMP is supported.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Image.Path = vValue

Alternate: sValue = PromptID.Image.Path

sValue (String) is the path to the image.

vValue (Variant) sets the path to the image.

Example:

```
imgPartImage.Image.Path = "C:\House.GIF"
RFPrompt("imgPartImage").Image.Path = "C:\House.GIF"
RFPrompt(2).Image.Path = "C:\House.GIF"
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

Visible

This property accesses the visible property associated with a specific prompt. Setting it to True makes a prompt visible, while setting it to False makes it invisible. Even though the prompt may be invisible, the GotFocus, OnEnter and LostFocus events will still be executed for this prompt. The screen must either be manually refreshed immediately or the event allowed to finish before the screen will be updated.

Syntax: PromptID.Image.Visible = True

True Keeps the image visible state for the prompt.

False Makes the image invisible.

Examples:

```
'This keeps the search button in the TextBox remains visible even after you click the submit button
Private Sub TextBox2_Click()
On Error Resume Next
TextBox2.Image.Visible=True
End Sub
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

Index

This property returns the prompt number and is read only. The language extension App.PromptNo returns the same value.

Group: Prompt-Specific Extensions

Syntax:

vValue = PromptID.Index

vValue (Variant) is the variable containing the prompt's number.

Example:

```
Dim vValue As Variant
vValue = txtPart.Index
vValue = RFPrompt("txtPart").Index
vValue = RFPrompt(2).Index
```

KeyBoardMode

This property sets the mode for a device touchscreen or soft input panel (SIP)/soft keyboard.

There are multiple sources for a keyboard's mode: a) The touchscreen/soft input panel/virtual keyboard that is native to the device's operating system, and the RFgen keyboard file that was imported to the Dev Studio Solution Explorer > Keyboard folder, or the RFgen keyboard that a developer customized with unique keys/layout/function calls in the Dev Studio Solution Explorer > Keyboard folder.

The options available are: kbAlpha, kbAlphaNum, kbCustom1, kbCustom2, kbCustom3, kbFullAlpha, kbFullAlphaNum, kbNone, kbNumeric, and kbSIP. The modes kbHide or kbShow are NO LONGER valid as a keyboard mode operations.

kbAlpha - displays the RFgen alphabetical soft keyboard (not the device's operating system alphabetical touchscreen).

kbAlphaNum - displays the RFgen alphabetical and numeric keyboard (not the device's operating system touchscreen).

kbCustom - displays the RFgen keyboard referenced under the Dev Studio > Solution Explorer > Keyboard folder.

kbNone - will NOT display a keyboard; "kbNone" is the equivalent of null for a keyboard mode.

kbNumeric - displays a RFgen numeric keyboard (not the device's operating system numeric keyboard).

kbSIP - displays the device's operating system keyboard (not a RFgen keyboard).

For more information, see Soft Input Panel Extensions.

Group: KeyBoardMode

Syntax:

promptID.KeyboardMode=<enValue>

Example

```
txtBox.KeyboardMode = kbAlpha
```

Label.Align

This property aligns the text of a left, center, right or vertically. The vertical option only applies to labels.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Label.Align = enValue

enValue (enAlign) an enumeration that contains AlignCenter, AlignLeft, AlignRight and AlignVertical.

Example:

```
txtBox.Label.Align = AlignCenter
```

```
RFPrompt("txtBox").Label.Align = AlignCenter
RFPrompt(2).Label.Align = AlignCenter
```

Version Supported: RFgen 5.0, 5.1, 5.2 and newer.

Label.Autosize

This property is only for the Label, Image and MenuList controls. The options are to size the control based on content, Fill the screen from its starting location on down, Horizontal size means with width of the screen, None for no auto sizing and Vertical which stretches on the vertical height of the control.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Label.Autosize = enValue

enValue (enSizeModes) an enumeration that contains AutosizeContent, AutosizeFill, AutosizeHorizontal, AutosizeNone and AutosizeVertical

Example

```
txtBox.Label.Autosize = AutosizeContent
RFPrompt("txtBox").Label.Autosize = AutosizeContent
RFPrompt(2).Label.Autosize = AutosizeContent
```

Label.BackColor

The Label.BackColor property accesses the prompt's label background color and is the primary back ground color for the label control.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Label.BackColor = vValue

vValue (Variant) sets the color.

nValue (Long) is the color.

Example:

```
txtPart.Label.BackColor1 = vbWhite
txtPart.Label.BackColor1 = RGB(255,255,0) 'yellow
txtPart.Label.BackColor1 = &HFF0000      'blue
txtPart.Label.BackColor1 = QBColor(5)    'magenta
```

```
RFPrompt("txtPart").Label.BackColor1 = vbWhite
RFPrompt(2).Label.BackColor1 = vbWhite
```

Note: "BackColor1" replaced "BackColor" in RFgen 5.0 and 5.1.

Version Supported: RFgen 4.0 and 4.1 and RFgen 5.2 and newer.

Label.BackGradient

This property uses either just the BackColor(1) for solid backgrounds or both BackColor properties to create gradients in one of several directions in graphical mode.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Label.BackGradient = enValue

enValue (enGradients) enumeration that contains GradientDefault, GradientDiagonalLeft, GradientDiagonalRight, GradientHorizontal, GradientNone, GradientVertical and GradientVerticalSplit

Example:

```
txtPart.Label.BackColor1 = RGB(0,0,255)
txtPart.Label.BackColor2 = vbWhite
txtPart.Label.BackGradient = GradientVertical
RFPrompt("txtPart").Label.BackGradient = GradientNone
RFPrompt(2).Label.BackGradient = GradientVertical
```

Label.BorderStyle

This property affects the border associated with a specific label. The different options are Active Border where the border is given an outline color when it has the focus, Default which takes its value from the current theme, No Border so only the label is visible, Standard is just the visible border without a color outline, Transparent hides the field allowing the text to be entered on the application's background and Visible On Focus where the field is transparent unless it has focus.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Label.BorderStyle = enValue

Alternate: enValue = PromptID.Label.BorderStyle

enValue (enBorderStyles) is the border style used to render the label object. The options are DisplayActiveBorder, DisplayDefault, DisplayNoBorder, DisplayStandard, DisplayTransparent, DisplayVisibleOnFocus

Example:

```
txtPart.Label.BorderStyle = DisplayVisibleOnFocus
RFPrompt("txtPart").Label.BorderStyle = DisplayDefault
RFPrompt(1).BorderStyle = DisplayTransparent
```

Label.Font.Bold

This property accesses the prompt's label bold option.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Label.Font.Bold = bValue

Alternate: bValue = PromptID.Label.Font.Bold

bValue (Boolean) is True or False for bold or not bold.

Example:

```
txtPart.Label.Font.Bold = True
RFPrompt("txtPart").Label.Font.Bold = True
RFPrompt(2).Label.Font.Bold = True
```

Label.Font.Italic

This property accesses the prompt's label italic option.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Label.Font.Italic = bValue

Alternate: bValue = PromptID.Label.Font.Italic

bValue (Boolean) is True or False for italics or no italics.

Example:

```
txtPart.Label.Font.Italic = True
RFPrompt("txtPart").Label.Font.Italic = True
RFPrompt(2).Label.Font.Italic = True
```

Label.Font.Size

This property accesses the prompt's label text size.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Label.Font.Size = nValue

Alternate: vValue = PromptID.Label.Font.Size

nValue (Long) sets the font size. Default is 10.

vValue (Variant) is the font size.

Example:

txtPart.Label.Font.Size = 16

RFPrompt("txtPart").Label.Font.Size = 16

RFPrompt(2).Label.Font.Size = 16

Label.Font.Underline

This property accesses the prompt's label underline option.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Label.Font.Underline = bValue

Alternate: bValue = PromptID.Label.Font.Underline

bValue (Boolean) True or False for underline or no underline.

Example:

```
txtPart.Label.Font.Underline = True
RFPrompt("txtPart").Label.Font.Underline = True
RFPrompt(2).Label.Font.Underline = True
```

Label.ForeColor

This property accesses the prompt's label fore (font) color.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Label.ForeColor = nValue

Alternate: vValue = PromptID.Label.ForeColor

nValue (Long) sets the color.

vValue (Variant) is the color.

Example:

```
txtPart.Label.ForeColor = RGB(255,255,0) 'yellow
txtPart.Label.ForeColor = QBColor(5)    'magenta
txtPart.Label.ForeColor = &HFF0000      'blue
txtPart.Label.ForeColor = vbWhite
RFPrompt("txtPart").Label.ForeColor = vbWhite
RFPrompt(2).Label.ForeColor = vbWhite
```

Label.Height

This property accesses the Height property for a label object associated with a specific prompt. This value can be changed at run time if there is a need to change prompt sizes on the screen. Depending on the client environment the default values may be in either pixels or characters.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Label.Height = nValue

Alternate: vValue = PromptID.Label.Height

nValue (Long) sets the display length for a prompt's label object.

vValue (Variant) is the display length for a prompt's label object.

Example:

```
Dim vValue As Variant
txtPart.Label.Height = 10
RFPrompt("txtPart").Label.Height = 10
vValue = RFPrompt(2).Label.Height
```

Label.Left

This property accesses the column position for a label object associated with a specific prompt. This value can be changed at run time if there is a need to move prompts around on the screen. Depending on the client environment the default values may be in either pixels or characters.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Label.Left = nValue

Alternate: vValue = PromptID.Label.Left

nValue (Long) sets the starting column for a prompt's label object.

vValue (Variant) is the starting column for a prompt's label object.

Example:

```
Dim vValue As Variant
vValue = txtPart.Label.Left
vValue = RFPrompt("txtPart").Label.Left
RFPrompt(2).Label.Left = 1
```

Label.Theme

This property gets or sets the theme for the label portion of the prompt.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Label.Theme = vValue

Alternate: vValue = PromptID.Label.Theme

vValue (Variant) is the name of the theme to set or retrieve

Example:

```
Dim vValue As Variant
txtPart.Label.Theme = "Standard"
vValue = txtPart.Label.Theme
vValue = RFPrompt("txtPart").Label.Theme
vValue = RFPrompt(2).Label.Theme
```

Label.Top

This property accesses the row position for a label object associated with a specific prompt. This value can be changed at run time if there is a need to move prompts around on the screen. Depending on the client environment the default values may be in either pixels or characters.

Group: Prompt-Specific Extensions

Syntax:

Syntax: PromptID.Label.Top = nValue

Alternate: vValue = PromptID.Label.Top

nValue (Long) sets the row to display a prompt's label object.

vValue (Variant) is the row to display a prompt's label object.

Example:

Dim vValue As Variant

vValue = txtPart.Label.Top

vValue = RFPrompt("txtPart").Label.Top

RFPrompt(2).Label.Top = 3

Label.Width

This property accesses the Width property for a label object associated with a specific prompt. This value can be changed at run time if there is a need to change prompt sizes on the screen. Depending on the client environment the default values may be in either pixels or characters.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Label.Width = nValue

Alternate: vValue = PromptID.Label.Width

nValue (Long) sets the display length for a prompt's label object.

vValue (Variant) is the display length for a prompt's label object.

Example:

Dim vValue As Variant

vValue = txtPart.Label.Width

vValue = RFPrompt("txtPart").Label.Width

RFPrompt(2).Label.Width = 10

Layout.Column( )

This method sets the sizemode, size, visibility, and minimum size, and maxiumum size for a specific column in a Layout control.

Group: Prompt-Specific Extensions

Syntax: PromptID.Layout.Column(x).[property name]

Column(x) "x" Is the index value of the column

Property Name The values are: Size, SizeMode, MaxSize, MinSize, and Visible.

Examples

```
Private Sub Layout1_GotFocus(ByRef Rsp As String, ByRef AllowChange As Boolean)
  On Error Resume Next
    Layout1.Layout.Column(2).SizeMode = stPixels    'This sets the size type for all the objects under column 2.
    Layout1.Layout.Column(2).Size = 20    'Make column 20 pixels wide. Use Layout1.Layout.Row(x).Size to set the height
    Layout1.Layout.Column(2).Visible = True    'Make the column content visible
    Layout1.Layout.Column(2).MaxSize = 18    'The maximum size this can autosized to is 18 pixels
    Layout1.Layout.Column(2).MinSize = 5    'The minimum size this can be autosized to is 5 pixels
    Label1.Parent    'This is the shortcut name for "Layout1"
  End Sub
```

Column Property Descriptions

Size - The SizeMode value. For example, if your SizeMode is a Percent, and you entered "50" for Column 1, then column 1 gets 50% of the space that's available in the Layout control. If the SizeMode was Pixels, then "50" is 50 pixels. If you set SizeMode to "AutoSize", this value is ignored.

(SizeMode) - This sets the metric / method that the Layout control uses to allocate space to the column.

- *AutoSize* – Column width is based on the largest item (Label, image etc) inside the column; the Width value is ignored. If the column is empty, the column will collapse (i.e. become 1 pixel wide)

- *Pixels* – Uses the Width value to set the column width.

- *Percent* – Uses the space left over from other columns or uses the Width value to calculate the allocation as a percentage of the Layout control. If you have columns where each column has at least one object (i.e. Label, Caption, button, TextBox etc), and want to use AutoSize, its best to have at least one column set to Percent – preferably one that contains TextBoxes.

MaxSize - The maximum size of the column can be expanded to if the Layout control is autosized or scaled

MinSize - The minimum size of the column can be reduced to if the Layout control is autosized or scaled.

Visible - True displays the contents of the column; False hides it.

Left

This property accesses the column position for the data field portion of a prompt object. This value can be changed at run time if there is a need to move prompts around on the screen. Depending on the client environment the default values may be in either pixels or characters.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Left = nValue

Alternate: vValue = PromptID.Left

nValue (Long) sets the starting column for a prompt's data field.

vValue (Variant) is the starting column for a prompt's data field.

Example:

```
Dim vValue As Variant
vValue = txtPart.Left
vValue = RFPrompt("txtPart").Left
RFPrompt(2).Left = 1
```

Version Supported: RFgen 5.0, 5.1, 5.2 and newer.

Map.CenterMap

Use this method to specify the center of a map/area using the Latitude and Longitude. This extension only functions with the Map control.

Group: Prompt-Specific Extensions

Syntax: bValue = PromptID.Map.CenterMap(sLat, sLng)

sLat: (String) is the latitude of the center of the map

sLong: (String) is the longitude of the center of the map

Example:

```
Dim SLat As String
Dim sLng As String
sLat = "38.575764"
sLng = "121.478851"
mMapArea.Map.CenterMap(sLat,sLng)
```

Version Supported: RFgen 5.1, 5.2 and newer.

Map.GetAddress

This method returns an address converted from the longitude and latitude geo-coordinates. This extension only works with the Map control.

Group: Prompt-Specific Extensions

Syntax: bValue = PromptID.Map.GetAddress(sLat, sLng, sAddrs)

bValue: (Boolean) True returns a value if successful; False if it fails

sLat: (String) is the Latitude of the address

sLng: (String) is the Longitude of the address

sAddrs: (String) is the address converted from the Latitude and Longitude

Example:

```
sLat = "38.575764"
sLng = "121.478851"
mMapArea.Map.GetAddress(sLat,sLng,sAddrs)
```

Version Supported: RFgen 5.1, 5.2 and newer.

Map.GetDirections

This returns the driving directions between a start and destination. This extension only works with the Map control.

Group: Prompt-Specific Extensions

Syntax: bVal = mapArea.Map.GetDirections(sOrigin, sDest, sDir, bGetMap, sDur, sDist)

bVal: (Boolean) Returns True if the conversion was retrieved; False if it failed.

sOrigin: (String) is the origination or start of the route

sDest: (String) is the route destination

sDir: (String Array) are the directions -- the list of roads between origination and destination

bGetMap: (Boolean) Returns True if the map was retrieved; False if it failed.

sDur: (String) is the duration of the distance traveled

sDist: (String) is the distance traveled

Example:

```
DIM sDir as String
```

mapArea.Map.GetDirections(txtOrig.Text, txtDest.Text, sDir, True)

Version Supported: RFgen 5.1, 5.2 and newer.

Map.GetLatLng

Use this method to return the Latitude and Longitude of a physical address. This extension only works with the Map control.

Group: Prompt-Specific Extensions

Syntax: bVal = PromptID.Map.GetLatLng (sAddr, sLat, sLng)

bValue (Boolean) True means the conversion was retrieved; False means it failed.

sAddr: (String) is the address of the location

sLat: (String) is the latitude of the location

sLng: (String) is the longitude of the location

Example:

```
MapArea.Map.GetLatLng(txtAddress.Text, textLat.Text,  textLng.Text, True)
```

Version Supported: RFgen 5.1, 5.2 and newer.

Map.PlanRoute

Use this method to obtain the most efficient driving route for multiple locations. This extension only works with the Map control.

Group: Prompt-Specific Extensions

Syntax: bVal = PromptID.Map.PlanRoute(sOrigin, sDest, sPoints, bGetMap)

bVal: (Boolean) True means the conversion was retrieved; False means it failed

sOrigin: (String) is the origination or start of the planned route

sDest: (String) is the destination of the planned route

sPoints: (String Array) Points are an array of delivery addresses and once its returned, the points will contain the sorted addresses.

bGetMap: (Boolean) Returns True if the map was retrieved; False if it failed.

Example:

```
Dim sPoints(3)As String
Dim sSrc as String
```

```
  Dim sDst as String
sSrc = "Services, 200 Plaza Dr, Folsom, CA 95630, USA"
sDst = "Services, 200 Plaza Dr, Folsom, CA 95630, USA"
sPoints(0) = "Services, 200 Plaza Dr, Folsom, CA 95630, USA"
sPoints(1) = "3321 Durock Rd, Cameron Park, CA 95682, USA"
sPoints(2) = "300 Automall Dr, Roseville, CA 95661, USA"
If not mapRoute.Map.PlanRoute(sSrc,sDst, sPoints, True) Then
App.MsgBox "Your route plan needs to be revised."
```

Version Supported: RFgen 5.1, 5.2 and newer.

Map.Zoom

Use this method to zoom in/out of the map area. This extension only functions with the Map control.

The Map control requires a Google license/key. We recommend you review the Google documentation for details on parameters that are supported. For example, as per the google documentation:

"Maps on Google Maps have an integer 'zoom level' which defines the resolution of the current view. Zoom levels between 0 (the lowest zoom level, in which the entire world can be seen on one map) and 21+ (down to streets and individual buildings) are possible within the default roadmap view. Building outlines, where available, appear on the map around zoom level 17. This value differs from area to area and can change over time as the data evolves."

Group: Prompt-Specific Extensions

Syntax: PromptID.Map.Zoom = nValue

nValue (Long) sets the Zoom size

Example:

```
Private Sub Button1_Click()
  On Error Resume Next
  mapDir.Map.Zoom = 3
End Sub

Private Sub Form_Load()
  On Error Resume Next
   Dim sPoints(3) As String
   Dim sSrc As String
   Dim sDst As String
```

```
sSrc = "Services, 200 Plaza Dr, Folsom, CA 95630, USA"
sDst = "Services, 200 Plaza Dr, Folsom, CA 95630, USA"
sPoints(0) = "Services, 200 Plaza Dr, Folsom, CA 95630, USA"
sPoints(1) = "3321 Durock Rd, Cameron Park, CA 95682, USA"
sPoints(2) = "300 Automall Dr, Roseville, CA 95661, USA"
If Not Map1.Map.PlanRoute(sSrc,sDst, sPoints, True) Then
App.MsgBox "Your route plan needs to be revised."
End If
End Sub
```

Version Supported: RFgen 5.1, 5.2 and newer.

MonitorKeys

If you define an OnKeyDown event for a prompt, the default behavior is to return any key pressed for processing at the scripting level. This method allows you to narrow down the scope of keys requested to improve performance.

If you have the OnKeyDown event in script and did not specify any keys, the default mode is vkAll.

Group: Prompt-Specific Extensions

Syntax:

PromptID.MonitorKeys(Mode,[Keycode]...

Modes:

vkNone = No keys are monitored; no keys will be sent to the script engine.

vkAll = All keys monitored; All keys will be preprocessed in the script engine before RFgen handles them.

vkCursor = The up, down, left, right, page up, page down, home, end keys will be preprocessed in the script engine before RFgen handles them.

vkEscape = The Escape key will be preprocessed in the script engine before RFgen handles it.

vkList =The list of virtual keycode values specified as optional parameters to be preprocessed in the script engine before RFgen handles them.

[keycode] = comma separated list of virtual key codes

PageNo

This property accesses the page number property associated with a specific prompt. This property is read-only at run time.

Group: Prompt-Specific Extensions

Syntax: vPage = PromptID.PageNo

vPage (Variant) is numeric and represents the page number on which the prompt will be displayed.

Example:

```
Dim iValue as Integer
iValue = txtPart.PageNo
iValue = RFPrompt("txtPart").PageNo
iValue = RFPrompt(2).PageNo
```

Password

This property, when set to True or False, will turn on or off asterisks (*) in the data field of a textbox that mask the data.

Group: Prompt-Specific Extensions

Syntax: PromptID.Password = bValue

bValue (Boolean) set to True or False

>

Examples: These reference the RFLogin form:

```
Password.Password = True
RFPrompt("Password").Password = True
RFPrompt(5).Password = True
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1. Replaced by MaskInput in 5.2.

Required

This property, when set to 'True', forces users to enter data into the prompt, while setting it to False allows users to skip the field. If the prompt never gets the focus, this property will not have a chance to be used.

Group: Prompt-Specific Extensions

Syntax:

PromptID.Required = bValue

Alternate: bValue = PromptID.Required

bValue (Boolean) is the required state for the prompt.

Example:

```
Dim bValue As Boolean
bValue = txtPart.Required
bValue = RFPrompt("txtPart").Required
RFPrompt("txtPart").Required = True
```

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

The RFPrompt

This control has been deprecated since 5.0 but is still used in legacy code.

ScrollBars

This property can be used either on the form control or a list control to enable or disable the scrollbars. The options are: Horizontal, Vertical, Both, None, or the default value from the selected theme.

Group: Prompt-Specific Extensions

Syntax: PromptID.ScrollBars = enValue

Alternate: Form.ScrollBars = enValue

Alternate: enValue = PromptID.ScrollBars

enValue (enScrollBars) is the option for enabling or disabling the scrollbars on the control.

Examples:

Form.ScrollBars = enScrollDefault

txtPart.ScrollBars = enScrollBoth

RFPrompt("txtPart").ScrollBars = enScrollVert

RFPrompt("txtPart").ScrollBars = enScrollNone

SelLength

In graphical mode, this property sets or returns the number of characters highlighted by the mouse for a specified text prompt. For example, if the text box contained '123456' and using the mouse you drag and select only '456', the property will return 3, the number of characters selected.

Group: Prompt-Specific Extensions

Syntax: vValue = PromptID.SelLength

Alternate: PromptID.SelLength = iValue

vValue (Variant) returns the numeric value

iValue (Integer) sets the numeric value and selects the specified number of characters in the field starting from the SelStart value

Examples:

Dim vValue as Variant

vValue = txtPart.SelLength

vValue = RFPrompt("txtPart").SelLength

RFPrompt("txtPart").SelLength = 3

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

SelStart

In graphical mode, this property sets or returns the position where the highlighting starts for a specified text prompt. For example, if the text box contained '123456' and using the mouse you drag and select only '56', the property will return 4, the position where the selection started.

Group: Prompt-Specific Extensions

Syntax: vValue = PromptID.SelStart

Alternate: PromptID.SelStart = iValue

Nbr (Variant) is numeric, or fieldname, and refers to a specific prompt.

vValue (Variant) returns the numeric value

iValue (Integer) sets the numeric value

Examples:

Dim vValue as Variant

vValue = txtPart.SelStart

vValue = RFPrompt("txtPart").SelStart

RFPrompt("txtPart").SelStart = 1

Version Supported: RFgen 4.0, 4.1, 5.0, 5.1, 5.2 and newer.

SetFocus

This method is equivalent to the App.SetFocus command but in this case the syntax check will catch errors because the prompt name must exist. This method is the preferred approach.

Group: Prompt-Specific Extensions

Syntax: [bValue =] PromptID.SetFocus([bSaveRSP], [bValidateRSP])

bValue (Boolean) Optional – Returns True or False depending on the success of the command

bSaveRSP (Boolean) Optional – set to True to save any changes to the current prompts value before switching focus to the new prompt.

bValidateRSP (Boolean) Optional – set to True to call the edit checks and to re-run the OnEnter event.

Example:

PartNo.SetFocus ' Cursor will go to prompt PartNo

PartNo.SetFocus(True, True) ' Cursor will go to "PartNo", save current Text property and execute edits and OnEnter event

Version Supported: RFgen 5.0, 5.1, 5.2 and newer.

Tab

This property is only used with the **TabControl** prompt which enables users to toggle between pages or groups of prompts at run time. At runtime, the **TabControl** determines which tab is active and which one is inactive. The following provides the methods for stylizing (coloring, shaping of the bezel and boarders) for the active tab.

The **Tab.Caption** resets the caption of the specified tab to the text assigned to it. The remainder of the functions Tab.Curxxx or Tab.Altxxx set values for the tabs when they are active or inactive respectively. If multiple tabs are used, RFgen automatically knows which are active or inactive, and applies these property values accordingly.

Group: Prompt-Specific Extensions

Syntax: PromptID.Tab <method or property>

bVal (Varient) sets the identifier for the active or inactive tab prompt in a TabControl.

PromptID.Tab. AltBtnBackColor1 = vValue

Alternate: nValue = PromptID.Tab. AltBtnBackColor1

vValue (Varient) sets the color

nValue (Long) is the color

Example:

```
TabControl.Tab.AltBtnBackColor1 = vbRed
TabControl.Tab. AltBtnBackColor2 = vbYellow
TabControl.Tab. AltBtnBackGradient = GradientVertical
TabControl.Tab. AltBtnBevel = 5
TabControl.Tab. AltBtnBorderColor = vbRed
TabControl.Tab. AltBtnBorderStyle = DisplayNoBorder
TabControl.Tab. AltBtnForeColor = vbBlue
TabControl.Tab.CurBtnBackColor1 = vbRed
TabControl.Tab.CurBtnBackColor2 = vbYellow
TabControl.Tab.CurBtnBackStyle = GradientVertical
TabControl.Tab.CurBtnBevel = 5
TabControl.Tab.CurBtnBorderColor = vbRed
TabControl.Tab.CurBtnBorderStyle = DisplayNoBorder
TabControl.Tab.CurBtnForeColor = vbBlue
```

Tab.AltBtnBackColor1

This method accesses the inactive tab button's background color in a TabControl prompt, and is the primary background color for the button.

Group: Prompt-Specific Extensions

Syntax: Refer to "Tab" for syntax and code examples for this property.

Tab.AltBtnBackColor2

This property accesses the inactive tab button's background color in a TabControl prompt, and is the secondary background color for the button.

Group: Prompt-Specific Extensions

Syntax: Refer to "Tab" for syntax and code examples for this property.

Tab.AltBtnBackStyle

This property sets the background coloring style for the inactive tab button(s) on the graphical TabControl. The style can be set to use just the AltBtnBackColor1 for a solid background, or both AltBtnBackColor1 and AltBtnBackColor2 to create a gradient (blended) color in several directions.

Group: Prompt-Specific Extensions

Syntax: Refer to "Tab" for syntax and code examples for this property.

Tab.AltBtnBevel

This method sets the roundness of inactive tab button's edges in a TabControl prompt.

Group: Prompt-Specific Extensions

Syntax: Refer to "Tab" for syntax and code examples for this property.

Tab.AltBtnBorderColor

This method accesses the inactive tab button's boarder color in a TabControl prompt.

Group: Prompt-Specific Extensions

Syntax: Refer to "Tab" for syntax and code examples for this property.

Tab.AltBtnBorderStyle

This method accesses the inactive tab button's boarder style in a TabControl prompt.

Group: Prompt-Specific Extensions

Syntax: Refer to "Tab" for syntax and code examples for this property.

Tab.AltBtnForeColor

This method accesses the inactive tab button's data field fore color in a TabControl prompt.

Group: Prompt-Specific Extensions

Syntax: Refer to "Tab" for syntax and code examples for this property.

Tab.Caption

This method resets the caption of the specified tab to the text assigned to it. Refer to the syntax and examples listed under "Tab". Note: This method can be applied to active an inactive tab.

Group: Prompt-Specific Extension

Syntax: PromptID.Tab.Caption = vValue

Alternate: sValue = PromptID.Tab.Caption

sValue (String) is the caption of the prompt's tab

vValue (Varient) is the caption to display for the prompt.

Example:

```
tabPlant.Tab.Caption(1) = "Changed"
```

Tab.CurBtnBackColor1

This method accesses the active tab button's background color in a TabControl prompt, and is the primary background color for the button. Refer to "Tab" for syntax and code examples for this property.

Tab.CurBtnBackColor2

This method accesses the inactive tab button's background color in a TabControl prompt, and is the secondary background color for the button. Refer to "Tab" for syntax and code examples for this property.

Tab.CurBtnBackStyle

This method accesses the active tab button's coloring style in a TabControl prompt, and uses either just the BackColor1 for solid backgrounds or both BackColor properties to create gradients in one of several directions. Refer to the examples listed under "Tab".

Syntax: PromptID.Tab.CurBtnBackStyle = enGradients

enGradients: GradientDefault, GradientDiagonalLeft, GradientDiagonalRight, GradientHorizontal, GradientNone, GradientVertical, and GradientVerticalSplit

Tab.CurBtnBevel

This method sets the roundness of active tab button's edges in a TabControl prompt. Refer to the syntax and examples listed under "Tab".

Tab.CurBtnBorderColor

This method accesses the active tab button's boarder color in a TabControl prompt. Refer to the syntax and examples listed under "Tab".

Tab.CurBtnBorderStyle

This method accesses the active tab button's boarder style in a TabControl prompt. Refer to the syntax and examples listed under "Tab".

Tab.CurBtnForeColor

This method accesses the inactive tab button's data field fore color in a TabControl prompt. Refer to the syntax and examples listed under "Tab".

Tab.OpenTab

This method switches the focus to the active tab in a TabControl prompt.

Syntax:

iVal (Index) the numeric identifier of the tab in the graphical TabControl. The tab index is one-based (not zero-based).

Example

```
'Make my second tab in my TabControl the active tab.
TabControl.Tab.OpenTab(2)
```

Tag

Prompt.tag acts like a extra value field for a prompt that is not visible to the user and is a property that specific prompt. This language extension is available for all prompts, but is not applicable to Rows within a ListBox.

Syntax: PromptID.Tab

Tag Data type is Varient.

Example:

```
txt1User.text = "Alfred"  ' This will show the User Name 'Alfred" in the textbox at runtime.
txt1User.tag = 20042  'This adds the additional value (the User ID value) but will not display in the textbox at runtime.
```

Supported Versions: RFgen 5.1 and newer.

Text

This property accesses the data contained by a text box object associated with a specific prompt. Other prompts also use this property such as the list objects. The property will return the highlighted entry in the list.

Syntax: PromptID.Text = vValue

Alternate: Value = PromptID.Text

vValue (Variant) is the data collected in a prompt object.

Examples:

Dim vValue As Variant

vValue = txtPart.Text

vValue = RFPrompt("txtPart").Text

RFPrompt(2).Text = "Car Parts"

Top

This property accesses the row position for a text box object associated with a specific prompt. This value can be changed at run time if there is a need to move prompts around on the screen. Depending on the client environment the default values may be in either pixels or characters.

Syntax: PromptID.Top = nValue

Alternate: vValue = PromptID.Top

nValue (Long) sets the row for a prompt's data field.

vValue (Variant) is the row for a prompt's data field.

Examples:

Dim iValue As Integer

iValue = txtPart.Top

iValue = RFPrompt("txtPart").Top

RFPrompt("txtPart").Top = 5

TreeView

This is not a property; Its a graphical prompt that lists data in a tree in a hierachial, nested tree format. Its methods are similar to the ListBox controla. Use the List property to associate data with this graphical prompt.

Example

```
  Dim Assembly_A As vbaRow
  Dim Assembly_B As vbaRow
  Dim SubAssembly_A As vbaRow
  Dim SubAssembly_B As vbaRow

  Set Assembly_A = TreeView1.List.AddItem(1, "Assembly A: Glass")
Set SubAssembly_A = Assembly_A.AddItem(2, "Location: A01")
```

```
Set Assembly_B = TreeView1.List.AddItem(1,"Assembly B: Frame")
Set SubAssembly_B = Assembly_B.AddItem(2, "Location: B01")
```

Type

This property will return a read-only enumeration describing what type of prompt is defined based on the prompt index name or number.

Syntax: enValue = PromptID.Type

enValue (enFieldType) the enumeration for the prompt type. Values are:
rfButton
rfButtonList
rfCheckBox
rfComboBox
rfDataGrid
rfDesktopIcons
rfForm
rfFrame
rfHostScreen
rfImage
rfImageList
rfLabel
rfListBox

rfMenuList
rfOptions
rfSignature
rfTextBox

rfVocollectTask

Examples:

```
Dim enValue As enFieldType
enValue = txtPart.Type

Dim i As Integer
For i = 1 To App.PromptCount
  If RFPrompt(i).Type = rfLabel Then Exit For
Next
```

Image.Visible

This property sets whether image is visible or not.

Group: Prompt-Specific Extensions

Syntax: Prompt.Image.Visible = bValue

bValue (Boolean) is the visible state for the image. True makes it visible; False makes it invisible.

Example:

```
Private Sub Button1_Click()
Button1.Image.Visible = False
'When the application loads at runtime the button is displayed.
'When the user taps the button, the image on the button disappears.
End Sub
```

Width

This property accesses the Width property for a prompt object. This value can be changed at run time if there is a need to change prompt sizes on the screen. Depending on the client environment the default values may be in either pixels or characters.

Group: Prompt Extensions

Syntax: PromptID.Width = nValue

Alternate: vValue = PromptID.Width

nValue (Long) sets the display length for a prompt object.

vValue (Variant) is the display length for a prompt object.

Example:

```
Dim vValue As Variant
vValue = txtPart.Width
vValue = RFPrompt("txtPart").Width
RFPrompt(2).Width = 10
```

scr

Screen Display Extensions

Screen display commands typically interact with the user at the GUI level for clients working on a client.

They include: Screen.Bell, Screen.Height, Screen.Refresh, Screen.Update, Screen.Width, and Screen.Zoom.

The following methods have been removed from RFgen as they were designed for Telnet operations and are no longer supported in our graphical environment.

- Device.PrinterOff
- Device.PrinterOn
- Screen.Clear
- Screen.ClearEOL
- Screen.ClearEOP
- Screen.DrawLine
- Screen.Print
- Screen.ResetCursor
- Screen.ReverseOn
- Screen.ReverseOff

Bell

This method was originally designed for telnet clients, and will cause the Client device terminal to beep. For WinCE devices, this extension will cause the waveform to play. Windows CE specifies five default waveforms whose values are 0, 16, 32, 48 and 64. Each of these sounds must be setup on the CE device.

Group: Screen Display

Syntax: Screen.Bell([iNbr])

iNbr (Integer) Optional – is the bell sound.

Examples:

```
Screen.Bell(0)  ' Graphical mode – OK sound

Screen.Bell(16) ' Graphical mode – Hand sound

Screen.Bell(32) ' Graphical mode – Question sound

Screen.Bell(48) ' Graphical mode – Exclamation sound

Screen.Bell(64) ' Graphical mode – Asterisk sound
```

Clear (for Telnet Clients)

This method has been removed from RFgen as it was designed for Telnet operations and is no longer supported in the RFgen graphical environment.

This command was used to clear the Client screen until the server needed to refresh the screen or unless a Screen.Refresh command was used to bring back any prompts that were in the area. Any text placed there with a Screen.Print command was removed permanently.

Group: Screen Display Extensions

Syntax: Screen.Clear([bSendNow])

bSendNow (Boolean) Optional – clears the screen immediately if this option is set to True.

ClearEOL (for Telnet Clients)

This method has been removed from RFgen as it was designed for Telnet operations and is no longer supported in the RFgen graphical environment.

This method was used to clear the Client screen from the current position to the end-of-line. Using a Screen.Refresh would have brought back any prompts that were in the area. Any text placed there with a Screen.Print command would have been remove permanently.

Syntax: Screen.ClearEOL

ClearEOP (for Telnet Clients)

This method has been removed from RFgen as it was designed for Telnet operations and is no longer supported in the RFgen graphical environment.

This method was used to clear the text boxes of all user-entered fields on the devices terminal screen from the current position to the bottom of the screen.
Using a Screen.Refresh will bring back any prompts that were in the area. Any text placed there with a Screen.Print command will be removed.

Syntax: Screen.ClearEOP

DrawLine (forTelnet Clients)

This method has been removed from RFgen as it was designed for Telnet operations and is no longer supported in the RFgen graphical environment.

This method was used to draw a vertical line, horizontal line, or create a box on device screen.

To display boxes within boxes, you must draw them from the inside to the outside. This command works in both character and graphical mode.

Syntax: Screen.DrawLine(nStartCol, nStartRow, nEndCol, nEndRow, [bSendNow])

nStartCol (Long) is the column (x coordinate) in which to start drawing the line/box.

nStartRow (Long) is the row (y coordinate) in which to start drawing the line/box.

nEndCol (Long) is the column (x coordinate) in which to finish drawing the line/box.

nEndRow (Long) is the row (y coordinate) in which to finish drawing the line/box.

bSendNow (Boolean) Set to True if the method should paint the lines on the screen immediately.

Examples:

Screen.DrawLine(0,10,20,10) Draws a horizontal line

Screen.DrawLine(0,0,0,16) Draws a vertical line

Screen.DrawLine(0,0,20,16) Draws a box around the screen

Screen.DrawLine(0,0,20,16, True) Draws immediately

Height (for Telnet Clients)

This method has been removed from RFgen as it was designed for Telnet operations and is no longer supported in the RFgen graphical environment.

It was used to return the height in pixels or rows for the current application. (See Screen.Width).

Syntax: Value = Screen.Height

Value (Variant) equals the number of available rows in the current application or the height in pixels.

Print (for Telnet Clients)

This method has been removed from RFgen as it was designed for Telnet operations and is no longer supported in the RFgen graphical environment.

This command was used to move the cursor to a specified row and column, print text in normal or reverse video, and optionally clear to the end-of-line. In graphical mode, the permanent objects, prompts on the screen, would have taken precedence and the text would have displayed behind the prompt.

Making a prompt invisible and then using Screen.Print may be an alternate solution.

As with all 'screen-printing' commands, you cannot use this command in the Form_Load event because the application is not created until the Form_Load event is finished.

Syntax: Screen.Print(nColumn, nRow, vText, [bReverse], [bClearEOL], [bSendNow])

nColumn (Long) is the column (x coordinate) in which to place the cursor.

nRow (Long) is the row (y coordinate) in which to place the cursor.

vText (Variant) is the text to print at the current row and column.

bReverse (Boolean) Optional – set to True to print text in reverse video. This only applies to a character based device, not graphical.

bClearEOL (Boolean) Optional – set to True to clear to the end-of-line after printing text.

bSendNow (Boolean) Optional – set to True to send the print packet to the Client device immediately

Example:

```
Screen.Print(0, 10, "F4 to Exit", , True, True)
```

Refresh (for Telnet Clients)

This method has been removed from RFgen as it was designed for Telnet operations and is no longer supported in the RFgen graphical environment.

This command was used to clear the Client screen and then repaint all standard prompts and data. (Note: any text displayed using Screen.Print statements will not be redisplayed.)

Syntax: Screen.Refresh

ResetCursor (for Telnet Clients)

This method has been removed from RFgen as it was designed for Telnet operations and is no longer supported in the RFgen graphical environment.

ReverseOff (for Telnet Clients)

This method has been removed from RFgen as it was designed for Telnet operations and is no longer supported in the RFgen graphical environment.

ReverseOn (for Telnet Clients)

This command is for internal use only and has no use for the user.

Update

This command will update (repaint) the screen with pending changes that have not yet been sent to the device. Compared to Screen.Refresh which repaints the entire screen instead of just the pending changes, Screen.Update is faster and is typically used with form loads.

Note: Any text displayed using Screen.Print statement will not be redisplayed.

Group: Screen Display Extensions

Syntax: Screen.Update

Example:

```
Private Sub Form_Load()   On Error GoTo ErrorHand
 Dim oList As New SearchList, i As Long
 Image1.Visible = True
 Screen.Update
 oList.SetColumn(1, "Lot Number",-1)
  For i = 0 To 25000
    oList.AddItem("Lot " & i, "Lot " & i)
  Next
 TextBox1.Text = oList.ShowList
 Image1.Visible = False
Exit Sub
```

Versions Supported: RFgen 5.2.4.1

Width (in Telnet Clients)

This method has been removed from RFgen as it was designed for Telnet operations and is no longer supported in the RFgen graphical environment.

Returns the width in pixels or columns for the current application. (See Screen.Height).

Syntax: vValue = Screen.Width

vValue (Variant) equals the number of available columns in the current application or the width in pixels.

Example:

```
If App.ClientType = "TE" Then RFPrompt("Description").Length = Screen.Width
```

In this case, the text box containing the description will be the same width as the current application. Since the graphical mode would return pixels, setting the length of a textbox to a very large number would not be appropriate.

Screen Mapping Extensions

Screen mapping commands deal specifically with placing and scraping text to and from a green screen / legacy system. The following details all of the Screen Mapping VBA Extensions.

BeginTrans

This function is provided to allow a series of commands to be sent to a single host session when connection pooling is enabled. It basically gets a connection from the pool and holds it until SM.CommitTrans is called. If connection pooling is disabled, it has no effect. It returns True if a connection handle was available and removed from the pool.

Syntax: [bOK =] SM.BeginTrans([vScreen])

bOK (Boolean) Optional – Returns True if a connection handle is available for use.
vScreen (Variant) Optional – the name of a screen macro to be called so that upon connection the host will attempt to go to this screen. If the screen macro does not exist the pooled handle will be released and the function will return a False.

Example:

```
Dim bOK As Boolean

bOK = SM.BeginTrans
```

CommitTrans

This subroutine basically returns the current handle to the connection pool (if it was previously retrieved by using the SM.BeginTrans function) and makes it available to other client sessions. It returns True if the

handle was successfully released back to the pool.

Syntax: [bOK =] SM.CommitTrans

bOK (Boolean) Optional – Returns True if the handle was released back to the connection pool.

Example:

```
Dim bOK As Boolean

bOK = SM.CommitTrans
```

Connected

This function is used to determine if the host is available for direct input or if transactions should be queued for later input. It also evaluates both the current state of the socket connection to the host and the state of any data sent but not yet received. If there is non-received data in the send buffer, this command will mark the connection as closed. It returns True if the host is currently available.

Syntax: bIsConnected = SM.Connected

bIsConnected (Boolean) Returns True if the host is currently available (e.g. the telnet connection is currently valid).

Example:

```
Dim bIsConnected As Boolean

bIsConnected = SM.Connected
```

CurScreen

This function returns the name of the current screen in the host session if it can be identified. The server identifies the screen by comparing the host screen to the Text Identifier grid and looks for an ID match. If a match is found, it returns the screen name otherwise this function returns an empty string.

Note: This function only works if the current screen has been defined as a macro that is linked to the main menu macro that is currently in use. For example if MainMenu1 links to TransactionScreen1 and Screen2 and MainMenu1 is either configured in the screen mapping connector as the default menu or the program performed the SM.SetBase("MainMenu1") command, then only Screen1 and Screen2 can be identified. If Screen3 is linked to MainMenu2 and you are on Screen3 but MainMenu1 is your base menu, Screen3 cannot be identified. Use the SM.GetText command to verify your location.

Syntax: sScreenName = SM.CurScreen()

sScreenName (String) Returns the name of the current screen or an empty string if screen is unknown.

Example:

```
Dim sScreenName As String

sScreenName = SM.CurScreen()
```

FindText

This function is used to determine if a specified text string is currently displayed on the host screen. It can be used to look for the text in a specific screen location or to search the entire host screen for the text. It returns True if it is found. Optionally, it can also return the screen coordinates where the text was found.

Syntax: bFound = SM.FindText(sText, iStartCol, iStartRow, [vFoundCol], [vFoundRow])

bFound (Boolean) Returns True if the specified text string was found.

sText(String) Text string to be searched for.

iStartCol(Integer) the screen column to search for the text. Specify a column position of '**-1**' (minus 1) to search for the text in any screen column position.

iStartRow(Integer) the screen row to search for the text. Specify a row position of '**-1**' (minus 1) to search for the text in any screen row.

vFoundCol(Variant) Optional – the column position where the text was found

vFoundRow(Variant) Optional – the row position where the text was found.

Example:

```
Dim bFound As Boolean

bFound = SM.FindText("UserID", 3, 5)
```

GetArea

This command gets the text off the screen in any rectangular area.

Syntax: [bOK =] SM.GetArea(iStartCol, iStartRow, iEndCol, iEndRow, sAreaText, [bTrimData], [bAppendData])

bOK (Boolean) Optional – Returns True / False depending on the success of the command

iStartCol (Integer) start column position coordinate

iStartRow (Integer) start row position coordinate

iEndCol (Integer) end column position coordinate

iEndRow (Integer) end row position coordinate

sAreaText (String) variable containing the captured text

bTrimData (Boolean) Optional – trims each line (row) of data. Default is False

bAppendData (Boolean) Optional – appends the new block of data to the data variable rather than replaces it. Default is False

Example:

```
Dim sText As String

SM.GetArea(5, 5, 10, 10, sText, True)
SM.GetArea(5, 11, 10, 15, sText, True, True)
SM.GetArea(5, 16, 10, 20, sText, True, True)
```

GetAttribute

This command returns an integer value describing the characteristics of an X,Y coordinate from a host screen. This command does not work with a VT host, only 5250 and 3270.

Syntax: iValue = SM.GetAttribute(iCol, iRow)

iValue (Integer) contains the value in the table below

-1 - error was returned

0 - Normal
1 - Bold
2 - Reverse Video
4 - Underline
8 - Half
16 - Protected
32 - Hidden
64 - Graphic

iCol (Integer) is the column position in question.

iRow (Integer) is the row position in question.

Example:

```
Dim iValue As Integer

iValue = SM.GetAttribute(5,18)
```

GetBackColor

This command returns an integer value representing the back color located at the specified coordinates. This command does not work with a VT host, only 5250 and 3270.

Syntax: iValue = SM.GetBackColor(iCol, iRow)

iValue (Integer) contains the integer value of the back color

-1 - Error was returned

0 - Black

1 - Blue

2 - Green

3 - Cyan

4 - Red

5 - Purple

6 - Yellow

7 - Gray

8 - Light Blue

9 - Light Green

10 - Light Cyan

11 - Light Red

12 - Light Purple

13 - Light Yellow

14 - Light Gray

15 - White

iCol (Integer) Is the current column position in question.

iRow (Integer) Is the current row position in question.

Example:

```
Dim iValue As Integer
iValue = SM.GetBackColor(3,22)
```

GetCursor

This method is used to determine where the cursor is currently located on the host screen.

Syntax: SM.GetCursor(vCol, vRow)

vCol (Variant) Returns the current column position of the cursor.

vRow (Variant) Returns the current row position of the cursor.

Example:

```
Dim vCol As Variant
```

```
Dim vRow As Variant
SM.GetCursor(vCol, vRow)
```

GetForeColor

This command returns an integer value representing the fore color located at the specified coordinates. This command does not work with a VT host, only 5250 and 3270.

Syntax: iValue = SM.GetForeColor(iCol, iRow)

iValue (Integer) contains the integer value of the fore color

-1 - Error was returned

0 - Black

1 - Blue

2 - Green

3 - Cyan

4 - Red

5 - Purple

6 - Yellow

7 - Gray

8 - Light Blue

9 - Light Green

10 - Light Cyan

11 - Light Red

12 - Light Purple

13 - Light Yellow

14 - Light Gray

15 - White

iCol (Integer) Is the current column position in question.

iRow (Integer) Is the current row position in question.

Example:

```
Dim iValue As Integer
iValue = SM.GetForeColor(3,22)
```

GetText

This method is used to retrieve text from the host screen. Note: you can retrieve the entire screen buffer by specifying a starting position of 1,1 and a length of 2,000.

Syntax: SM.GetText(iCol, iRow, iLength, sScreenText, [bTrim])

iCol	(Integer) The starting screen column position to get text.
iRow	(Integer) The starting screen row position to get the text.
iLength	(Integer) The number of screen columns to retrieve text from. Note: on DBCS systems, this may not be the number of characters returned as some characters require two screen columns.
sScreenText	(String) The text being returned.
bTrim	(Boolean) Optional – if True, the string will be returned will all padding removed. Default is False.

Example:

```
Dim sText As String

SM.GetText(3, 6, 10, sText, True)
```

GoToScreen

This function attempts to navigate the host menu system to move to the requested application screen. It will return True if the desired screen is successfully displayed. Its method of operation (simplified) is as follows:

1. Test if the current screen is the requested screen.
2. Call the current screens "ReturnToMainMenu" method.
3. Call the requested screens "GoToScreen" method.

If the screen you are trying to get to is not part of the linked macros from the current main menu then the server will not navigate properly. The current main menu is defined in the screen mapping connector or by using the SM.SetBase command.

Syntax: [bOK =] SM.GoToScreen(sScreenName)

bOK	(Boolean) Optional – Returns True if the host session was successfully moved to the requested screen.
sScreenName	(String) The name of the screen to activate on the host session.

Example:

```
Dim bOK As Boolean
```

bOK = SM.GoToScreen("UserLogin")

IsScreen

This function returns a True or False by comparing the specified screen and optionally the page number against the host screen's current page.

Syntax: [bIsScreen =] SM.IsScreen(sScreenName)

bIsScreen (Boolean) Optional – Returns True if the requested screen is the current active screen

sScreenName (String) The name of the screen to be evaluated.

Example:

Dim bIsScreen As Boolean

bIsScreen = SM.IsScreen("invTrans")

LogOff

This function is used internally to logoff the session just prior to termination of the process. It works by sending the user to the base screen and executes the 'LogOff" macro. It returns True if the session was logged off properly. Note: this function is not required to be called by the user as the server calls it automatically when the user disconnects.

Syntax: [bOK =] SM.LogOff

bOK (Boolean) Optional – Returns True if the session was successfully logged off.

Example:

Dim bSuccess As Boolean

bSuccess = SM.LogOff

LogOn

This function is used to log in to the host system. It returns True if the session was logged in properly. The host system must already be logged off or you must use the SM.LogOff command. A user and password may be specified but the host does not use this information. This is for validation later in the programming. Note: this function is not always required to be called by the user as the server calls it automatically.

Syntax: [bOK =] SM.LogOn([vUser], [vPassword], [vMenu])

bOK (Boolean) Optional – Returns True is the session was successfully logged on.

vUser (Variant) Optional – value accessed by SM.SessionUser

vPassword (Variant) Optional – value accessed by SM.SessionPwd

vMenu (Variant) Optional – value specifying the menu to log in to

Example:

```
Dim bSuccess As Boolean
bSuccess = SM.LogOn
bSuccess = SM.LogOn("UserName", "Password", "MainMenu")
```

PadInput

This command adds spaces to the end of a string until the total length of the string is the size specified in the command. Using a number too small will truncate the string.

Syntax: SM.PadInput(sText, iLength)

sText (String) the string of characters to be padded

iLength (Integer) the number of spaces to add to the end of the string

Example:

```
Dim sPartNo as String
sPartNo = "12345"
SM.PadInput(sPartNo, 8)
```

Result is "12345 " with 3 spaces at the end

PingHost

This command sends a single ping packet to the host and waits for a response. It works against the current connection and can be used in transaction or navigation macros, used against a local connection, or you can get a pooled connection (SM.BeginTrans) and then use it to test the connection.

Syntax: bOK = SM.PingHost

bOK (Boolean) returns True / False if the server can be reached

Example:

```
Dim bOK As Boolean
bOK = SM.PingHost
```

ResetConnection

This command will reset the active host session in an effort to re-establish a locked-up connection.

Syntax: SM.ResetConnection

Example:

SM.ResetConnection

SendCTRL

This function sends the <CTRL> code with the specified character. It returns True if the key was successfully entered. Note: reasons for a negative or False value might be that keyboard entry is inhibited, or that the cursor was not in an input field.

Syntax: [bOK =] SM.SendCTRL(sKey)

bOK (Boolean) Optional – Returns True if the key was successfully sent to the host session.

sKey (String) Valid characters are: A-Z, space, [, /,], ~, ?, 0-9

Example:

```
Dim bOK As Boolean
bOK = SM.SendCTRL("C")
```

SendCTRLAlt

This function sends the <CTRL> code with the specified character to a specific location in host session's screen. It returns True if the key was successfully entered. Reasons for a negative or False value might be that keyboard entry is inhibited, or that the coordinates were not an input field. Note: the ability to move the cursor to a specific location is only supported for tn5250 and tn3270 sessions.

Syntax: [bOK =] SM.SendCTRLAlt(iCol, iRow, sKey)

bOK (Boolean) Optional – Returns True if the key was successfully sent to the host session.

iCol (Integer) The screen column position to input the key.

iRow (Integer) The screen row position to input the key.

sKey (String) Valid characters are: A-Z, space, [, /,], ~, ?, 0-9

Example:

```
Dim bOK As Boolean
bOK = SM.SendCTRLAlt(8, 12, "C")
```

SendKey

This function sends the selected key to the current cursor location in the host session's screen. It returns True if the key was successfully entered. Note: reasons for a negative or False value might be that keyboard entry is inhibited, or that the cursor was not in an input field.

Syntax: [bOK =] SM.SendKey(enKey)

bOK (Boolean) Optional – Returns True if the key was successfully sent to the host session.

enKey (enCommandKeys) Select from the drop-down list the desired key to send to the host session. Options are:

KeyAttention	KeyBackspace
KeyBackTab	KeyBreak
KeyClear	KeyCursorDown
KeyCursorLeft	KeyCursorRight
KeyCursorUp	KeyDelete
KeyDo	KeyDuplicate
KeyEnd	KeyEnter
KeyEscape	KeyF1 – KeyF24
KeyFieldAdvance	KeyFieldBack
KeyFieldErase	KeyFieldExit
KeyFieldMark	KeyFieldMinus
KeyFieldPlus	KeyFind
KeyHelp	KeyHome
KeyInsert	KeyPA1 – KeyPA3
KeyPageDown	KeyPageUp
KeyRemove	KeyReplace
KeyReset	KeySelect
KeySystemRequest	KeyTab

Example:

```
Dim bOK As Boolean

bOK = SM.SendKey(KeyEnter)
```

SendKeyAlt

This function sends the selected key to a specific location in host session's screen. It returns True if the key was successfully entered. Reasons for a negative or False value might be that keyboard entry is inhibited, or that the coordinates were not an input field. Note: the ability to move the cursor to a specific location is only supported for tn5250 and tn3270 sessions.

Syntax: [bOK =] SM.SendKeyAlt(iCol, iRow, enKey)

bOK (Boolean) Optional – Returns True if the key was successfully sent to the host session.

iCol (Integer) The screen column position to input the key.

iRow (Integer) The screen row position to input the key.

enKey (enCommandKeys) Select from the drop-down list the desired key to send to the host session. For a list see SM.SendKey.

Example:

```
Dim bOK As Boolean

bOK = SM.SendKeyAlt(8, 12, KeyEnter)
```

SendText

This function is used to send text to the current cursor location in the host session's screen. It returns True if the text was successfully entered. Note: reasons for a negative or False value might be that keyboard entry is inhibited, that the cursor was not in an input field, or that the text exceeded the input field's length.

Syntax: [bOK =] SM.SendText(sText, [iPadSize], [vPadChar])

bOK (Boolean) Optional – Returns True if the text was successfully sent to the host session.

sText (String) The desired text to send to the host session.

iPadSize (Integer) Optional – total length of padded string

vPadChar (Variant) Optional – the character to use for padding

Examples:

```
Dim bOK As Boolean

bOK = SM.SendText("SAM")

SM.SendText("SAM", 10, " ")
```

SendTextAlt

This function is used to send text to a specific cursor location in the host session's screen. It returns True if the text was successfully entered. Reasons for a negative or False value might be that keyboard entry is inhibited, that the coordinates were not an input field, or that the text exceeded the input field's length. Note: the ability to move the cursor to a specific location is only supported for tn5250 and tn3270 sessions.

Syntax: [bOK =] SM.SendTextAlt(iCol, iRow, sText, [iPadSize], [vPadChar])

bOK (Boolean) Optional – Returns True if the text was successfully sent to the host session.

iCol (Integer) The screen column position to input the text.

iRow (Integer) The screen row position to input the text.

sText (String) The desired text to send to the host session.

iPadSize (Integer) Optional – total length of padded string

vPadChar (Variant) Optional – the character to use for padding

Example:

```
Dim bOK As Boolean
bOK = SM.SendTextAlt(10, 8, "SAM")
```

SessionID

This function is used return the current session or pool number. If Connection Pooling is disabled, the Transaction Manager may have to establish its own connection if macros are queued. If this is the case the returned integer will be 0. If Connection Pooling is enabled, the pool number (1, 2, 3 etc.) will be returned.

Syntax: iValue = SM.SessionID

iValue (Integer) stores the pool number used by the client

Example:

```
Dim iValue as Integer
iValue = SM.SessionID
```

SessionPwd

This function is used to set or return the Password associated with a host session. For pooled connections the password can be defined under Connections / Connection X (Screen Mapping) / Pooling option. For non-pooled connections the SM.SessionPwd can assign the password.

Syntax: sValue = SM.SessionPwd

Alternate: SM.SessionPwd = sValue

sValue (String) variable containing the password

Example:

```
Dim sValue as String
sValue = SM.SessionPwd
```

SessionUser

This function is used to set or return the User ID associated with a host session. For pooled connections the user can be defined under the Connections / Connection X (Screen Mapping) / Connection Pooling option.

For non-pooled connections the SM.SessionUser can assign the user ID.

Syntax: sValue = SM.SessionUser

Alternate: SM.SessionUser = sValue

sValue (String) variable containing the user ID

Example:

```
Dim sValue As String
sValue = SM.SessionUser
```

SetBase

This function is used to switch to an alternate Main Menu. This command must be run while on a menu screen and not an application screen.

Syntax: [bOK =] SM.SetBase(sMenu)

bOK (Boolean) Optional – returns True / False depending on the success of the command

sMenu (String) the name of the new base menu

Example:

```
Dim bOK As Boolean
bOK = SM.SetBase("MainMenu")
```

SetCursor

This function is used to move the cursor to a specified location on the host screen. It returns True if the cursor was successfully moved to the requested location. Note: the ability to move the cursor to a specific location is only supported for tn5250 and tn3270 sessions.

Syntax: [bOK =] SM.SetCursor(iCol, iRow)

bOK (Boolean) Optional – Returns True if the cursor was successfully moved in the host session.

iCol (Integer) The desired new column position for the cursor.

iRow (Integer) The desired new row position for the cursor.

Example:

```
Dim bOK As Boolean
bOK = SM.SetCursor(2, 9)
```

SetDelay

This function is typically used for debugging purposes against a vt220 host session. It allows users to set a delay, in milliseconds, for all screen mapping VBA extensions that change what is on the host screen. This allows the user to watch in slow motion, all screen updates as the result of a macro. Note: the user could also accomplish this by single stepping through the macro's VBA code in the test environments VBA debug window.

Syntax: SM.SetDelay(nMilliSeconds)

nMilliSeconds (Long) The delay to set in milliseconds.

Example:

```
SM.SetDelay(1000)
```

SetSession

This function is used when you are connecting to multiple hosts via Screen Mapping. It allows you to specify which screen mapping session is active. Note: the first screen mapping connection is set as the active session by default.

Syntax: [bOK =] SM.SetSession(vSource)

bOK (Boolean) Optional – Returns True if the requested session is defined as a valid Screen Mapping Connection.

vSource (Variant) The connection number or the name of the host session (as displayed in the status bar).

Example:

```
Dim bOK As Boolean
bOK = SM.SetSession(2)
```

SetTimeout

This function is used with all other SM commands, providing a maximum length of time before the other SM commands fail after no response from the host system. The internal default is 5 seconds.

Syntax: SM.SetTimeout(nSeconds)

nSeconds (Long) The number of seconds

Example:

```
SM.SetTimeout(10)
```

WaitForCursor

This function is used to time your commands to the host session. With it you can delay sending text or keys to the host session or retrieve data from the host session until the cursor is in a specific location. It basically allows you to wait for the cursor to arrive at a specific position in the host screen for a certain amount of time. Once the cursor reaches the desired location this function will immediately return with a value of True, otherwise it will timeout and return False.

Syntax: [bOK =] SM.WaitForCursor(iCol, iRow, iSeconds)

bOK (Boolean) Optional – Immediately returns True if the cursor stopped at the desired location.

iCol (Integer) The column position to wait for cursor.

iRow (Integer) The row position to wait for cursor.

iSeconds (Integer) The maximum number of seconds to wait for the cursor to reach the requested position.

Example:

```
Dim bOK As Boolean

bOK = SM.WaitForCursor(6, 14, 5)
```

WaitForCursorMove

This function is also used to time your commands to the host session. With this command, you specify only an amount of time in seconds. If the cursor has changed positions within that time, a True result is returned. Otherwise it will timeout and return False.

Syntax: [bOK =] SM.WaitForCursorMove(iSeconds)

bOK (Boolean) Optional – Immediately returns True if the cursor has changed locations.

iSeconds (Integer) – The maximum number of seconds to wait for the cursor to change locations.

Example:

```
Dim bOK As Boolean

bOK = SM.WaitForCursorMove(5)
```

WaitForHost

This function is used to time your commands to a vt220 host session. With it you can delay sending text or keys to the host session or retrieve data from the host session until the host has responded to the last

command sent. Whenever input data is sent to a host session, the server sets a switch that indicates whether the host has responded. This function basically allows you to wait for a certain amount of time until the host processes and responds to the last input command. Once the host response has been received and processed by the server this function will immediately return with a value of True, otherwise it will timeout and return False.

Note: There is a difference with the VT environment where this command will return a 'True' after the last command finishes even if the host is not ready for the keyboard input. This is because in the VT environment, the keyboard is never locked and in the 5250 and 3270 environments, this commands waits for the keyboard to be unlocked before returning any values.

Syntax: [bReplyReceived =] SM.WaitForHost(iSeconds)

bReplyReceived (Boolean) Optional – Immediately returns True once the host responds to your last input command.

iSeconds (Integer) The maximum number of seconds to wait for the host to respond to your last input command.

Example:

```
Dim bReplyReceived As Boolean

bReplyReceived = SM.WaitForHost(5)
```

WaitForScreen

This function is used to confirm that we have reached the desired host application screen. With it you can positively confirm that the 'GoToScreen' or 'ReturnToMainMenu' functions have succeeded. It allows you to wait for a certain amount of time for the desired screen to be positively identified. Once the ID match has occurred, this function will immediately return with a value of True, otherwise it will timeout and return False.

Syntax: [bOK =] SM.WaitForScreen(sScreenName, iSeconds)

bOK (Boolean) Optional – Immediately returns True if the host session is now in the desired screen.

sScreenName (String) The host screen that we wish to confirm is active.

iSeconds (Integer) The maximum number of seconds to wait for the screen to be identified.

Example:

```
Dim bOK As Boolean

bOK = SM.WaitForScreen("InventoryMovements", 5)
```

WaitForText

This function is used to time your commands to the host session. With it you can delay sending text or keys to the host session or retrieve data from the host session until a specific text string has appeared on the screen. It allows you to wait for a text string to appear anywhere on the screen, or only at a specific position. Once the text appears, this function will immediately return with a value of True, otherwise it will timeout and return False.

Syntax: [bFound =] SM.WaitForText(sText, iSeconds, [iCol], [iRow])

bFound (Boolean) Optional – Immediately returns True once the specified text appears on the screen.

sText (String) The text to find.

iSeconds (Integer) The maximum number of seconds to wait for the screen to be identified.

iCol (Integer) Optional – the column position to look for the text. Use '-1' for any Column

iRow (Integer) Optional – the row position to look for the text. Use '-1' for any row.

Example:

```
Dim bFound As Boolean

bFound = SM.WaitForText("Enter Next Task Code:", 5, -1, -1)
```

WaitForWrite

This function waits for a specified number of seconds for data to be entered at a specific location and returns a True or False. If data was written within the wait time, True is returned. If the number of seconds expires first, False is returned.

Syntax: [bOK =] SM.WaitForWrite(iCol, iRow, iSeconds)

bOK (Boolean) Optional – Returns True if data was written at the specified coordinates within the specified time frame

iCol (Integer) The column position to watch.

iRow (Integer) The row position to watch.

iSeconds (Integer) The maximum number of seconds to wait for data to be written.

Example:

```
Dim bOK As Boolean

bOK = SM.WaitForWrite(24,13,5)
```

SearchList Object

A SearchList is a full screen display of selected items; i.e., the data entry device screen is cleared and the contents of the list are displayed. A ListBox is different in that it displays selected items in the original prompt space allocated for the data in the application.

Alternatively, a SearchList can also be designed to display selected items in a "freestyle format". For example, the data entry device screen is replaced by a series of panels as opposed to a traditional table format.

Example

For a basic example, see Example Searchlist with a Button (in the textbox).

AddItem

This method adds an item to the list.

Group: SearchList Object

Syntax: oList.AddItem(vSelValue, vDispCols)

vSelValue (Variant) The value to be returned when this item is chosen. If the user adds the pipe symbol to the selection value variable, it will cause additional columns to be created in the display.

vDispCols (param array of Variants) a comma separated list of values to be displayed in each column

Example:

```
Dim oMyList as New SearchList
oMyList.AddItem(123, "1st Col Description", "2nd Col Description")
```

Supported Versions: RFgen 5.0 and higher.

BindControl

The method connects a control within a panel to a column number. See also *SetBind*.

Group: SearchList Object

Syntax: oMyList.BindControl(vControl, vColumn)

vControl (String) The value to be returned when this is chosen.

vColumn (String) The value to be returned when this is chosen.

Example:

```
Dim oMyList As New SearchList
oList.SQL = "select * from RFgen.cyclecounts"
' SetBind() connects the SearchList to a panel within a form
oMyList.SetBind("SearchListTest", "Panel1")
' BindControl() connects a control within the panel to a column number
oMyList.BindControl("Label1", 1)
oMyList.BindControl("Label2", 2)
oMyList.BindControl("Label3", 3)
oMyList.BindControl("Label4", 4)
oMyList.ShowList
```

Supported Versions: RFgen 5.1, removed in 5.2.

Cell

This method is used to read or change values or properties of a specific cell within the SearchList object.

Group: SearchList Object

Syntax:oMyList.Cell(Row, Col).<method or property>

Row (Long) specifies the row number in the grid

Col (Long) specifies the column number in the grid

Example:

```
oMyList.Cell(2, 2).Value = "1969"
```

Supported Versions: RFgen 5.0 and higher.

Cell(x,y).BackColor1

This method is used to read or change the primary background color of a specific cell within the SearchList object.

Group: SearchList Object

Syntax: oMyList.Cell(x,y).BackColor1 = vValue

Alternate: Sets the look and feel for alternate rows. lValue = oMyList.Cell(x,y).BackColor1

lValue (Long) is the color.

vValue (Variant) sets the color.

Example:

```
oMyList.Cell(2, 2).BackColor1 = RGB(255,255,0) 'yellow
oMyList.Cell(2, 2).BackColor1 = &HFF0000  'blue
oMyList.Cell(2, 2).BackColor1 = QBColor(5) 'magenta
oMyList.Cell(2, 2).BackColor1 = vbWhite
```

Supported Versions: RFgen 5.0 and higher.

Cell(x,y).BackColor2

This method is used to read or change the secondary background color of a specific cell within the SearchList object. It is used to produce gradients from one color to another.

Group: SearchList Object

Syntax: oMyList.Cell(x,y).BackColor2 = vValue

Alternate: Sets the look and feel for alternate rows. lValue = oMyList.Cell(x,y).BackColor2

lValue (Long) is the color.

vValue (Variant) sets the color.

Example:

```
oMyList.Cell(2, 2).BackGradient = GradientVertical
oMyList.Cell(2, 2).BackColor2 = RGB(255,255,0) 'yellow
oMyList.Cell(2, 2).BackColor2 = &HFF0000  'blue
oMyList.Cell(2, 2).BackColor2 = QBColor(5  'magenta
oMyList.Cell(2, 2).BackColor2 = vbWhite
```

Supported Versions: RFgen 5.0 and higher.

Cell(x,y).BackGradient

This property uses either just the BackColor(1) for solid backgrounds or both BackColor properties to create gradients in one of several directions for a specific cell within the SearchList object.

Group: SearchList Object

Syntax: oMyList.Cell(x,y).BackGradient = enValue

enValue (enGradients) enumeration that contains GradientDefault, GradientDiagonalLeft, GradientDiagonalRight, GradientHorizontal, GradientNone, GradientVertical and GradientVerticalSplit

Examples:

```
oMyList.Cell(2, 2).BackColor1 = RGB(0,0,255)
oMyList.Cell(2, 2).BackColor2 = vbWhite
oMyList.Cell(2, 2).BackGradient = GradientVertical
```

Supported Versions: RFgen 5.0 and higher.

Cell(x,y).Bold

This property accesses the prompt's data field bold option for a specific cell within the SearchList object.

Group: SearchList Object

Syntax: oMyList.Cell(x,y).Bold = bValue

Alternate: bValue = oMyList.Cell(x,y).Bold

bValue (Boolean) is True or False for bold or not bold.

Examples:

```
oMyList.Cell(2, 2).Bold = True
```

Supported Versions: RFgen 5.0 and higher.

Cell(x,y).ForeColor

This method is used to read or change the forecolor of a specific cell within the SearchList object.

Group: SearchList Object

Syntax: oMyList.Cell(x,y).ForeColor = vValue

Alternate: Sets the look and feel for alternate rows. lValue = oMyList.Cell(x,y).ForeColor

lValue (Long) is the color.

vValue (Variant) sets the color.

Example:

```
oMyList.Cell(2, 2).ForeColor = RGB(255,255,0) 'yellow
oMyList.Cell(2, 2).ForeColor = &HFF0000     'blue
```

```
oMyList.Cell(2, 2).ForeColor = QBColor(5)   'magenta
oMyList.Cell(2, 2).ForeColor = vbWhite
```

Supported Versions: RFgen 5.0 and higher.

Cell(x,y).Italic

This property accesses the prompt's data field italic option for a specific cell within the SearchList object.

Group: SearchList Object

Syntax: oMyList.Cell(x,y).Italic = bValue

Alternate: bValue = oMyList.Cell(x,y).Italic

bValue (Boolean) is True or False for italic or not italic.

Examples:

```
oMyList.Cell(2, 2).Italic = True
```

Supported Versions: RFgen 5.0 and higher.

Cell(x,y).Underline

This property accesses the prompt's data field underline option for a specific cell within the SearchList object.

Group: SearchList Object

Syntax: oMyList.Cell(x,y).Underline = bValue

Alternate: bValue = oMyList.Cell(x,y).Underline

bValue (Boolean) is True or False for underline or not underline.

Examples:

```
oMyList.Cell(2, 2).Underline = True
```

Supported Versions: RFgen 5.0 and higher.

Cell(x,y).Value

This property accesses the prompt's data field value for a specific cell within the SearchList object.

Group: SearchList Object

Syntax: oMyList.Cell(x,y).Value = vValue

Alternate: vValue = oMyList.Cell(x,y).Value

vValue (Variant) is the value within the cell.

Examples:

oMyList.Cell(2, 2).Value = "1"

vValue = oMyList.Cell(2, 2).Value

Supported Versions: RFgen 5.0 and higher.

Clear

This method clears the list's contents and optionally clears the column formatting within the Searchlist object.

Group: SearchList Object

Syntax: oMyList.Clear([bClear])

bClear (Boolean) Optional – Set to True to also clear the format of the columns initially set using the SetColumn method. The default is False

Examples:

oMyList.Clear

oMyList.Clear(True)

Supported Versions: RFgen 5.0 and higher.

Column

This method is used to read or change properties of a specific column within the Searchlist object.

Group: SearchList Object

Syntax: oMyList.Column(Col).<method or property>

Col (Long) specifies the column number in the Searchlist object

Examples:

oMyList.Column(2).Width = 10

Supported Versions: RFgen 5.0 and higher.

Column(x).Align

This property aligns the text of the column within the Searchlist object. The options are: BottomCenter, BottomLeft, BottomRight, CenterCenter, CenterLeft, CenterRight, TextCenter, TextLeft, TextRight., TopCenter, TopLeft, TopRight

Group: SearchList Object

Syntax: oMyList.Column(x).Align = enValue

enValue (enColAlign) an enumeration that contains BottomCenter, BottomLeft, BottomRight, CenterCenter, CenterLeft, CenterRight, TextCenter, TextLeft, TextRight., TopCenter, TopLeft, TopRight

Examples:

oMyList.Column(2).Align = TextCenter

Supported Versions: RFgen 5.0 and higher.

Column(x).Autosize

This property will size the column based on the widest value in that column.

Group: SearchList Object

Syntax: oMyList.Column(x).Autosize = bValue

bValue (Boolean) set to True or False to autosize the width of the column.

Examples:

oMyList.Column(2).Autosize = True

Supported Versions: RFgen 5.0 and higher.

Column(x).BackColor1

This method is used to read or change the primary background color of the whole column within the Search-list object.

Group: SearchList Object

Syntax: oMyList.Column(x).BackColor1 = vValue

Alternate: lValue = oMyList.Column(x).BackColor1

lValue (Long) is the color.

vValue (Variant) sets the color.

Examples:

```
oMyList.Column(2).BackColor1 = RGB(255,255,0) 'yellow
oMyList.Column(2).BackColor1 = &HFF0000      'blue
oMyList.Column(2).BackColor1 = QBColor(5)    'magenta
oMyList.Column(2).BackColor1 = vbWhite
```

Supported Versions: RFgen 5.0 and higher.

Column(x).BackColor2

This method is used to read or change the secondary background color of the whole column within the Searchlist object. It is used to produce gradients from one color to another.

Group: SearchList Object

Syntax:oMyList.Column(x).BackColor2 = vValue

Alternate: lValue = oMyList.Column(x).BackColor2

lValue (Long) is the color.

vValue (Variant) sets the color.

Examples:

```
oMyList.Column(2).BackGradient = GradientVertical
oMyList.Column(2).BackColor2 = RGB(255,255,0) 'yellow
oMyList.Column(2).BackColor2 = &HFF0000      'blue
oMyList.Column(2).BackColor2 = QBColor(5)    'magenta
oMyList.Column(2).BackColor2 = vbWhite
```

Supported Versions: RFgen 5.0 and higher.

Column(x).BackGradient

This property uses either just the BackColor(1) for solid backgrounds or both BackColor properties to create gradients in one of several directions for the whole column within the Searchlist object.

Group: SearchList Object

Syntax: oMyList.Column(x).BackGradient = enValue

enValue (enGradients) enumeration that contains GradientDefault, GradientDiagonalLeft, GradientDiagonalRight, GradientHorizontal, GradientNone, GradientVertical and GradientVerticalSplit

Example:

```
oMyList.Column(2).BackColor1 = RGB(0,0,255)
oMyList.Column(2).BackColor2 = vbWhite
oMyList.Column(2).BackGradient = GradientVertical
```

Supported Versions: RFgen 5.0 and higher.

Column(x).Bold

This property accesses the column's bold option for the whole column within the Searchlist object.

Group: SearchList Object

Syntax: oMyList.Column(x).Bold = bValue

Alternate: bValue = oMyList.Column(x).Bold

bValue (Boolean) is True or False for bold or not bold.

Example:

oMyList.Column(2).Bold = True

Supported Versions: RFgen 5.0 and higher.

Column(x).Caption

This property accesses the caption associated with the specified column.

Syntax: oMyList.Column(x).Caption = vValue

Alternate: sValue = oMyList.Column(x).Caption

sValue (String) is the title of the column.

vValue (Variant) sets the title of the column.

Example:

oMyList.Column(2).Caption = "Model"

Column(x).DisplayOnly

This property accesses the display only property associated with the specified column. Setting it to True makes all the cells in that column unchangeable.

Syntax: oMyList.Column(x).DisplayOnly = bValue

Alternate: bValue = oMyList.Column(x).DisplayOnly

bValue (Boolean) is the display only state for the prompt.

Example:

oMyList.Column(2).DisplayOnly = True

Supported Versions: RFgen 5.0 and higher.

Column(x).Expanded

This property is intended for the Tree control only and is not to be used with the SearchList object.

Column(x).FontSize

This property accesses the font size of the entire search list display. The Default (set in the Theme) in 10.

Syntax: oMyList.Column(x).FontSize = lValue

Alternate: lValue = oMyList.Column(x).FontSize

lValue (Long) is the font size for the search list display

Example:

```
oMyList.Column(2).FontSize = 16
```

Supported Versions: RFgen 5.0 and higher.

Column(x).ForeColor

This property accesses the column's fore color property associated with the specified column.

Syntax: oMyList.Column(x).ForeColor = lValue

Alternate: vValue = oMyList.Column(x).ForeColor

vValue (Variant) is the color.

lValue (Long) sets the color.

Examples:

```
oMyList.Column(2).ForeColor = RGB(255,255,0) 'yellow
oMyList.Column(2).ForeColor = &HFF0000      'blue
oMyList.Column(2).ForeColor = QBColor(5)    'magenta
oMyList.Column(2).ForeColor = vbWhite
```

Supported Versions: RFgen 5.0 and higher.

Column(x).Format

This property affects the format of the whole specified column. It is only an extension of the Format VBA command.

Syntax: oMyList.Column(x).Format = sValue

Alternate: sValue = oMyList.Column(x).Format

sValue (String) is the format mask to use when displaying data for the prompt.

Examples:

oMyList.Column(2).Format = "hh:mm"

c - General Date

dddddd - Long Date

ddddd - Short Date

ttttt - Long Time

hh:mm AMPM - Medium Time

hh:mm - Short Time

$#,##0.00 or ($#,##0.00) - Currency 0.00 - Fixed

#,##0.00 - Standard 0.00% - Percent 0.00E+00 - Scientific

Yes/No - Return "No" if zero, else return "Yes"

True/False - Return "True" if zero, else return "False"

On/Off - Return "On" if zero, else return "Off"

For further examples get help on the VB FORMAT command.

Column(x).ImageHeight

This property sets the images in this column to a specific height.

Syntax: oMyList.Column(x).ImageHeight = lValue

Alternate: lValue = oMyList.Column(x).ImageHeight

lValue (Long) is the pixel height size for images in this column

Example:

oMyList.Column(1).ImageHeight = 5

Column(x).ImageWidth

This property sets the images in this column to a specific width.

Syntax: oMyList.Column(x).ImageWidth = lValue

Alternate: lValue = oMyList.Column(x).ImageWidth

lValue (Long) is the pixel width size for images in this column

Example:

oMyList.Column(1).ImageWidth = 5

Column(x).Indent

This property is intended for the Tree control only and is not to be used with the SearchList object.

Column(x).Italic

This property accesses the column's italic option for the whole column within the Searchlist object.

Syntax: oMyList.Column(x).Italic = bValue

Alternate: bValue = oMyList.Column(x).Italic

bValue (Boolean) is True or False for italic or not italic.

Example:

oMyList.Column(2).Italic = True

Column(x).MarginBottom

This property pads the bottom of all the cells in a specified column. It also contributes to the overall height of the entire row.

Syntax: oMyList.Column(x).MarginBottom = lValue

Alternate: lValue = oMyList.Column(x).MarginBottom

lValue (Long) the padding in pixels between the contents of a cell and the bottom of the cell.

Example:

oMyList.Column(1).MarginBottom = 2

Column(x).MarginLeft

This property pads the left of all the cells in a specified column.

Syntax: oMyList.Column(x).MarginLeft = lValue

Alternate: lValue = oMyList.Column(x).MarginLeft

lValue (Long) the padding in pixels between the contents of a cell and the left of the cell.

Example:

oMyList.Column(1).MarginLeft = 2

Column(x).MarginRight

This property pads the right of all the cells in a specified column.

Syntax: oMyList.Column(x).MarginRight = lValue

Alternate: lValue = oMyList.Column(x).MarginRight

lValue (Long) the padding in pixels between the contents of a cell and the right of the cell.

Example:

oMyList.Column(1).MarginRight = 2

Column(x).MarginTop

This property pads the top of all the cells in a specified column. It also contributes to the overall height of the entire row.

Syntax: oMyList.Column(x).MarginTop = lValue

Alternate: lValue = oMyList.Column(x).MarginTop

lValue (Long) the padding in pixels between the contents of a cell and the top of the cell.

Example:

oMyList.Column(1).MarginTop = 2

Column(x).ScaleDecimals

This property formats the numeric values in the specified column if the database does not store decimals. This option will position a decimal at a position from the right side. A comma will be used for large numbers. This field is locale specific and will use the appropriate characters for each region.

Syntax: oMyList.Column(x).ScaleDecimals = vValue

Alternate: vValue = oMyList.Column(x).ScaleDecimals

vValue (Variant) is the decimal position.

Example:

oMyList.Column(1).ScaleDecimals = 2

Column(x).Style

This property gets or sets the type of column. The values are Text, Image, and Check box.

Syntax: oMyList.Column(x).Style = enStyle

Alternate: enStyle = oMyList.Column(x).Style

enStyle (enColumnStyle) Contains ColumnStyleCheck, ColumnStyleImage, and ColumnStyleText

Examples:

oMyList.Column(1).Style = ColumnStyleCheck

oMyList.Column(1).Style = ColumnStyleText

Column(x).TrimSpaces

This property formats the values in the specified column by deleting the leading and trailing spaces in the data.

Syntax: oMyList.Column(x).TrimSpaces = bValue

Alternate: bValue = oMyList.Column(x).TrimSpaces

bValue (Boolean) set to True to trim all space from the data.

Example:

oMyList.Column(1).TrimSpaces = True

Column(x).Underline

This property accesses the column's underline option for the whole column within the Searchlist object.

Syntax: oMyList.Column(x).Underline = bValue

Alternate: bValue = oMyList.Column(x).Underline

bValue (Boolean) is True or False for underline or not underline.

Example:

oMyList.Column(2).Underline = True

Column(x).Visible

This property makes visible or invisible the whole column within the Searchlist object.

Syntax: oMyList.Column(x).Visible = bValue

Alternate: bValue = oMyList.Column(x).Visible

bValue (Boolean) is True or False for column visibility.

Example:

oMyList.Column(2).Visible = True

Column(x).Width

This property sets or returns the width of the column within the Searchlist object.

Syntax: oMyList.Column(x).Width = vValue

Alternate: vValue = oMyList.Column(x).Visible

vValue (Variant) sets or gets the width of the specified column.

Examples:

oMyList.Column(2).Width = 25

Columns

This property returns the number of columns in the Searchlist object.

Group: SearchList Object

Syntax:

vValue = oMyList.Columns

vValue (Variant) gets the number of columns in the Searchlist object.

Examples:

```
Dim iCnt As Integer
iCnt = oMyList.Columns
```

Supported Versions: RFgen 5.0 and higher.

Count

This function returns the number of items in the Searchlist object.

Syntax: vValue = oMyList.Count

vValue (Variant) the number of items in the list

Example:

Dim nValue as Long

nValue = oMyList.Count

Index

This property returns or sets the current list index property.

Group: SearchList

Syntax: oMyList.Index = lValue

Alternate: vValue = oMyList.Index

lValue (Long) sets the item index in the list

vValue (Variant) gets the item index in the list

Example:

```
Dim nValue As Long
nValue = oMyList.Index
oMyList.Index = 5
```

Supported Versions: RFgen 5.0 and higher.

List

The SearchList object can be used to generate the formatted list and then output it to a prompt's list property (like a Listbox or Combobox) or be used in methods like App.ShowList.

Syntax: sList = oMyList.List

sList (String) contains the formatted array of values for use in other methods

Examples:

```
oMyList.MaxRows = 100
oMyList.ReturnAllRows = False
oMyList.ShowEmptyList = False
oMyList.SQL = "select * from PLANTS"
oMyList.SetColumn 1, "ID", 5, TextLeft,True
oMyList.SetColumn 2, "NAME", 21, TextLeft, True

cboPlants.List.Data = oMyList.List
```

or

```
Rsp = App.ShowList(oMyList.List)
```

Supported Versions: RFgen 5.2 and higher.

MaxRows

This property limits how many rows will be allowed in the list. If the database will return several thousand records, you may want to limit this list to 500. An alternative is to further restrict the search criteria if using SQL.

Syntax: oMyList.MaxRows = nNum

Alternate: nNum = oMyList.MaxRows

nNum (Long) is a numeric value to limit the rows in the list.

Example:

```
oMyList.MaxRows = 100
```

Supported Versions: RFgen 5.0 and higher.

Normalize

This property, when set to True, will trim the preceding and trailing spaces from the column data in the list. This should only be necessary if the database stores space-padded data values.

Syntax: oMyList.Normalize = bVal

Alternate: bVal = oMyList.Normalize

bVal (Boolean) set to True to trim all data in the list

Examples:

Dim bValue As Boolean

```
oMyList.Normalize = True
```

bValue = oMyList.Normalize

ReturnAllRows

This property when set to True instructs the list to return all the values for all the columns instead of the first value of the first column when a row is selected.

Syntax: oMyList.ReturnAllRows = bValue

Alternate: bValue = oMyList.ReturnAllRows

bValue (Boolean) a True or False value *(default = False)*

Examples:

```
Dim bValue As Boolean
```

```
bValue = oMyList.ReturnAllRows
oMyList.ReturnAllRows = True
```

Supported Versions: RFgen 5.0 and higher.

SetBind

This method connects the SearchList object to a panel within a form. See also, *BindControl*.

Syntax: List.Set(vForm, vControl)

vForm (String) The value to be returned when this is chosen.

vControl (String) The value to be returned when this control is chosen.

Example:

```
Dim oMyList As New SearchList
oList.SQL = "select * from RFgen.cyclecounts"

' SetBind() connects the SearchList to a panel within a form
oMyList.SetBind("SearchListTest", "Panel1")

' BindControl() connects a control within the panel to a column number
oMyList.BindControl("Label1", 1)
oMyList.BindControl("Label2", 2)
oMyList.BindControl("Label3", 3)
oMyList.BindControl("Label4", 4)

oMyList.ShowList
```

SetColumn

This method formats the returned data to fit the device's screen and desired layout. This statement should be used for each column to be displayed.

Group: SeachList

Syntax: oMyList.SetColumn(nColumn, sHeading, vDefWidth, [enAlign], [bTrimSpaces], [nDecimals])

nColumn	(Long) the column number to be affected by the formatting
sHeading	(String) the title that will appear across the top of the column
vDefWidth	(Long) defines the width of the column in pixels
enAlign	(enColAlign) Optional – how to align the column; either Left, Right or Center; the default is Left justified

bTrimSpaces (Boolean) Optional – True means that leading and trailing spaces will be removed from the display of the data in this column

nDecimals (Long) Optional – if the value is numeric and the database does not store decimals, use this to position a decimal at a position from the right side. A comma will be used for large numbers. This field is locale specific and will use the appropriate characters for each region.

Example:

```
Dim oMyList as New SearchList
oMyList.oList.MaxRows = 100
oMyList.ReturnAllRows = False
oMyList.ShowEmptyList = False
oMyList.Normalize = True
oMyList.SQL = "select * from PLANTS"
oMyList.SetColumn 1, "ID", 5, TextLeft, True
oMyList.SetColumn 2, "NAME", 21, TextLeft, True
sName = oMyList.ShowList
```

Supported Versions: RFgen 5.0 and higher.

ShowEmptyList

This property displays the blank list to the user even when there are no entries. The whole screen will be temporarily cleared and the user must press enter to acknowledge there were no entries.

Group: SearchList

Syntax: oMyList.ShowEmptyList = bValue

Alternate: bValue = oMyList.ShowEmptyList

bValue (Boolean) is either True or False.

Example:

```
oMyList.ShowEmptyList = True
```

Supported Versions: RFgen 5.0 and higher.

ShowList

This function displays the list to the user. The whole screen will be temporarily cleared and the list will be displayed until a choice is made.

Syntax: sValue = oMyList.ShowList

sValue (String) is a Chr(1) delimited list of the values in the columns based on the row selected.

Example:

```
Dim sValue as String
sValue = oMyList.ShowList
```

SQL

This property will contain the SQL statement to be used in creating the list.

Group: SearchList

Syntax: oMyList.SQL = sSQL

Alternate: sSQL = oMyList.SQL

sSQL (String) the SQL statement to be executed

Example:

```
Dim sSQL As String
sSQL = "Select * from ItemMaster"
oMyList.SQL = sSQL
```

Supported Versions: RFgen 5.0 and higher.

Value

This property returns the value in the specified row for the Searchlist.

Group: SearchList

Syntax: vValue = oMyList.Value(Row)

Row (Long) is the row number in the list

vValue (Variant) is given the value assigned to the row

Examples:

```
Dim vValue As Variant
vValue = oMyList.Value(1)
```

Supported Versions: RFgen 5.0 and higher.

Server-Based Extensions

Server-based commands are used by the mobile device when not in a Thin-client state to send and receive data to and from the server.

The extensions available for the server extensions are:

Server.CallMacro, Server.CommandTimeout, Server.Connect

Server.Disconnect, Server.ExecuteSQL, Server.GetTable

Server.GeoGetAddress, Server.GeoGetDirections, Server.GeoGetLatLng, Server.GeoGetMap

Server.IsConnected, Server.MakeList, Server.Ping

Server.QueueMacro, Server.ReadFile, Server.SendQueue

Server.SendTable, Server.SetCredentials, Server.SetHost

Server.SetVPN, Server.SetWAN, Server.ShowProgress

Server.SyncApps, Server.SyncAppsEx, Server.WriteFile

CallMacro

This function is used to call a screen mapping or transaction macro and pass the macros required parameters. Using this function gives you the ability to "store-and-forward" transactions while the host is off-line for backup or other reasons. It returns 'True' if the macro was successfully completed.

Group: Server Extensions

Syntax: [enOK =] Server.CallMacro(sMacroName, bQueueOffline, [vParams])

enOK (enMacroResults) Optional – Returns 1 of the following 4 values:

MacroFailed
MacroNotProcessed
MacroQueued
MacroSucceeded

sMacroName (String) – This is the name of the macro to be called.

bQueueOffline (Boolean) – This determines whether the macro can be queued for later processing if the host is not currently available.

vParams (Variant) – Optional: A series of passing parameters as required by the selected macro. Note: these parameters are the fields you defined when you record the macro.

Example:

```
Dim bOK As enMacroResults
```

```
bOK = Server.CallMacro("PICK", True, "Sam", "12")
```

Supported Versions: RFgen 4.0 and newer.

CommandTimeout

This command limits the number of seconds any Server command waits for a response from the server. All Server commands will return with a response or failure message no later than the specified number of seconds. Set the parameter to zero (0) to disable this feature and go back to the default for each command.

Group: Server Extension

Syntax:Server.CommandTimeout(nTimeout)

nTimeout (Long) parameter to specify a number of seconds

Example:

Server.CommandTimeout(10)

Supported Versions: RFgen 4.0 and newer.

Connect

This command connects to the remote server via TCP/IP. Default values are stored in the registry when the files are installed based on the profile.

Group: Server Extensions

Syntax: [bOK =] Server.Connect

bOK (Boolean) Optional – return a True if the mobile device is able to make a connection to the host within 30 seconds or less.

Example:

```
Dim bOK as Boolean
bOK = Server.Connect
```

Version Supported: RFgen 4.0 and newer.

Disconnect

This command disconnects from the remote server. You should always use this command when you know the connection is no longer needed. Using the Form_Unload event may be a good place.

Group: Server Extensions

Syntax: [bOK =] Server.Disconnect

bOK (Boolean) Optional – return a True if the mobile device disconnects from the host.

Example:

```
Dim bOK as Boolean
bOK = Server.Disconnect
```

Supported Versions: RFgen 5.1 and newer.

ExecuteSQL

This function will execute a pass-through SQL statement on the host through the connection maintained by the server. Any results from the statement will be returned in a text-delimited object. (Note: this means that the item is a static view of the database and cannot be updated.)

Group: Server Extensions

Syntax: [bOK =] Server.ExecuteSQL(sSQL, [vColumns], [vRows])

bOK (Boolean) Optional – is a return value; a value of True means the SQL statement processed normally.

sSQL (String) is the SQL statement to be sent to the database. As this is a pass through statement, its syntax must be understood by the database, as no pre-processing will occur.

vColumns (Variant) – Optional – is a string representation of the columns returned by the SQL statement.

vRows (Variant) – Optional – is a string representation of the static result of an SQL statement.

Example:

```
Dim bOK As Boolean
Dim sSQL As String
Dim sCols As String
Dim sRows As String
sSQL = "select * from Inventory"
bOK = Server.ExecuteSQL(sSQL, sCols, sRows)
In the case of an insert or an update, the sCols and sRows variables would not be necessary.
```

Supported Versions: RFgen 4.0 and newer.

GeoGetAddress

This method returns an address converted from the longitude and latitude geo-coordinates. This extension only works with the Map control.

Group: Server Extensions

Syntax:

bValue = PromptID.Server.GeoGetAddress(sLat, sLng, sAddrs)

bValue: (Boolean) True returns a value if successful; False if it fails

sLat: (String) is the Latitude of the address

sLng: (String) is the Longitude of the address

sAddrs: (String) is the address converted from the Latitude and Longitude

Example:

```
sLat = "38.575764"
sLng = "121.478851"
mMapArea.Server.GeoGetAddress(sLat,sLng,sAddrs)
```

Version Supported: RFgen 5.1 and newer.

GeoGetDirections

This method to display the driving directions between a start and destination. This extension is used with the Map control.

Group: Server Extensions

Syntax:

bVal = promptID.Server.GetDirections(sOrg, sDst, sRoads, [Map], [Duration], [Dist])

bVal: (Boolean) Returns True if the directions were retrieved; False if it failed.

sOrg: (String) is the origination or start of the route

sDst: (String) is the route destination

sRoads: (String Array) is the list of roads between origination and destination

[Map]: Optional - is an image of the directions

[Dur]: Optional - is the duration of the distance traveled by car

[Dist]: Optional – is the distance traveled

Example:

```
DIM sDir as String
server.GeoGetDirections(txtOrig.Text, txtDest.Text, sRoads, True)
```

Version Supported: RFgen 5.1 and newer.

GeoGetLatLng

Use this method to return the Latitude and Longitude of a physical address. This extension is used with the Map control.

Group: Server Extensions

Syntax:

MapControlPromptID.map.

bVal = Server.GeoGetLatLng (sAddr, sLat, sLng)

bValue (Boolean) True means the conversion was retrieved; False means it failed.

sAddr: (String) is the address of the location

sLat: (String) is the latitude of the location

sLng: (String) is the longitude of the location

Example:

```
Server.GeoGetLatLng(txtAddress.Text, textLat.Text, textLng.Text, True)
```

Version Supported: RFgen 5.1 and newer.

GeoGetMap

This method returns a map representing the last Geo command from Google maps. For instance, if you execute GeoGetDirections, GeoGetMap will get a bitmap image representing those directions from Google Maps. This extension is only used with the Map control.

Group: Prompt-Specific Extensions

Syntax:

promptID.GeoGetMap

Example:

Server.GeoGetMap( )

Supported Versions: RFgen 5.2 and newer.

GeoPlanRoute

This method returns the best route when there are multiple stops.

Use this method to obtain the most efficient driving route for multiple locations. This server extension is used with the Map control.

Group: Server Extensions

Syntax:

bVal = Server.GeoPlanRoute(sOrg, sDst, sPoints,[Map])

bVal: (Boolean) True means the route was retrieved; False means it failed

sOrg: (String) is the origination or start of the planned route

sDst: (String) is the destination of the planned route

sPoints: (String Array) Points are an array of delivery addresses and once its returned, the points will contain the sorted addresses.

Map: (Boolean) Optional - returns True if the map from Google Maps was retrieved; False if it failed.

Example:

```
DIM sPlanRoute as String

server.GeoPlnRoute (txtOrig.Text, txtDest.Text, sPoints(), Map, True)
```

Version Supported: RFgen 5.1 and newer.

GetTable

This command executes the SQL statement on the remote server and copies the results to an existing local table.

Group: Server Extensions

Syntax: [bOK =] Server.GetTable(sSQL, sLocalTable, [vKeyFields])

bOK (Boolean) Optional – is a return value; a value of True means the SQL statement processed normally.

sSQL (String) is the SQL statement to be sent to the database. As this is a pass through statement, its syntax must be understood by the database, as no pre-processing will occur.

sLocalTable (String) is the name of the table already created on the mobile device.

vKeyFields (Variant) Optional – is 1 or more key fields to represent a unique record. One key would be represented by "PartNo" and multiple keys look like "PartNo,Onhand".

Example:

```
Dim bOK as Boolean

bOK = Server.GetTable("Select * from Inventory", "Inv2", "PartNo")
```

Note: In the event that you need to update data using the Key field you should pay close attention to how the table has been populated. To optimize update performance you should try to match the order in which the data was populated in the table initially. For instance assume table INVENTORY has a unique ID column. Also assume that you will need to update a small subset or rows on the device database using the Server.GetTable() language extension. You would want to initially populate the table using the GetTable function with a SQL statement similar to:

```
Server.GetTable("SELECT * FROM INVENTORY ORDER BY Id", "INVENTORY")
```

To optimize performance while updating records you would want to match the data that way it was downloaded by using the ORDER BY clause in your update like:

```
Server.GetTable("SELECT Id, Column1, Column2 FROM INVENTORY WHERE LastWrite> '07/22/2013' ORDER BY Id",
"INVENTORY", "Id")
```

The ORDER BY clause will ensure that the records in the initial download and the updates are in the same order and will optimize the databases ability to seek to the next record to update.

Supported Versions: RFgen 4.0 and newer.

IsConnected

This command returns a Boolean value of True or False depending on if the Server.Connect command previously executed. If the connection was successful, the value returned is True. If the connection failed, the value returned is false.

Group: Server Extensions

Syntax: bOK = Server.IsConnected

bOK (Boolean) a value of True means there is already an open connection to use.

Example

```
Dim bOK as Boolean

bOK = Server.IsConnected
```

Supported Versions: RFgen 5.1 and newer.

MakeList

This command returns a dynamic array string containing the results of a SQL statement that was executed on the server from a mobile device. For this command to be successful the server must be in wireless range for a connection to be established.

Group: Server Extensions

Syntax: [bOK =] Server.MakeList(sSQL, sList, [bRtnAllCols], [bNormalize])

bOK (Boolean) Optional – is a return value; a value of True means the SQL statement processed normally.

sSQL (String) is the SQL 'SELECT' statement to be sent to the database.

sList (String) is the list to be returned for display (i.e., set RSP = Value to display the list on the Client device.)

bRtnAllCols (Boolean) Optional – when set to True will return all the columns as the potential key, not just the first column. Default is False.

bNormalize (Boolean) Optional – when set to True will trim the spaces from the data so that it will display consistently. Default is False.

Example:

```
Dim sSQL As String
Dim sMyList As String
sSQL = "select PartNo from ItemMaster"
sMyList = Server.MakeList(sSQL, True, True)
Rsp = App.ShowList(sMyList)
Or
Rsp = App.ShowList(Server.MakeList(sSQL, True, True))
```

Supported Versions: RFgen 4.0 and newer.

Ping

This command returns a True / False regarding the ability to reach the server. Based on the result you may choose to continue with the Server.Connect or go to a disconnected state. Default values are stored in the registry when the files are installed.

Group: Server Extensions

Syntax: bOK = Server.Ping([vServer])

bOK (Boolean) returns a True if the mobile device could connect to the server.

vServer (Variant) Optional – to specify the name or IP address of the server. If this is not specified, the registry settings will be used.

Example:

```
Dim bOK as Boolean
bOK = Server.Ping("192.168.123.45")
```

Supported Versions: RFgen 4.0 and newer.

QueueMacro

This function is used to queue any transaction macro and pass its required passing parameters directly to the server while in mobile mode. Unlike the TM.CallMacro with the queue property set to True, this command will not check for a valid host connection, or move to a linked screen. It returns 'True' if the macro was successfully queued. These macros can be created on the Transactions tree.

Group: Server Extensions

Syntax: [nSeq =] Server.QueueMacro(sMacroName, [vParams])

nSeq (Long) Optional – Returns the sequence number of the macro when it is successfully queued.

sMacroName (String) – This is the name of the macro to be called.

vParams (Variant) Optional – A series of passing parameters as required by the selected macro. Note: these parameters are the fields you defined when you wrote the macro.

Example:

```
Dim nSeq As Long
nSeq = Server.QueueMacro("ChangeUser", "Sam", "1234")
```

Supported Versions: RFgen 4.0 and newer.

ReadFile

This command will read data from a file on the server. String data or binary data can be read.

Group: Server Extensions

Syntax: [bOK =] Server.ReadFile(sFileName, vData)

bOK (Boolean) Optional – the success or failure of the command.

sFileName (String) specifies where on the server the file should be found

vData (Variant) contains the data read from the file. String or binary data is allowed.

Example:

```
Dim bOK As Boolean
Dim vData As Variant
Dim sFile As String
sFile = "C:\Program Files\MyFile.txt"
bOK = Server.ReadFile(sFile, vData)
```

Supported Versions: RFgen 4.0 and newer.

SendQueue

This function sends any locally queued transactions to the remote server queue for processing. Upon successfully transferring the local queue to the server, the local queue is cleared.

Group: Server Extensions

Syntax: [bOK =] Server.SendQueue([vDestQueue])

bOK (Boolean) Optional – Returns True if the queued macros were successfully sent to the host for processing.

vDestQueue (Variant) Optional – an option to send the queue to a queue with a different name

Example:

```
Dim bOK As Boolean
bOK = Server.SendQueue
```

Supported Versions: RFgen 4.0 and newer.

SendTable

This function executes the SQL statement locally and copies the results to an existing table on the remote server.

Group: Server Extensions

Syntax: [bOK =] Server.SendTable(sSQL, sRemoteTable, [vKeyFields])

bOK (Boolean) Optional – is a return value; a value of True means the SQL statement processed normally.

sSQL (String) is the SQL statement to be sent to the database. As this is a pass through statement, its syntax must be understood by the database, as no pre-processing will occur.

sRemoteTable (String) is the name of the table already created on the mobile device.

vKeyFields (Variant) Optional – is one or more key fields to represent a unique record. One key would be represented by "PartNo" and multiple keys look like "PartNo,Onhand".

Example:

```
Dim bOK as Boolean
bOK = Server.SendTable("Select * from Inv2", "Inventory", "PartNo")
```

Supported Versions: RFgen 4.0 and newer.

SetCredentials

This function sets or changes the Domain, User or Password values for connecting to the server's network when using the NTLM connection security authentication option.

Group: Server Extensions

Syntax: [bOK =] Server.SetCredentials([vDomain], [vUserID], [vPassword])

bOK	(Boolean) Optional – is a return value; a value of True means the method processed normally.
vDomain	(Variant) Optional – is the name of the domain where the credentials will be validated
vUserID	(Variant) Optional – is the name of the user
vPassword	(Variant) Optional – is the password of the user

Example:

```
Dim bOK as Boolean
bOK = Server.SetCredentials("RFGEN","MIKE","PASS")
```

Supported Versions: RFgen 5.0 and newer.

SetHost

This function sets the default settings for connecting to the host server. Typically Server.Connect follows this command.

Group: Server Extensions

Syntax: [bOK =] Server.SetHost([vServerName], [nPort])

bOK	(Boolean) Optional – is a return value; a value of True means the method processed normally.
vServerName	(Variant) Optional – is the name or IP address of the server
nPort	(Long) Optional – is the port that facilitates the data portion of the communication.

Example:

```
Dim bOK as Boolean
bOK = Server.SetHost("192.168.1.100", 21097)
```

Supported Versions: RFgen 4.0 and newer.

SetVPN

This function lets the user change the credentials used when using the operating system's VPN settings to establish server access.

Group: Server Extensions

Syntax: [bOK =] Server.SetVPN(bEnabled, [vVPNName], [vUserID], [vPassword])

bOK	(Boolean) Optional – is a return value; a value of True means the method processed normally.
bEnabled	(Boolean) enables or disables the configured VPN
vVPNName	(Variant) Optional – is the name of the VPN as configured in the operating system
vUserID	(Variant) Optional – is the user ID to be used for making that VPN connection if it is different that the already configured user ID in the VPN setup
vPassword	(Variant) Optional – is the password to be used for making that VPN connection if it is different that the already configured password in the VPN setup

Example:

```
Dim bOK as Boolean
bOK = Server.SetVPN(True, "MyISP")
```

Supported Versions: RFgen 5.0 and newer.

SetWAN

This function lets the user change the credentials used when using the operating system's GPRS (cellular) settings to establish internet access.

Group: Server Extensions

Syntax: [bOK =] Server.SetWAN(bEnabled, [vWANName], [vUserID], [vPassword])

bOK	(Boolean) Optional – is a return value; a value of True means the method processed normally.
bEnabled	(Boolean) enables or disables the configured WAN
vWANName	(Variant) Optional – is the name of the WAN as configured in the operating system
vUserID	(Variant) Optional – is the user ID to be used for making that WAN connection if it is different that the already configured user ID in the WAN setup

vPassword (Variant) Optional – is the password to be used for making that WAN connection if it is different that the already configured password in the WAN setup

Example:

```
Dim bOK as Boolean
bOK = Server.SetWAN(True, "MyWAN")
```

Supported Versions: RFgen 5.0 and newer.

ShowProgress

This command enables or disables a popup progress box for all Server commands. The elapsed number of seconds and the current activity are displayed.

Group: Server Extensions

Syntax: Server.ShowProgress(bShow)

bShow (Boolean) True or False for turning on or off the progress bar

Example:

```
Server.ShowProgress(True)
```

Supported Versions: RFgen 4.0 and newer.

SyncApps

This command will update a mobile device configured for mobile operation with changes made to application related items such as menus, users, applications, macros etc. If an administrator changes a device profile to include or exclude items, this command will compare what is on the device with what is in the profile and request any changed or missing items. Depending on the size of the change the Server.ShowProgress command may be useful.

Group: Server Extensions

Syntax: [bOK =] Server.SyncApps([vProfile])

bOK (Boolean) Optional – returns True or False for the success of the command

vProfile (Variant) Optional – is the name of the profile used to compare what is on the device with the server list of objects.

Example:

```
Dim bOK As Boolean
bOK = Server.SyncApps
Server.SyncApps("CEProfile1")
```

Supported Versions: RFgen 4.0 and newer.

SyncAppsEx

This command is similar to Server.SyncApps except that it includes a parameter to set an error message.

Group: Server Extensions

Syntax: [bOK =] Server.SyncAppsEx([vProfile])

bOK (Boolean) Optional – returns True or False for the success of the command

vProfile (String) Optional – is the name of the profile used to compare what is on the device with the server list of objects.

[ErrMsg] Optional – if an error occurs this parameter will be sent to a detailed error message

Example:

```
Dim bOK As Boolean
Dim sErr As String
bOK = Server.SyncAppsEx("CEProfile1", sErr)
If bOK = False Then
  App.MsgBox(sErr)
End If
```

Supported Versions: RFgen 5.1 and newer.

WriteFile

This command will write a file to the server hard drive and can be used from both the Thin client mode and Mobile client mode.

Group: Server Extensions

Syntax: [bOK =] Server.WriteFile(sFileName, vData, bOverwrite)

bOK (Boolean) Optional – returns True or False for the success of the command

sFileName (String) is the path and file name for the file being created or overwritten.

vData (Variant) the contents of the file which can be either a string or byte array

bOverwrite (Boolean) set to True, the server will overwrite an existing file, False will return a failure because the file already existed

Example:

```
Dim bOK As Boolean
Dim sData As String
```

```
Dim sPath As String
sPath = "C:\MyFile.txt"
sData = "Sample Text"
bOK = Server.WriteFile(sPath, sData, True)
```

Supported Versions: RFgen 4.0 and newer.

SigToImg

The **SigToImg** is a special type of function that takes the array of points (based 64 encoded array) returned from Signature.Text, and makes it into a bitmap image.

The image can then be passed from an offline client to the server, speed performance when passing it through a thin client, or be saved to a batch client.

Group: SigToImg

Syntax: SigToImg(Points, Img())

SigToImg (Boolean) Returns True if the array of points from Signature.Text were returned; False if the array of points were not returned.

Points (String) The points from the Signature entered in the Signature prompt/control.

Img (Byte) The bitmap that contains the image created from the array.

Example:

```
Dim bImage() As Byte
SigToImg(Signature1.Text, bImage)
```

Socket Object

The Socket object is provided to the programmer for creating and managing their own WinSock connection from within the solution. This object is only a pass-through to the standard Windows WinSock control so additional documentation about how this works can be easily found on the Internet. There are three sections for this object, properties, methods and events.

This object is declared in a similar manner to:

```
Dim WithEvents oSocket As Socket
```

The properties for the Socket object are:

BytesReceived, LocalHostName, LocalIP, Protocol, RemoteHost, RemoteHostIP, RemotePort, and State.

The methods for the Socket object are:

sktClose, sktConnect, sktGetData, sktPeekData, and sktSendData.

The events for the Socket object are:

OnClose, OnConnect, OnDataArrival, OnError, OnSendComplete, and OnSendProgress.

BytesReceived

This function returns the number of bytes currently in the receive buffer. This is a read-only property and is unavailable at design time. The value returned is a long integer.

Syntax: nValue = oSocket.BytesReceived

nValue (Long) the number of bytes

Group: Socket Object

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Dim nBytes As Long
Set oSocket = New Socket
nBytes = oSocket.BytesReceived
```

LocalHostName

The LocalHostName function returns the name of the local host system. This is read-only property and is unavailable at the design time. The value returned is a string.

Syntax: sValue = oSocket.LocalHostName

sValue (String) the name of the local host

Group: Socket Object

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Dim sName As String
Set oSocket = New Socket
sName = oSocket.LocalHostName
```

LocalIP

The LocalIP function returns the local host system IP address in the form of a string, such as 11.0.0.127. This property is read-only and is unavailable at design time.

Syntax: sValue = oSocket.LocalIP

sValue (String) the IP address of the local host

Group: Socket Object

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Dim sIP As String
Set oSocket = New Socket
SIP = oSocket.LocalIP
```

Protocol

This property can get or set the protocol used to either TCP or UDP.

Syntax: nValue = oSocket.Protocol

Alternate: oSocket.Protocol = enValue

enValue (enWinsockProtocols) the numeric value representing TCP or UDP. TCP = 0, UDP = 1. There are built in constants to represent these value as well. They are sckTCPProtocol and sckUDPProtocol.

nValue (Long) the numeric value representing TCP or UDP. TCP = 0, UDP = 1.

Group: Socket Object

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Dim wProtocols As enWinsockProtocols
    Set oSocket = New Socket
    oSocket.Protocol = sckTCPProtocol
    wProtocols = oSocket.Protocol
```

RemoteHost

The RemoteHost property returns or sets the remote host. This can be both read from and written to and is available both in design time and runtime. The value returned is a string and can be specified either as an IP address or as a DNS name.

Syntax: sValue = oSocket.RemoteHost

Alternate: oSocket.RemoteHost = Value

sValue (String) the name of the host system the client will be connecting to.

Group: Socket Object

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Dim sName As String
Set oSocket = New Socket
sName = oSocket.RemoteHost
```

RemoteHostIP

This property returns or sets the remote host IP address.

Syntax: sValue = oSocket.RemoteHostIP

Alternate: oSocket.RemoteHostIP = Value

sValue (String) the IP address of the host system the client will be connecting to.

Group : Socket Object

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Dim sIP As String
Set oSocket = New Socket
sIP = oSocket.RemoteHostIP
```

RemotePort

This property returns or sets the remote port number.

Syntax: nValue = oSocket.RemotePort

Alternate: oSocket.RemotePort = Value

nValue (Long) the port number used to access the host system the client will be connecting to.

Group : Socket Object

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Dim nPort As Long
    Set oSocket = New Socket
    oSocket.RemotePort = 11908
```

```
nPort = oSocket.RemotePort
```

State

This property returns the state of the control as expressed by an enumerated list. This is read-only property and is unavailable at design time.

Syntax: nValue = oSocket.State

nValue (Long) the number assigned to the different socket states
0 = sckClosed
1 = sckOpen
2 = sckListening
3 = sckConnectionPending
4 = sckResolvingHost
5 = sckHostResolved
6 = sckConnecting
7 = sckConnected
8 = sckClosing
9 = sckError

Group : Socket Object

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Private Sub oSocket_OnConnect()
 On Error Resume Next
 '
Set oSocket = New Socket
 If oSocket.State <> sckConnected Then
  App.MsgBox("Connect failed.")
End Sub
```

sktClose

This method terminates a TCP connection from either the client or server applications. If the object is declared using the WithEvents option, then an OnClose event is available in the script environment but will not fire with the use of this method but only if the connection is closed by the server.

Syntax: oSocket.sktClose

Group : Socket Object

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
    Set oSocket = New Socket
    If oSocket.State = sckConnected Then oSocket.sktClose
```

sktConnect

This method requests a connection to a remote computer. If the object is declared using the WithEvents option, then an OnConnect event is available in the script environment.

Syntax: oSocket.sktConnect([vRemoteHost], [vRemotePort])

vRemoteHost	(Variant) Optional – allows a connection to a host that is not specified in the RemoteHost property
vRemotePort	(Variant) Optional – allows a connection to a host that is not specified in the RemotePort property

Group : Socket Object

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Set oSocket = New Socket
    oSocket.Protocol   = sckTCPProtocol
    oSocket.RemoteHost = "127.0.0.1"
    oSocket.RemotePort = 21097
    oSocket.sktConnect

Private Sub oSocket_OnConnect()
 On Error Resume Next
 '
    Set oSocket = New Socket
 Do
    Loop Until oSocket.State <> sckConnecting
    If oSocket.State <> sckConnected Then App.MsgBox "Connect failed."
End Sub
```

sktGetData

This method retrieves the current block of data from the buffer and then stores it in a variable of the variant type. If the object is declared using the WithEvents option, then an OnDataArrival event is available in the script environment.

Syntax: oSocket.sktGetData(vData, [vType], [vMaxLen])

vData (Variant) Where retrieved data will be stored after the method returns successfully. If there is not enough data available for requested type, vData will be set to Empty.

vType (Variant) Optional – specifies the type of data being retrieved
Byte = vbByte
Integer = vbInteger
Long = vbLong
Single = vbSingle
Double = vbDouble
Currency = vbCurrency
Date = vbDate
Boolean = vbBoolean
SCODE = vbError
String = vbString
Byte Array = vbArray + vbByte

vMaxLen (Variant) Optional – Specifies the desired size when receiving a byte array or a string. If this parameter is missing for byte array or string, all available data will be retrieved. If provided for data types other than byte array and string, this parameter is ignored.

Group : Socket Object

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Private Sub oSocket_OnDataArrival(ByVal bytesTotal As Long)
 On Error Resume Next
 '
    Dim sData As String
    Set oSocket = New Socket
    oSocket.sktGetData(sData, vbString, bytesTotal)
End Sub
```

sktPeekData

This method operates in a fashion similar to the sktGetData method. However, it does not remove data from the input queue. It is used to see what data is coming before using the sktGetData method to retrieve and delete the data from the receive buffer. This method works only for TCP connections.

Syntax: oSocket.sktPeekData(vData, [vType], [vMaxLen])

vData (Variant) Where retrieved data will be stored after the method returns successfully. If there is not enough data available for requested type, vData will be set to Empty.

vType (Variant) Optional – specifies the type of data being retrieved
Byte = vbByte
Integer = vbInteger
Long = vbLong
Single = vbSingle
Double = vbDouble
Currency = vbCurrency
Date = vbDate
Boolean = vbBoolean
SCODE = vbError
String = vbString
Byte Array = vbArray + vbByte

vMaxLen (Variant) Optional – Specifies the desired size when receiving a byte array or a string. If this parameter is missing for byte array or string, all available data will be retrieved. If provided for data types other than byte array and string, this parameter is ignored.

Group : Socket Object

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Private Sub oSocket_OnDataArrival(ByVal bytesTotal As Long)
 On Error Resume Next
 '
 Dim sData As String
 Set oSocket = New Socket
 oSocket.sktPeekData(sData, vbString, bytesTotal)
End Sub
```

sktSendData

This method dispatches data to the remote computer. When a UNICODE string is passed in, it is converted to an ANSI string before being sent out on the network.

Syntax: oSocket.sktSendData(vData)

vData (Variant) Data to be sent. For binary data, byte array should be used.

Group : Socket Object

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Set oSocket = New Socket
```

```
oSocket.sktSendData("Hello world.")
```

OnClose

This is a Socket Object event that occurs when the connection has been closed by the remote host. This does not occur when the sktClose method has been called.

Group: Event

Applies to: Socket Object

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Private Sub oSocket_OnClose()
 On Error Resume Next
End Sub
```

OnConnect

This is a Socket Object event that occurs when a connection is successfully established.

Group: Socket Object

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Private Sub oSocket_OnConnect()
 On Error Resume Next
End Sub
```

OnDataArrival

This is a Socket Object event that occurs when data arrives and is used to process incoming data.

Group: Socket Object

Example:

```
Dim WithEvents oSocket As Socket      ' Module level Dim
Private Sub oSocket_OnDataArrival(ByVal bytesTotal As Long)
 On Error Resume Next
End Sub
```

OnError

This is a Socket Object event that occurs when there is an error, such as a PC not existing.

Group: Socket Object

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Private Sub oSocket_OnError(ByVal Number As Long, ByVal Description As String)
 On Error Resume Next
End Sub
```

OnSendComplete

This event is for Socket Objects. It occurs when all the data has been sent to its destination.

Group: Events

Applies To: Socket Object events

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Private Sub oSocket_OnSendComplete()
 On Error Resume Next
End Sub
```

OnSendProgress

This is a Socket Object event that occurs while the data is being sent between client and server.

Group: Events

Applies To: Socket Object events

Example:

```
Dim WithEvents oSocket As Socket ' module level Dim
Private Sub oSocket_OnSendProgress(ByVal bytesSent As Long, ByVal bytesRemaining As Long)
 On Error Resume Next
End Sub
```

Soft Input Panel (SIP) Extensions

The soft input panel (SIP) is the virtual keyboard that pops up when a field on the mobile (graphical capable) device allows for text input. The SIP object is a global override. Setting any property with SIP takes precedence over everything else.

Language extensions "SIP.Keyboard" and "SIP.Show" are now global in scope. Setting these to true, false, or a keyboard type will maintain that state for the session or until a SIP.Keyboard kbNone is received.

Use "SIP Reset" to remove the global overrides.

The extensions available for this are:

SIP.GetCurrentType, SIP.GetTypes, SIP.Keyboard, SIP.Mode, SIP.Reset, SIP.SetType, SIP.Show, and SIP.Visible.

Related Topics

See Configuration > Menus and Keys > Show/Hide SIP

GetCurrentType

This method returns a textual representation of the current input panel such as "Keyboard" or "Transcriber".

Group: Soft Input Panel

Syntax: [sValue =] SIP.GetCurrentType

sValue (String) – is the variable containing the result.

Example:

```
Dim sVal as String
sVal = SIP.GetCurrentType
```

In this case sVal will have a value like "Keyboard".

GetTypes

This method returns a pipe (|) delimited list of available input types that are supported on the device. Not all devices will have the same list of available types. Here are a few examples:

Letter Recognizer	Block Recognizer	Phone Keypad
Compact QWERTY	Full QWERTY	WinCE Handwriting
WinCE Keyboard	Keyboard	Kana
Kensaku	Romaji	Tegaki

Group: Soft Input Panel

Syntax: sList = SIP.GetTypes

sList (String) – Contains a list of all available types of input that are supported on the device.

Example:

```
Dim sList as String
```

```
sList = SIP.GetTypes
```

Keyboard

This property is used to set one or modes together to create the proper input. For standard English use the Roman mode. For other variations a combination may be required.

Language extensions "SIP.Keyboard" and "SIP.Show" are now global in scope. Setting these to true, false, or a keyboard type will maintain that state for the session or until a SIP.Keyboard kbNone is received.

Group: Soft Input Panel

Syntax: SIP.Keyboard

Examples:

Mode

This property sets the type of character to be displayed in the SIP/touchscreen for a given language. For standard English use the Roman mode. Other variations a combination may be required.

Group: Soft Input Panel

Syntax: [bOK =] SIP.Mode(enMode)

bOK (Boolean) – Optional – Returns True if the command was successful.

enMode (enSIPMode) – are the individual properties that can combine to make up the proper input panel. They are:

enSIP_CHARCODE
enSIP_CHINESE

enSIP_FULLSHAPE

enSIP_HANGUL

enSIP_JAPANESE

enSIP_KATAKANA

enSIP_LANGUAGE

enSIP_NATIVE

enSIP_ROMAN

Examples:

```
Dim bOK as Boolean

bOK = SIP.Mode(enSIP_ROMAN)

bOK = SIP.Mode(enSIP_ROMAN or enSIP_FULLSHAPE or enSIP_KATAKANA or enSIP_NATIVE)
```

In the second example, several modes are ORed together to create the Katakana input panel.

Reset

SIP.Reset removes all soft keyboard overrides and returns to the prompt-level control.

Group: Soft Input Panel

Syntax: To be supplied.

Example:

```
To be supplied.
```

SetType

This method sets the current input panel. The Input Method parameter can either be text as it is returned in the GetTypes method or it can be a zero-based number referring to the same list. This list is the same as the dropdown list on the mobile device where you choose between the styles of recognition.

Group: Soft Input Panel

Syntax: [bOK =] SIP.SetType(vIM)

bOK (Boolean) – Optional – Returns True if the command was successful.

vIM (Variant) – set to a number or a string representing one of the available recognition styles.

Example:

```
Dim bOK as Boolean
bOK = SIP.SetType(0)' 0 may not represent the keyboard
bOK = SIP.SetType("Keyboard")
```

Show

This method is used to display or hide the Soft Input Panel (SIP). "SIP.Keyboard" and "SIP.Show" are global in scope.

Setting these to true, false, or a keyboard type will maintain that state for the session or until a "SIP.Keyboard kbNone" is received.

Group: Soft Input Panel

Syntax: [bOK =] SIP.Show(bShow)

bOK (Boolean) – Optional – Returns True if the command was successful.

iShow (Long) – Sets the keyboard mode (no keyboard, show or hide) for the specified RFgen keyboard types.

Long (Long) The enumeration for the RFgen keyboard type. Values are:

KBRD_NONE The default value with SIP.Show to prevent a keyboard/soft input panel from displaying. Is not the same as hiding a keyboard.

KBRD_SIP Use this value with SIP.Show to display the keyboard/soft input panel.

KBRD_SIP Use this value with SIP.Show to hide a keyboard/soft input panel.

KBRD_FULL_ALPHA

KBRD_FULL_ALPHA_NUMERIC

KBRD_ALPHA

KBRD_ALPHA_NUMERIC

KBRD_CUSTOM1

KBRD_CUSTOM2

KBRD_CUSTOM3

KBRD_CUSTOM4

Example:

```
Dim bOK as Boolean
bOK = SIP.Show (KBRD_FULL_ALPHA)
```

Visible

For situations where the keyboard was implemented as a function instead of a property, this helps you to determine if the keyboard is visible or not. "SIP.Visible" is a read-only function; It does not make a keyboard visible.

Group: Soft Input Panel

Syntax: SIP.Visible()

( ) Boolean. True means the keyboard is visible; False means its not.

Example:

```
'Create a variable and then use SIP.Visible to receive a true or false value.
```

Versions Supported: RFgen 5.2.3.1

Smtp Extension

This language extension gives the user the ability to send e-mails from the scripting environment. In the Application Services configuration window is a place where some defaults may be set so the script environment doesn't need to repeat static information.

The extensions available for smtp are:

smtp.attach, smtp.bcc, smtp.cc, smtp.clear, smtp.from, smtp.host, smtp.message, smtp.port, smtp.send, smtp.subject and smtp.to

Attach

Specify a path and a file you wish to attach. If you would like to attach more than one file just separate the path + file names with a comma like:

"C:\Log.txt, C:\DebugLog.txt"

Syntax: Smtp.Attach = sPath

sPath (String) is the path and the name of the file to be attached to the outgoing message.

Example – see Smtp.Send

Bcc

This property is used to specify a recipient to be "Blind Carbon Copied" .

Syntax: Smtp.Bcc = sEmail

sEmail (String) is the email address of the recipient

Example – see Smtp.Send

Cc

This property is used to specify a recipient to be "Carbon Copied".

Syntax:

Smtp.Cc = sEmail

sEmail (String) is the email address of the recipient.

Example:

See Smtp.Send

Clear

This property is used to clear the contents of a smtp object.

Syntax: Smtp.Clear()

Example – see Smtp.Send

From

This property is used to specify the sending person's email address.

Syntax: Smtp.From = sEmail

sEmail (String) is the email address of the sender

Example – see Smtp.Send

Host

This property is the host email server that will process the request. In the Application Services configuration there is a place to default this value so it isn't necessary in the script.

Syntax: Smtp.Host = sServer

sServer (String) is the IP address or DNS name of the outgoing mail server

Example – see Smtp.Send

Message

This property contains the message body of the email. To include Carriage Return + Line Feed (Enter keys) in the text use the convention /r/n in the text itself.

Syntax: Smtp.Message = sBody

sBody (String) is all the text in the body of the email

Example – see Smtp.Send

Port

This property is the host email port that will be used to communicate with the host email server. In the Application Services configuration there is a place to default this value so it isn't necessary in the script. Be sure that this port isn't being blocked by firewalls or virus protection.

Syntax: Smtp.Port = nPort

nPort (Long) is the port number of the outgoing email server

Example – see Smtp.Send

Send

This method submits the Smtp properties to the mail server for processing. Note that the Host and Port properties are optional if already specified in the Application Services configuration screen.

Syntax: Smtp.Send()

Example:

Smtp.Host = "smtp.company.com"

Smtp.Port = 25

Smtp.From = "MySelf@company.com"

Smtp.Cc = "User@company.com"

Smtp.Bcc = "Boss@company.com"

Smtp.To = "Ex-Employee@company.com"

Smtp.Subject = "I'm so sorry"

Smtp.Message = "Good bye./r/nCall me!"

Smtp.Attach = "C:\PinkSlip.PDF, C:\GetWell.PDF"

Smtp.Send

Smtp.Clear()

Subject

This property contains the subject of the email.

Syntax: Smtp.Subject = sText

sText (String) is the subject of the email

Example – see Smtp.Send

To

This property is used to specify the recipient's email address.

Syntax: Smtp.To = sEmail

sEmail (String) is the email address of the recipient

Example – see Smtp.Send

Stored Procedure Extensions

These *obsolete* commands relate specifically to stored procedures and have been replaced with the Embedded Procedure Object.

CallAction (not used with Sybase)

This command is OBSOLETE. It relates specifically to stored procedures and has been replaced with the Embedded Procedures structure.

This function will call the stored procedure as specified, and will return the output from the procedure.

Syntax: vReturnVal = SP.CallAction(sName, vParams)

vReturnVal (Variant) is a return value number from the stored procedure. Typically '0' (zero) specifies that no errors occurred during processing.

sName (String) is the name of the stored procedure to call.

vParams (Variant) passing parameters as required by the stored procedure. For stored procedures that return values as passing parameters, any output parameters must be declared as Variant and be set to empty.

Example:

```
Dim vReturnVal As Variant

vReturnVal = SP.CallAction ("SPname", Rsp)
```

CallProc (must be used with Sybase)

This command is OBSOLETE. It relates specifically to stored procedures and has been replaced with the Embedded Procedures structure.

This function will call the stored procedure as specified, and will return the output from the procedure. An unlimited number of passing parameters Pn (e.g., P1, P2, etc.) may be specified.

Syntax: vErrNo = SP.CallProc(sName, sOptions, sColumns, sRows, vParams)

vErrNo (Variant) is a return value from the stored procedure. Typically '0' (zero) specifies that no errors occurred during processing.

sName (String) is the name of the stored procedure to call.

sOptions (String) are the options for the stored procedure to use. Allowable options are:

A = Action - no rows returned

S = Select - rows are returned

O = Open - an alternative required for some databases (i.e., for Sybase: 'A' and 'S' are not useful, use only 'O').

Note: each type may optionally be suffixed by the letters 'Y' or N' (for Yes or No) to specify additional possible database requirements, as follows:

1st letter (Y or N): a 'Return' value is specified in the stored procedure. 1st letter defaults (if not specified) are 'Y' for type 'A', 'N' for type 'S', and 'N' for type 'O'),

2nd letter (Y or N): to declare the passing parameters as 'input 'or 'output'. The 2nd letter default (if not specified) is 'Y' for all 3 type ('A', 'S', and 'O'). A tip: consider 'N' for Oracle databases).

Examples:

Type='A'

Type='S'

Type='ONN'

sColumns (String) is a string representation of the columns contained in the associated Records item (below).

sRows (String) is a string representation of any results returned by the stored procedure.

vParams (Variant) passing parameters as required by the stored procedure. For stored procedures that return values as passing parameters, any output parameters must be declared as Variant and be set to empty.

Example:

```
Dim sColumns As String
Dim sRows As String
Dim vPn As Variant
Dim vErrNo As Variant
vPn = Rsp ' User must provide value
vErrNo= SP.CallProc("byroyalty", "SNY", sColumns, sRows, vPn)
```

The Records variable now contains a resultset from which values can be extracted, using DB.Extract.

CallSelect (not used with Sybase)

This command is OBSOLETE. It relates specifically to stored procedures and has been replaced with the Embedded Procedures structure.

This function will call the stored procedure as specified, and will return the output from the procedure.

Syntax: vErrNo = SP.CallSelect(sSQL, sColumns, sRows, [vParams])

vErrNo (Variant) is a return value from the stored procedure. Typically '0' (zero) specifies that no errors occurred during processing.

sSQL (String) is the name of the stored procedure to call as a string or string variable, along with any passing parameters for the procedure (i.e., "GetCust '1001'").

sColumns (String) is a string representation of the columns contained in the associated Records item (below).

sRows (String) is a string representation of any results returned by the stored procedure.

vParams (Variant) Optional – additional parameters for the CallSelect

Example:

```
Dim vErrNo As Variant
Dim sSQL As String
Dim sColumns As String
Dim sRows As String
sSQL = "SPQuery" ' Name of stored procedure
vErrNo = SP.CallSelect(sSQL, sColumns, sRows)
```

Synchronization Overview

The Sync commands provide more control over how often the data in a table is synchronized with other databases. The synchronization runs as a background process and is designed to synchronize records where a change occurred. The process includes changes was not captured in the prior synchronization process.

The default server for client synchronization defaults to the server defined in the client profile. If you want to use a different server, use Sync.SetHost.

The Sync commands available are:

SetHost

SyncNow

SyncSetTabelFilter

SyncStart

SyncStop

Instead of running continuously, the **Sync.SyncStart** process can be set to run after a specified period of time. For example if it takes 5 minutes for your device to synchronize, and you set the elaspe time of 1 minute, then the synchronization process will kick-off every 6 minutes.

If you need the synchronization background process to stop, use the **Sync.SyncStop( )**command.

If you set a command to stop the synchronization process, after it was initiated, the system will stop gracefully. (It will compete the process it committed to instead of stopping abruptly.)

If you want the process to synchronize immediately instead of waiting for its preset cycle, you can use the **SyncNow** command. This works in two ways. For example, if you have an offline solution and don't have a sync process running in the background, but need the table data synchronized when the user taps a button, the sync.now process will launch the Sync process, but only once. However, if the synch process running in the background, and is just waiting for its next start time, then the **SyncNow** command will wake up the process and make it start immediately. The SyncNow completes its synchronization then terminates. The background process executed from Sync.Start will resume its course once SyncNow terminates.

SyncNow Technical Details

If the thread is running but its actually waiting for its start time or interval, the Sync.Now( ) will make the thread go immediately. If that thread is not running, then it will start running, but will only make one pass. After it completes that pass, the Sync.Now process will come back and terminate itself.

There is also a **Sync.CommandTimout** command you can use to specify how long the sync process will wait for the host to respond/complete a download. For example, you don't want to take more than 5 minutes to download a table and you know you have a million records to download or the device is not getting a response from the server.

The command timeout will adjust how long it keeps going before it gives up.

The timeout is set in seconds.

Using filters to specify what is synchronized

Sync.SetTableFilter - This adds a Where clause to the SQL query used to synchronize the offline data. This helps narrow the scope of synchronization actions so that they are very specific to what you want to synchronize. There are no "rules" on how to use so that your usage is flexible. For example, if your customer has

3 locations where an offline client will be used to collect data, you use this command to build a filter that would only synchronize data updates for the end user's specific location.

Supported Versions: RFgen 5.2.5.6 and newer. RFgen 6.0 and newer.

CommandTimeout

This sets in seconds how long the sync process will wait on the host to complete a download to complete or response to a request before it gives up and stops running the process.

The timeout is set in seconds.

Group: Sync Extension

Syntax: Sync.CommandTimout (Timeout)

Timeout (Long)The number of seconds

Example:

Sync.CommandTimeout(10)

Supported Versions: RFgen 5.2.5.5 and newer.

SetHost

If you have a need to synchronize your client with a server other than the one already defined in your client's profile, you can use this command to specify a different server and port number to be used for synchronization.

The timeout is set in seconds.

Group: Sync Extension

Syntax: Sync.SetHost (ServerName, port) as Boolean

b if able to set the hostname then returns True; False if it fails

Servername The name of the server to receive synchronization requests

port (Long) The port id of the target server

Example:

Sync.SetHost(MyBackupRFgen-2019.Server,21098)

Supported Versions: RFgen 5.2.5.5 and newer.

SyncNow

The sync process runs as a background process based on the time specified in the Sync.Start command.

If you need the synchronization process to start immediately (like when a User taps a Sync Right NOW button), and the synchronization thread is NOT actively running in the background, Sync.SyncNow will cause the process to run immediately for one pass then terminate itself.

If a synchronization process thread is present, but its waiting for its next interval to run, then Sync.SyncNow will trigger the process to wake up and run, then return the thread to its wait-for the-next-interval state.

Group: Sync Extension

Syntax: Sync.StartNow

There are no parameters for this extension.

Example:

Sync.StartNow

Supported Versions: RFgen 5.2.5.5 and newer.

SyncStart

The synchronization process runs as a background process.

SyncStart sets when this background process comes alive. The launch of the process starts after last sync process completes and after thetimeperiod specified in the command has elasped.

For example, if this command was **Sync.SyncStart(10**), this means the synchronization process will run 10 minutes after the prior synchronization process has completed. So even if some processes take longer to complete a synch, the start of the next sync is always 10 minutes following the completion of the last sync process.

If for some reason the synchronization process never completes (i.e.have a runaway process), then the successive sync will NOT launch unless had Sync.Command Timeout in your script.

Group: Sync Extension

Syntax: Sync.SyncStart(Interval)

Interval (Long)The number of minutes to wait before launching the process

Example:

Sync.SyncStart(10)

Supported Versions: RFgen 5.2.5.5and newer.

SyncStop

If the synchronization process is running, you use this command to stop the sync process. It stops the sync background process only after finding a safe place to stop. (It does not just stop immediately if in the middle of a process.)

Group: Sync Extension

Syntax: Sync.SyncStop

There are not parameters for this command.

Example:

Supported Versions: RFgen 5.2.5.5 and newer.

System Error Extensions

This object contains application error information that is also written to the error log. This object does not trap VB errors such as a type mismatch but application level issues such as a macro failing. For the VB errors see the Err object.

The extensions available for System Error are:

SysErr.AddError, SysErr.AppName, SysErr.Clear

SysErr.Count, SysErr.Description, SysErr.DevGUID

SysErr.IpAddress, SysErr.ErrNative, SysErr.Number, and SysErr.UserName.

AddError

Adds an error to the collection in the SysErr object. You must specify the error number, native error number and a description of the error.

Group: System Error

Syntax: SysErr.AddError(nErrNo, nNativeError, sDesc)

nErrNo (Long) the VBA error number

nNativeError (Long) the native database error number

sDesc (String) the description of the error

Example:

```
SysErr.AddError(1269, -2847638354, "TCP NOT AVAIL")
```

AppName

This property contains the name of the current application when the error occurred.

Group: System Error

Syntax: sName = SysErr.AppName

sName (String) the name of the faulting application as shown in the error log

Example:

```
Dim sName As String
```

```
sName = SysErr.AppName
```

Clear

Clears all property settings of the SysErr properties. Use this command after trapping and dealing with an error so the next error can be trapped.

Group: System Error

Syntax: SysErr.Clear

Example:

```
SysErr.Clear
```

Count

This property returns the number of errors. Occasionally an ODBC driver may return multiple errors and the first one may not be the most descriptive or accurate for solving the problem. Concatenating all the errors for the message box will give the most complete description of the problem.

Group: System Error

Syntax: vValue = SysErr.Count

vValue (Variant) is the number of errors in the collection.

Example:

```
Dim vCount As Variant
vCount = SysErr.Count
```

Description

This property returns a string description of the ODBC error. Using the SysErr.Count command you may specify the number of the error in a loop and extract each error for display to the user.

Group: System Error

Syntax: sDesc = SysErr.Description([vIndex])

sDesc (String) is the error description.

vIndex (Variant) Optional – is the error number to view. Note: this value is zero-based and defaults to 1.

Examples:

```
Dim sDesc As String
    sDesc = SysErr.Description(0)
```

```
Dim i As Integer
    For i = 0 To SysErr.Count - 1
    sDesc = sDesc & SysErr.Description(i) & vbCrLf
Next
```

DevGUID

This property contains the device GUID value used by the server to uniquely identify a device with a session. This value can be displayed from the Enterprise Dashboard if necessary.

Group: System Error

Syntax: sValue = SysErr.DevGUID

sValue (String) is the device GUID identifier

Example:

```
Dim sValue As String
sValue = SysErr.DevGUID
```

IpAddress

This property contains the IP address of the device. This value can be displayed from the Enterprise Dashboard if necessary.

Group: System Error

Syntax: sValue = SysErr.IpAddress

sValue (String) is the IP address of the device

Example:

```
Dim sIP As String
sIP = SysErr.IpAddress
```

NativeError

This property returns a numeric value specifying the native database error number.

Group: System Error

Syntax: vValue = SysErr.ErrNative([vIndex])

vValue (Variant) is the database error number.

vIndex (Long) Optional – is the error number to view. Note: this value is zero-based and defaults to 1.

Example:

```
Dim vError As Variant
vError = SysErr.NativeError(0)
```

Number

This property returns a numeric value specifying the VBA error number.

Group: System Error

Syntax: vError = SysErr.Number([vIndex])

vError (Variant) is the VBA error number.

vIndex (Long) Optional – is the error number to view. Note: this value is zero-based and defaults to 1.

Example:

```
Dim vError As Variant
vError = SysErr.Number(0)
```

UserName

This property contains the user logged in that experienced the error.

Group: System Error

Syntax: sName = SysErr.UserName

sName (String) is the user name of the device when the error occurred

Example:

```
Dim sUser As String
sUser = SysErr.UserName
```

System Level Extensions

System level commands get or set environmental properties for users, data connections or other global system settings.

The extensions available for the system level commands are:

SYS.CheckBoxNoClickFromCode, SYS.Color, SYS.ConnectionProperty, SYS.DeleteProperty,

SYS.DisableTimeout, SYS.GetConnection, SYS.GetProperty,

SYS.SelectKeyboard, SYS.SendMessage, SYS.SetProperty,

SYS.SetTimeout, SYS.UserList, and SYS.ValidateWinUser

CheckBoxNoClickFromCode

This property is a global internal flag, when set to True, will prevent the Click event from executing when code is used to toggle the Checkbox. The Click event will still execute if a user toggles the Checkbox from the user interface. This would typically be used to initialize a form, check or uncheck the Checkboxes based on database data and not trigger the Click event.

Group: System

Syntax: [bValue] = SYS.CheckBoxNoClickFromCode = bValue

bValue (Boolean) Optional – returns the state of this function

Example:

```
SYS.CheckBoxNoClickFromCode = True
bValue = SYS.CheckBoxNoClickFromCode
```

Versions Supported: RFgen 5.0 and newer.

Color

This function can be used to assign a color from a specific palette of system colors in Solution Explorer > Themes > [Theme Name] > Application: SystemColors.

Group: System

Syntax: SYS.Color(iIndex)

iIndex is a value from 1 to 16.

Example:

```
Label1.ForeColor=SYS.Color(1)  'Where 1 corresponds to Color1 in Themes > Application > SystemColor.
```

Versions Supported: RFgen 5.2 and newer.

ConnectionProperty

This function will return a characteristic of a data source. Some properties only refer to certain types of data connections. For example, dcDatabase would refer to an ODBC connection where dcHostPort would refer to a screen mapping connection. This is read-only.

Group: System

Syntax: vValue = SYS.ConnectionProperty(vSource, enProp)

vValue (Variant) set to the value of the specified property

vSource (Variant) is the string representation or the numeric representation of the database, screen mapping host or ERP session

enProp (enDCProps) the following list of properties is available

dcAppGroup	(SAP)	Application Group field
dcAppName	(SM)	For Windows Console mode, this is the Applciation Executable field
dcAutoWrap	(SM)	Wrap text at end-of-line (VT220 only)
dcBackColor	(SM)	Back color of telnet emulator, it returns a numeric value corresponding to VB colors
dcBlockSize	(DB2)	Block Size field
dcCatalog	(DB2)	Initial Catalog field
dcCCSID	(DB2)	CCSID Conversion flag, returns a 0 (False) or a 1 (True)
dcCFIT	(SAP)	Conversion Fault Character field
dcClientNo	(SAP)	Application server's Client number
dcCmdLine	(SM)	For Windows Console mode, this is the Command Line field
dcCompany	(Dynamics)	Company field
dcConfigFile	(ERP)	Axapta and Dynamics connections, this is the Configuration File field
dcConnCheck	(DB)	Validation Check field for all database connections. Returns a 0 (No) or a 1 (Yes)
dcConnectMode	(All)	Returns User Database, Enterprise Connection, Screen Mapping Connection, or Web Service
dcConnectTimeout	(Web)	Connect Timeout field
dcConnectType	(All)	Returns the type of connection. Values are: Access, Axapta, DB2, Dynamics, JDE Enterprise One, ODBC, OleDB, Oracle, SAP R/3, SQL Server, SQLite, TN3270, TN5250, VT220, Web Services, Windows Exe.
dcConnString	(DB)	Returns the provider string used to make the connection to the database. A portion of the string may be the path and file name of a file or one or more configuration fields combined together.
dcCursorType	(DB)	The cursor type or locking state used to make the connection (eg: Optimistic / Pessimistic)

dcDatabase	(DB)	For DB2 it is the Default Library field, for SQL Server it is the Database Name field.
dcDatabaseOwner		(For future use)
dcDatabaseType	(DB)	For specific database types it returns the database name. For OleDB it returns the Database field value.
dcDataStream	(Other)	For SAP it returns the Data Format field, 0 (ASCII) or 1 (Unicode). For Screen Mapping, vt220 mode it returns the Data Stream field, 0 (Standard) or 1 (UTF-8)
dcDestructiveBS	(SM)	For VT220 only, it returns the Destructive Backspace Boolean check box option
dcDeviceName	(SM)	Device Name field for TN5250 and TN3270 only
dcDownTime	(All)	Returns the Schedule Downtime configuration for all data connector types.
dcDriver		(For future use)
dcDSN	(DB)	For ODBC type only, it returns the Data Source field value.
dcEBCDIC	(SM)	Returns the Code Page field for TN5250 and TN3270 only
dcErrorCodes	(DB)	Reset on Error field values are returned for all database connector types.
dcExtendedProps	(DB)	Returns the Extended Properties list for all database connectors except ODBC.
dcFileName	(DB)	Returns the Database File name field value for Access, SQL Server and SQLite data connectors
dcFontSize	(SM)	Font Size field used to display the host screen in the telnet emulator or the HostScreen control for Windows Console mode.
dcForceKeyPacket	(SM)	For VT220 only, it returns the Send Whole Key Packets Boolean check box option
dcForeColor	(SM)	Display ForeColor used to display the host screen in the telnet emulator or the HostScreen control for Windows Console mode.
dcGWHost	(SAP)	Gateway Host field
dcGWService	(SAP)	Gateway Service field
dcIgnoreCert	(Web)	Ignore Invalid Certificates Boolean check box field

dcIndexDatabase	(DB)	Index Selected Databases Boolean check box field for all database connector types.
dcIndexList	(DB)	Tables Specified for Indexing list of values all database connector types.
dcIsJDE	(DB)	Returns the JD Edwards Database Indexing Boolean check box on the Indexing Options tab for all database connector types.
dcLanguage	(SAP)	Returns the Language field, used to log in to the ERP system
dcLibraryList	(DB2)	Returns the Catelog Library List field value
dcLocale	(SM)	Returns the numeric value of the Display Language field
dcLocalEcho	(SM)	For VT220 only, it returns the Echo Characters Locally Boolean check box option
dcLogicalName	(SQL Svr)	Returns the Logical Name field value for the SQL Server database type only
dcLoginMode	(All)	Returns Login Mode option on the Login Options tab (0 = Automatic, 1 = Manual)
dcMsgServer	(SAP)	Message Server field
dcOnCCE	(SAP)	returns the Character Conversion option (0 = Abort, 1 = Copy, 2 = Replace)
dcPackageName	(DB2)	Returns the SQL Package Name field value
dcPadFields	(SM)	For TN5250 and TN3270 only, returns the Pad fields When Sending Data Boolean check box value
dcPassword	(All)	Returns the Password field from most of the data connectors. Some data connector configurations do not have a password field.
dcPoolData	(All)	Returns the list or users and passwords in the Connection Pooling Named Users grid
dcPooled	(All)	Returns the Pooling Status field value (0 = Disabled, 1 = Enabled)
dcPoolMax	(All)	Returns the Allocated Licenses number
dcPort	(SM)	For TN5250, TN3270, and VT220 only, this returns the Telnet Port field value
dcProvider	(DB)	Returns the Provider Name field value for database types that have one.
dcQueryGoal	(DB2)	Returns the Query Optimize Goal field value

dcQueryTimeOut	(DB)	Returns the Query Timeout value configured for any database connector
dcR3Name	(SAP)	R/3 Name field
dcReadDataRows	(ERP)	Maxium SQL Rows to be returned from the ERP connector
dcReceiveTimeout	(Web)	Returns the Receive Timeout field value
dcResetOnFailure		(For future use)
dcRouter	(SAP)	Router String field
dcSendCRLF	(SM)	For VT220 only, Returns Send Return + Line Feed Boolean field value
dcSendTimeout	(Web)	Send Timeout field
dcServer	(All)	Returns the Host Server / Name field for any connector that has one
dcSessionTimeout	(All)	Returns the Session Timeout field for all connection types
dcSourceId	(All)	The Source Id field that names the connection
dcStartMenu	(SM)	Returns Session Default Main Menu option
dcStartMenuLocked	(SM)	Returns 1 if the data connection is locked to a specific menu rather than the default of any main menu. This is found in the Login Options tab, Connection Pooling Named Users grid but only applies at runtime.
dcSystemDatabase		(For future use)
dcSystemNo	(SAP)	System Number field
dcTraceMode	(SM)	Returns the Enable Packet Trace Boolean check box field value
dcTrimFields	(SM)	For TN5250 and TN3270 only, returns the Trim Fields When Retriving Data Boolean field value
dcTrusted		(For future use)
dcUseFile	(DB)	Returns 1 for SQLite and Access and SQL Server only if a file is selected instead of a database server.
dcUseHTTPS	(Web)	Returns the Connect Using HTTPS Boolean field value
dcUser	(All)	Returns the User ID field for any connector that uses login credentials.
dcUseSSH	(SM)	For VT220 only, returns the Connect via SSH Boolean check box option

dcViewOwner	(DB)	Returns the Include Owner Name in Indexes Boolean check box field
dcViewSource	(DB)	Returns the Include Source Name in Indexes Boolean check box field
dcWinDomain	(ERP)	For Axapta and Dynamics only, returns the Windows Domain field
dcWinProxy	(ERP)	For Dynamics only, returns the Proxy Domain field
dcWinPwd	(ERP)	For Axapta and Dynamics only, returns the Proxy Password or Domain Password field
dcWinUser	(ERP)	For Axapta and Dynamics only, returns the Proxy User or Domain User field
dcWorkDir	(SM)	For the Windows Console Type only, returns the Working Directory field value.

Example:

```
Dim vValue As Variant
vValue = SYS.ConnectionProperty(1, dcUser)
vValue = SYS.ConnectionProperty("RFSample", dcUser)
```

Versions Supported: RFgen 4.0 and newer.

DeleteProperty

This function will delete a property (or all properties) contained within the collection specified by the key. SYS.SetProperty is used to add new properties to a collection.

Group: System

Syntax: SYS.DeleteProperty(vKey,v ID)

vKey (Variant) is the main key. It is the name of the collection of properties that are to be grouped together.

vID (Variant) is the property key. If an '*' (asterisk) is used, all properties within the collection will be deleted.

Example:

```
SYS.DeleteProperty("Counts", "LastIssue")
SYS.DeleteProperty("Counts", *)
```

Versions Supported: RFgen 4.0 and newer.

DisableTimeout

This function will allow the session to continue even when the server's Client Inactivity Timeout value has been reached. In essence, this session will never time out.

Group: System

Syntax: SYS.DisableTimeout( )

bValue (Boolean) set to True to enable (eliminate the Client Inactivity Timer) or False to return to normal monitoring.

Example:

```
SYS.DisableTimeout(True)
SYS.DisableTimeout(False)
```

Versions Supported: RFgen 4.1 and newer.

EnvironmentProperty

This function will return the value of the assigned property set in the Configuration / Environment Properties / System Environment Properties grid.

Syntax: vValue = SYS.EnvironmentProperty(vProperty)

vValue (Variant) is the value stored in the System Environment Properties for the Property specified.

vProperty (Variant) is the property specified in the Environment Properties grid.

Example:

```
Dim vPlant As Variant
vPlant = SYS.EnvironmentProperty("PlantName")
```

GetConnection

This is a specialized method similar to the EmbeddedProc object that could be used for getting the connection object to Microsoft Dynamics. The Dynamics client must be installed on the same machine. Use the ERP.BeginTrans and ERP.CommitTrans before and after the use of this method. This exposes the business functions but the programmer must know how to use them.

Group: System

Syntax: oValue = SYS.GetConnection(vSource)

oValue (Object) An object declaredConn as Axapta3 or Object, if you choose to use late binding. It will return the ADO, RDO, OLEDB, ERP, or Web connection object

vSource (Variant) data source name (DSN) or the data source number

Example:

```
Dim oConn As Axapta3
Dim oRecord As IAxaptaRecord
Set oConn = SYS.GetConnection("Dynamics")
Set oRecord = oConn.CreateRecord("INVENTJOURNALTABLE")
If oRecord Is Nothing Then
  App.MsgBox "Error: Axapta Record object failed."
  Exit Sub
End If
App.MsgBox "Record company: " & oRecord.company
```

Supported Versions: RFgen 4.0 and higher.

GetProperty

This function will return a property value that has been set using SYS.SetProperty. SYS.GetProperty and SYS.SetProperty can be used in place of making function calls to an INI file or to the System Registry. The keys and data are stored in the solution MDB file.

Group: System

Syntax: Syntax: vValue = SYS.GetProperty(vKey, vID)

vValue (Variant) is the property value.

vKey (Variant) is the main key. It is the name of the collection of properties that are to be grouped together.

vID (Variant) is the property key. It is the reference to the value being returned. If an '*' (asterisk) is used as the ID, a Chr(1) delimited list of properties contained within the collection will be returned.

Example:

```
Dim vTransfersCt As Variant
Dim vIssuesCt As Variant
vTransfersCt = SYS.GetProperty("Counts", "LastTfr")
vIssuesCt = SYS.GetProperty("Counts", "LastIssue")
```

Versions Supported: RFgen 4.0 and newer.

GetUserProperty

This function can access a user property before the user attempts to login. This is useful for evaluating a potential user's properties prior to actual login so you have the opportunity to change how the user is handled, how the client behaves etc.

The values such as the user Id and property come from the Solution Explorer > Users group.

Note: setting the App.User property logs the fact that the user signed in, so assigning it before the login was successful will result in irregular data in the logfile.

Group: System

Syntax: Syntax: vValue = SYS.GetUserProperty()

User (Varient) The user whom you to obtain the property from

PropID (Varient) The property ID of the user.

Varient The data type that is set for the value.

Example:

'This will cause a message box to display the user's property in the RFLogin app.

```
Private Sub Form_Load()
  SYS.GetUserProperty
    Dim var As String
  var=SYS.GetUserProperty("ADM", "Language")
App.MsgBox "" & var
```

Versions Supported: RFgen 5.2.4.2 and newer.

SelectKeyboard

This function was removed in 5.2.1.5. If your application used this function in a prior release of RFgen, when you upgrade "Sys.SelectKeyboard" will be converted to "SIP.Keyboard".

This function enables the end-user to open a specific keyboard.

Group: System

Syntax: enKyBoardMode = SYS.SelectKeyboard

eKeyBoard = (enKeyBoardMode) The keyboard modes available.

kbAlph = Alphabetical keyboard, no punctuation or numbers

kbAlphNum = Alphabetical and Numeric keyboard, no punctuation symbols

kbCustom( ) = the customized keyboard created from the RFgen Mobile Development Studio > Soft Keyboards node

kbFullAlpha = the keyboard with all 26 alphabetical characters and punctuation symbols

kbFullAlpha = the keyboard with all 26 alphabetical characters, punctuation symbols, and numbers

kbNone = prevents/suppress the system keyboards. Disables use of any system keyboards

kbNumeric = numbers only keyboard

kbSIP = the Soft Input Panel keyboard (characters are arranged on keyboard to provide a smaller keyboard)

Example:

```
SYS.SelectKeyboard = kbFullAlpha
```

Supported Versions: RFgen 5.0 and 5.1; Removed in 5.2.

SendMessage

This function sends a message to a specific device number currently connected to the server. Device numbers of users can be obtained by using the SYS.UserList command.

Group: System

Syntax: SYS.SendMessage(iDevNo, sMessage)

iDevNo (Integer) the ID of the mobile device

sMessage (String) the message to send to the other device

Example:

```
SYS.SendMessage(4, "Please see Sam immediately.")
```

Supported Versions: RFgen 4.0 and higher.

SetProcOptionFields

This function can be used to change the table and fields used by the JD Edwards JDEProcOpt objects. The default table and fields are: **F983051,** VRPID, VRVERS, and VRPODATA.

Group: JDE Processing Option

Syntax: Sys.SetProcOptionFields(ByVal F983051 As String, [ByVal VRPID], [ByVal VRVERS], [ByVal VRPODATA])

ByVal F983051 is the JD Edwards Table that contains the specified fields and/or processing options.

ByVal VRPID is field that contains Program IDs as a string

ByVal VRVERS is the field that contains the version of the JDE program as a string

ByVal VRPODATA is the field that contains the processing data option as binary (BLOB)

Examples:

```
'Reset the table and field names to the defaults
Sys.SetProcOptionFields("F983051", "VRPID", "VRVERS", "VRPODATA")
```

Supported Versions: RFgen 5.0 and higher.

SetProperty

This function will save a property value for future retrieval using SYS.GetProperty. Property values will persist between sessions because they are saved in the solution MDB file. SYS.GetProperty and SYS.SetProperty can be used in place of making function calls to an INI file or to the System Registry. Use this command with caution. The solution database is not designed for heavy adding and deleting of properties. Microsoft Access tends to grow over time as records are added and deleted.

Group: System

Syntax: SYS.SetProperty(vKey, vID, vValue)

vKey (Variant) is the main key. This will be the identifier for all like properties and their values.

vID (Variant) is the sub key. This key is referenced to obtain its value

vValue (Variant) is the property value.

Example:

```
Dim nIssuesCt as Long
nIssuesCt = 200
SYS.SetProperty("Counts", "LastTxr", 500)
SYS.SetProperty("Counts", "LastIssue", nIssuesCt)
```

Supported Versions: RFgen 4.0 and higher.

SetTimeout

This function sets the number of minutes before a Client connection times out if there is no activity.

Group: System

Syntax: SYS.SetTimeout( )

ByVal (Long) the period of time in minutes.

Example:

```
SYS.SetTimeout(1)    'Times the client out in 1 minute.
SYS.SetTimeout(5)    'Times the client out in 5 minutes.
```

Versions Supported: 5.2.3.1 and newer.

UserList

This function will return the device number, user and application currently in use. The purpose of this command is to log or locate a user doing a transaction, possibly for sending just the one device a message over the network. *(See SYS.SendMessage)*

Group: System

Syntax: sUserList = SYS.UserList([bOnlyLoggedIn])

sUserList (String) contains the device number, logged in user and application of all devices in use at the time the command was issued.

bOnlyLoggedIn (Boolean) Optional – when set to True only returns entries for devices that have a logged in user. The default is False and this returns all connected devices to the server regardless of a user logged in. This could be used to generate a list of all connected devices so that they may all receive a message on the screen. The default is False.

Example:

```
Dim sUserList As String
sUserList = SYS.UserList
sUserList = App.ShowList(SYS.UserList(True),True)
```

Supported Versions: RFgen 4.0 and higher.

ValidateWinUser

This function will validate user credentials against the Windows domain or local system. The user account must not be inactive.

Group: System

Syntax: bOK = SYS.ValidateWinUser(sDomain, sUser, gsPassword)

bOK (Boolean) returns True if the validation was a success

sDomain (String) Enter the domain to be searched. If the local PC is used, specify a single period for the domain or enter the PC name.

sUser	(String) Enter the user name to be validated.
gsPassword	(String) Enter the password to be validated. Passwords are case sensitive. Users that do not have a password will not work.

Example:

```
Dim bOK As Boolean
SYS.ValidateWinUser(".", App.User, gsPass)
'In this example the password was saved in a global string variable when it was used on the login screen.
```

Supported Versions: RFgen 4.0 and higher.

TTS (Text-To-Speech)

The Text-To-Speech (TTS) is a feature that converts written text into synthetic speech (audio data) on Android and iOS devices. Its supported on any version of Android or iOS device that the RFgen client software supports. (The API is provided by the Android OS.) TTS can also take text and translate it outloud.

Note:

• The ability to capability to capture user speech is not supported yet.

• The speak option can be called by any user input, it can be set to a button press or after a scan event is called.

The available commands are: TTS.PauseTime, TTS.ReadRate, TTS.Repeat, TTS.SetLanguage, TTS.Speak, TTS.Spell, TTS.StopSpeak, and TTS.Volume.

Group: TTS

Syntax:

To be supplied.

Examples:

```
Private Sub Button1_Click()
On Error Resume Next
TTS.Speak "This is a Test"
TTS.PauseTime=3
End Sub
```

Versions Supported: RFgen 5.2.2.0 and newer.

Devices Supported: Android and iOS, any OS RFgen Supports.

TTS.PauseTime

The pause is the "endpoint" between the end of a sentenance and the beginning of the next sentence. The pause time is in milliseconds, and the default value is 100 milliseconds. When a device reads a text message outloud, the device will pause between sentences (or other series or listings) using the value set for TTS.PauseTime.

Group: TTS

Syntax: TTS.PauseTime

Example:

```
Private Sub Button5_Click()
On Error Resume Next
TTS.PauseTime = TextBox1.Text
End Sub
```

Versions Supported: RFgen 5.2.2.0 and newer.

Devices Supported: Android, iOS.

TTS.ReadRate

When convertingwritten text into human-sounding speech, the conversion takes the speed for which words are uttered into consideration. This parameter sets the cadence of synthesized voice converted from written text. The read rate defaults to 100, which is the device's normal rate of speaking. It can be slowed to 50, or sped up to 200 for twice the normal rate. The default read rate is what the OS defines it to be.

Group: TTS

Syntax:

TTS.ReadRate=<integer>

Example:

```
Private Sub Button5_Click()
TTS.ReadRate=50
End Sub
```

Versions Supported: RFgen 5.2.2.0 and newer.

Devices Supported: Android, iOS.

TTS.Repeat

The repeat option can be called by any user input to re-read text previously read out loud by the Android device. The button is the preferred option to call this option.

Group: TTS

Syntax: TTS.Repeat

Example:

```
Private Sub btnRepeat_Click()
TTS.Repeat
End Sub
```

Versions Supported: RFgen 5.2.2.0 and newer.

Devices Supported: Android, iOS.

TTS.SetLanguage

The TTS.SetLanguage defaults to the language that is configured in the operating system of the Android device.The default can be changed via code.

Group: TTS

Syntax: TTS.SetLanguage

Example:

```
Private Sub Button5_Click()
On Error Resume Next
TTS.SetLanguage = "JA-JP"
End Sub
```

Versions Supported: RFgen 5.2.2.0 and newer.

Devices Supported: Android, iOS.

TTS.Speak

The speak option makes the Android device read the text outloud when its called by any type of user input. It can be set to a button press or after a scan event is called. The ability to capture the user's speech is not supported yet. It will speak in the language that is set by default in the device's OS configuration or the language called by TTS.SetLanguage.

Note: Not all devices have a built-in speaker which is required to use TTS.

Group: TTS

Syntax: TTS.Speak

Speak (String) – The sentence or phase to be read outloud.

Example:

```
Private Sub Speak_Click()
On Error Resume Next
TTS.Speak
End Sub
```

Versions Supported: RFgen 5.2.2.0 and newer.

Devices Supported: Android, iOS.

TTS.Spell

The option makes the Android device spell the letters of a text outloud. While it can be called any user input, the button is most preferred.

Group: TTS

Syntax: TTS.PauseTime

Example:

```
Private Sub btnSpell_Click()
On Error Resume Next
TTS.Spell = "テスト"
TTS.Spell = "Orientation"
End Sub
```

Versions Supported: RFgen 5.2.2.0 and newer.

Devices Supported: Android, iOS.

TTS.StopSpeak

The StopSpeak option stops the Android device from reading the text outloud when its called by any type of user input. It can be set to a button press or after a Speak event is called. The ability to capture user speech is not supported yet.

Group: TTS

Syntax: TTS.StopSpeak

Example:

```
Private Sub btnStopSpeaking_Click()
On Error Resume Next
TTS.StopSpeak
```

```
End Sub
```

Versions Supported: RFgen 5.2.2.0 and newer.

Devices Supported: Android, iOS.

TTS.Volume

The volume is set to 100 by default. It is based off of percentages, where 0 is silent and 100 is maximum volume.

Group: TTS

Syntax: TTS.Volume

Example:

```
TTS.Volume = 50
```

Versions Supported: RFgen 5.2.2.0 and newer.

Devices Supported: Android, iOS.

Transaction Management Extensions

Commands relating to Transaction Management capabilities and queuing are called from the TM object.

The extensions available for this object are:

TM.AbortTrans, TM.CheckStatus, TM.GetItems

TM.GetItemsEx, TM.MacroName, TM.MoveQueue

TM.QueueMacro,TM.QueueName, TM.SeqNo

AbortTrans

This function can be used inside a Transaction macro to halt processing of the transaction if conditions warrant it. This command will notify the Transaction Manager that the current macro processing was aborted for a reason other than failure. The Transaction Manager will then keep this transaction at the top of the queue and try again on the next cycle. For example, if an SM command comes back with a failed status or the BeginTrans command fails, TM.AbortTrans can be executed to give the transaction macro's Boolean value a False value. In the application's script, you will know if the called macro was successful.

Group: Transaction Management

Syntax: TM.AbortTrans

Example:

TM.AbortTrans

CheckStatus

This method looks up the status of a queued or processed transaction and returns what happened to that transaction. The return options are that the macro cannot be found, it failed, it succeeded, it is still queued and the macro couldn't be executed.

Group: Transaction Management

Syntax: enResult = TM.CheckStatus(sQueue, nSeqNo)

enResult	(enMacroResults) the enumeration describing what was found. Possible values are MacroFailed, MacroNotFound, MacroNotProcessed, MacroQueued and MacroSucceeded.
sQueue	(String) the name of the queue to be searched
nSeqNo	(Long) the sequence number to look for

Example:

```
Dim enValue As enMacroResults

enValue = TM.CheckStatus("RFQueue", 100)
```

GetItems

This method loads a set of macros to be evaluated from the Transaction Management tables. Selecting the category, date and user, this command will return a DataRecord containing the macros and their data. If a list of macros with a more complex selection criteria is desired use TM.GetItemsEx.

Group: Transaction Management

Syntax: [bOK =] TM.GetItems(enStatus, vTranDate, sUserID, oTran)

bOK	(Boolean) Optional – True or False for the success of the command
enStatus	(enItemStatus) There are 4 options for the ItemStatus parameter:
	tmAllItems all macros in all queues, in the queue, failed, or completed
	tmCompleted all macros in all queues that were completed successfully
	tmFailed all macros in all queues that were completed unsuccessfully
	tmInProcess all macros in all queues that are not yet processed
vTranDate	(Variant) specifies a single day to be retrieved
sUserID	(String) specifies the user that performed the transaction
oTran	(DataRecord) the variable declared as a DataRecord that contains the list of macros and their data. For more information on how the DataRecord object is used see the DataRecord section of the manual.

Example:

```
Dim bOK As Boolean
```

```
Dim oList As DataRecord
bOK = TM.GetItems(tmFailed, Date, "SAM", oList)
```

This command will return all the failed macros on the current day that was performed by the user Sam.

GetItemsEx

This method loads a set of macros to be evaluated from the Transaction Management tables. Specify a SQL statement that will retrieve the desired recordset and take advantage of the complexity allowed by ODBC. Knowledge of the table and field names is required.

Group: Transaction Management

Syntax: [bOK =] TM.GetItemsEx(sSQL, oTran)

bOK (Boolean) Optional – True or False for the success of the command
sSQL (String) specifies the SQL statement for retrieving a list of macros
oTran (DataRecord) the variable declared as a DataRecord that contains the list of macros and their data. For more information on how the DataRecord object is used see the DataRecord section of the manual.

The table names required depend on the name of the queue configured in the Transaction Management setup. For the RFQueue the tables created are:

RFQueue – used to store the InProcess items
RFQueue_Completed – used to store successfully completed items
RFQueue_Failed – used to store failed transaction macros

Substitute the RFQueue name for alternate queues that may have been defined. The fields that can / should be referenced through SQL within the tables are shown below.

RFQueue table:

Field Name	Data Type	Description
SeqNo	Number	the sequence number
TranDate	Number	when the macro was queued (20151231)
TranTime	Number	when the macro was queued (235959)
Source	Text	linked source for the macro if it exists
Name	Text	name of the macro
Record	Text	contains the passed values to the macro
FormId	Text	form that queued the macro
UserId	Text	user that queued the macro
DeviceNo	Number	device number assigned to the client
IPAddress	Text	IP address of the client device
DevGuid	Text	assigned GUID of the client device
ExecDate	Number	date the macro executed (20151231)
ExecTime	Number	time the macro executed (235959)

Status Number integer status of the macro

The **RFQueue_Completed** table and **RFQueue_Failed** table have the same design as RFQueue.

Example:

```
Dim oList As DataRecord
TM.GetItemsEx("Select * from RFQueue where UserId = 'SAM'", oList)
```

MacroName

This returns the name of the macro currently in process by the Transaction Manager. This command is only available while the transaction macro is being executed so it must be used either in the macro itself or in any global function calls that may be used to process generic macros such as a logging feature.

Group: Transaction Management

Syntax: sValue = TM.MacroName

sValue (String) the name of the macro being executed

Example:

```
Dim sName As String
sName = TM.MacroName
```

MoveQueue

This function will take a queue from one database and guarantee its delivery to another database. If another instance of the server is monitoring the second database, that instance will become responsible for executing the queued transactions. The Transaction Management database connection must be capable of seeing both databases.

Group: Transaction Management

Syntax: [bValue =] TM.MoveQueue(sFromQueue, sToQueue)

bValue (Boolean) Optional – the success or failure to move the queue from one database to another.

sFromQueue (String) the source queue to be moved

sToQueue (String) the destination queue to receive the transactions

Example:

```
Dim bValue As Boolean
bValue = TM.MoveQueue("RFQueue", "HostQueue")
bValue = TM.MoveQueue("RFQueue", "AltDB.HostQueue")
```

QueueMacro

This function is used to queue Data Transaction macros and pass any required parameters. It returns the sequence number of the macro after it has been successfully queued. These macros can be created on the Transactions tree.

Group: Transaction Management

Syntax: [nSeq =] TM.QueueMacro(sMacroName, [vParams])

nSeq (Long) Optional – the sequence number of the macro after it has been successfully queued.

sMacroName (String) is the name of the macro to be called.

vParams (Variant) Optional – A series of passing parameters as required by the selected macro. Note: these parameters are the fields you defined when you wrote the macro.

Example:

```
Dim nSeq As Long
nSeq = TM.QueueMacro("ChangeUser", "Sam", "1234")
```

QueueName

This function will return the name of the queue currently being processed if this command is executed from within the macro itself.

Group: Transaction Management

Syntax: sValue = TM.QueueName

sValue (String) is the name of the queue

Example:

```
Dim sValue As String
sValue = TM.QueueName
```

SeqNo

This function will return the sequence number of the transaction currently being processed if this command is executed from within the macro itself.

Group: Transaction Management

Syntax: Syntax: nValue = TM.SeqNo

nValue (Long) is the sequence number

Example:

```
Dim nValue As Long
nValue = TM.SeqNo
```

Web Object

The Web object is provided to the programmer for managing the Web data connector configured as one of the data connections.

The extensions available for the web object are:

oWeb.ConnectTimeout, oWeb.DataSource, oWeb.Execute,

oWeb.HeaderValue, oWeb.QueryType, oWeb.ReceiveTimeout,

oWeb.Reply, oWeb.Request, and oWeb.SendTimeout

ConnectTimeout

The ConnectTimeout property returns or sets the timeout to connect for the HTTP request. When the language extension is created it will initialize this value to what is configured in the data connection configuration.

Group: Web Object

Syntax: nValue = oWeb.ConnectTimeout

Alternate: oWeb.ConnectTimeout = Value

nValue (Long) the number of milliseconds the system will wait for a connection

Example:

```
Dim oWeb As Web
Dim nTimeout As Long
nTimeout = oWeb.ConnectTimeout
oWeb.ConnectTimeout = 10000
```

DataSource

The DataSource property returns or sets an index of the data connection that should be used. If a DataSource is not specified the language extension will use the first data connection that is configured as a WEB connection.

Group: Web Object

Syntax: vValue = oWeb.DataSource

Alternate: oWeb.DataSource = vValue

vValue (Variant) the number or name of the data connection object.

Example:

```
Dim oWeb As Web
Dim vConnID As Variant
oWeb.DataSource = 2
oWeb.DataSource = "MyWebServer"
vConnID = oWeb.DataSource
```

EndPoint

HeaderValue

The HeaderValue property returns or sets what the HTTP or HTTPS header values are.

Group: Web Object

Syntax: oWeb.HeaderValue(sIndex) = bValue

Alternate: bValue = oWeb.HeaderValue(sIndex)

bValue (Boolean) contains a True for False for the success of the HeaderValue assignment.

sIndex (String) the property name for the HTTP header

Example:

```
Dim oWeb As Web
oWEB.HeaderValue("Content-Type") = "text/xml; charset=utf-8"
oWEB.HeaderValue("SOAPAction") = """http://xml.namespaces.xerox.com/im /xip/services/xcllmswebservice/wsdl/getDistributionRefurbInfo"""
```

InParam

For future use.

OutParam

For future use.

QueryType

This property sets how the Web object will communicate with the web server. Get, Put and Post are the available options.

Group: Web Object

Syntax: oWeb.QueryType = enValue

Alternate: enValue = oWeb.QueryType

enValue (QueryType) contains three options:

HTTP_GET – it is for obtaining a resource, without changing anything on the server.

HTTP_POST – sends data to a specific URI and expects the resource at that URI to handle the request. The web server at this point can determine what to do with the data in the context of the specified resource.

HTTP_PUT – puts a file or resource at a specific URI, and exactly at that URI. If there's already a file or resource at that URI, PUT replaces that file or resource. If there is no file or resource there, PUT creates one.

The fundamental difference between the POST and PUT requests is reflected in the different meaning of the Request-URI. The URI in a POST request identifies the resource that will handle the enclosed entity. That resource might be a data-accepting process, a gateway to some other protocol, or a separate entity that accepts annotations. In contrast, the URI in a PUT request identifies the entity enclosed with the request -- the user agent knows what URI is intended and the server MUST NOT attempt to apply the request to some other resource. If the server desires that the request be applied to a different URI, it MUST send a 301 (Moved Permanently) response; the user agent MAY then make its own decision regarding whether or not to redirect the request.

Example:

```
Dim enType As QueryType
Dim oWeb As Web
oWEB.QueryType = HTTP_POST
enType = oWEB.QueryType
```

ReceiveTimeout

The ReceiveTimeout property returns or sets the timeout to receive a reply to the HTTP request. If a reply has not been completely received before the timeout then the request will be terminated. When this language extension is created it will initialize this value to what is configured in the data connection configuration.

Group: Web Object

Syntax: nValue = oWeb.ReceiveTimeout

Alternate: oWeb.ReceiveTimeout = nValue

nValue (Long) the number of milliseconds the system will wait for a connection

Example:

```
Dim oWeb As Web
Dim nTimeout As Long
nTimeout = oWeb.ReceiveTimeout
oWeb.ReceiveTimeout = 20000
```

Reply

The Reply property returns the data portion of the HTTP response. This is where the data will be held when the request returns. Keeping the Data and Reply properties separate allows Execute to be called multiple times without requiring the Data property to be reset.

Group: Web Object

Syntax: sValue = oWeb.Reply

sValue (String) the data being returned to the client from the server

Example:

```
Dim oWeb As Web
Dim sValue As String
sValue = oWeb.Reply
```

Request

The Request property returns or sets the path to the ASP page (if executed against a Windows server) that is then appended to the URL that is used for connecting.

Group: Web Object

Syntax: oWeb.Request = sValue

Alternate: sValue = oWeb.Request

sValue (String) part of the URL that is the path to the ASP page

Example:

```
Dim oWeb As Web
Dim sRequest As String
oWeb.Request = "/data/DoWork.asp?num=100620?MySQLTable"
sRequest = oWeb.Request
```

SendTimeout

The SendTimeout property returns or sets the timeout value to send the HTTP request. If the send is not complete before the timeout then the request will be terminated. When the language extension is created it will initialize this value to what is configured in the data connection configuration.

Group: Web Object

Syntax: nValue = oWeb.SendTimeout

Alternate: oWeb.SendTimeout = nValue

nValue (Long) the number of milliseconds the system will wait for submitting a HTTP request

Example:

```
Dim oWeb As Web
Dim nTimeout As Long
nTimeout = oWeb.SendTimeout
oWeb.SendTimeout = 30000
```

Execute

The Execute method will execute the configured oWeb object. vData is optional and is equivalent to the XML data in a SOAP request.

Group: Web Object

Syntax: bValue = oWeb.Execute([vData])

bValue (Boolean) is True or False depending on the success of the submission to the web server.

vData (Variant) Optional – an object that could contain the individual properties.

Example:

```
Dim oWeb As Web
If oWeb.Execute Then
 RFPrompt("txtReply").Text = oWeb.Reply
End If
```

Login

The Login method is for future use.

Logout

The Logout method is for future use.

WWB.NET: Overview

WinWrap Basic (WWB) is a third-party resource and component that enables RFgen to provide its customers with additional scripting features when coding an application in the Mobile Development Studio. The WWB help is provided for .NET and COM.

WWB.NET	Use **'#Language "WWB.NET"** for Visual Basic.NET(TM) compatibility.
Declaration	**'#Language**, **'#Reference**, **'#Uses**, **Attribute**, **Class Module**, **Code Module**, **Const**, **Declare**, **Delegate**, **Dim**, **Enum...End Enum**, **Event**, **Function...End Function**, **Imports**, **Object Module**, **Option**, **Private**, **Property...End Property**, **Public**, **ReDim**, **Static**, **Structure...End Structure**, **Sub...End Sub**, **WithEvents**.
Data Type	**Any**, **Boolean**, **Byte**, **Char**, **Date**, **Decimal**, **Double**, **Integer**, **Long**, **Object**, **SByte**, **Short**, **Single**, **String**, **UInteger**, **ULong**, **UShort**, *obj type*, *user dialog*, *user enum*, *user structure*.
Assignment	**Assign**: (=, +=, -=, *=, /=, \=, ^=, <<=, >>=), **Erase**
Flow Control	**AddHandler**, **Call**, **CallByName**, **Do...Loop**, **End**, **Exit**, **For...Next**, **For Each...Next**, **GoTo**, **If...ElseIf...Else...End If**, **MacroCheck**, **MacroCheckThis**, **MacroRun**, **MacroRunThis**, **ModuleLoad**, **ModuleLoadThis**, **RaiseEvent**, **RemoveHandler**, **Return**, **Select Case...End Select**, **Stop**, **While...End While**.
Error Handling	**Err**, **Error**, **ErrorToString**, **On Error**, **Resume**, **Throw**, **Try**, **Using**
Conversion	**CBool**, **CByte**, **CChar**, **CDate**, **CDec**, **CDbl**, **CInt**, **CLng**, **CObj**, **CSByte**, **CShort**, **CSng**, **CStr**, **CType**, **CUInt**, **CULng**, **CUShort**, **CVErr**, **DirectCast**, **TryCast**, **Val**.
Variable Info	**IsArray**, **IsDate**, **IsDBNull**, **IsError**, **IsNothing**, **IsNumeric**, **IsReference**, **LBound**, **SystemTypeName**, **TypeName**, **UBound**, **VarType**, **VbTypeName**.
Constant	**False**, **Nothing**, **True**, **Win16**, **Win32**, **Win64**.
Math	**Abs**, **Atn**, **Cos**, **Exp**, **Fix**, **Int**, **Log**, **Randomize**, **Rnd**, **Round**, **Sgn**, **Sin**, **Sqr**, **Tan**.
String	**Asc**, **AscW**, **Chr**, **ChrW**, **Format**, **GetChar**, **Hex**, **InStr**, **InStrRev**, **Join**, **LCase**, **Left**, **Len**, **LSet**, **LTrim**, **Mid**, **Oct**, **Replace**, **Right**, **RSet**, **RTrim**, **Space**, **Split**, **Str**, **StrComp**, **StrConv**, **StrDup**, **StrReverse**, **Trim**, **UCase**.
Object	**CreateObject**, **GetObject**, **Me**, **With...End With**.
Time/Date	**DateAdd**, **DateDiff**, **DatePart**, **DateSerial**, **DateValue**, **Day**, **Hour**, **Minute**, **Month**, **MonthName**, **Now**, **Second**, **Timer**, **TimeSerial**, **TimeValue**, **Weekday**, **WeekdayName**, **Year**.
File	**ChDir**, **ChDrive**, **CurDir**, **Dir**, **EOF**, **FileAttr**, **FileClose FileCopy**, **FileDateTime**, **FileLen**, **FileOpen FreeFile**, **Get**, **GetAttr**, **Input**, **InputString**, **Kill**, **Line Input**, **LineInput**, **Loc**, **Lock**, **LOF**, **MkDir**, **Print**,

	PrintLine, **Put**, **Reset**, **Rename**, **RmDir**, **Seek**, **Seek**, **SetAttr**, **Unlock**, **Write**, **WriteLine**.
User Input	**Dialog**, **GetFilePath**, **InputBox**, **MsgBox**. **ShowPopupMenu**
User Dialog	**Begin Dialog...End Dialog**, **CancelButton**, **CheckBox**, **ComboBox**, **DropListBox**, **GroupBox**, **ListBox**, **MultiListBox**, **OKButton**, **OptionButton**, **OptionGroup**, **Picture**, **PushButton**, **Text**, **TextBox**.
Dialog Function	**DialogFunc**, **DlgControlId**, **DlgCount**, **DlgEnable**, **DlgEnd**, **DlgFocus**, **DlgListBoxArray**, **DlgName**, **DlgNumber**, **DlgSetPicture**, **DlgText**, **DlgType**, **DlgValue**, **DlgVisible**.
DDE	**DDEExecute**, **DDEInitiate**, **DDEPoke**, **DDERequest**, **DDETerminate**, **DDETerminateAll**.
Settings	**DeleteSetting**, **GetAllSettings**, **GetSetting**, **SaveSetting**
Miscellaneous	**AboutWinWrapBasic**, **AppActivate**, **Assign**, **Attribute**, **Beep**, **CallersLine**, **Choose**, **Clipboard**, **Command**, **Decode64**, **Decode64B**, **Decrypt64**, **Decrypt64B**, **Debug**.Print, **DoEvents**, **Encode64**, **Encode64B**, **Encrypt64**, **Encrypt64B**, **Environ**, **Eval**, **IIf**, **GetLocale**, **KeyName**, **MacroDir**, **QBColor**, **Rem**, **RGB**, **SendKeys**, **SetLocale**, **Shell**, **Wait**.
Operator	**Operators**: +, -, ^, *, /, \, Mod, +, -, <<, >>, &, =, <>, <, >, <=, >=, **Like**, **New**, **TypeOf**, Not, And, AndAlso, Or, OrElse, Xor, Eqv, Imp, **Is**, **IsNot**, **GetType**, **AddressOf**.

AboutWinWrapBasic Instruction

Syntax AboutWinWrapBasic [*Timeout*]

Group Miscellaneous

Description Show the WinWrap Basic about box.

Parameter	Description
Timeout	This numeric value is the maximum number of seconds to show the about box. A value less than or equal to zero displays the about box until the user closes it. If this value is omitted then a three second timeout is used.

Example

```
'#Language "WWB.NET"
Sub Main
  AboutWinWrapBasic
End Sub
```

Terms

arglist [| *expr* | *param*:=*expr*][, ...]

A list of zero or more *expr*s that are assigned to the parameters of the *procedure*.

- A positional parameter may be skipped by omitting the expression. Only optional parameters may be skipped.
- Positional parameter assignment is done with *expr*. Each parameter is assigned in turn. By name parameter assignment may follow.
- By name parameter assignment is done with *param*:=*expr*. All following parameters must be assigned by name.

arrayvar A variable that holds an array of values. A *Object* variable can hold an array. Dynamic arrays can be **ReDim**ensioned.

As [New] type As type
-or-

As New *objtype*[(Of *objtype*[, ...])][([*param*[, ...]])]

Dim, **Private**, **Public** and **Static** statements may declare variable types using *As type* or As New *objtype*. A variable declared using As New *objtype* is created using a list of zero or more *param*s.

As type Variable and parameter types, as well as, function and property results may be specified using As type: **Boolean**, **Byte**, **Char**, **Date**, **Decimal**, **Double**, **Integer**, **Long**, **Object**, **SByte**, **Short**, **Single**, **String**, **UInteger**, **ULong**, **UShort**, *objtype*, *user delegate*, *user dialog*, *user enum*, *user structure*.

attribute A file attribute is zero or more of the following values added together.

Attribute	Value	Description
vbNormal	0	Normal file.
vbReadOnly	1	Read-only file.
vbHidden	2	Hidden file.
vbSystem	4	System file.
vbVolume	8	Volume label.
vbDirectory	16	MS-DOS directory.
vbArchive	32	File has changes since last backup.
vbAlias	64	File is an alias.

big-endian Multiple byte data values (not strings) are stored with the highest order byte first. For example, the long integer &H01020304 is stored as this sequence of four bytes: &H01, &H02, &H03 and &H04. A Binary or Random file written using **Put** uses *little-endian* format so that it can be read using **Get** on any machine. (Big-endian machines, like the Power-PC, reverse the bytes as they are read by **Get** or written by **Put**.)

bytearray A variable that holds an array of byte values.

caseexpr An expression which specifies a single value or a range. Refer to the **Select Case** statement.

charlist A group of one or more characters enclosed by [ ] as part of **Like** operator's right string expression.

- This list contains single characters and/or character ranges which describe the characters in the list.
- A range of characters is indicated with a hyphen (-) between two characters. The first character must be ordinally less than or equal to the second character.
- Special pattern characters like ?, *, # and [can be matched as literal characters.
- The] character can not be part of charlist, but it can be part of the pattern outside the charlist.

condexpr An expression that returns a numeric result. If the result is zero then the conditional is **False**. If the result is non-zero then the conditional is **True**.

```
0 'false
-1 'true
X > 20 'true if X is greater than 20
' = "hello" 'true if ' equals "hello"
```

dateexpr An expression that returns a **date** result. Use #literal-date# to express a date value.

```
#1/1/2000# ' Jan 1, 2000
Now+7 ' seven days from now
DateSerial(Year(Now)+1,Month(Now),Day(Now))
  ' one year from now
```

dialogfunc A dialog function executes while a *user dialog* is visible.

dimension [*lower* To] *upper*

Array dimension.

dlgvar A dialog variable holds values for fields in the dialog. Dialog variables are declared using **Dim** dlgvar As *user dialog*.

expr An simple or complex expression that returns the appropriate result.

- Simple: *var*, *cond*, *date*, *num*, *str*, *obj*, *field*, *method*, **function** (result) or **property** (result).
- Complex: One or more simple expressions with parenthesises and **Operators**.

field Use .field to access individual fields in a dialog variable.

```
dlg.LastNam'
dlg.ZipCode
```

initialvalue Initial value for a variable. Use { *expr*, ... } to create an array value.

instruction A single command.

```
Beep
Debug.Print "Hello"
Today = Date
```

Multiple instructions may be used instead of a single instruction by separating the single instructions with colons.

```
X = 1:Debug.Print X
If X = 1 Then Debug.Print "X=";X:Stop
Beep ' must resume from Stop to get to here
```

label An identifier that *name*s a statement. Identifiers start with a letter. Following chars may be a letter, an underscore or a digit.

little-endian Multiple byte data values (not strings) are stored with the lowest order byte first. For example, the long integer &H01020304 is stored as this sequence of four bytes: &H04, &H03, &H02 and &H01. A Binary or Random file written using **Put** uses little-endian format so that it can be read using **Get** on any machine. (*Big-endian* machines, like the Power-PC, reverse the bytes as they are read by **Get** or written by **Put**.)

macro A macro is like an application. Execution starts at the macro's Sub **Main**.

method An object provides methods and *properties*. Methods can be called as subs (the return value is ignored), or used as functions (the return value is used).

If the method name contains characters that are not legal in a *name*, surround the method name with [].

```
App.[Title$]
```

module A file with **public** symbols that are accessible by other modules/*macros* via the **'#Uses** special comment.

- A module is loaded on demand.
- A **code module** is a code library.
- An **object module** or **class module** implements an object.
- A module may also access other modules with its own **'#Uses** special comments.

name An identifier that names a variable or a user defined *procedure*. Identifiers start with a letter. Following chars may be a letter, an underscore or a digit.

```
Count
DaysTill2000
Get_Data
```

num An expression that returns a numeric result. Use &O to express an octal number. Use &H to express a hex number. The interpretation &H and &O is affected by the **'#Language** setting. Specific types of numbers can be indicated by using one of the endings shown in the table below:

Ending	Description
S	The number is a **Short** value.
I or %	The number is a **Integer** value. The interpretation of the type is controlled by the **'#Language** setting.
L or &	The number is a **Long** value. The interpretation of the type is controlled by the **'#Language** setting.
US	The number is a **UShort** value.

UI	The number is a **UInteger** value. The interpretation of the type is controlled by the **'#Language** setting.
UL	The number is a **ULong** value. The interpretation of the type is controlled by the **'#Language** setting.
D	The number is a **Decimal** value.
F or !	The number is a **Single** value.
R or #	The number is a **Double** value.
@	The number is a **Decimal** value. The interpretation of the type is controlled by the **'#Language** setting.

numstr An expression that returns a *numeric* or *string* result.

numvar A variable that holds one numeric value.

objexpr A expression that returns a reference to an object or *module*.

```
CreateObject("WinWrap.CDemoApplication")
```

objtype A specific type defined by your application, another application or by an **object module** or **class module**.

objvar A variable that holds a *objexpr* which references an object. Object variables are declared using As *Object* in a **Dim**, **Private** or **Public** statement.

param [[Optional] [| ByVal | ByRef] | [ByVal] ParamArray] *param*[*type*][( )] [*As type*] [= *defaultvalue*]

The *param* receives the value of the associated expression in the **Declare**, **Sub**, **Function** or **Property** call. (See *arglist*.)

- An Optional *param* may be omitted from the call. It must also have a *defaultvalue*. The parameter receives the defaultvalue if a value is not specified by the call.
- All parameters following an Optional parameter must also be Optional.
- ParamArray may be used on the final *param*. It must be an array of **Object** type. It must not follow any Optional parameters. The ParamArray receives all the expressions at the end of the call as an array. If **LBound** (*param*) > **UBound**(*param*) then the ParamArray didn't receive any expressions.
- If the *param* is not ByVal and the expression is merely a variable then the *param* is a reference to that variable (ByRef). (Changing *param* changes the variable.) Otherwise, the parameter variable is local to the *procedure*, so changing its value does not affect the caller.
- Use *param*( ) to specify an array parameter. An array parameter must be referenced and can not be passed by value. The bounds of the parameter array are available via **LBound( )** and **UBound( )**.

precedence When several operators are used in an expression, each operator is evaluated in a predetermined order. Operators are evaluated in this order:

- ^ (power)
- - (negate)
- * (multiply), / (divide)

- \ (integer divide)
- Mod (integer remainder)
- + (add), - (difference)
- << (shift left), >> (shift right)
- & (string concatenate)
- = (equal), <> (not equal), < (less than) > (greater than), <= (less than or equal to), >= (greater than or equal to), **Like**, (string similarity) **New**, (object creation) **TypeOf**, (object type) **Is**, (object equivalence) **IsNot** (object non-equivalence)
- Not (bitwise invert)
- And (bitwise and), AndAlso (short-circuit logical and)
- Or (bitwise or), OrElse (short-circuit logical or)
- Xor (bitwise exclusive-or)

Operators shown on the same line are evaluated from left to right.

procedure A **sub**routine, **function** or **property**.

property An object provides *method*s and properties. Properties may be used as values (like a function call) or changed (using assignment syntax).

If the property name contains characters that are not legal in a *name*, surround the property name with [].

```
App.[Title$]
```

statement Zero or more *instruction*s. A statement is at least one line long. **Begin Dialog**, **Do**, **For**, **If** (multiline), **Select Case**, **While** and **With** statements are always more than one line long. A single line statement continues on the next line if it ends a line with a space and an underscore ' _'.

```
' = "This long string is easier to read, " + _
    "if it is broken across two lines."
Debug.Print '
```

str An expression that returns a string result.

```
"Hello"
'
' + " Goodbye"
' & " Goodbye"
Mi'(',2)
```

A string constant may have a 'C' suffix

```
' = "x"C
```

The 'C' suffix indicates that the string is one character long.

strarray A variable that holds an array of string values.

streamnum An expression that returns a numeric result. Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

strvar A variable that holds one string value.

```
FirstNam'
```

type Variable and parameter types, as well as, function and property results may be specified using a type character as the last character in their name.

Type char	As Type
%	**Integer**, The interpretation of the type is controlled by the **'#Language** setting.
&	**Long**, The interpretation of the type is controlled by the **'#Language** setting.
!	**Single**
#	**Double**
@	**Decimal**, The interpretation of the type is controlled by the **'#Language** setting.
$	**String**

user delegate User defined delegates are defined with **Delegate**.

user dialog User defined dialogs are defined with **Begin Dialog**.

user enum User defined enums are defined with **Enum**.

user structure User defined structures are defined with **Structure**.

userstructurevar A user defined structure variable holds values for elements of the user defined structure. User defined structures are defined using **Structure**.

- Declare with **Dim**, **Private**, **Public** or **Static**.
- Declare as a parameter of **Sub**, **Function** or **Property** definition.

var A variable holds either a string, a numeric value or an array of values depending on its type.

vardeclaration *name*[*type*][([*dimension*[, ...]])][*As [New] type*]

The *name* declares a variable.

variantvar A variant variable can hold any type of value or it can hold an array.

#Language Special Comment

Syntax '#Language "WWB.NET"

Group Declaration

Description Selects Visual Basic.NET(TM) compatibility.

If this special comment is not included then the macro/module is parsed using '#Language "WWB-COM" rules without the enhancements.

Language reference by **group**:

- *Declaration*, *Data Type*, *Assignment*
- *Flow Control*, *Error Handling*
- *Conversion*, *Variable Info*, *Constant*
- *Math*, *String*, *Object*, *Time/Date*, *File*
- *User Input*, *User Dialog*, *Dialog Function*

- *DDE, Settings, Miscellaneous*
- *Operator*

These VB.NET keywords are recognized, but not supported: FileGet, FileGetObject, FilePut, FilePutObject, Inherits, Interface, MyBase, MyClass, MustInherit, MustOverride, Namespace, Narrowing, NotInheritable, NotOverridable, Operator, Overloads, Overridable, Overrides, Partial, Protected, SyncLock, Widening.

Version Available for WinWrap Basic version 9.1 or higher. (Not supported in Windows CE Applications.)

#Reference Special Comment

Syntax

```
'#Reference {uuid}#vermajor.verminor#lcid#[path[#name]]
-or-
'#Reference #assembly
```

Description The Reference comment indicates that the current *macro*/*module* references the COM type library (or .NET assembly) identified. Reference comment lines must be the first lines in the macro/module (following the global **Attribute**s). Reference comments are in reverse priority (from lowest to highest). The IDE does not display the reference comments.

Parameter	Description
uuid	Type library's universally unique identifier.
vermajor	Type library's major version number.
verminor	Type library's minor version number.
lcid	Type library's locale identifier.
path	Type library's path.
name	Type library's name.
assembly	Fully qualified assembly name.

Examples

```
'#Reference {00025E01-0000-0000-C000-000000000046}#4.0#0#C:\PROGRAM
FILES\COMMON FILES\MICROSOFT SHARED\DAO\DAO350.DLL#Microsoft DAO
3.5 Object Library
```

For a .Net assembly determine the fully qualified name first using the GACUTIL.exe /lr <assembly name> or other means and then:

```
'#Reference #Class1, Version=1.0.0.0, Culture=neutral, PublicKeyToken-
n=3d01b60ac60ef95e, processorArchitecture=MSIL
```

Note that the status line at the bottom of the VBA Modules window will display an error if something is not correct. The Reference line entered will disappear after pressing enter at the end of the line.

The GACUTIL.exe /i can be used to install an assembly into the Global Assembly Cache. To verify, find the DLL's name in \Windows\Assembly. Changes to the

DLL must be re-installed to the Global Assembly Cache to take effect. If RFgen is already running it may need to be restarted to see the change.

#Uses Special Comment

Syntax '#Uses "*module*" [Only:[**Win16**|**Win32**|Win64]] ...
-or-
'$Include: "*module*"

Group Declaration

Description The Uses comment indicates that the current *macro*/*module* uses public and friend symbols from the *module*. The Only option indicates that the module is only loaded for that Windows platform.

Parameter	Description
module	Public and Friend symbols from this module are accessible. If the module name is a relative path then the path is relative to the macro/module containing the Uses comment. For example, if module "A:\B\C\D.BAS" has this uses comment: '#Uses "E.BAS" then it uses "A:\B\C\E.BAS".

See Also **Class Module**, **Code Module**, **Object Module**.

Example

```
'Macro A.WWB
'#Language "WWB.NET"
'#Uses "B.BAS"
Sub Main
   Debug.Print BFun'("Hello") '"HELLO"
End Sub

'Module B.BAS
'#Language "WWB.NET"
Public Function BFun'(')
   BFun' = UCase(')
End Function
```

Abs Function

Syntax Abs(*Num*)

Group Math

Description Return the absolute value.

Parameter	Description
Num	Return the absolute value of this numeric value. '

See Also **Sgn**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Abs(9)  ' 9
  Debug.Print Abs(0)  ' 0
  Debug.Print Abs(-9) ' 9
End Sub
```

AddHandler Instruction

Syntax AddHandler *expr.name*, *user delegate*

Group Flow Control

Description Add a *user delegate* to the *expr.name*'s list of handlers.

Parameter	Description
expr	This is an expression which returns an object reference.
name	This is an event name.
user delegate	This is an expression which returns a delegate.

See Also **RemoveHandler**.

Example

```
'#Reference #System.Windows.Forms, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089, processorArchitecture=MSIL
'#Language "WWB.NET"
Imports System.Windows.Forms

Sub Main
  Using t As New Timer
    AddHandler t.Tick, AddressOf OnTick
    t.Interval = 1000
    t.Enabled = True
    Wait 5
    RemoveHandler t.Tick, AddressOf OnTick
  End Using
End Sub

Private Sub OnTick
  Debug.Print Now
End Sub
```

AddressOf Operator

Syntax AddressOf [*expr.*]*name*

Group Operator

Description Return a delegate for the *expr macro/module*'s Sub or Function matching *name*.

A delegate's Sub or Function is called using the delegate's Invoke method.

Note: The AddressOf operator returns an object which holds a weak reference to the macro/module. If no other references to the macro/module exist then the macro/module reference is deleted and using the result of the AddressOf operator causes a run-time error.

Parameter	Description
expr	This is a macro/module reference. If omitted then the current macro/module's reference (Me) is used.
name	Return a delegate for this Sub or Function.

See Also **Delegate**.

Example

```
'#Language "WWB.NET"
Delegate Function OpType(ByVal v1 As Object, ByVal v2 As Object) As Object

Sub Main
  Debug.Print DoOp(AddressOf Add,1,2) ' 3
  Debug.Print DoOp(AddressOf Subtract,1,2) '-1
End Sub

Function DoOp(ByVal op As OpType, _
        ByVal v1 As Object, ByVal v2 As Object) As Object
  DoOp = op.Invoke(v1,v2)
End Function

Function Add(ByVal v1 As Object, ByVal v2 As Object) As Object
  Add = v1+v2
End Function

Function Subtract(ByVal v1 As Object, ByVal v2 As Object) As Object
  Subtract = v1-v2
End Function
```

Any Data Type

Syntax **Declare** ...(... v As Any ...)...

Group Data Type

Description Any variable expression (**Declare** only).

AppActivate Instruction

Syntax AppActivate *Titl'*
-or-
AppActivate *TaskID*

Group Miscellaneous

Description Form 1: Activate the application top-level window titled *Titl'*. If no window by that title exists then the first window with at title that starts with *Titl'* is

activated. If no window matches then an error occurs.

Form 2: Activate the application top-level window for task *TaskID*. If no window for that task exists then an error occurs.

Parameter	Description
Titl'	The name shown in the title bar of the window.
TaskID	This numeric value is the task identifier.

See Also **SendKeys**, **Shell( )**.

Example

```
'#Language "WWB.NET"
Sub Main
   ' make ProgMan the active application
   AppActivate "Program Manager"
End Sub
```

Asc Function

Syntax Asc(')

Group String

Description Return the ASCII value.

Note: A similar function, AscW, returns the Unicode value.

Parameter	Description
'	Return the ASCII value of the first char in this string value.

See Also **Ch'( )**.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print Asc("A") ' 65
End Sub
```

Assign Instruction

Syntax Assign *lhs*, *rhs*[, *Depth*]

Group Miscellaneous

Description Assign the value of the *rhs* to the *lhs*.

Parameter	Description
lhs	This string represents the variable expression to be set.
rhs	Evaluate this string value and assign it to the *lhs* expression.
Depth	This integer value indicates how deep into the stack to locate the local variables. If Depth = 0 then use the current *procedure*. If this value is omitted then the depth is 0.

See Also **Eval**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim X As String
  Assign "X", """Hello"""
  Debug.Print X 'Hello
  A
  Debug.Print X 'Bye
End Sub
Sub A
  Dim X As String
  Assign "X", """Welcome"""
  Assign "X", """Bye""", 1
  Debug.Print Eval("X") 'Welcome
  Debug.Print Eval("X",1) 'Bye
End Sub
```

Assign Operators

Syntax *var* [Op] *expr*

Group Assignment

Description Assign a value to *var*.

Op	Description
=	Assign the value of *expr* to *var*.
+=	The same as: *var* = *var* + (*expr*).
-=	The same as: *var* = *var* - (*expr*).
*=	The same as: *var* = *var* * (*expr*).
/=	The same as: *var* = *var* / (*expr*).
\=	The same as: *var* = *var* \ (*expr*).
^=	The same as: *var* = *var* ^ (*expr*).
<<=	The same as: *var* = *var* << (*expr*).
>>=	The same as: *var* = *var* >> (*expr*).
&=	The same as: *var* = *var* & (*expr*).

Example

```
'#Language "WWB.NET"
Sub Main
  X = 1
  X = X*2
  Debug.Print X ' 2
End Sub
```

Atn Function

Syntax Atn(*Num*)

Group Math

Description Return the arc tangent. This is the number of radians. There are 2*Pi radians in a full circle.

Parameter	Description
Num	Return the arc tangent of this numeric value.

See Also **Cos**, **Sin**, **Tan**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Atn(1)*4 ' 3.1415926535898
End Sub
```

Attribute Definintion/Statement

Syntax

```
Attribute attributename = value
Attribute varname.attributename = value
Attribute procname.attributename = value
```

Group Declaration

Description All attribute definitions and statements are ignored except for:

- Form 1: Module level attribute

  ```
  Attribute VB_Name = "name"
  Attribute VB_GlobalNameSpace = bool
  Attribute VB_Creatable = bool
  Attribute VB_PredeclaredId = bool
  Attribute VB_Exposed = bool
  Attribute VB_HelpID = int
  Attribute VB_Description = "text"
  ```

  VB_Name - Declares the name of the **class module** or **object module**.
  VB_GlobalNameSpace - Declares the class module as a global class. (ignored)
  VB_Creatable - Declares the module as creatable (True), non-creatable (False). (ignored)
  VB_PredeclaredId - Declares the module as a predeclared identifier (True). (ignored)
  VB_Exposed - Declares the module as public (True). (ignored)
  VB_HelpID - Declares the module's help context displayed by the object browser.
  VB_Description - Declares the module's help text displayed by the object browser.

- Form 2: Macro/Module level variable attribute

  ```
  Public varname As type
  Attribute varname.VB_VarUserMemId = 0
  Attribute varname.VB_VarHelpID = int
  Attribute varname.VB_VarDescription = "text"
  ```

  VB_VarUserMemID - Declares **Public** varname as the default property for a **class module** or **object module**.
  VB_VarHelpID - Declares the variable's help context displayed by the

object browser.
VB_VarDescription - Declares the variable's help text displayed by the object browser.

- Form 3: User defined procedure attribute

```
[Sub | Function | Property [Get|Let|Set]] procname ...
Attribute procname.VB_UserMemId = 0
Attribute procname.VB_HelpID = int
Attribute procname.VB_Description = "text"
  ...
End [Sub | Function | Property]
```

VB_UserMemID - Declares **Property** procname as the default property for a **class module** or **object module**.
VB_HelpID - Declares the procedure's help context displayed by the object browser.
VB_Description - Declares the procedure's help text displayed by the object browser.

HelpFile Each macro/module can define the HelpFile for the object browser:

```
'#HelpFile "helpfile"
```

where "helpfile" is a full path to the help file associated with the help text and help context.

Beep Instruction

Syntax Beep

Group Miscellaneous

Description Sound the bell.

Example

```
'#Language "WWB.NET"
Sub Main
  Beep ' beep the bell
End Sub
```

Begin Dialog Definition

Syntax

```
Begin Dialog name [X, Y,] DX, DY[, Titl'] _
      [, .dialogfunc]
  User Dialog Item
  [User Dialog Item]...
End Dialog
```

Group User Dialog

Description Define a *user dialog* type to be used later in a **Dim** As *name* statement.

Parameter	Description

X	This numeric value is the distance from the left edge of the screen to the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font. If this is omitted then the dialog will be centered.
Y	This numeric value is the distance from the top edge of the screen to the top edge of the dialog box. It is measured in 1/12 ths of the average character width for the dialog's font. If this is omitted then the dialog will be centered.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Titl'	This string value is the title of the user dialog. If this is omitted then there is no title.
dialogfunc	This is the function name that implements the **DialogFunc** for this *user dialog*. If this is omitted then the UserDialog doesn't have a dialogfunc.
User Dialog Item	One of: **CancelButton**, **CheckBox**, **ComboBox**, **DropListBox**, **GroupBox**, **ListBox**, **MultiListBox**, **OKButton**, **OptionButton**, **OptionGroup**, **PushButton**, **Text**, **TextBox**.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  Dialog dlg ' show dialog (wait for ok)
End Sub
```

Boolean Data Type

Syntax Dim v As Boolean

Group Data Type

Description A **True** or **False** value.

Byte Data Type

Syntax Dim v As Byte

Group Data Type

Description An 8 bit unsigned integer value.

Call Instruction

Syntax Call *name*[(*arglist*)]
-or-
name [*arglist*]

Group Flow Control

Description Evaluate the *arglist* and call subroutine (or function) *name* with those values. Sub (or function) *name* must be previously defined by either a **Sub**, **Function** or **Property** definition. If *name* is a function then the result is discarded. If Call is omitted and *name* is a subroutine then the *arglist* must not be enclosed in parens.

See Also **Declare**, **Sub**.

Example

```
'#Language "WWB.NET"
Sub Show(Titl',Value)
  Debug.Print Titl'; "="; Value
End Sub

Sub Main
  Call Show("2000/9",2000/9) ' 222.2222222222
  Show "1<2",1<2          'True
End Sub
```

CallByName Instruction

Syntax CallByName(*Obj*,*ProcName*,*CallType*,[*expr*[, ...]])

Group Flow Control

Description Call an *Obj*'s method/property, *ProcName*, by name. Pass the *expr*s to the method/property.

Parameter	Description
Obj	Call the method/property for this object reference.
ProcName	This string value is the name of the method/property to be called.
CallType	Type of method/property call. See table below.
expr	These expressions are passed to the obj's method/property.

CallType	Value	Effect
vbMethod	1	Call or evaluate the method.
vbGet	2	Evaluate the property's value.
vbLet	4	Assign the property's value.
vbSet	8	Set the property's reference.

Example

```
'#Language "WWB.NET"
Sub Main
  On Error Resume Next
  CallByName Err, "Raise", vbMethod, 1
  Debug.Print CallByName(Err, "Number", vbGet) ' 1
End Sub
```

CallersLine Function

Syntax CallersLine[(*Depth*)]

Group Miscellaneous

Description Return the caller's line as a text string.

The text format is: "[macroname|subname#linenum] linetext".

Parameter	Description
Depth	This integer value indicates how deep into the stack to get the caller's line. If Depth = -1 then return the current line. If Depth = 0 then return the calling sub-routine's current line, etc.. If Depth is greater than or equal to the call stack depth then a null string is returned. If this value is omitted then the depth is 0.

Example

```
'#Language "WWB.NET"
Sub Main
  A
End Sub
Sub A
  Debug.Print CallersLine "[(untitled 1)|Main#  2] A"
End Sub
```

CancelButton Dialog Item Definition

Syntax CancelButton *X*, *Y*, *DX*, *DY*[, .*Field*]

Group User Dialog

Description Define a cancel button item. Pressing the Cancel button from a **Dialog** instruction causes a run-time error. (**Dialog**( ) function call returns 0.)

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Field	This identifier is the name of the field. The *dialogfunc* receives this name as *string*. If this is omitted then the field name is "Cancel".

See Also **Begin Dialog**.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,30,"Please push the Cancel button"
```

```
    OKButton 40,90,40,20
    CancelButton 110,90,60,20
  End Dialog
  Dim dlg As UserDialog
  Dialog dlg ' show dialog (wait for cancel)
  Debug.Print "Cancel was not pressed"
End Sub
```

CBool Function

Syntax CBool(*expr*)

Group Conversion

Description Convert to a **Boolean** value. Zero converts to **False**, while all other values convert to **True**.

Parameter	Description
expr	Convert a number or string value to a boolean value.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print CBool(-1) 'True
  Debug.Print CBool(0)  'False
  Debug.Print CBool(1)  'True
End Sub
```

CByte Function

Syntax CByte(*expr*)

Group Conversion

Description Convert to an 8 bit unsigned integer **Byte** value.

Parameter	Description
expr	Convert a number or string value to an unsigned byte value.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print CByte(1.6) ' 2
End Sub
```

CChar Function

Syntax CChar(*expr*)

Group Conversion

Description Convert to a **Char** value.

Parameter	Description
expr	Convert a number or string value to an single character value. This is the first character of the string conversion.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print CChar(33) "'!"
End Sub
```

CDate Function

Syntax CDate(*expr*)

Group Conversion

Description Convert to a **Date** value.

Parameter	Description
expr	Convert a number or string value to a date value.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print CDate(2) ' 1/1/00
End Sub
```

CDbl Function

Syntax CDbl(*expr*)

Group Conversion

Description Convert to a **Double** precision real.

Parameter	Description
expr	Convert a number or string value to a double precision real.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print CDbl("1E6") ' 1000000
End Sub
```

CDec Function

Syntax CDec(*expr*)

Group Conversion

Description Convert to a **Decimal** (96 bit scaled real).

Parameter	Description

expr	Convert a number or string value to a 96 bit scaled real.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print CDec("1E16")+0.1 ' 10000000000000000.1
End Sub
```

Char Data Type

Syntax Dim v As Char

Group Data Type

Description A one character value.

ChDir Instruction

Syntax ChDir *Di'*

Group File

Description Change the current directory to *Di'*.

Pocket PC Not supported.

Parameter	Description
Di'	This string value is the path and name of the directory.

See Also **ChDrive**, **CurDi'( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  ChDir "C:\"
  Debug.Print CurDi'() '"C:\"
End Sub
```

ChDrive Instruction

Syntax ChDrive *Driv'*

Group File

Description Change the current drive to *Driv'*.

Pocket PC Not supported.

Parameter	Description
Driv'	This string value is the drive letter.

See Also **ChDir**, **CurDi'( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  ChDrive "B"
  Debug.Print CurDi'() '"B:\"
End Sub
```

CheckBox Dialog Item Definition

Syntax CheckBox *X*, *Y*, *DX*, *DY*, *Titl'*, *.Field*[, *Options*]

Group User Dialog

Description Define a checkbox item.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Field	The value of the check box is accessed via this field. Unchecked is 0, checked is 1 and grayed is 2.
Options	This numeric value controls the type of check box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option	Description
0	Check box is either check or unchecked.
1	Check box is either check, unchecked or grayed, and it switches between checked and unchecked when clicked.
2	Check box is either check, unchecked or grayed, and it cycles through all three states as the button is clicked.

See Also **Begin Dialog**.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    CheckBox 10,25,180,15,"&Check box",.Check
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  dlg.Check = 1
  Dialog dlg ' show dialog (wait for ok)
  Debug.Print dlg.Check
End Sub
```

Choose Function

Syntax Choose(*Index*, *expr*[, ...])

Group Flow Control

Description Return the value of the *expr* indicated by *Index*.

Parameter	Description
Index	The numeric value indicates which *expr* to return. If this value is less than one or greater than the number of *expr*s then System.DBNull.Value is returned.
expr	All expressions are evaluated.

See Also **If**, **Select Case**, **IIf( )**.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print Choose(2,"Hi","there") '"there"
End Sub
```

Ch' Function

Syntax Ch'(*Num*)

Group String

Description Return a one char string for the ASCII value.

Note: A similar function, ChrW, returns a single char Unicode string.

Parameter	Description
Num	Return one char string for this ASCII numeric value.

See Also **Asc( )**.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print Ch'(48) '"0"
End Sub
```

CInt Function

Syntax CInt(*expr*)

Group Conversion

Description Convert to an **Integer**. If *expr* is too big (or too small) to fit then an overflow error occurs.

Parameter	Description
expr	Convert a number or string value to an **Integer**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print CInt(1.6) ' 2
End Sub
```

Class Module

Group Declaration

Description A class *module* implements an object.

Note: The Class statement is not supported. Use a class module instead.

- Has a set of **Public** *procedure*s accessible from other *macros* and *modules*.
- These public symbols are accessed via an object variable.
- Has an optional set of **Event**s that can be raised.
- Public **Const**s, **Structure**s, arrays, fixed length strings are not allowed.
- Has an optional Private Sub **Class_Initialize** which is called when an instance is created.
- Has an optional Private Sub **Class_Terminate** which is called when an instance is destroyed.
- A class module is similar to a **object module** except that no instance is automatically created.
- To create an instance use:

```
Dim Obj As classname
Obj = New classname
```

See Also **Code Module**, **Object Module**, **Uses**, **Class_Initialize**, **Class_Terminate**.

Example

```
'A.WWB
'#Language "WWB.NET"
'#Uses "File.CLS"
Sub Main
  Dim File As New File
  File.Attach "C:\AUTOEXEC.BAT"
  Debug.Print File.ReadLine
End Sub

'File.CLS
'File|New Module|Class Module
'Edit|Properties|Name=File
'#Language "WWB.NET"
Option Explicit
Dim FN As Integer
Public Sub Attach(FileName As String)
  FN = FreeFile
  FileOpen FN, FileName, OpenMode.Input
End Sub
```

```
Public Sub Detach()
  If FN <> 0 Then FileClose'FN
  FN = 0
End Sub
Public Function ReadLine() As String
  ReadLine = LineInput(FN)
End Function

Private Sub Class_Initialize()
  Debug.Print "Class_Initialize"
End Sub

Private Sub Class_Terminate()
  Debug.Print "Class_Terminate"
  Detach
End Sub
```

Class_Initialize Sub

Syntax

```
Private Sub Class_Initialize()
  ...
End Sub
```

Group Declaration

Description **Class module** initialization subroutine. Each time a new instance is created for a class module the Class_Initialize sub is called. If Class_Initialize is not defined then no special initialization occurs.

See Also **Code Module**, **Class_Terminate**.

Class_Terminate Sub

Syntax

```
Private Sub Class_Terminate()
  ...
End Sub
```

Group Declaration

Description **Class module** termination subroutine. Each time an instance is destroyed for a class module the Class_Terminate sub is called. If Class_Terminate is not defined then no special termination occurs.

See Also **Code Module**, **Class_Initialize**.

Clipboard Instruction/Function

Syntax

```
Clipboard Tex'
-or-
Clipboar'[( )]
```

Group Miscellaneous

Description Form 1: Set the clipboard to *Tex'*. This is like the Edit|Copy menu command.

Form 2: Return the text in the clipboard.

Parameter	Description
Tex'	Put this string value into the clipboard.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Clipboar'()
  Clipboard "Hello"
  Debug.Print Clipboar'() "'Hello"
End Sub
```

CLng Function

Syntax CLng(*expr*)

Group Conversion

Description Convert to a **Long**. If *expr* is too big (or too small) to fit then an overflow error occurs.

Parameter	Description
expr	Convert a number or string value to a **Long**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print CLng(1.6) ' 2
End Sub
```

CObj Function

Syntax CObj(*expr*)

Group Conversion

Description Convert to an **Object**. The object contains the value.

Parameter	Description
expr	Convert to an object. If the parameter is already an object, return it.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print CObj(Sqr(2)) ' 1.4142135381699
End Sub
```

Code Module

Group Declaration

Description A Code *module* implements a code library.
Note: The Module statement is not supported. Use a code module instead.

- Has a set of **Public** *procedure*s accessible from other *macros* and *modules*.
- These public symbols are accessed directly.
- May be initialized by a Private Sub **Main**.

See Also **Class Module**, **Object Module**, **Uses**, **Main**.

Example

```
'A.WWB
'#Language "WWB.NET"
'#Uses "Module1.BAS"
Sub Main
   Debug.Print Value "'Hello"
End Sub

'Module1.BAS
'File|New Module|Code Module
'Edit|Properties|Name=Module1
'#Language "WWB.NET"
Option Explicit
Private mValue As String
Property Get Value() As String
   Value = mValue
End Property
'this sub is called when the module is first loaded
Private Sub Main
   mValue = "Hello"
End Sub
```

ComboBox Dialog Item Definition

Syntax ComboBox *X*, *Y*, *DX*, *DY*, *StrArra'( )*, *.Fiel*[, *Options*]

Group User Dialog

Description Define a combobox item. Combo boxes combine the functionality of an edit box and a list box.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.

DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
StrArra'( )	This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.
Fiel'	The value of the combo box is accessed via this field. This is the text in the edit box.
Options	This numeric value controls the type of combo box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option	Description
0	List is not sorted.
2	List is sorted.

See Also **Begin Dialog**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim combo'(3)
  combo'(0) = "Combo 0"
  combo'(1) = "Combo 1"
  combo'(2) = "Combo 2"
  combo'(3) = "Combo 3"
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    ComboBox 10,25,180,60,combo'(),.comb'
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  dlg.comb' = "none"
  Dialog dlg ' show dialog (wait for ok)
  Debug.Print dlg.comb'
End Sub
```

Comman' Function

Syntax Comman'

Group Miscellaneous

Description Contains the value of the **MacroRun** parameters.

See Also **MacroRun**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print "Command line parameter is: """;
  Debug.Print Comman';
  Debug.Print """"
End Sub
```

Const Definition

Syntax [| **Private** | **Public**] _
Const *name*[*type*] [*As Type*] = *expr*[, ...]

Group Declaration

Description Define *name* as the value of *expr*. The *expr* may be refer other constants or built-in functions. If the type of the constants is not specified, the type of *expr* is used. Constants defined outside a **Sub**, **Function** or **Property** block are available in the entire *macro*/*module*.

Private is assumed if neither **Private** or **Public** is specified.

Note: Const statement in a **Sub**, **Function** or **Property** block may not use **Private** or **Public**.

Example

```
'#Language "WWB.NET"
Sub Main
   Const Pi = 4*Atn(1), e = Exp(1)
   Debug.Print Pi ' 3.14159265358979
   Debug.Print e  ' 2.71828182845905
End Sub
```

Cos Function

Syntax Cos(*Num*)

Group Math

Description Return the cosine.

Parameter	Description
Num	Return the cosine of this numeric value. This is the number of radians. There are 2*Pi radians in a full circle.

See Also **Atn**, **Sin**, **Tan**.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print Cos(1) ' 0.54030230586814
End Sub
```

CreateObject Function

Syntax CreateObject(*Clas'*)

Group Object

Description Create a new object of type *Clas'*.

Parameter	Description
Clas'	This string value is the application's registered class name. If this application is not currently active it will be started.

See Also **Objects**.

Example

```
'#Language "WWB.NET"
Sub Main
   Dim App As Object
  'App = CreateObject("WinWrap.CppDemoApplication")
   App.Move 20,30 ' move icon to 20,30
  'App = Nothing
   App.Quit      ' run-time error (no object)
End Sub
```

CSByte Function

Syntax CSByte(*expr*)

Group Conversion

Description Convert to an 8 bit signed integer **SByte** value.

Parameter	Description
expr	Convert a number or string value to a signed byte value.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print CSByte(1.6) ' 2
End Sub
```

CShort Function

Syntax CShort(*expr*)

Group Conversion

Description Convert to a 16 bit signed integer **Short** value. If *expr* is too big (or too small) to fit then an overflow error occurs.

Parameter	Description
expr	Convert a number or string value to a 16 bit signed integer.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print CShort(1.6) ' 2
End Sub
```

CSng Function

Syntax CSng(*expr*)

Group Conversion

Description Convert to a **Single** precision real. If *expr* is too big (or too small) to fit then an overflow error occurs.

Parameter	Description
expr	Convert a number or string value to a single precision real.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print CSng(Sqr(2)) ' 1.4142135381699
End Sub
```

CStr Function

Syntax CStr(*expr*)

Group Conversion

Description Convert to a **String**.

Parameter	Description
expr	Convert a number or string value to a string value using the current locale (**GetLocale**).

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print CStr(Sqr(2)) '"1.4142135623731" - US English locale
End Sub
```

CurDi' Function

Syntax CurDi'([*Driv'*])

Group File

Description Return the current directory for *Driv'*.

Pocket PC Not supported.

Parameter	Description
Driv'	This string value is the drive letter. If this is omitted or null then return the current directory for the current drive.

See Also **ChDir**, **ChDrive**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print CurDi'()
End Sub
```

CType Function

Syntax CType(*expr*, *objtype*)

Group Conversion

Description Convert an *expr* to the specified *objtype* type. Conversion include inheritance, implementation and value conversion. If the *expr* can't be converted, a "type mismatch error" will occur.

Parameter	Description
expr	Convert the value of this expression.
objtype	Convert to this type.

See Also **DirectCast**, **TryCast**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print CType(1.1, Integer) ' 1
  Dim V As Object
  V = Err
  Debug.Print TypeName(CType(V, ErrObject)) ' ErrObject
End Sub
```

CUInt Function

Syntax CUInt(*expr*)

Group Conversion

Description Convert to an **UInteger**. If *expr* is too big (or too small) to fit then an overflow error occurs.

Parameter	Description
expr	Convert a number or string value to an **UInteger**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print CUInt(1.6) ' 2
End Sub
```

CULng Function

Syntax CULng(*expr*)

Group Conversion

Description Convert to a **ULong**. If *expr* is too big (or too small) to fit then an overflow error occurs.

Parameter	Description
expr	Convert a number or string value to a **ULong**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print CULng(1.6) ' 2
End Sub
```

CUShort Function

Syntax CUShort(*expr*)

Group Conversion

Description Convert to a 16 bit unsigned integer **UShort** value. If *expr* is too big (or too small) to fit then an overflow error occurs.

Parameter	Description
expr	Convert a number or string value to a 16 bit unsigned integer.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print CUShort(1.6) ' 2
End Sub
```

Date Data Type

Syntax Dim v As Date

Group Data Type

Description A 64 bit real value. The whole part represents the date, while the fractional part is the time of day. (December 30, 1899 = 0.) Use #date# as a literal date value in an expression.

DateAdd Function

Syntax DateAdd(*interval*, *number*, *dateexpr*)

Group Time/Date

Description Return a **Date** value a number of intervals from another date.

Parameter	Description
interval	This string value indicates which kind of interval to add.

number	Add this many intervals. Use a negative value to get an earlier date.
dateexpr	Calculate the new date relative to this date value. '

Interval	Description
yyyy	Year
q	Quarter
m	Month
y	Day of year
d	Day
w	Weekday
ww	Week
h	Hour
n	Minute
s	Second

See Also **DateDiff**, **DatePart**.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print DateAdd("yyyy",1,#1/1/2000#) '1/1/2001
End Sub
```

DateDiff Function

Syntax DateDiff(*interval*, *dateexpr1*, *dateexpr2*)

Group Time/Date

Description Return the number of intervals between two dates.

Parameter	Description
interval	This string value indicates which kind of interval to subtract.
dateexpr1	Calculate the from this date value to dateexpr2. '
dateexpr2	Calculate the from dateexpr1 to this date value. '

Interval	Description
yyyy	Year
q	Quarter
m	Month
y	Day of year
d	Day
w	Weekday
ww	Week
h	Hour
n	Minute
s	Second

See Also **DateAdd**, **DatePart**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print DateDiff("yyyy",#1/1/1990#,#1/1/2000#) ' 10
End Sub
```

DatePart Function

Syntax DatePart(*interval*, *dateexpr*)

Group Time/Date

Description Return the number from the date corresponding to the interval.

Parameter	Description
interval	This string value indicates which kind of interval to extract.
dateexpr	Get the interval from this date value. '

Interval	Description (return value range)
yyyy	Year (100-9999)
q	Quarter (1-4)
m	Month (1-12)
y	Day of year (1-366)
d	Day (1-31)
w	Weekday (1-7)
ww	Week (1-53)
h	Hour (0-23)
n	Minute (0-59)
s	Second (0-59)

See Also **DateAdd**, **DateDiff**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print DatePart("yyyy",#1/1/2000#) ' 2000
End Sub
```

DateSerial Function

Syntax DateSerial(*Year*, *Month*, *Day*)

Group Time/Date

Description Return a **Date** value.

Parameter	Description
Year	This numeric value is the year (0 to 9999). (0 to 99 are interpreted by the operating system.)
Month	This numeric value is the month (1 to 12).
Day	This numeric value is the day (1 to 31).

See Also **DateValue**, **TimeSerial**, **TimeValue**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print DateSerial(2000,7,4) '7/4/2000
End Sub
```

DateValue Function

Syntax DateValue(*Dat'*)

Group Time/Date

Description Return the day part of the date encoded as a string.

Parameter	Description
Dat'	Convert this string value to the day part of date it represents.

See Also **DateSerial**, **TimeSerial**, **TimeValue**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print DateValue("1/1/2000 12:00:01 AM")
    '1/1/2000
End Sub
```

Day Function

Syntax Day(*dateexpr*)

Group Time/Date

Description Return the day of the month (1 to 31).

Parameter	Description
dateexpr	Return the day of the month for this date value. '

See Also **Date( )**, **Month( )**, **Weekday( )**, **Year( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Day(#1/1/1900#) ' 1
  Debug.Print Day(#1/2/1900#) ' 2
End Sub
```

DDEExecute Instruction

Syntax DDEExecute *ChanNum*, *Comman'*[, *Timeout*]

Group DDE

Description Send the DDE Execute *Comman'* string via DDE *ChanNum*.

Pocket PC Not supported.

Parameter	Description
ChanNum	This is the channel number returned by the **DDEInitiate** function. Up to 10 channels may be used at one time.
Comman'	Send this command value to the server application. The interpretation of this value is defined by the server application.
Timeout	The command will generate an error if the number of seconds specified by the timeout is exceeded before the command has completed. The default is five seconds.

Example

```
'#Language "WWB.NET"
Sub Main
  ChanNum = DDEInitiate("PROGMAN","PROGMAN")
  DDEExecute ChanNum,"[CreateGroup(XXX)]"
  DDETerminate ChanNum
End Sub
```

DDEInitiate Function

Syntax DDEInitiate(*Ap'*, *Topi'*)

Group DDE

Description Initiate a DDE conversation with *Ap'* using *Topi'*. If the conversation is successfully started then the return value is a channel number that can be used with other DDE instructions and functions.

Pocket PC Not supported.

Parameter	Description
Ap'	Locate this server application.
Topi'	This is the server application's topic. The interpretation of this value is defined by the server application.

Example

```
'#Language "WWB.NET"
Sub Main
  ChanNum = DDEInitiate("PROGMAN","PROGMAN")
  DDEExecute ChanNum,"[CreateGroup(XXX)]"
  DDETerminate ChanNum
End Sub
```

DDEPoke Instruction

Syntax DDEPoke *ChanNum*, *Ite'*, *Dat'*[, *Timeout*]

Group DDE

Description Poke *Dat'* to the *Ite'* via DDE *ChanNum*.

Pocket PC Not supported.

Parameter	Description

ChanNum	This is the channel number returned by the **DDEInitiate** function. Up to 10 channels may be used at one time.
Ite'	This is the server application's item. The interpretation of this value is defined by the server application.
Dat'	Send this data value to the server application. The interpretation of this value is defined by the server application.
Timeout	The command will generate an error if the number of seconds specified by the timeout is exceeded before the command has completed. The default is five seconds.

Example

```
'#Language "WWB.NET"
Sub Main
  ChanNum = DDEInitiate("PROGMAN","PROGMAN")
  DDEPoke ChanNum,"Group","XXX"
  DDETerminate ChanNum
End Sub
```

DDEReques' Function

Syntax DDEReques'(*ChanNum*, *Ite*[, *Timeout*])

Group DDE

Description Request information for *Ite'*. If the request is not satisfied then the return value will be a null string.

Pocket PC Not supported.

Parameter	Description
ChanNum	This is the channel number returned by the **DDEInitiate** function. Up to 10 channels may be used at one time.
Ite'	This is the server application's item. The interpretation of this value is defined by the server application.
Timeout	The command will generate an error if the number of seconds specified by the timeout is exceeded before the command has completed. The default is five seconds.

Example

```
'#Language "WWB.NET"
Sub Main
  ChanNum = DDEInitiate("PROGMAN","PROGMAN")
  Debug.Print DDEReques'(ChanNum,"Groups")
  DDETerminate ChanNum
End Sub
```

DDETerminate Instruction

Syntax DDETerminate *ChanNum*

Group DDE

Description Terminate DDE *ChanNum*.

Pocket PC Not supported.

Parameter	Description
ChanNum	This is the channel number returned by the **DDEInitiate** function. Up to 10 channels may be used at one time.

Example

```
'#Language "WWB.NET"
Sub Main
  ChanNum = DDEInitiate("PROGMAN","PROGMAN")
  DDEExecute ChanNum,"[CreateGroup(XXX)]"
  DDETerminate ChanNum
End Sub
```

DDETerminateAll Instruction

Syntax DDETerminateAll

Group DDE

Description Terminate all open DDE channels.

Pocket PC Not supported.

Example

```
'#Language "WWB.NET"
Sub Main
  ChanNum = DDEInitiate("PROGMAN","PROGMAN")
  DDEExecute ChanNum,"[CreateGroup(XXX)]"
  DDETerminateAll
End Sub
```

Debug Object

Syntax

```
Debug.Clear
-or-
Debug.Print [expr[; ...][;]]
```

Group Miscellaneous

Description Form 1: Clear the output window.

Form 2: Print the *expr*(s) to the output window. Use ; to separate expressions. A *num* is it automatically converted to a string before printing (just like **St'( )**). If the instruction does not end with a ; then a newline is printed at the end.

Example

```
'#Language "WWB.NET"
Sub Main
  X = 4
  Debug.Print "X/2="; X/2 ' 2
  Debug.Print "Start..."; ' don't print a newline
  Debug.Print "Finish" ' print a newline
End Sub
```

Decimal Data Type

Syntax **Dim** v As Decimal

Group Data Type

Description A 96 bit scaled real value. A decimal number is of the form: s*m*10^-p where

- s - sign (+1 or -1)
- m - mantissa, unsigned binary value of 96 bits (0 to 79,228,162,514,264,337,593,543,950,335)
- p - scaling power (0 to +28)

Declare Definition

Syntax

```
[ | Private | Public ] _
Declare Sub name Lib "dll name" _
   [Alias "module name"] [([param[, ...]])]
-or-
[ | Private | Public ] _
Declare Function name[type] Lib "dll name" _
   [Alias "module name"] [([param[, ...]])] [As type[()]]
```

Group Declaration

Description Interface to a DLL defined subroutine or function. The values of the calling *arg-list* are assigned to the *param*s.

WARNING! Be very careful when declaring DLL subroutines or functions. If you make a mistake and declare the paramenters or result incorrectly then Windows might halt. Save any open documents before testing new DLL declarations.

Err.LastDLLError returns the error code for that last DLL call.

Access If no access is specified then **Public** is assumed.

Pocket PC Not supported.

Parameter	Description
name	This is the name of the subroutine or function being defined. If Alias "module name" is omitted then this is the module name, too.
"dll name"	This is the DLL file where the module's code is.
"module name"	This is the name of the module in the DLL file. If this is #number then it is the ordinal number of the module. If it is omitted then *name* is the module name. The DLL is searched for the specified module name. If this module exists, it is used. All As String parameters are converted from Unicode to ASCII prior to calling the DLL and from ASCII to Unicode afterwards. (Use "Unicode:module name" to prevent ASCII to Unicode conversion.) If the module does not exist, one or two other module names are tried: 1) For Windows NT only: The module name with a "W" appended is tried. All As

	String parameters are passed as Unicode to calling the DLL. 2) For Windows NT and Windows 95: The module name with an "A" appended is tried. All As String parameters are converted from Unicode to ASCII prior to calling the DLL and from ASCII to Unicode afterwards. If none of these module names is found a run-time error occurs.
params	A list of zero or more *params* that are used by the DLL subroutine or function. (Note: A ByVal string's value may be modified by the DLL.)

See Also **Function**, **Sub**, **Call**.

Example

```
'#Language "WWB.NET"
Declare Function GetActiveWindow& Lib "user32" ()
Declare Function GetWindowTextLengthA& Lib "user32" _
  (ByVal hwnd&)
Declare Sub GetWindowTextA Lib "user32" _
  (ByVal hwnd&, ByVal lps', ByVal cbMax&)

Function ActiveWindowTitl'()
  ActiveWindow = GetActiveWindow()
  TitleLen = GetWindowTextLengthA(ActiveWindow)
  Titl' = Spac'(TitleLen)
  GetWindowTextA ActiveWindow,Titl',TitleLen+1
  ActiveWindowTitl' = Titl'
End Function

Sub Main
  Debug.Print ActiveWindowTitl'()
End Sub
```

Decode64 Function

Syntax Decode6'(*Data*)
-or-
Decode64B(*Data*)

Group Miscellaneous

Description Return a string using the base 64 decoding algorithm.

Decode64B returns a **Byte** array.

Parameter	Description
Data	Return this string's base 64 decoding.

See Also **Encode64**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Decode64("SGVsbG8gV29ybGQhIQ==") '"Hello World!!"
End Sub
```

Decrypt64 Function

Syntax Decrypt6'(*Data* [,*Password*])
-or-
Decrypt64B(*Data* [,*Password*])

Group Miscellaneous

Description Return a string using the RC4 stream decryption algorithm.

Decrypt64B returns a **Byte** array.

Parameter	Description
Data	Return this string's decryption. (The string is first decoded using base 64 decoding.)
Password	Decrypt using this password.

See Also **Decode64**, **Encode64**, **Encrypt64**.

Example
```
'#Language "WWB.NET"
Sub Main
  Debug.Print Decrypt64("Y4GFrF+k1YUHwjEzsg==", "abc") '"Hello World!!"
End Sub
```

Delegate Definition

Syntax
```
[ | Private | Public ] _
Delegate Sub name [([param[, ...]])]
-or-
[ | Private | Public ] _
Delegate Function name[type] _
  [([param[, ...]])] [As type[()]]
```

Group Declaration

Description Define a new *user delegate* (subroutine or function pointer type).

A delegate's Sub or Function is called using the delegate's Invoke method.

Access If no access is specified then **Public** is assumed.

Parameter	Description
name	This is the name of the delegate being defined.
params	A list of zero or more *params* that are used by the delegate's subroutine or function.

See Also **AddressOf**.

Example
```
'#Language "WWB.NET"
Delegate Function OpType(ByVal v1 As Object, ByVal v2 As Object) As Object
```

```
Sub Main
  Debug.Print DoOp(AddressOf Add,1,2) ' 3
  Debug.Print DoOp(AddressOf Subtract,1,2) '-1
End Sub

Function DoOp(ByVal op As OpType, _
        ByVal v1 As Object, ByVal v2 As Object) As Object
  DoOp = op.Invoke(v1,v2)
End Function

Function Add(ByVal v1 As Object, ByVal v2 As Object) As Object
  Add = v1+v2
End Function

Function Subtract(ByVal v1 As Object, ByVal v2 As Object) As Object
  Subtract = v1-v2
End Function
```

DeleteSetting Instruction

Syntax DeleteSetting *AppNam'*, *Sectio'*[, *Ke'*]

Group Settings

Description Delete the settings for *Key* in *Section* in project *AppName*. Win16 and Win32s store settings in a .ini file named *AppName*. Win32/Win64 store settings in the registration database under "HKEY_CURRENT_USER\ Software\ VB and VBA Program Settings\ *AppName*\ *Section*\ *Key*". If *AppName* starts with "..\" then "VB and VBA Program Settings\" is omitted.

Parameter	Description
AppNam'	This string value is the name of the project which has this *Section* and *Key*.
Sectio'	This string value is the name of the section of the project settings.
Ke'	This string value is the name of the key in the section of the project settings. If this is omitted then delete the entire section.

Example

```
'#Language "WWB.NET"
Sub Main
  SaveSetting "MyApp","Font","Size",10
  DeleteSetting "MyApp","Font","Size"
End Sub
```

Dialog Instruction/Function

Syntax Dialog *dialogvar*[, *default*]
-or-
Dialog(*dialogvar*[, *default*])

Group User Input

Description Display the dialog associated with *dialogvar*. The initial values of the dialog fields are provided by *dialogvar*. If the **OK button** or any **push button** is pressed then the fields in dialog are copied to the *dialogvar*. The Dialog( ) function returns a value indicating which button was pressed. (See the result table below.)

Parameter	Description
dlgvar	This variable that holds the values of the fields in a dialog. Use *.field* to access individual fields in a dialog variable.
default	This numeric value indicates which button is the default button. (Pressing the Enter key on a non-button pushes the default button.) Use -2 to indicate that there is no default button. Other possible values are shown in the result table below. If this value is omitted then the first **PushButton**, **OKButton** or **CancelButton** is the default button.

Result	Description
-1	**OK button** was pressed.
0	**Cancel button** was pressed.
>0	Nth **push button** was pressed.

See Also **Begin Dialog**.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  Dialog dlg ' show dialog (wait for ok)
End Sub
```

DialogFunc Prototype

Syntax

```
Function dialogfunc(DlgItem$, Action%, SuppValue&) _
  As Boolean
  Select Case Actio'
  Case 1 ' Dialog box initialization
    ...
  Case 2 ' Value changing or button pressed
    ...
  Case 3 ' TextBox or ComboBox text changed
    ...
  Case 4 ' Focus changed
    ...
  Case 5 ' Idle
    ...
  Case 6 ' Function key
    ...
  End Select
End Function
```

Group Dialog Function

Description A *dialogfunc* implements the dynamic dialog capabilities.

Parameter	Description
DlgItem	This string value is the name of the user dialog item's *field*.
Action	This numeric value indicates what action the dialog function is being asked to do.
SuppValue	This numeric value provides additional information for some actions.

Action	Description
1	Dialog box initialization. *DlgItem* is a null string. *SuppValue* is the dialog's window handle. Set *dialogfunc* = **True** to terminate the dialog.
2	**CheckBox**, **DropListBox**, **ListBox**, **MultiListBox** or **OptionGroup**: *DlgItem*'s value has changed. *SuppValue* is the new value. **CancelButton**, **OKButton** or **PushButton**: *DlgItem*'s button was pushed. *SuppValue* is meaningless. Set *dialogfunc* = **True** to prevent the dialog from closing.
3	**ComboBox** or **TextBox**: *DlgItem*'s text changed and losing focus. *SuppValue* is the number of characters.
4	Item *DlgItem* is gaining focus. *SuppValue* is the item that is losing focus. (The first item is 0, second is 1, etc.)
5	Idle processing. *DlgItem* is a null string. *SuppValue* is zero. Set *dialogfunc* = **True** to continue receiving idle actions. The idle action is called as often as possible. Use Wait .1 to reduce the number of idle calls to 10 per second.
6	Function key (F1-F24) was pressed. *DlgItem* has the focus. *SuppValue* is the function key number and the shift/control/alt key state. Regular function keys range from 1 to 24. Shift function keys have &H100 added. Control function keys have &H200 added. Alt function keys have &H400 added. (Alt-F4 closes the dialog and is never passed to the Dialog Function.)

See Also **Begin Dialog**.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
    PushButton 110,90,60,20,"&Hello"
  End Dialog
  Dim dlg As UserDialog
  Debug.Print Dialog(dlg)
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue&) As Boolean
  Debug.Print "Action="; Actio'
  If Actio' <> 1 And Actio' <> 5 Then
    Debug.Print DlgIte"; "="""; DlgText(DlgIte'); """"
  End If
  Debug.Print "SuppValue="; SuppValue&
  Select Case Actio'
  Case 1 ' Dialog box initialization
```

```
    Beep
  Case 2 ' Value changing or button pressed
    If DlgIte' = "Hello" Then
      MsgBox "Hello"
      DialogFunc = True 'do not exit the dialog
    End If
  Case 4 ' Focus changed
    Debug.Print "DlgFocus="""; DlgFocus(); """"
  Case 6 ' Function key
    If SuppValue& And &H100 Then Debug.Print "Shift-";
    If SuppValue& And &H200 Then Debug.Print "Ctrl-";
    If SuppValue& And &H400 Then Debug.Print "Alt-";
    Debug.Print "F" & (SuppValue And &HFF)
  End Select
End Function
```

Dim Definition

Syntax Dim [**WithEvents**] *vardeclaration* [= *initialvalue*][, ...]

Group Declaration

Description Dimension var array(s) using the *dimension*s to establish the minimum and maximum index value for each dimension. If the *dimension*s are omitted then a scalar (single value) variable is defined. A dynamic array is declared using ( ) without any *dimension*s. It must be **ReDim**ensioned before it can be used.

See Also **Begin Dialog**, **Dialog**, **Option** Base, **Private**, **Public**, **ReDim**, **Static**, **WithEvents**.

Example

```
'#Language "WWB.NET"
Sub DoIt(Size)
  Dim C0, C1(), C2(2,3)
  ReDim C1(Size+1) ' dynamic array
  C0 = 1
  C1(0) = 2
  C2(0,0) = 3
  Debug.Print C0; C1(0); C2(0,0) ' 1 2 3
End Sub

Sub Main
  DoIt 1
End Sub
```

Di' Function

Syntax Di'([*Patter*'][, *AttribMask*])

Group File

Description Scan a directory for the first file matching *Patter'*.

Parameter	Description
Patter'	This string value is the path and name of the file search pattern. If this is omitted then continue scanning with the previous pattern. Each *macro* has its own independent search. A path relative to the current directory can be used.
AttribMask	This numeric value controls which files are found. A file with an *attribute* that matches will be found.

See Also **GetAttr( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  ' = Di'("*.*")
  While ' <> ""
    Debug.Print '
    ' = Di'()
  End While
End Sub
```

DirectCast Function

Syntax DirectCast(*expr*, *objtype*)

Group Conversion

Description Return *expr*'s type is related to *objtype* type. If it is not, a "type mismatch error" will occur.

Parameter	Description
expr	Cast the value of this expression.
objtype	Cast to this type.

See Also **CType**, **TryCast**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim V As Object
  V = Err
  Debug.Print TypeName(DirectCast(V, ErrObject)) ' ErrObject
End Sub
```

DlgControlId Function

Syntax DlgControlId(*DlgItem*)

Group Dialog Function

Description Return the *field*'s window id.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter	Description

DlgItem	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
    PushButton 110,90,60,20,"&Hello"
  End Dialog
  Dim dlg As UserDialog
  Debug.Print Dialog(dlg)
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue&) As Boolean
  Debug.Print "Action="; Actio'
  Select Case Actio'
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
    If DlgIte' = "Hello" Then
      DialogFunc = True 'do not exit the dialog
    End If
  Case 4 ' Focus changed
    Debug.Print "DlgFocus="""; DlgFocus(); """"
    Debug.Print "DlgControlId("; DlgIte'; ")=";
    Debug.Print DlgControlId(DlgIte')
  End Select
End Function
```

DlgCount Function

Syntax DlgCount()

Group Dialog Function

Description Return the number of dialog items in the dialog.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
  End Dialog
  Dim dlg As UserDialog
  Dialog dlg
End Sub
```

```
Function DialogFunc(DlgItem$, Action%, SuppValue&) As Boolean
  Debug.Print "Action="; Actio'
  Select Case Actio'
  Case 1 ' Dialog box initialization
    Beep
    Debug.Print "DlgCount="; DlgCount() ' 3
  End Select
End Function
```

DlgEnable Instruction/Function

Syntax

DlgEnable *DlgItem*[, *Enable*]
-or-
DlgEnable(*DlgItem*)

Group

Dialog Function

Description

Instruction: Enable or disable *DlgItem*.

Function: Return **True** if *DlgItem* is enabled.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter	Description
DlgItem	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name. **Note:** Use -1 to enable or disable all the dialog items at once.
Enable	It this numeric value is **True** then enable *DlgItem*. Otherwise, disable it. If this omitted then toggle it.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
    PushButton 110,90,60,20,"&Disable"
  End Dialog
  Dim dlg As UserDialog
  Debug.Print Dialog(dlg)
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue&) As Boolean
  Debug.Print "Action="; Actio'
  Select Case Actio'
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
    Select Case DlgIte'
    Case "Disable"
      DlgText DlgIte',"&Enable"
      DlgEnable "Text",False
```

```
      DialogFunc = True 'do not exit the dialog
    Case "Enable"
      DlgText DlgIte','&Disable"
      DlgEnable "Text",True
      DialogFunc = True 'do not exit the dialog
    End Select
  End Select
End Function
```

DlgEnd Instruction

Syntax DlgEnd *ReturnCode*

Group Dialog Function

Description Set the return code for the **Dialog** Function and close the user dialog.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter	Description
ReturnCode	Return this numeric value.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 210,120,.DialogFunc
    Text 10,10,190,15,"Please push the Close button"
    OKButton 30,90,60,20
    CheckBox 120,90,60,20,"&Close",.CheckBox1
  End Dialog
  Dim dlg As UserDialog
  Debug.Print Dialog(dlg)
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue&) As Boolean
  Debug.Print "Action="; Actio'
  Select Case Actio'
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
    Select Case DlgIte'
    Case "CheckBox1"
      DlgEnd 1000
    End Select
  End Select
End Function
```

DlgFocus Instruction/Function

Syntax DlgFocus *DlgItem*
-or-
DlgFocu'()

Group Dialog Function

Description Instruction: Move the focus to this *DlgItem*.

Function: Return the *field* name which has the focus as a string.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter	Description
DlgItem	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
    PushButton 110,90,60,20,"&Hello"
  End Dialog
  Dim dlg As UserDialog
  Debug.Print Dialog(dlg)
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue&) As Boolean
  Debug.Print "Action="; Actio'
  Select Case Actio'
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
    If DlgIte' = "Hello" Then
      MsgBox "Hello"
      DialogFunc = True 'do not exit the dialog
    End If
  Case 4 ' Focus changed
    Debug.Print "DlgFocus="""; DlgFocus(); """"
  End Select
End Function
```

DlgListBoxArray Instruction/Function

Syntax DlgListBoxArray *DlgItem*, *StrArra'( )*
-or-
DlgListBoxArray(*DlgItem*[, *StrArra'( )*])

Group Dialog Function

Description Instruction: Set the list entries for *DlgItem*.

Function: Return the number entries in *DlgItem*'s list.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

The *DlgItem* should refer to a **ComboBox**, **DropListBox**, **ListBox** or **MultiListBox**.

Parameter	Description
DlgItem	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name.
StrArra'( )	Set the list entries of *DlgItem*. This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.

Example

```
'#Language "WWB.NET"
Dim lists$()

Sub Main
  ReDim list'(0)
  list'(0) = "List 0"
  Begin Dialog UserDialog 200,119,.DialogFunc
    Text 10,7,180,14,"Please push the OK button"
    ListBox 10,21,180,63,lists(),.list
    OKButton 30,91,40,21
    PushButton 110,91,60,21,"&Change"
  End Dialog
  Dim dlg As UserDialog
  dlg.list = 2
  Dialog dlg ' show dialog (wait for ok)
  Debug.Print dlg.list
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue&) As Boolean
  Select Case Actio'
  Case 2 ' Value changing or button pressed
    If DlgIte' = "Change" Then
      Dim N As Integer
      N = UBound(list')+1
      ReDim Preserve list'(N)
      list'(N) = "List " & N
      DlgListBoxArray "list",list'()
      DialogFunc = True 'do not exit the dialog
    End If
  End Select
End Function
```

DlgName Function

Syntax DlgNam'(*DlgItem*)

Group Dialog Function

Description Return the *field* name of the *DlgItem* number.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter	Description
DlgItem	This numeric value is the dialog item number. The first item is 0, second is 1, etc.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
  End Dialog
  Dim dlg As UserDialog
  Dialog dlg
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue&) As Boolean
  Debug.Print "Action="; Actio'
  Select Case Actio'
  Case 1 ' Dialog box initialization
    Beep
    For I = 0 To DlgCount()-1
      Debug.Print I; DlgName(I)
    Next I
  End Select
End Function
```

DlgNumber Function

Syntax DlgNumber(*DlgIte'*)

Group Dialog Function

Description Return the number of the *DlgIte'*. The first item is 0, second is 1, etc.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter	Description
DlgIte'	This string value is the dialog item's *field* name.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
  End Dialog
  Dim dlg As UserDialog
  Dialog dlg
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue&) As Boolean
  Debug.Print "Action="; Actio'
  Select Case Actio'
  Case 1 ' Dialog box initialization
    Beep
  Case 4 ' Focus changed
```

```
    Debug.Print DlgIte'; "="; DlgNumber(DlgIte')
  End Select
End Function
```

DlgSetPicture Instruction

Syntax DlgSetPicture *DlgItem*, *FileName*, *Type*

Group Dialog Function

Description Instruction: Set the file name for *DlgItem*.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter	Description
DlgItem	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name.
FileName	Set the file name of *DlgItem* to this string value.
Type	This numeric value indicates the type of bitmap used. See below.

Type	Effect
0	*FileName* is the name of the bitmap file. If the file does not exist then "(missing picture)" is displayed.
3	The clipboard's bitmap is displayed. If the clipboard does not contain a bitmap then "(missing picture)" is displayed.
16	Same a 0, but instead of displaying "(missing picture)" a run-time error occurs.
19	Same a 3, but instead of displaying "(missing picture)" a run-time error occurs.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Picture 10,10,180,75,"",0,.Picture
    OKButton 30,90,60,20
    PushButton 110,90,60,20,"&View"
  End Dialog
  Dim dlg As UserDialog
  Debug.Print Dialog(dlg)
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue&) As Boolean
  Debug.Print "Action="; Actio'
  Select Case Actio'
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
    Select Case DlgIte'
    Case "View"
      FileName = GetFilePath("Bitmap","BMP")
      DlgSetPicture "Picture",FileName,0
      DialogFunc = True 'do not exit the dialog
    End Select
```

```
  End Select
End Function
```

DlgText Instruction/Function

Syntax DlgText *DlgItem*, *Text*
-or-
DlgTex'(*DlgItem*)

Group Dialog Function

Description Instruction: Set the text for *DlgItem*.

Function: Return the text from *DlgItem*.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter	Description
DlgItem	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name. **Note:** Use -1 to access the dialog's title.
Text	Set the text of *DlgItem* to this string value.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
    PushButton 110,90,60,20,"&Now"
  End Dialog
  Dim dlg As UserDialog
  Debug.Print Dialog(dlg)
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue&) As Boolean
  Debug.Print "Action="; Actio'
  Select Case Actio'
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
    Select Case DlgIte'
    Case "Now"
      DlgText "Text",CStr(Now)
      DialogFunc = True 'do not exit the dialog
    End Select
  End Select
End Function
```

DlgType Function

Syntax DlgTyp'(*DlgItem*)

Group Dialog Function

Description Return a string value indicating the type of the *DlgItem*. One of: "**CancelButton**", "**CheckBox**", "**ComboBox**", "**DropListBox**", "**GroupBox**", "**ListBox**", "**MultiListBox**", "**OKButton**", "**OptionButton**", "**OptionGroup**", "**PushButton**", "**Text**", "**TextBox**".

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter	Description
DlgItem	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
  End Dialog
  Dim dlg As UserDialog
  Dialog dlg
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue&) As Boolean
  Debug.Print "Action="; Actio'
  Select Case Actio'
  Case 1 ' Dialog box initialization
    Beep
    For I = 0 To DlgCount()-1
      Debug.Print I; DlgType(I)
    Next I
  End Select
End Function
```

DlgValue Instruction/Function

Syntax DlgValue *DlgItem*, *Value*
-or-
DlgValue(*DlgItem*)

Group Dialog Function

Description Instruction: Set the numeric value(s) *DlgItem*.

Function: Return the numeric value(s) for *DlgItem*. (A MultiListBox user dialog

item returns an array.)

This instruction/function must be called directly or indirectly from a *dialogfunc*. The *DlgItem* should refer to a **CheckBox**, **ComboBox**, **DropListBox**, **ListBox**, **MultiListBox** or **OptionGroup**.

Parameter	Description
DlgItem	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name.
Value	Set the text of *DlgItem* to this numeric value. (A MultiListBox user dialog item uses an array.)

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 150,147,.DialogFunc
    GroupBox 10,7,130,77,"Direction",.Field1
    PushButton 100,28,30,21,"&Up"
    PushButton 100,56,30,21,"&Dn"
    OptionGroup .Direction
      OptionButton 20,21,80,14,"&North",.North
      OptionButton 20,35,80,14,"&South",.South
      OptionButton 20,49,80,14,"&East",.East
      OptionButton 20,63,80,14,"&West",.West
    OKButton 10,91,130,21
    CancelButton 10,119,130,21
  End Dialog
  Dim dlg As UserDialog
  Dialog dlg
  MsgBox "Direction=" & dlg.Direction
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue&) As Boolean
  Select Case Actio'
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
    Select Case DlgIte'
    Case "Up"
      DlgValue "Direction",0
      DialogFunc = True 'do not exit the dialog
    Case "Dn"
      DlgValue "Direction",1
      DialogFunc = True 'do not exit the dialog
    End Select
  End Select
End Function
```

DlgVisible Instruction/Function

Syntax

DlgVisible *DlgItem*[, *Visible*]
-or-
DlgVisible(*DlgItem*)

Group Dialog Function

Description Instruction: Show or hide *DlgItem*.

Function: Return **True** if *DlgItem* is visible.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter	Description
DlgItem	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name.
Enable	It this numeric value is **True** then show *DlgItem*. Otherwise, hide it. If this omitted then toggle it.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
    PushButton 110,90,60,20,"&Hide"
  End Dialog
  Dim dlg As UserDialog
  Debug.Print Dialog(dlg)
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue&) As Boolean
  Debug.Print "Action="; Actio'
  Select Case Actio'
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
    Select Case DlgIte'
    Case "Hide"
      DlgText DlgIte','&Show"
      DlgVisible "Text",False
      DialogFunc = True 'do not exit the dialog
    Case "Show"
      DlgText DlgIte','&Hide"
      DlgVisible "Text",True
      DialogFunc = True 'do not exit the dialog
    End Select
  End Select
End Function
```

Do Statement

Syntax

```
Do
  statements
Loop
-or-
Do {Until|While} condexpr
  statements
Loop
-or-
Do
  statements
Loop {Until|While} condexpr
```

Group Flow Control

Description Form 1: Do *statements* forever. The loop can be exited by using **Exit** or **Goto**.

Form 2: Check for loop termination before executing the loop the first time.

Form 3: Execute the loop once and then check for loop termination.

Loop Termination:

- Until *condexpr*: Do *statements* until *condexpr* is **True**.
- While *condexpr*: Do *statements* while *condexpr* is **True**.

See Also **For**, **For Each**, **Exit** Do, **While**.

Example

```
'#Language "WWB.NET"
Sub Main
  I = 2
  Do
    I = I*2
  Loop Until I > 10
  Debug.Print I ' 16
End Sub
```

DoEvents Instruction

Syntax

```
DoEvents
```

Group Miscellaneous

Description This instruction allows other applications to process events.

Example

```
'#Language "WWB.NET"
Sub Main
  DoEvents ' let other apps work
End Sub
```

Double Data Type

Syntax **Dim** v As Double

Group Data Type

Description A 64 bit real value.

DropListBox Dialog Item Definition

Syntax DropListBox *X*, *Y*, *DX*, *DY*, *StrArra'( )*, *.Field*[, *Options*]

Group User Dialog

Description Define a drop-down listbox item.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
StrArra'( )	This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.
Field	The value of the drop-down list box is accessed via this field. It is the index of the *StrArra'( )* var.
Options	This numeric value controls the type of drop-down list box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option	Description
0	Text box is not editable and list is not sorted.
1	Text box is editable and list is not sorted.
2	Text box is not editable and list is sorted.
3	Text box is editable and list is sorted.

See Also **Begin Dialog**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim list'(3)
  list'(0) = "List 0"
  list'(1) = "List 1"
  list'(2) = "List 2"
  list'(3) = "List 3"
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    DropListBox 10,25,180,60,list'(),.list1
```

```
    DropListBox 10,50,180,60,list'(),.list2,1
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  dlg.list1 = 2      ' list1 is a numeric field
  dlg.list2 = "xxx"   ' list2 is a string field
  Dialog dlg ' show dialog (wait for ok)
  Debug.Print list'(dlg.list1)
  Debug.Print dlg.list2
End Sub
```

Encode64 Function

Syntax Encode6'(*Data*)
-or-
Encode64'(*Data*)

Group Miscellaneous

Description Return a string using the base 64 encoding algorithm.

Parameter	Description
Data	Return this string's base 64 encoding.

See Also **Decode64**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Encode64("Hello World!!") '"SGVsbG8gV29ybGQhIQ=="
End Sub
```

Encrypt64 Function

Syntax Encrypt6'(*Data* [,*Password*])

Group Miscellaneous

Description Return a string using the RC4 stream encryption algorithm. (The string is also encoded using base 64 encoding.)

Parameter	Description
Data	Return this string's encryption.
Password	Encrypt using this password.

See Also **Decode64**, **Decrypt64**, **Encode64**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Encrypt64("Hello World!!", "abc") '"Y4GFrF+k1YUHwjEzsg=="
End Sub
```

End Instruction

Syntax End

Group Flow Control

Description The end instruction causes the *macro* to terminate immediately. If the macro was run by another macro using the **MacroRun** instruction then that macro continues on the instruction following the **MacroRun**.

Example

```
'#Language "WWB.NET"
Sub DoSub
   ' = UCas'(InputBo'("Enter End:"))
   If ' = "END" Then End
   Debug.Print "End was not entered."
End Sub

Sub Main
   Debug.Print "Before DoSub"
   DoSub
   Debug.Print "After DoSub"
End Sub
```

Enum Definition

Syntax

```
[ | Private | Public ] _
Enum name
   elem [ = value]
  [...]
End Enum
```

Group Declaration

Description Define a new *user enum*. Each *elem* defines an element of the enum. If *value* is given then that is the element's value. The value can be any constant integer expression. If *value* is omitted then the element's value is one more than the previous element's value. If there is no previous element then zero is used.

Access If no access is specified then **Public** is assumed.

Example

```
'#Language "WWB.NET"
Enum Days
   Monday
   Tuesday
   Wednesday
   Thursday
   Friday
   Saturday
   Sunday
End Enum
```

```
Sub Main
  Dim D As Days
  For D = Days.Monday To Days.Friday
    Debug.Print D ' 0 through 4
  Next D
End Sub
```

Environ Function

Syntax Enviro'(*Index*)
-or-
Enviro'(*Name*)

Group Miscellaneous

Description Return an environment string.

Pocket PC Not supported.

Parameter	Description
Index	Return this environment string's value. If there is no environment string at this index a null string is returned. Indexes start at one.
Name	Return this environment string's value. If the environment string can't be found a null string is returned.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Environ("Path")
End Sub
```

EOF Function

Syntax EOF(*StreamNum*)

Group File

Description Return **True** if *StreamNum* is at the end of the file.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

Example

```
'#Language "WWB.NET"
Sub Main
  FileOpen 1, "XXX", OpenMode.Input
   While Not EOF(1)
    Dim L As String
    L = LineInput(1)
    Debug.Print L
  End While
```

```
  FileClose'1
End Sub
```

Erase Instruction

Syntax Erase *arrayvar*[, ...]
-or-
Erase *userstructurevar.elem*[, ...]

Group Assignment

Description Reset *arrayvar* or *user defined structure* array element to zero. (Dynamic arrays are reset to undimensioned arrays.) String arrays values are set to a null string. *arrayvar* must be declared as an array.

- Declare with **Dim**, **Private**, **Public** or **Static**.
- Declare as a parameter of **Sub**, **Function** or **Property** definition.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim '(2)
  '(1) = 1
  Erase '
  Debug.Print '(1) ' 0
End Sub
```

Err Object

Syntax Err

Group Error Handling

Description Set Err to zero to clear the last error event. Err in an expression returns the last error code. Add vbObjectError to your error number in ActiveX Automation objects. Use Err.Raise or **Error** to trigger an error event.

Err[.Number]

This is the error code for the last error event. Set it to zero (or use Err.Clear) to clear the last error condition. Use **Error** or Err.Raise to trigger an error event. This is the default property.

Err.Description

This string is the description of the last error event.

Err.Source

This string is the error source file name of the last error event.

Err.HelpFile

This string is the help file name of the last error event.

Err.HelpContext

This number is the help context id of the last error event.

Err.Clear

Clear the last error event.

Err.Raise [Number:=]*errorcode* _
[, [Source:=]*source*] _
[, [Description:=]*errordesc*] _
[, [HelpFile:=]*helpfile*] _
[, [HelpContext:=]*context*]

Raise an error event.

Err.LastDLLError

Returns the error code for the last DLL call (see **Declare**).

Example

```
'#Language "WWB.NET"
Sub Main
  On Error GoTo Problem
  Err = 1 ' set to error #1 (handler not triggered)
  Exit Sub

  Problem: ' error handler
  Error Err ' halt macro with message
End Sub
```

Error Instruction

Syntax Error *ErrorCode*

Group Error Handling

Description Signal error *ErrorCode*. This triggers error handling just like a real error. The current *procedure*'s error handler is activated, unless it is already active or there isn't one. In that case the calling *procedure*'s error handler is tried. (Use **Err**.Raise to provide complete error information.)

Parameter	Description
ErrorCode	This is the error number.

Example

```
'#Language "WWB.NET"
Sub Main
  On Error GoTo Problem
  Error 1 ' simulate error #1
  Exit Sub

Problem: ' error handler
```

```
  Debug.Print "Err.Description="; Err.Description
  Resume Next
End Sub
```

ErrorToString Function

Syntax ErrorToString(*ErrorCode*)

Group Error Handling

Description The ErrorToString function returns the error text string.

Parameter	Description
ErrorCode	This is the error number.

Example

```
'#Language "WWB.NET"
Sub Main
  On Error GoTo Problem
  Error 1 ' simulate error #1
  Exit Sub

Problem: ' error handler
  Debug.Print "Description="; ErrorToString(Err.Num)
  Resume Next
End Sub
```

Eval Function

Syntax Eval(*Expr*[, *Depth*])

Group Miscellaneous

Description Return the value of the string expression as evaluated.

Parameter	Description
Expr	Evaluate this string value.
Depth	This integer value indicates how deep into the stack to locate the local variables. If Depth = 0 then use the current *procedure*. If this value is omitted then the depth is 0.

See Also **Assign**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim X As String
  X = "Hello"
  Debug.Print Eval("X") 'Hello
  A
End Sub
Sub A
  Dim X As String
```

```
  X = "Bye"
  Debug.Print Eval("X") 'Bye
  Debug.Print Eval("X",1) 'Hello
End Sub
```

Event Definition

Syntax [| **Public**] _
Event *name*[([*param*[, ...]])]

Group Declaration

Description User defined event. The event defines a sub that can be defined using **WithEvents**.. The values of the calling *arglist* are assigned to the *param*s.

Access If no access is specified then **Public** is assumed.

See Also **RaiseEvent**.

Example

```
' Class1
'#Language "WWB.NET"
Event Changing(ByVal OldValue As String, ByVal NewValue As String)

Private Value_ As String

Property Get Value As String
  Value = Value_
End Property

Property Let Value(ByVal NewValue As String)
  RaiseEvent Changing(Value_, NewValue)
  Value_ = NewValue
End Property

'#Uses "Class1.cls"

Dim WithEvents c1 As Class1

Sub Main
  'c1 = New Class1
  c1.Value = "Hello"
  c1.Value = "Goodbye"
End Sub

Sub c1_Changing(ByVal OldValue As String, ByVal NewValue As String) Handles
c1.Changing
  Debug.Print "OldValue=""" & OldValue & """, NewValue=""" & NewValue & """"
End Sub
```

Exit Instruction

Syntax Exit {All|Do|For|Function|Property|Sub|Try|While}

Group Flow Control

Description The exit instruction causes the *macro* to continue with out doing some or all of the remaining instructions.

Exit	Description
All	Exit all *macros*.
Do	Exit the **Do** loop.
For	Exit the **For** of **For Each** loop.
Function	Exit the **Function** block. Note: This instruction clears the **Err** and sets **Error`$'** to null.
Property	Exit the **Property** block. Note: This instruction clears the **Err** and sets **Error`$'** to null.
Sub	Exit the **Sub** block. Note: This instruction clears the **Err** and sets **Error`$'** to null.
Try	Exit the **Try** block.
While	Exit the **While** loop.

Example

```
'#Language "WWB.NET"
Sub Main
  ' = InputBo'("Enter Do, For, While, Sub or All:")
  Debug.Print "Before DoSub"
  DoSub UCas'(')
  Debug.Print "After DoSub"
End Sub

Sub DoSub(')
  Do
    If ' = "DO" Then Exit Do
    I = I+1
  Loop While I < 10
  If I = 0 Then Debug.Print "Do was entered"

  For I = 1 To 10
    If ' = "FOR" Then Exit For
  Next I
  If I = 1 Then Debug.Print "For was entered"

  I = 10
  While I > 0
    If ' = "WHILE" Then Exit While
    I = I-1
  End While
  If I = 10 Then Debug.Print "While was entered"

  If ' = "SUB" Then Exit Sub
  Debug.Print "Sub was not entered."
  If ' = "ALL" Then Exit All
  Debug.Print "All was not entered."
End Sub
```

Exp Function

Syntax Exp(*Num*)

Group Math

Description Return the exponential.

Parameter	Description
Num	Return e raised to the power of this numeric value. The value e is approximately 2.718282.

See Also **Log**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Exp(1) ' 2.718281828459
End Sub
```

False Keyword

Group Constant

Description A *condexpr* is false when its value is zero. A function that returns False returns the value 0.

FileAttr Function

Syntax FileAttr(*StreamNum*, *ReturnValue*)

Group File

Description Return *StreamNum*'s open mode or file handle.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
ReturnValue	1 - return the mode used to open the file: 1=Input, 2=Output, 4=Random, 8=A-ppend, 32=Binary 2 - return the file handle

See Also **Open**.

Example

```
'#Language "WWB.NET"
Sub Main
  FileOpen 1, "XXX", OpenMode.Output
  Debug.Print FileAttr(1,1) ' 2
  FileClose'1
End Sub
```

FileClose Instruction

Syntax FileClose [*StreamNum*][, ...]

Group File

Description Close *StreamNum*s.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros. If this is omitted then all open streams for the current *macro*/*module* are closed.

See Also **FileOpen**, **Reset**.

Example

```
'#Language "WWB.NET"
Sub Main
  ' read the first line of XXX and print it
  FileOpen 1, "XXX", OpenMode.Input
  Dim L As String
  LineInput 1, L
  Debug.Print L
  FileClose 1
End Sub
```

FileCopy Instruction

Syntax FileCopy *FromNam'*, *ToNam'*

Group File

Description Copy a file.

Parameter	Description
FromNam'	This string value is the path and name of the source file. A path relative to the current directory can be used.
ToNam'	This string value is the path and name of the destination file. A path relative to the current directory can be used.

Example

```
'#Language "WWB.NET"
Sub Main
  FileCopy "C:\AUTOEXEC.BAT","C:\AUTOEXEC.BAK"
End Sub
```

FileDateTime Function

Syntax FileDateTime(*Nam'*)

Group File

Description Return the date and time file *Nam'* was last changed as a **Date** value. If the file does not exist then a run-time error occurs.

Parameter	Description
Nam'	This string value is the path and name of the file. A path relative to the current directory can be used.

Example

```
'#Language "WWB.NET"
Sub Main
  ' = Di'("*.*")
  While ' <> ""
    Debug.Print '; " "; FileDateTime(')
    ' = Di'()
  End While
End Sub
```

FileLen Function

Syntax FileLen(*Nam'*)

Group File

Description Return the length of file *Nam'*. If the file does not exist then a run-time error occurs.

Parameter	Description
Nam'	This string value is the path and name of the file. A path relative to the current directory can be used.

Example

```
'#Language "WWB.NET"
Sub Main
  ' = Di'("*.*")
  While ' <> ""
    Debug.Print '; " "; FileLen(')
    ' = Di'()
  End While
End Sub
```

FileOpen Instruction

Syntax FileOpen *StreamNum*, *Nam'*, *mode*, [*access*], [*lock*], [*RecordLen*]

Group File

Description Open file *Nam'* for *mode* as *StreamNum*.

Reading Unicode Files To read a Unicode (UTF-16 or UTF-8) file, just open using Input mode. All the input is converted automatically.

Writing Unicode Files To write a Unicode (UTF-16 or UTF-8) file, open using Output or Append mode and write the appropriate Byte Order Mark (BOM). All the printed output is converted automatically.

To create a Unicode (UTF-16) text file use:

```
FileOpen fn, FileName, OpenMode.Output
Print fn, vbUTF16BOM ' first char is the UTF-16 Byte Order Mark
... ' everything automatically converted to UTF-16
FileClose fn
```

To create a Unicode (UTF-8) text file use:

```
FileOpen fn, FileName, OpenMode.Output
Print fn, vbUTF8BOM ' first three chars are the UTF-8 Byte Order Mark
... ' everything automatically converted to UTF-8
FileClose fn
```

Parameter	Description
Nam'	This string value is the path and name of the file. A path relative to the current directory can be used.
mode	May be Input (1), Output (2), Append (8), Binary (4) or Random (32).
access	May be Read (1), Write (2) or Read Write (3). If omitted then Read Write is used.
lock	May be Shared (0), Lock Read (2), Lock Write (1) or Lock Read Write (3). If omitted then Shared is used.
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
RecordLen	This numeric value is the record length for Random mode files. Other file modes ignore this value.

See Also **FileClose**, **FileAttr**, **FreeFile**, **Reset**.

Example

```
'#Language "WWB.NET"
Sub Main
   FileOpen 1, "XXX", OpenMode.Output
   PrintLine 1, "1,2,""Hello"""
   FileClose 1
End Sub
```

Fix Function

Syntax Fix(*Num*)

Group Math

Description Return the integer value.

Parameter	Description
Num	Return the integer portion of this numeric value. The number is truncated. Positive numbers return the next lower integer. Negative numbers return the next higher integer. '

See Also **Int**.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print Fix(9.9)  ' 9
   Debug.Print Fix(0)    ' 0
```

```
  Debug.Print Fix(-9.9) '-9
End Sub
```

For Statement

Syntax

```
For Num [As type] = First To Last [Step Inc]
  statements
Next [Num]
```

Group Flow Control

Description Execute *statements* while *Num* is in the range *First* to *Last*.

Parameter	Description
Num	This is the iteration variable.
As type	Optional. The interation variable can be declared here which eliminates the need for a separate **Dim** statement.
First	Set *Num* to this value initially.
Last	Continue looping while *Num* is in the range. See *Step* below.
Step	If this numeric value is greater than zero then the for loop continues as long as *Num* is less than or equal to *Last*. If this numeric value is less than zero then the for loop continues as long as *Num* is greater than or equal to *Last*. If this is omitted then one is used.

See Also **Do**, **For Each**, **Exit** For, **While**.

Example

```
'#Language "WWB.NET"
Sub Main
  For I = 1 To 2000 Step 100
    Debug.Print I; I+I; I*I
  Next I
End Sub
```

For Each Statement

Syntax

```
For Each var [As type] In items
  statements
Next [var]
```

Group Flow Control

Description Execute *statements* for each item in *items*.

Parameter	Description
var	This is the iteration variable.
As type	Optional. The interation variable can be declared here which eliminates the need for a separate **Dim** statement.
items	This is the collection of items to be done.

See Also **Do**, **For**, **Exit** For, **While**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim Document As Object
  For Each Document In App.Documents
    Debug.Print Document.Title
  Next Document
End Sub
```

Forma' Function

Syntax

Forma'(*expr*[, *for*'], [*firstday*], _
[*firstweek*])

Group

String

Description

Return the formatted string representation of *expr*.

Parameter	Description
expr	Return the formatted string representation of this numeric value.
form	Format *expr* using to this string value. If this is omitted then return the *expr* as a string.
firstday	Format using this day as the first day of the week. If this is omitted then the vbSunday is used.
firstweek	Format using this week as the first week of the year. If this is omitted then the vbFirstJan1 is used.

firstday	Value	Description
vbUseSystemFirstDay	0	Use the systems first day of the week.
vbSunday	1	Sunday (default)
vbMonday	2	Monday
vbTuesday	3	Tuesday
vbWednesday	4	Wednesday
vbThursday	5	Thursday
vbFriday	6	Friday
vbSaturday	7	Saturday

firstweek	Value	Description
vbUseSystem	0	Use the systems first week of the year.
vbFirstJan1	1	The week that January 1 occurs in. This is the default value.
2	vbFirstFourDays	The first week that has at least four days in the year.
3	vbFirstFullWeek	The first week that entirely in the year.

See Also

Predefined Date Format, **Predefined Number Format**, **User defined Date Format**, **User defined Number Format**, **User defined Text Format**.

Format Predefined Date

Description The following predefined date formats may be used with the **Format** function. Predefined formats may not be combined with user defined formats or other predefined formats.

Form	Description
General Date	Same as **user defined date format** "c"
Long Date	Same as **user defined date format** "dddddd"
Medium Date	**Not supported at this time.**
Short Date	Same as **user defined date format** "ddddd"
Long Time	Same as **user defined date format** "ttttt"
Medium Time	Same as **user defined date format** "hh:mm AMPM"
Short Time	Same as **user defined date format** "hh:mm"

Format Predefined Number

Description The following predefined number formats may be used with the **Format** function. Predefined formats may not be combined with user defined formats or other predefined formats.

Form	Description
General Number	Return number as is.
Currency	Same as **user defined number format** '#,##0.00;'#,##0.00)" **Not locale dependent at this time.**
Fixed	Same as **user defined number format** "0.00".
Standard	Same as **user defined number format** "#,##0.00".
Percent	Same as **user defined number format** "0.00%".
Scientific	Same as **user defined number format** "0.00E+00".
Yes/No	Return "No" if zero, else return "Yes".
True/False	Return "True" if zero, else return "False".
On/Off	Return "On" if zero, else return "Off".

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Forma'(2.145,"Standard") ' 2.15
End Sub
```

Format User Defined Date

Description The following date formats may be used with the **Format** function. Date formats may be combined to create the user defined date format. User defined date formats may not be combined with other user defined formats or predefined formats.

Parameter	Description
:	insert localized time separator
/	insert localized date separator
c	insert ddddd ttttt, insert date only if t=0, insert time only if d=0
d	insert day number without leading zero
dd	insert day number with leading zero
ddd	insert abbreviated day name
dddd	insert full day name
ddddd	insert date according to Short Date format
dddddd	insert date according to Long Date format
w	insert day of week number
ww	insert week of year number
m	insert month number without leading zero insert minute number without leading zero (if follows h or hh)
mm	insert month number with leading zero insert minute number with leading zero (if follows h or hh)
mmm	insert abbreviated month name
mmmm	insert full month name
q	insert quarter number
y	insert day of year number
yy	insert year number (two digits)
yyyy	insert year number (four digits, no leading zeros)
h	insert hour number without leading zero
hh	insert hour number with leading zero
n	insert minute number without leading zero
nn	insert minute number with leading zero
s	insert second number without leading zero
ss	insert second number with leading zero
ttttt	insert time according to time format
AM/PM	use 12 hour clock and insert AM (hours 0 to 11) and PM (12 to 23)
am/pm	use 12 hour clock and insert am (hours 0 to 11) and pm (12 to 23)
A/P	use 12 hour clock and insert A (hours 0 to 11) and P (12 to 23)
a/p	use 12 hour clock and insert a (hours 0 to 11) and p (12 to 23)
AMPM	use 12 hour clock and insert localized AM/PM strings
\c	insert character c
"text"	insert literal text

Example

Format User Defined Number

Description The following number formats may be used with the **Format** function. Number formats may be combined to create the user defined number format. User defined number formats may not be combined with other user defined formats or predefined formats.

User defined number formats can contain up to four sections separated by ';':

- form - format for non-negative expr, '-'format for negative expr, empty and null expr return ""
- form;negform - negform: format for negative expr
- form;negform;zeroform - zeroform: format for zero expr
- form;negform;zeroform;nullform - nullform: format for null expr

Parameter	Description
#	digit, don't include leading/trailing zero digits (all the digits left of decimal point are returned) eg. Format(19,"###") returns "19" eg. Format(19,"#") returns "19"
0	digit, include leading/trailing zero digits eg. Format(19,"000") returns "019" eg. Format(19,"0") returns "19"
.	decimal, insert localized decimal point eg. Format(19.9,"###.00") returns "19.90" eg. Format(19.9,"###.##") returns "19.9"
,	thousands, insert localized thousand separator every 3 digits "xxx," or "xxx,." mean divide expr by 1000 prior to formatting two adjacent commas ",," means divide expr by 1000 again eg. Format(1900000,"0,,") returns "2" eg. Format(1900000,"0,,.0") returns "1.9"
%	percent, insert %, multiply expr by 100 prior to formatting
:	insert localized time separator
/	insert localized date separator
E+ e+ E- e-	use exponential notation, insert E (or e) and the signed exponent eg. Format(1000,"0.00E+00") returns "1.00E+03" eg. Format(.001,"0.00E+00") returns "1.00E-03"
- +' ( ) space	insert literal char eg. Format(10,'#") returns '10"
\c	insert character c eg. Format(19,"\####\#") returns "#19#"
"text"	insert literal text eg. Format(19,"""##""###""##""") returns "##19##"

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print Forma'(2.145,"#.00") ' 2.15
End Sub
```

Format User Defined Text

Description The following text formats may be used with the **Format** function. Text formats may be combined to create the user defined text format. User defined text formats may not be combined with other user defined formats or pre-defined formats.

User defined text formats can contain one or two sections separated by ';':

- form - format for all strings
- form;nullform - nullform: format for empty and null strings

Parameter	Description
@	char placeholder, insert char or space
&	char placeholder, insert char or nothing
<	all chars lowercase
>	all chars uppercase
!	fill placeholder from left-to-right (default is right-to-left)
\c	insert character c
"text"	insert literal text

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print Format("123","ab@c")  '" ab1c23"
   Debug.Print Format("123","!ab@c") '" ab3c"
End Sub
```

FreeFile Function

Syntax FreeFile[( )]

Group File

Description Return the next unused shared stream number (greater than or equal to 256). Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print FreeFile ' 256
   FN = FreeFile
   FileOpen FN, "XXX", OpenMode.Output
   Debug.Print FreeFile ' 257
   FileClose'FN
   Debug.Print FreeFile ' 256
End Sub
```

Friend Keyword

Group Declaration

Description Friend **Function**s, **Property**s and **Sub**s in a *module* are available in all other *macros*/modules that access it. Friends are not accessible via **Object** variables.

Function Definition

Syntax

```
[ | Private | Public | Friend ] _
[ Default ] _
Function name[type][([param[, ...]])] _
  [As type[()]] [Handles var.eventname]
  statements
End Function
```

Group Declaration

Description User defined function. The function defines a set of *statement*s to be executed when it is called. The values of the calling *arglist* are assigned to the *param*s. Assigning to *name*[*type*] sets the value of the function result.

Access If no access is specified then **Public** is assumed.

Handles The Function provides an event handler for the **WithEvents** variable's (*var*) event (*eventname*).

See Also **Declare**, **Property**, **Sub**, **WithEvents**.

Example

```
'#Language "WWB.NET"
Function Power(X,Y)
  P = 1
  For I = 1 To Y
    P = P*X
  Next I
  Power = P
End Function

Sub Main
  Debug.Print Power(2,8) ' 256
End Sub
```

Get Instruction

Syntax Get'*StreamNum*, [*RecordNum*], *var*

Group File

Description Get a variable's value from *StreamNum*.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
RecordNum	For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1. If this is omitted then the current position (or record number) is used.
var	This variable value is read from the file. For a fixed length variable (like **Long**) the number of bytes required to restore the variable are read. For a **Variant** variable two bytes are read which describe its type and then the variable value is

read accordingly. For a *user structure* variable each field is read in sequence. For an array variable each element is read in sequence. For a dynamic array variable the number of dimensions and range of each dimension is read prior to reading the array values. All binary data values are read from the file in *little-endian* format.

Note: When reading a string (or a dynamic array) from a Binary mode file the length (or array dimension) information is not read. The current string length determines how much string data is read. The current array dimension determines how may array elements are read.

See Also **Open**, **Put**.

Example

```
'#Language "WWB.NET"
Sub Main
   Dim V As Variant
   FileOpen 1, "SAVE_V.DAT", OpenMode.Binary, OpenAccess.Read
   Get'1, , V
   FileClose'1
End Sub
```

GetAllSettings Function

Syntax GetAllSettings(*AppNam'*, *Sectio'*)

Group Settings

Description Get all of *Section*'s settings in project *AppName*. Settings are returned in a **Object**. **Nothing** is returned if there are no keys in the section. Otherwise, the Object contains a two dimension array: (I,0) is the key and (I,1) is the setting. Win16 and Win32s store settings in a .ini file named *AppName*. Win32/Win64 store settings in the registration database under "HKEY_CURRENT_USER\ Software\ VB and VBA Program Settings\ *AppName*\ *Section*\ *Key*". If *AppName* starts with "..\" then "VB and VBA Program Settings\" is omitted.

Parameter	Description
AppNam'	This string value is the name of the project which has this *Section* and *Key*.
Sectio'	This string value is the name of the section of the project settings.

Example

```
'#Language "WWB.NET"
Sub Main
   SaveSetting "MyApp","Font","Size",10
   SaveSetting "MyApp","Font","Name","Courier"
   Settings = GetAllSettings("MyApp","Font")
   For I = LBound(Settings) To UBound(Settings)
      Debug.Print Settings(I,0); "="; Settings(I,1)
   Next I
   DeleteSetting "MyApp","Font"
End Sub
```

GetAttr Function

Syntax GetAttr(*Nam'*)

Group File

Description Return the *attribute*s for file *Nam'*. If the file does not exist then a run-time error occurs.

Parameter	Description
Nam'	This string value is the path and name of the file. A path relative to the current directory can be used.

Example

```
'#Language "WWB.NET"
Sub Main
  ' = Di'("*.*")
  While ' <> ""
    Debug.Print '; " "; GetAttr(')
    ' = Di'()
  End While
End Sub
```

GetChar Function

Syntax GetChar(*Str*, *Index*)

Group String

Description Return the character at *Index*. The first char is at Index=1.

Parameter	Description
Str	Index into this string value.
Index	This is the index into the string value.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print GetChar("abcd",2) '"b"
End Sub
```

GetFilePat' Function

Syntax GetFilePat'([*DefNam'*], [*DefEx'*], [*DefDi'*], _
[*Titl'*], [*Option*])

Group User Input

Description Put up a dialog box and get a file path from the user. The returned string is a complete path and file name. If the cancel button is pressed then a null string is returned.

Parameter	Description
DefNam'	Set the initial File Name in the to this string value. If this is omitted then *.*DefEx'* is used.
DefEx'	Initially show files whose extension matches this string value. (Multiple extensions can be specified by using ";" as the separator.) If this is omitted then * is used. A "filter" may be specified using "description|*.ext|". Multiple descriptions can be used by repeating this format. (e.g. "Bitmap files|*.bmp|PNG files|*.png|JPEG files|*.jpg")
DefDi'	This string value is the initial directory. If this is omitted then the current directory is used.
Titl'	This string value is the title of the dialog. If this is omitted then "Get File Path" is used.
Option	This numeric value determines the file selection options. If this is omitted then zero is used. See table below.

Option	Effect
0	Only allow the user to select a file that exists.
1	Confirm creation when the user selects a file that does not exist.
2	Allow the user to select any file whether it exists or not.
3	Confirm overwrite when the user selects a file that exists.
+4	Selecting a different directory changes the application's current directory.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print GetFilePat'()
End Sub
```

GetObject Function

Syntax GetObject([*Fil'*][, *Clas'*])

Group Object

Description Get an existing object of type *Clas'* from *Fil'*.

Pocket PC Not supported.

Parameter	Description
Fil'	This is the file where the object resides. If this is omitted then the currently active object for *Clas'* is returned.
Clas'	This string value is the application's registered class name. If this application is not currently active it will be started. If this is omitted then the application associated with the file's extension will be started.

Example

```
'#Language "WWB.NET"
Sub Main
   Dim App As Object
  'App = GetObject(,"WinWrap.CppDemoApplication")
   App.Move 20,30 ' move icon to 20,30
  'App = Nothing
```

```
    App.Quit     ' run-time error (no object)
End Sub
```

GetLocale Function

Syntax GetLocale

Group Miscellaneous

Description Get the locale ID for the current thread.

See Also **SetLocale**.

Example

```
'#Language "WWB.NET"
Sub Main
    SetLocale &H409 ' English, US
    Debug.Print Hex(GetLocale) '"409"
End Sub
```

GetSetting Function

Syntax GetSettin'(*AppNam'*, *Sectio'*, *Ke'*[, *Defaul'*])

Group Settings

Description Get the setting for *Key* in *Section* in project *AppName*. Win16 and Win32s store settings in a .ini file named *AppName*. Win32/Win64 store settings in the registration database under "HKEY_CURRENT_USER\ Software\ VB and VBA Program Settings\ *AppName*\ *Section*\ *Key*". If *AppName* starts with "..\" then "VB and VBA Program Settings\" is omitted.

Parameter	Description
AppNam'	This string value is the name of the project which has this *Section* and *Key*.
Sectio'	This string value is the name of the section of the project settings.
Ke'	This string value is the name of the key in the section of the project settings.
Defaul'	Return this string value if no setting has been saved. If this is omitted then a null string is used.

Example

```
'#Language "WWB.NET"
Sub Main
    SaveSetting "MyApp","Font","Size",10
    Debug.Print GetSetting("MyApp","Font","Size") ' 10
End Sub
```

GetType Operator

Syntax GetType(*objtype*)

Group Operator

Description Return the .NET Type object for *objtype*.

This operator does not support user defined **Enum**s, **Structure**s, **Class Module**s or **Object Module**s.

See Also **Objects**.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print GetType(Integer).Name "'Integer"
End Sub
```

Goto Instruction

Syntax GoTo *label*

Group Flow Control

Description Go to the *label* and continue execution from there. Only *label*s in the current user defined *procedure* are accessible.

Example

```
'#Language "WWB.NET"
Sub Main
   X = 2
Loop:
   X = X*X
   If X < 100 Then GoTo Loop
   Debug.Print X ' 256
End Sub
```

GroupBox Dialog Item Definition

Syntax GroupBox *X*, *Y*, *DX*, *DY*, *Titl*[, .*Field*]

Group User Dialog

Description Define a groupbox item.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Titl'	This string value is the title of the group box.
Field	This identifier is the name of the field. The *dialogfunc* receives this name as *string*. If this identifer is omitted then the first two words of the title are used.

See Also **Begin Dialog**.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    GroupBox 10,25,180,60,"Group box"
    OKButton 80,90,40,20
    End Dialog
  Dim dlg As UserDialog
  Dialog dlg ' show dialog (wait for ok)
End Sub
```

He' Function

Syntax He'(*Num*)

Group String

Description Return a hex string.

Parameter	Description
Num	Return a hex encoded string for this numeric value.

See Also **Oc'( ), St'( ), Val( ).**

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print He'(15) 'F
End Sub
```

Hour Function

Syntax Hour(*dateexpr*)

Group Time/Date

Description Return the hour of the day (0 to 23).

Parameter	Description
dateexpr	Return the hour of the day for this date value. '

See Also **Minute( ), Second( ), Time( ).**

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Hour(#12:00:01 AM#) ' 0
End Sub
```

If Statement

Syntax

If *condexpr* Then [*instruction*] [Else *instruction*]
-or-
If *condexpr* Then
  *statement*s
[ElseIf *condexpr* Then
  *statement*s]...
[Else
  *statement*s]
End If

Group Flow Control

Description Form 1: Single line if statement. Execute the *instruction* following the Then if *condexpr* is **True**. Otherwise, execute the *instruction* following the Else. The Else portion is optional.

Form 2: The multiple line if is useful for complex ifs. Each if *condexpr* is checked in turn. The first **True** one causes the following *statement*s to be executed. If all are **False** then the Else's *statement*s are executed. The ElseIf and Else portions are optional.

Form 3: If *objexpr*'s type is the same type or a type descended from *objtype* the Then portion is executed.

See Also **Select Case**, **Choose( )**, **IIf( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  S = InputBox("Enter hello, goodbye, dinner or sleep:")
  S = UCase(S)
  If S = "HELLO" Then Debug.Print "come in"
  If S = "GOODBYE" Then Debug.Print "see you later"
  If S = "DINNER" Then
    Debug.Print "Please come in."
    Debug.Print "Dinner will be ready soon."
  ElseIf S = "SLEEP" Then
    Debug.Print "Sorry."
    Debug.Print "We are full for the night"
  End If
End Sub
```

IIf Function

Syntax IIf(*condexpr*, *TruePart*, *FalsePart*)

Group Miscellaneous

Description Return the value of the parameter indicated by *condexpr*. Both *TruePart* and *FalsePart* are evaluated.

Parameter	Description
condexpr	If this value is **True** then return *TruePart*. Otherwise, return *FalsePart*.
TruePart	Return this value if *condexpr* is **True**.
FalsePart	Return this value if *condexpr* is **False**.

See Also **If**, **Select Case**, **Choose( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Iif(1 > 0,"True","False") "'True"
End Sub
```

Imports Definition

Syntax Imports [alias =] prefix

Group Declaration

Description The Imports statement provides a convenient shorthand for using identifiers provided by references.

Parameter	Description
alias	Use this identifier as a substitute for "prefix". If omitted, no alternate identifier is defined.
prefix	Use this "prefix" before identifiers when searching references.

Example

```
'#Language "WWB.NET"
Imports System.Text
Imports StrBuf = System.Text.StringBuffer

Sub Main
  Dim sb As New StringBuffer ' System.Text.StringBuffer
  Dim sb2 As New StrBuf ' System.Text.StringBuffer
End Sub
```

Input Instruction

Syntax Input'*StreamNum*, *var*[, ...]

Group File

Description Get input from *StreamNum* and assign it to *var*s. Input values are comma delimited. Leading and trailing spaces are ignored. If the first char (following the leading spaces) is a quote (") then the string is terminated by an ending quote. Special values #NULL#, #FALSE#, #TRUE#, #date# and #ERROR number# are converted to their appropriate value and data type.

See Also **Line Input**, **Print**, **Write**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim A, B, '
  FileOpen 1, "XXX", OpenMode.Input
  Input'1, A, B, '
  Debug.Print A, B, '
  FileClose'1
End Sub
```

InputBo' Function

Syntax InputBo'(*Prompt'*[, *Titl'*][, *Defaul'*][, *XPos*, *YPos*])

Group User Input

Description Display an input box where the user can enter a line of text. Pressing the OK button returns the string entered. Pressing the Cancel button returns a null string.

Parameter	Description
Promp'	Use this string value as the prompt in the input box.
Titl'	Use this string value as the title of the input box. If this is omitted then the input box does not have a title.
Defaul'	Use this string value as the initial value in the input box. If this is omitted then the initial value is blank.
XPos	When the dialog is put up the left edge will be at this screen position. If this is omitted then the dialog will be centered.
YPos	When the dialog is put up the top edge will be at this screen position. If this is omitted then the dialog will be centered.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim '
  ' = InputBo'("Enter some text:", _
      "Input Box Example","asdf")
  Debug.Print '
End Sub
```

InputStrin' Function

Syntax InputStrin'(*StreamNum*, *N*)

Group File

Description Return *N* chars from *StreamNum*.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
N	Read this many chars. If fewer than that many chars are left before the end of file then a run-time error occurs.

Example

```
'#Language "WWB.NET"
Sub Main
  FileOpen 1, "XXX", OpenMode.Input
  Dim L, T
  L = LOF(1)
  T = InputString(1,L)
  FileClose 1
  Debug.Print T;
End Sub
```

InStr Function

Syntax InStr([*Index*,]*S'*, *S'*)

Group String

Description Return the index where *S'* first matches *S'*. If no match is found return 0.

Parameter	Description
Index	Start searching for *S'* at this index in *S'*. If this is omitted then start searching from the beginning of *S'*.
S'	Search for *S'* in this string value. '
S'	Search *S'* for this string value. '

See Also **InStrRev( ), Lef'( ), Len( ), Mi'( ), Replac'( ), Righ'( ).**

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print InStr("Hello","l") ' 3
End Sub
```

InStrRev Function

Syntax InStrRev(*S'*, *S'*[, *Index*])

Group String

Description Return the index where *S'* last matches *S'*. If no match is found return 0.

Parameter	Description
S'	Search for *S'* in this string value. '
S'	Search *S'* for this string value. '
Index	Start searching for *S'* ending at this index in *S'*. If this is omitted then start searching from the end of *S'*.

See Also **Lef'( ), Len( ), Mi'( ), Replac'( ), Righ'( ).**

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print InStrRev("Hello","l") ' 4
End Sub
```

Int Function

Syntax Int(*Num*)

Group Math

Description Return the integer value.

Parameter	Description
Num	Return the largest integer which is less than or equal to this numeric value. '

See Also **Fix**.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print Int(9.9)  ' 9
   Debug.Print Int(0)    ' 0
   Debug.Print Int(-9.9) '-10
End Sub
```

Integer Data Type

Syntax **Dim** v As Integer

Group Data Type

Description A 32 bit signed integer value.

Is Operator

Syntax *expr* Is *expr*
-or-
expr **IsNot** *expr*

Group Operator

Description Return the **True** if both *expr*s refer to the same object.

The IsNot operator inverts the result.

See Also **Objects**.

Example

```
'#Language "WWB.NET"
Sub Main
   Dim X As Object
   Dim Y As Object
   Debug.Print X Is Y ' True
   Debug.Print X IsNot Y ' False
End Sub
```

IsArray Function

Syntax IsArray(*expr*)

Group Variable Info

Description Return the **True** if *expr* is an array of values.

Parameter	Description
expr	A array variable or a variant var can contain multiple of values.

See Also **TypeName**, **VarType**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim X As Object, Y(2) As Integer
  Debug.Print IsArray(X) 'False
  X = Y
  Debug.Print IsArray(X) 'True
End Sub
```

IsDate Function

Syntax IsDate(*expr*)

Group Variable Info

Description Return the **True** if *expr* is a valid date.

Parameter	Description
expr	A variant expression to test for a valid date.

See Also **TypeName**, **VarType**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim X As Object
  X = 1
  Debug.Print IsDate(X) 'False
  X = Now
  Debug.Print IsDate(X) 'True
End Sub
```

IsDBNull Function

Syntax IsDBNull(*expr*)

Group Variable Info

Description Return the **True** if *expr* is System.DBNull.Value.

Parameter	Description

expr	A variant expression to test for System.DBNull.Value.

See Also **IsDBNull**, **TypeName**, **VarType**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim X As Object
  Debug.Print IsEmpty(X) 'True
  Debug.Print IsDBNull(X)  'False
  X = 1
  Debug.Print IsDBNull(X)  'False
  X = "1"
  Debug.Print IsDBNull(X)  'False
  X = System.DBNull.Value
  Debug.Print IsDBNull(X)  'True
  X = X*2
  Debug.Print IsDBNull(X)  'True
End Sub
```

IsError Function

Syntax IsError(*expr*)

Group Variable Info

Description Return the **True** if *expr* is an error code.

Parameter	Description
expr	A variant expression to test for an error code value.

See Also **TypeName**, **VarType**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim X As Object
  Debug.Print IsError(X) 'False
  X = CVErr(1)
  Debug.Print IsError(X) 'True
End Sub
```

IsNot Operator

Syntax *expr* IsNot *expr*

Group Operator

Description Return the **False** if both *expr*s refer to the same object.

See Also **Objects**.

Example

```
'#Language "WWB.NET"
Sub Main
```

```
  Dim X As Object
  Dim Y As Object
  Debug.Print X IsNot Y ' False
End Sub
```

IsNothing Function

Syntax IsNothing(*expr*)

Group Variable Info

Description Return the **True** if *expr* contains an object reference which is **Nothing**.

Parameter	Description
var	If *objexpr* reference is **Nothing** return **True**.

See Also **TypeName**, **VarType**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim X As Object
  X = 1
  Debug.Print IsNothing(X) 'False
  X = "1"
  Debug.Print IsNothing(X) 'False
 'X = Nothing
  Debug.Print IsNothing(X) 'True
End Sub
```

IsNumeric Function

Syntax IsNumeric(*expr*)

Group Variable Info

Description Return the **True** if *expr* is a numeric value.

Parameter	Description
expr	A variant expression is a numeric value if it is *numeric* or string value that represents a number.

See Also **TypeName**, **VarType**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim X As Object
  X = 1
  Debug.Print IsNumeric(X) 'True
  X = "1"
  Debug.Print IsNumeric(X) 'True
  X = "A"
```

```
  Debug.Print IsNumeric(X) 'False
End Sub
```

IsReference Function

Syntax IsReference(*expr*)

Group Variable Info

Description Return the **True** if *expr* contains an object reference.

Parameter	Description
var	If *expr* is an object reference return **True**.

See Also **TypeName**, **VarType**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim X As Object
  X = 1
  Debug.Print IsReference(X) 'False
  X = "1"
  Debug.Print IsReference(X) 'False
 'X = Nothing
  Debug.Print IsReference(X) 'True
End Sub
```

Join Function

Syntax Join(*StrArray*, [*Sep*])

Group Miscellaneous

Description Return a string by concatenating all the values in the array with Sep in between each one.

Parameter	Description
StrArray	Concatenate values from this array.
Sep	Use this string value to separate the values. (Default: " ")

See Also **Split( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Join(Array(1,2,3)) "'1 2 3"
End Sub
```

KeyName Function

Syntax KeyName(*Key*)

Group Miscellaneous

Description Return the key name for a key number. This is the name used by **SendKeys**.

Parameter	Description
Key	Key number.

See Also **SendKeys**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print KeyName(&H270)  "^{F1}"
End Sub
```

Kill Instruction

Syntax Kill *Nam'*

Group File

Description Delete the file named by *Nam'*.

Parameter	Description
Nam'	This string value is the path and name of the file. A path relative to the current directory can be used.

Example

```
'#Language "WWB.NET"
Sub Main
  Kill "XXX"
End Sub
```

LBound Function

Syntax LBound(*arrayvar*[, *dimension*])

Group Variable Info

Description Return the lowest index.

Parameter	Description
arrayvar	Return the lowest index for this array variable.
dimension	Return the lowest index for this dimension of *arrayvar*. If this is omitted then return the lowest index for the first dimension.

See Also **UBound( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim A(-1 To 3,2 To 6)
  Debug.Print LBound(A)   '-1
  Debug.Print LBound(A,1) '-1
```

```
  Debug.Print LBound(A,2) ' 2
End Sub
```

LCas' Function

Syntax LCas'(')

Group String

Description Return a string from ' where all the uppercase letters have been lowercased.

Parameter	Description
'	Return the string value of this after all chars have been converted to lowercase. '

See Also **StrComp( ), StrCon'( ), UCas'( ).**

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print LCas'("Hello") '"hello"
End Sub
```

Lef' Function

Syntax Lef'(', *Len*)

Group String

Description Return a string from ' with only the *Len* chars.

Parameter	Description
'	Return the left portion of this string value. '
Len	Return this many chars. If ' is shorter than that then just return '.

See Also **InStr( ), InStrRev( ), Len( ), Mi'( ), Replac'( ), Righ'( ).**

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Lef'("Hello",2) '"He"
End Sub
```

Len Function

Syntax Len(')
-or-
Len(*usertypevar*)

Group String

Description Return the number of characters in '.

Parameter	Description
'	Return the number of chars in this string value. '

usertypevar	Return the number of bytes required to store this user variable. If the user type has any dynamic **String** and **Variant** elements the length returned may not be as big as the actual number of bytes required.

See Also **InStr( )**, **InStrRev( )**, **Lef'( )**, **Mi'( )**, **Replac'( )**, **Righ'( )**.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print Len("Hello") ' 5
End Sub
```

Like Operator

Syntax *str1* Like *str2*

Group Operator

Description Return the **True** if *str1* matches pattern *str2*. The pattern in *str2* is one or more of the special character sequences shown in the following table.

Char(s)	Description
?	Match any single character.
*	Match zero or more characters.
#	Match a single digit (0-9).
[*charlist*]	Match any char in the list.
[!*charlist*]	Match any char not in the list.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print "abcdfgcdefg" Like "" ' False
   Debug.Print "abcdfgcdefg" Like "a*g" ' True
   Debug.Print "abcdfgcdefg" Like "a*cde*g" ' True
   Debug.Print "abcdfgcdefg" Like "a*cd*cd*g" ' True
   Debug.Print "abcdfgcdefg" Like "a*cd*cd*g" ' True
   Debug.Print "00aa" Like "####" ' False
   Debug.Print "00aa" Like "????" ' True
   Debug.Print "00aa" Like "##??" ' True
   Debug.Print "00aa" Like "*##*" ' True
   Debug.Print "hk" Like "hk*" ' True
End Sub
```

LineInput Function

Syntax LineInput(*StreamNum*)

Group File

Description Get a line of input from *StreamNum*.

See Also **Input**, **Print**, **Write**.

Example

```
'#Language "WWB.NET"
Sub Main
  FileOpen 1, "XXX", OpenMode.Input
  Dim S As String
  S = LineInput(1)
  Debug.Print S
  FileClose 1
End Sub
```

ListBox Dialog Item Definition

Syntax ListBox *X*, *Y*, *DX*, *DY*, *StrArra'( )*, *.Field*[, *Options*]

Group User Dialog

Description Define a listbox item.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
StrArra'( )	This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.
Field	The value of the list box is accessed via this field. It is the index of the *StrArra'( )* var.
Options	This numeric value controls the type of list box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option	Description
0	List is not sorted.
1	List is not sorted and horizontally scrollable.
2	List is sorted.
3	List is sorted and horizontally scrollable.

See Also **Begin Dialog**, **MultiListBox**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim list'(3)
  list'(0) = "List 0"
  list'(1) = "List 1"
  list'(2) = "List 2"
  list'(3) = "List 3"
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    ListBox 10,25,180,60,list'(),.list
    OKButton 80,90,40,20
```

```
  End Dialog
  Dim dlg As UserDialog
  dlg.list = 2
  Dialog dlg ' show dialog (wait for ok)
  Debug.Print dlg.list
End Sub
```

Loc Function

Syntax Loc(*StreamNum*)

Group File

Description Return *StreamNum* file position. For Random mode files this is the current record number minus one. For Binary mode files it is the current byte position minus one. Otherwise, it is the current byte position minus one divided by 128. The first position in the file is 0.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

Example

```
'#Language "WWB.NET"
Sub Main
  FileOpen 1, "XXX" OpenMode.Input
  L = Loc(1)
  FileClose'1
  Debug.Print L ' 0
End Sub
```

Lock Instruction

Syntax Lock *StreamNum*
-or-
Lock *StreamNum*, *RecordNum*
-or-
Lock *StreamNum*, [*start*] To *end*

Group File

Description Form 1: Lock all of *StreamNum*.

Form 2: Lock a record (or byte) of *StreamNum*.

Form 3: Lock a range of records (or bytes) of *StreamNum*. If *start* is omitted then lock starting at the first record (or byte).

Note: Be sure to **Unlock** for each Lock instruction.

Note: For sequential files (Input, Output and Append) lock always affects the entire file.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
RecordNum	For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1.
start	First record (or byte) in the range.
end	Last record (or byte) in the range.

See Also **Open**, **Unlock**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim V As Variant
  FileOpen 1, "SAVE_V.DAT" OpenMode.Binary
  Lock'1
  Get'1, 1, V
  V = "Hello"
  Put'1, 1, V
  Unlock'1
  FileClose'1
End Sub
```

LOF Function

Syntax LOF(*StreamNum*)

Group File

Description Return *StreamNum* file length (in bytes).

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

Example

```
'#Language "WWB.NET"
Sub Main
  FileOpen 1, "XXX" OpenMode.Input
  L = LOF(1)
  FileClose'1
  Debug.Print L
End Sub
```

Log Function

Syntax Log(*Num*)

Group Math

Description Return the natural logarithm.

Parameter	Description
Num	Return the natural logarithm of this numeric value. The value e is approximately 2.718282.

See Also **Exp**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Log(1) ' 0
End Sub
```

Long Data Type

Syntax **Dim** v As Long

Group Data Type

Description A 64 bit signed integer value. The number of bits is controlled by the **#Language** setting.

LSe' Function

Syntax LSe'(', *Len*)

Group String

Description Return a string from ' with only the *Len* chars.

Parameter	Description
'	Return the left portion of this string value.
Len	Return this many chars. If ' is shorter than that then fill the remainder with spaces.

See Also **RSe'( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print LSe'("Hello",6) '"Hello "
End Sub
```

LTri' Function

Syntax LTri'(')

Group String

Description Return the string with 's leading spaces removed.

Parameter	Description
'	Copy this string without the leading spaces. '

See Also **RTri'( )**, **Tri'( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print "."; LTri'("  x  "); "." "".x  ."
End Sub
```

MacroCheck Function

Syntax MacroCheck(*MacroNam'*)

Group Flow Control

Description Check the syntax of a *macro/module*. Does not execute the macro/module. Returns an Err object if there is a syntax error, otherwise Nothing is returned.

Parameter	Description
MacroNam'	Check the macro named by this string value.

See Also **MacroCheckThis**, **MacroRun**, **MacroRunThis**, **ModuleLoad**, **ModuleLoadThis**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim E As ErrObject
  'E = MacroCheck("Demo")
  If Not E Is Nothing Then Debug.Print E.Description
End Sub
```

MacroCheckThis Function

Syntax MacroCheckThis(*MacroCod'*)

Group Flow Control

Description Check the syntax the *macro/module* code in *MacroCode*. The macro/module code is not executed. Returns an Err object if there is a syntax error, otherwise Nothing is returned.

Parameter	Description
MacroNam'	Check the macro code in this string value.

See Also **MacroCheck**, **MacroRun**, **MacroRunThis**, **ModuleLoad**, **ModuleLoadThis**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim E As ErrObject
  'E = MacroCheckThis("bad macro")
  If Not E Is Nothing Then Debug.Print E.Description
End Sub
```

MacroDi' Function

Syntax MacroDi'

Group Flow Control

Description Return the directory of the current macro. A run-time error occurs if the current macro has never been saved.

See Also **MacroRun**.

Example

```
'#Language "WWB.NET"
Sub Main
   ' open the file called Data that is in the
   ' same directory as the macro
   FileOpen 1, MacroDir & "\Data", OpenMode.Input
   Dim S As String
   S = LineInput(1)
   Debug.Print '
   FileClose'1
End Sub
```

MacroRun Instruction

Syntax MacroRun *MacroNam'*[, *Comman'*]

Group Flow Control

Description Play a *macro*. Execution will continue at the following statement after the macro has completed.

Parameter	Description
MacroNam'	Run the macro named by this string value.
Comman'	Pass this string value as the macro's **Comman'** value.

See Also **Comman'**, **MacroCheck**, **MacroCheckThis**, **MacroDi'**, **MacroRunThis**, **ModuleLoad**, **ModuleLoadThis**.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print "Before Demo"
   MacroRun "Demo"
   Debug.Print "After Demo"
End Sub
```

MacroRunThis Instruction

Syntax MacroRunThis *MacroCod'*

Group Flow Control

Description Play the *macro* code in *MacroCode*. Execution will continue at the following statement after the macro code has completed. The macro code can be either a single line or a complete macro.

Parameter	Description
MacroNam'	Run the macro code in this string value.

See Also **Comman'**, **MacroCheck**, **MacroCheckThis**, **MacroDi'**, **MacroRun**, **ModuleLoad**, **ModuleLoadThis**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print "Before Demo"
  MacroRunThis "MsgBox ""Hello"""
  Debug.Print "After Demo"
End Sub
```

Main Sub

Syntax

```
Sub Main()
  ...
End Sub
-or-
Private Sub Main()
  ...
End Sub
```

Group Declaration

Description Form 1: Each *macro* must define Sub Main. A macro is a "program". Running a macro starts the Sub Main and continues to execute until the subroutine finishes.

Form 2: A code *module* may define a Private Sub Main. This Sub Main is the **code module** initialization subroutine. If Main is not defined then no special initialization occurs.

See Also **Code Module**.

Me Object

Syntax Me

Group Object

Description Me references the current macro/module. It can be used like any other *object variable*, except that it's reference can't be changed.

Example

```
'#Language "WWB.NET"
Sub Main
```

```
  DoIt
  Me.DoIt ' calls the same sub
End Sub
Sub DoIt
  MsgBox "Hello"
End Sub
```

Mi' Function/Assignment

Syntax Mi'(', *Index*[, *Len*])
-or-
Mi'(*strvar*, *Index*[, *Len*]) = '

Group String

Description Function: Return the substring of 'starting at *Index* for *Len* chars.

Instruction: Assign 'to the substring in *strvar* starting at *Index* for *Len* chars.

Parameter	Description (Mid Function)
'	Copy chars from this string value. '
Index	Start copying chars starting at this index value. If the string is not that long then return a null string.
Len	Copy this many chars. If the 'does not have that many chars starting at *Index* then copy the remainder of '.

Parameter	Description (Mid Assignment)
strvar	Change part of this string.
Index	Change *strvar* starting at this index value. If the string is not that long then it is not changed.
Len	The number of chars copied is smallest of: the value of *Len*, the length of 'and the remaining length of *strvar*. (If this value is omitted then the number of chars copied is the smallest of: the length of 'and the remaining length of *strvar*.)
'	Copy chars from this string value.

See Also **InStr( ), Lef'( ), Len( ), Replac'( ), Righ'( ).**

Example

```
'#Language "WWB.NET"
Sub Main
  ' = "Hello There"
  Mi'(',7) = "??????????"
  Debug.Print ' '"Hello ?????"
  Debug.Print Mi'("Hello",2,1) '"e"
End Sub
```

Minute Function

Syntax Minute(*dateexpr*)

Group Time/Date

Description Return the minute of the hour (0 to 59).

Parameter	Description
dateexpr	Return the minute of the hour for this date value. '

See Also **Hour( )**, **Second( )**, **Time( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Minute(#12:00:01 AM#) ' 0
End Sub
```

MkDir Instruction

Syntax MkDir *Nam'*

Group File

Description Make directory *Nam'*.

Parameter	Description
Nam'	This string value is the path and name of the directory. A path relative to the current directory can be used.

See Also **RmDir**.

Example

```
'#Language "WWB.NET"
Sub Main
  MkDir "C:\WWTEMP"
End Sub
```

ModuleLoad Function

Syntax ModuleLoad(*ModuleNam'*, *CreateNew*)

Group Flow Control

Description Load a *module*. Does not execute the module. Macro's can not be loaded. Returns an object if successful, otherwise Nothing is returned.

Parameter	Description
ModuleNam'	Load the module named by this string value.
CreateNew	Return a new instance if True. Otherwise return the default instance. A class module does not have a default instance. A code module can not have a new instance.

See Also **MacroCheck**, **MacroCheckThis**, **MacroRun**, **MacroRunThis**, **ModuleLoadThis**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim Obj As Object
  'Obj = ModuleLoad("Demo")
```

```
  Obj.DoIt ' call Demo's DoIt method
End Sub
```

ModuleLoadThis Function

Syntax ModuleLoadThis(*ModuleCod'*, *CreateNew*)

Group Flow Control

Description Load *ModuleCode* as a *module*. Does not execute the module. Macro's can not be loaded. Returns an object if successful, otherwise Nothing is returned.

Parameter	Description
ModuleCod'	Load the module code in this string value.
CreateNew	Return a new instance if True. Otherwise return the default instance. A class module does not have a default instance. A code module can not have a new instance.

See Also **MacroCheck**, **MacroCheckThis**, **MacroRun**, **MacroRunThis**, **ModuleLoad**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim Obj As Object
  'Obj = ModuleLoadThis("Sub DoIt" & vbCrLf & "End Sub", False)
  Obj.DoIt ' call Demo's DoIt method
End Sub
```

Month Function

Syntax Month(*dateexpr*)

Group Time/Date

Description Return the month of the year (1 to 12).

Parameter	Description
dateexpr	Return the month of the year for this date value. '

See Also **Date( )**, **Day( )**, **MonthName( )**, **Weekday( )**, **Year( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Month(#1/1/1900#) ' 1
  Debug.Print Month(#2/1/1900#) ' 2
End Sub
```

MonthName Function

Syntax MonthName(NumZ{month}[, CondZ{abbrev}])

Group Time/Date

Description Return the localized name of the month.

Parameter	Description
month	Return the localized name of this month. (1-12)
abbrev	If this conditional value is **True** then return the abbreviated form of the month name.

See Also **Month( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print MonthName(1) 'January
  Debug.Print MonthName(Month(Now))
End Sub
```

MsgBox Instruction/Function

Syntax MsgBox *Messag'*[, *Type*][, *Titl'*]
-or-
MsgBox(*Messag'*[, *Type*][, *Titl'*])

Group User Input

Description Show a message box titled *Titl'*. *Type* controls what the message box looks like (choose one value from each category). Use MsgBox( ) if you need to know what button was pressed. The result indicates which button was pressed.

Result	Value	Button Pressed
vbOK	1	OK button
vbCancel	2	Cancel button
vbAbort	3	Abort button
vbRetry	4	Retry button
vbIgnore	5	Ignore button
vbYes	6	Yes button
vbNo	7	No button

Parameter	Description
Messag'	This string value is the text that is shown in the message box.
Type	This numeric value controls the type of message box. Choose one value from each of the following tables.
Titl'	This string value is the title of the message box.

Button	Value	Effect
vbOkOnly	0	OK button
vbOkCancel	1	OK and Cancel buttons
vbAbortRetryIgnore	2	Abort, Retry, Ignore buttons
vbYesNoCancel	3	Yes, No, Cancel buttons
vbYesNo	4	Yes and No buttons
vbRetryCancel	5	Retry and Cancel buttons

Icon	Value	Effect
	0	No icon
vbCritical	16	Stop icon
vbQuestion	32	Question icon
vbExclamation	48	Attention icon
vbInformation	64	Information icon

Default	Value	Effect
vbDefaultButton1	0	First button
vbDefaultButton2	256	Second button
vbDefaultButton3	512	Third button

Mode	Value	Effect
vbApplicationModal	0	Application modal
vbSystemModal	4096	System modal
vbMsgBoxSetForeground	&h10000	Show message box in front of all other windows

Example

```
'#Language "WWB.NET"
Sub Main
  MsgBox "Please press OK button"
  If MsgBox("Please press OK button",vbOkCancel) = vbOK Then
    Debug.Print "OK was pressed"
  Else
  Debug.Print "Cancel was pressed"
  End If
End Sub
```

MultiListBox Dialog Item Definition

Syntax MultiListBox *X*, *Y*, *DX*, *DY*, *StrArra'( )*, *.Field*[, *Options*]

Group User Dialog

Description Define a multiple selection listbox item.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
StrArra'( )	This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.
Field	The values of the list box are accessed via this field. It is the index of the *StrArra'( )* var.
Options	This numeric value controls the type of list box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option	Description

0	List is not sorted.
1	List is not sorted and horizontally scrollable.
2	List is sorted.
3	List is sorted and horizontally scrollable.

See Also **Begin Dialog**, **ListBox**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim list'(3)
  list'(0) = "List 0"
  list'(1) = "List 1"
  list'(2) = "List 2"
  list'(3) = "List 3"
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    MultiListBox 10,25,180,60,list'(),.list
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  Dim selects() As Integer = {0,2}
  dlg.list = selects
  Dialog dlg ' show dialog (wait for ok)
  Dim i As Integer
  For i = LBound(dlg.list) To UBound(dlg.list)
    Debug.Print dlg.list(i);
  Next i
  Debug.Print
End Sub
```

New Operator

Syntax

New *objtype*[([*param*[, ...]])]
-or-
New *objtype*() { *expr*[, ...] \verb,},

Group Operator

Description Returns a new instance of *objtype*. Or, use { *expr*, ... } to create an array value.

Parameter	Description
objtype	This is the new object's type.
params	A list of zero or more *params* used during the object creation.

See Also **Objects**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim obj As Object
  'obj = New System.Collections.Hashtable
End Sub
```

Nothing Keyword

Group Constant

Description An *objexpr* that does not refer to any object.

Now Function

Syntax Now

Group Time/Date

Description Return the current date and time as a **Date** value.

See Also **Date**, **Time**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Now ' example: 1/1/1995 10:05:32 AM
End Sub
```

Object Data Type

Syntax **Dim** v As Object

Group Data Type

Description An object reference value. (see **Objects**) An object reference may also appear to have a data value, see table below:

Data	Description
get-reference	Use in any *object expression*.
set-reference	Use **Assign** to change the reference.
get-value	Use the reference as a data value. If it is not a data value, a "type mismatch error" will occur.
assign-value	Use **Assign** to change the value.

Object Module

Group Declaration

Description An object *module* implements an object.

- Has a set of **Public** *procedure*s accessible from other *macros* and *modules*.
- These public symbols are accessed via the name of the object module or an object variable.
- Public **Const**s, **Structure**s, arrays, fixed length strings are not allowed.

- Has an optional Private Sub **Object_Initialize** which is called when an instance is created.
- Has an optional Private Sub **Object_Terminate** which is called when an instance is destroyed.
- An object module is similar to a **class module** except that one instance is automatically created. That instance has the same name as the object module's name.
- To create additional instances use:

  Dim Obj As objectname
  Obj = **New** objectname

See Also **Class Module**, **Code Module**, **Uses**, **Object_Initialize**, **Object_Terminate**.

Example

```
'A.WWB
'#Language "WWB.NET"
'#Uses "System.OBM"
Sub Main
  Debug.Print Hex(System.Version)
End Sub

'System.OBM
'File|New Module|Object Module
'Edit|Properties|Name=System
'#Language "WWB.NET"
Option Explicit
Declare Function GetVersion16 Lib "Kernel" _
  Alias "GetVersion" () As Long
Declare Function GetVersion Lib "Kernel32" () As PortInt

Public Function Version() As PortInt
  If Win16 Then
    Version = GetVersion16
  Else
    Version = GetVersion
  End If
End Function
```

Object_Initialize Sub

Syntax

```
Private Sub Object_Initialize()
  ...
End Sub
```

Group Declaration

Description Object module initialization subroutine. Each time a new instance is created for a Object module the Object_Initialize sub is called. If Object_Initialize is not defined then no special initialization occurs.

Note: Object_Initialize is also called for the instance that is automatically created.

See Also **Object Module**, **Object_Terminate**.

Object_Terminate Sub

Syntax

```
Private Sub Object_Terminate()
  ...
End Sub
```

Group Declaration

Description Object module termination subroutine. Each time an instance is destroyed for a Object module the Object_Terminate sub is called. If Object_Terminate is not defined then no special termination occurs.

See Also **Object Module**, **Object_Initialize**.

Oc' Function

Syntax Oc'(*Num*)

Group String

Description Return a octal string.

Parameter	Description
Num	Return an octal encoded string for this numeric value.

See Also **He'( )**, **St'( )**, **Val( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Oc'(15) '17
End Sub
```

OKButton Dialog Item Definition

Syntax OKButton *X*, *Y*, *DX*, *DY*[, .*Field*]

Group User Dialog

Description Define an OK button item. Pressing the OK button updates the *dlgvar* field values and closes the dialog. (**Dialog**( ) function call returns -1.)

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.

DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Field	This identifier is the name of the field. The *dialogfunc* receives this name as *string*. If this is omitted then the field name is "OK".

See Also **Begin Dialog**.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,30,"Please push the OK button"
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  Dialog dlg ' show dialog (wait for ok)
End Sub
```

On Error Instruction

Syntax

```
On Error GoTo 0
-or-
On Error GoTo label
-or-
On Error Resume Next
```

Group Error Handling

Description Form 1: Disable the error handler (default).

Form 2: Send error conditions to an error handler.

Form 3: Error conditions continue execution at the next statement.

On Error sets or disables the error handler. Each user defined *procedure* has its own error handler. The default is to terminate the *macro* on any error. The **Err** object's properties are set whenever an error occurs. Once an error has occurred and the error handler is executing any further errors will terminate the macro, unless the **Err** object has been cleared.

Note: This instruction clears the **Err** and sets **Error`$'** to null.

Example

```
'#Language "WWB.NET"
Sub Main
  On Error Resume Next
  Err.Raise 1
  Debug.Print "RESUMING, Err="; Err
  On Error GoTo X
  Err.Raise 1
  Exit Sub
```

```
X:  Debug.Print "Err="; Err
    Err.Clear
    Debug.Print "Err="; Err
    Resume Next
End Sub
```

Operators

Syntax

^ Not * / \ Mod + - << >> & < <= > >= = <> **Is IsNot**
And AndAlso Or OrElse Xor

Description

These operators are available for numbers n1 and n2 or strings s1 and s2. If any value in an expression is System.DBNull.Value then the expression's value is System.DBNull.Value. The order of operator evaluation is controlled by operator *precedence*.

Operator	Description
- *n1*	Negate *n1*.
n1 ^ *n2*	Raise *n1* to the power of *n2*.
n1 * *n2*	Multiply *n1* by *n2*.
n1 / *n2*	Divide *n1* by *n2*.
n1 \ *n2*	Divide the integer value of *n1* by the integer value of *n2*.
n1 Mod *n2*	Remainder of the integer value of *n1* after dividing by the integer value of *n2*.
n1 + *n2*	Add *n1* to *n2*.
s1 + *s2*	Concatenate *s1* with *s2*.
n1 - *n2*	Difference of *n1* and *n2*.
n1 << *n2*	Shift *n1* by *n2* bits to the left. The number of bits to shift is indicated by the lowest bits of *n2*.
n1 >> *n2*	Shift *n1* by *n2* bits to the right. The number of bits to shift is indicated by the lowest bits of *n2*. (For signed numbers the sign bit is propagated to the right.)
s1 & *s2*	Concatenate *s1* with *s2*.
n1 < *n2*	Return **True** if *n1* is less than *n2*.
n1 <= *n2*	Return **True** if *n1* is less than or equal to *n2*.
n1 > *n2*	Return **True** if *n1* is greater than *n2*.
n1 >= *n2*	Return **True** if *n1* is greater than or equal to *n2*.
n1 = *n2*	Return **True** if *n1* is equal to *n2*.
n1 <> *n2*	Return **True** if *n1* is not equal to *n2*.
s1 < *s2*	Return **True** if *s1* is less than *s2*.
s1 <= *s2*	Return **True** if *s1* is less than or equal to *s2*.
s1 > *s2*	Return **True** if *s1* is greater than *s2*.
s1 >= *s2*	Return **True** if *s1* is greater than or equal to *s2*.
s1 = *s2*	Return **True** if *s1* is equal to *s2*.
s1 <> *s2*	Return **True** if *s1* is not equal to *s2*.
Not *n1*	Bitwise invert the integer value of *n1*. Only Not **True** is **False**.
n1 And *n2*	Bitwise and the integer value of *n1* with the integer value *n2*.
n1 AndAlso *n2*	Logical and of *n1* with the *n2*. If *n1* is **False**, *n2* is not evaluated.
n1 Or *n2*	Bitwise or the integer value of *n1* with the integer value *n2*.
n1 OrElse *n2*	Logical or of *n1* with the *n2*. If *n1* is **True**, *n2* is not evaluated.
n1 Xor *n2*	Bitwise exclusive-or the integer value of *n1* with the integer value *n2*.

Example

```
'#Language "WWB.NET"
Sub Main
  N1 = 10
  N2 = 3
  S' = "asdfg"
  S' = "hjkl"
  Debug.Print -N1      '-10
  Debug.Print N1 ^ N2   ' 1000
  Debug.Print Not N1    '-11
  Debug.Print N1 * N2   ' 30
  Debug.Print N1 / N2   ' 3.3333333333333
  Debug.Print N1 \ N2   ' 3
  Debug.Print N1 Mod N2  ' 1
  Debug.Print N1 + N2   ' 13
  Debug.Print S' + S'  '"asdfghjkl"
  Debug.Print N1 - N2   ' 7
  Debug.Print N1 << N2  ' 80
  Debug.Print N1 >> N2  ' 1
  Debug.Print N1 & N2   '"103"
  Debug.Print N1 < N2   'False
  Debug.Print N1 <= N2  'False
  Debug.Print N1 > N2   'True
  Debug.Print N1 >= N2  'True
  Debug.Print N1 = N2   'False
  Debug.Print N1 <> N2  'True
  Debug.Print S' < S'  'True
  Debug.Print S' <= S' 'True
  Debug.Print S' > S'  'False
  Debug.Print S' >= S' 'False
  Debug.Print S' = S'  'False
  Debug.Print S' <> S' 'True
  Debug.Print N1 And N2    ' 2
  Debug.Print N1 AndAlso N2 ' True
  Debug.Print N1 Or N2     ' 11
  Debug.Print N1 OrElse N2  ' True
  Debug.Print N1 Xor N2    ' 9
End Sub
```

Option Definition

Syntax

Option Compare [Binary | **Text**]
-or-
Option Explicit [On | Off]
-or-
Option Strict [On | Off]
-or-
Option **Private** Module

Group

Declaration

Description

Option Compare: Set the default comparison mode for <, <=, =, >, >=, <>, **Like** and **StrComp**.

- Option Compare Binary - compare string text using binary data (default)

- Option Compare Text - compare string text using the collation rules

Option Explicit On: Require all variables to be declared prior to use. Variables are declared using **Dim**, **Private**, **Public**, **Static** or as a parameter of **Sub**, **Function** or **Property** blocks. (This is the default.)

Option Strict: Ignored.

Option Private: Public symbols defined by the module are only accessible from the same project.

Example

```
'#Language "WWB.NET"
Option Explicit Off

Sub Main
  A = 1 ' don't need to declare
End Sub
```

OptionButton Dialog Item Definition

Syntax OptionButton *X*, *Y*, *DX*, *DY*, *Titl*'[, .*Field*]

Group User Dialog

Description Define an option button item.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Titl'	The value of this string is the title of the option button.

See Also **Begin Dialog**, **OptionGroup**.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    OptionGroup .options
      OptionButton 10,30,180,15,"Option &0"
      OptionButton 10,45,180,15,"Option &1"
      OptionButton 10,60,180,15,"Option &2"
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  dlg.options = 2
  Dialog dlg ' show dialog (wait for ok)
```

```
  Debug.Print dlg.options
End Sub
```

OptionGroup Dialog Item Definition

Syntax

OptionGroup .*Field*
OptionButton *X*, *Y*, *DX*, *DY*, *Titl*'[, .*Field*]
OptionButton *X*, *Y*, *DX*, *DY*, *Titl*'[, .*Field*]
...

Group

User Dialog

Description

Define a optiongroup and option button items.

Parameter	Description
Field	The value of the option group is accessed via this field. This first option button is 0, the second is 1, etc.
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Titl'	The value of this string is the title of the option button.

See Also

Begin Dialog, **OptionButton**.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    OptionGroup .options
      OptionButton 10,30,180,15,"Option &0"
      OptionButton 10,45,180,15,"Option &1"
      OptionButton 10,60,180,15,"Option &2"
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  dlg.options = 2
  Dialog dlg ' show dialog (wait for ok)
  Debug.Print dlg.options
End Sub
```

Picture Dialog Item Definition

Syntax

Picture *X*, *Y*, *DX*, *DY*, *FileNam'*, *Type*[, .*Field*]

Group

User Dialog

Description Define a picture item. The bitmap is automatically sized to fit the item's entire area.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
FileNam'	The value of this string is the .BMP file shown in the picture control.
Type	This numeric value indicates the type of bitmap used. See below.
Field	This identifier is the name of the field. The *dialogfunc* receives this name as *string*. If this identifer is omitted then the first two words of the title are used.

Type	Effect
0	*FileName* is the name of the bitmap file. If the file does not exist then "(missing picture)" is displayed.
3	The clipboard's bitmap is displayed. Not supported.
+16	Instead of displaying "(missing picture)" a run-time error occurs.

See Also **Begin Dialog**.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120
    Picture 10,10,180,75,"SAMPLE.BMP",0
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  Dialog dlg ' show dialog (wait for ok)
End Sub
```

Print Instruction

Syntax Print *StreamNum*, [*expr*[, ...]]

Group File

Description Print the *expr*(s) to *StreamNum*. A *num* is it automatically converted to a string before printing (just like **St'( )**).

See Also **Input**, **LineInput**, **PrintLine**, **Write**, **WriteLine**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim A, B, '
  A = 1
  B = 2
  ' = "Hello"
```

```
  FileOpen 1, "XXX", OpenMode.Output
  Print'1, A, ",", B, ",""", ', """"
  FileClose'1
End Sub
```

PrintLine Instruction

Syntax PrintLine *StreamNum*, [*expr*[, ...]]

Group File

Description Print the *expr*(s) to *StreamNum*. A *num* is it automatically converted to a string before printing (just like **St'( )**). A newline is printed at the end.

See Also **Input**, **LineInput**, **Print**, **Write**, **WriteLine**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim A, B, C
  A = 1
  B = 2
  C = "Hello"
  FileOpen 1, "XXX", OpenMode.Output
  PrintLine 1, A, ",", B, ",""", C, """"
  FileClose 1
End Sub
```

Private Definition

Syntax Private [**WithEvents**] *vardeclaration* [= *initialvalue*][, ...]

Group Declaration

Description Create arrays (or simple variables) which are available to the entire *macro*/*module*, but not other macros/modules. Dimension var array(s) using the *dimension*s to establish the minimum and maximum index value for each dimension. If the *dimension*s are omitted then a scalar (single value) variable is defined. A dynamic array is declared using ( ) without any *dimension*s. It must be **ReDim**ensioned before it can be used. The Private statement must be placed outside of **Sub**, **Function** or **Property** blocks.

See Also **Dim**, **Option** Base, **Public**, **ReDim**, **Static**, **WithEvents**.

Example

```
'#Language "WWB.NET"
Private A0, A1(1), A2(1,1)

Sub Init
  A0 = 1
  A1(0) = 2
  A2(0,0) = 3
End Sub
```

```
Sub Main
  Init
  Debug.Print A0; A1(0); A2(0,0) ' 1 2 3
End Sub
```

Private Keyword

Group Declaration

Description **Private Const**s, **Declare**s, **Function**s, **Property**s, **Sub**s and **Structure**s are only available in the current *macro/module*.

Property Definition

Syntax

```
[ | Private | Public | Friend ] _
[ Default ] [ | ReadOnly | WriteOnly ] _
Property name[type][([param])] [As type[()]]
  [Get
    statements
  End Get]
  [Set(param)
    statements
  End Set]
End Property
```

Group Declaration

Description User defined property. The property defines a set of *statement*s to be executed when its value is used or changed. A property acts like a variable, except that getting its value calls Property's Get block and changing its value calls Property's Set block. The values of the calling *arglist* are assigned to the *param*s.

- At a minimum, either a Get or Set block must be specified.
- If only the Get block is specified the property must be marked as ReadOnly.
- If only the Set block is specified the property must be marked as WriteOnly.
- If both the Get and Set blocks are specified the property must not be marked as ReadOnly or WriteOnly.
- The Set block's parameter type must match the Property's return type.

Access If no access is specified then **Public** is assumed.

See Also **Function**, **Sub**.

Example

```
'#Language "WWB.NET"
Dim X_Value As String

Property X() As String
```

```
  Get
    X = X_Value
  End Get
  Set(ByVal Value As String)
    X_Value = Value
  End Set
End Property

Sub Main
  X = "Hello"
  Debug.Print X '"Hello"
End Sub
```

Public Definition

Syntax Public [**WithEvents**] *vardeclaration* [= *initialvalue*][, ...]

Group Declaration

Description Create arrays (or simple variables) which are available to the entire *macro*/*module* and other macros/modules. Dimension var array(s) using the *dimension*s to establish the minimum and maximum index value for each dimension. If the *dimension*s are omitted then a scalar (single value) variable is defined. A dynamic array is declared using ( ) without any *dimension*s. It must be **ReDim**ensioned before it can be used. The Public statement must be placed outside of **Sub**, **Function** or **Property** blocks.

See Also **Dim**, **Option** Base, **Private**, **ReDim**, **Static**, **WithEvents**.

Example

```
'#Language "WWB.NET"
Public A0, A1(1), A2(1,1)

Sub Init
  A0 = 1
  A1(0) = 2
  A2(0,0) = 3
End Sub

Sub Main
  Init
  Debug.Print A0; A1(0); A2(0,0) ' 1 2 3
End Sub
```

Public Keyword

Group Declaration

Description **Public Const**s, **Declare**s, **Function**s, **Property**s, **Sub**s and **Structure**s in a *module* are available in all other *macro*s/modules that access it.

PushButton Dialog Item Definition

Syntax PushButton *X*, *Y*, *DX*, *DY*, *Titl*'[, .*Field*]

Group User Dialog

Description Define a push button item. Pressing the push button updates the *dlgvar* field values and closes the dialog. (**Dialog**( ) function call returns the push button's ordinal number in the dialog. The first push button returns 1.)

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Titl'	The value of this string is the title of the push button control.
Field	This identifier is the name of the field. The *dialogfunc* receives this name as *string*. If this identifer is omitted then the first two words of the title are used.

See Also **Begin Dialog**.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,30,"Please push the DoIt button"
    OKButton 40,90,40,20
    PushButton 110,90,60,20,"&Do It"
  End Dialog
  Dim dlg As UserDialog
  Debug.Print Dialog(dlg)
End Sub
```

Put Instruction

Syntax Put'*StreamNum*, [*RecordNum*], *var*

Group File

Description Write a variable's value to *StreamNum*.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
RecordNum	For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1. If this is omitted then the current position (or record number) is used.

var	This variable value is written to the file. For a fixed length variable (like **Long**) the number of bytes required to store the variable are written. For a **Variant** variable two bytes which describe its type are written and then the variable value is written accordingly. For a *user structure* variable each field is written in sequence. For an array variable each element is written in sequence. For a dynamic array variable the number of dimensions and range of each dimension is written prior to writing the array values. All binary data values are written to the file in *little-endian* format. Note: When a writing string (or a dynamic array) to a Binary mode file the string length (or array dimension) information is not written. Only the string data or array elements are written.

See Also **Get**, **Open**.

Example

```
'#Language "WWB.NET"
Sub Main
   Dim V As Variant
   FileOpen 1, "SAVE_V.DAT", OpenMode.Binary, OpenAccess.Access
   Put'1, , V
   FileClose'1
End Sub
```

QBColor Function

Syntax QBColor(*num*)

Group Miscellaneous

Description Return the appropriate color defined by Quick Basic.

num	color
0	black
1	blue
2	green
3	cyan
4	red
5	magenta
6	yellow
7	white
8	gray
9	light blue
10	light green
11	light cyan
12	light red
13	light magenta
14	light yellow
15	bright white

See Also **RGB( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Hex(QBColor(1))  "'800000"
  Debug.Print Hex(QBColor(7))  "'C0C0C0"
  Debug.Print Hex(QBColor(8))  "'808080"
  Debug.Print Hex(QBColor(9))  "'FF0000"
  Debug.Print Hex(QBColor(10)) "'FF00"
  Debug.Print Hex(QBColor(12)) "'FF"
  Debug.Print Hex(QBColor(15)) "'FFFFFF"
End Sub
```

RaiseEvent Instruction

Syntax RaiseEvent *name*[(*arglist*)]

Group Flow Control

Description Raise an **Event**. Evaluate the *arglist* and call *name* with those values.

See Also **Event**, **WithEvents**.

Example

```
'Class1.cls
'#Language "WWB.NET"
Event Changing(ByVal OldValue As String, ByVal NewValue As String)

Private Value_ As String

Property Get Value As String
  Value = Value_
End Property

Property Let Value(ByVal NewValue As String)
  RaiseEvent Changing(Value_, NewValue)
  Value_ = NewValue
End Property

'Macro.bas
'#Uses "Class1.cls"
'#Language "WWB.NET"

Dim WithEvents c1 As Class1

Sub Main
  'c1 = New Class1
  c1.Value = "Hello"
  c1.Value = "Goodbye"
End Sub

Sub c1_Changing(ByVal OldValue As String, ByVal NewValue As String) Handles
c1.Changing
  Debug.Print "OldValue=""" & OldValue & """, NewValue=""" & NewValue & """"
End Sub
```

Randomize Instruction

Syntax Randomize [*Seed*]

Group Math

Description Randomize the random number generator.

Parameter	Description
Seed	This numeric value sets the initial seed for the random number generator. If this value is omitted then the current time is used as the seed.

See Also **Rnd( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Randomize
  Debug.Print Rnd ' 0.??????????????
End Sub
```

ReDim Instruction

Syntax ReDim [Preserve] *vardeclaration* [*As type*][, ...]

Group Declaration

Description Redimension a dynamic *arrayvar* or *user defined structure* array element. Use Preserve to keep the array values. Otherwise, the array values will all be reset. When using preserve only the last index of the array may change, but the number of indexes may not. (A one-dimensional array can't be redimensioned as a two-dimensional array.)

See Also **Dim**, **Option** Base, **Private**, **Public**, **Static**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim X()
  ReDim X(3)
  Debug.Print UBound(X) ' 3
  ReDim X(200)
  Debug.Print UBound(X) ' 200
End Sub
```

Rem Instruction

Syntax Rem ...
-or-
'...

Group Miscellaneous

Description Both forms are comments. The Rem form is an instruction. The ' form can be used at the end of any line. All text from either ' or Rem to the end of the line is part of the comment. That text is not executed.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print "Hello" ' prints to the output window
   Rem the macro terminates at Main's End Sub
End Sub
```

RemoveHandler Instruction

Syntax RemoveHandler *expr.name*, *delegate*

Group Flow Control

Description Remove a *delegate* from the *expr.name*'s list of handlers.

Parameter	Description
expr	This is an expression which returns an object reference.
name	This is an event name.
delegate	This is an expression which returns a delegate.

See Also **AddHandler**.

Example

```
'#Reference #System.Windows.Forms, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089, processorArchitecture=MSIL
'#Language "WWB.NET"
Imports System.Windows.Forms

Sub Main
   Using t As New Timer
      AddHandler t.Tick, AddressOf OnTick
      t.Interval = 1000
      t.Enabled = True
      Wait 5
      RemoveHandler t.Tick, AddressOf OnTick
   End Using
End Sub

Private Sub OnTick(ByVal sender As Object, ByVal e As System.EventArgs)
   Debug.Print Now
End Sub
```

Rename Instruction

Syntax Rename *OldNam'*, *NewNam'*

Group File

Description Rename file *OldNam'* as *NewNam'*.

Parameter	Description
OldNam'	This string value is the path and name of the file. A path relative to the current directory can be used.
NewNam'	This is the new file name (and path). A path relative to the current directory can be used.

Example

```
'#Language "WWB.NET"
Sub Main
   Rename "AUTOEXEC.BAK", "AUTOEXEC.SAV"
End Sub
```

Replac' Function

Syntax Replac'(', *Pa'*, *Re'*, [*Index*], [*Count*])

Group String

Description Replace *Pa'* with *Re'* in '.

Parameter	Description
'	This string value is searched. Replacements are made in the string returned by Replace.
Pa'	This string value is the pattern to look for.
Re'	This string value is the replacement.
Index	This numeric value is the starting index in '. Replace(S,Pat,Rep,N) is equivalent to Replace(Mid(S,N),Pat,Rep). If this is omitted use 1.
Count	This numeric value is the maximum number of replacements that will be done. If this is omitted use -1 (which means replace all occurrences).

See Also **InStr( )**, **InStrRev( )**, **Lef'( )**, **Len( )**, **Mi'( )**, **Righ'( )**.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print Replac'("abcabc","b","B")    '"aBcaBc"
   Debug.Print Replac'("abcabc","b","B",,1) '"aBcabc"
   Debug.Print Replac'("abcabc","b","B",3)  '"caBc"
   Debug.Print Replac'("abcabc","b","B",9)  ""
End Sub
```

Reset Instruction

Syntax Reset

Group File

Description Close all open streams for the current *macro*/*module*.

See Also **Close**, **Open**.

Example

```
'#Language "WWB.NET"
Sub Main
```

```
  ' read the first line of XXX and print it
  FileOpen 1, "XXX", OpenMode.Input
  Dim L As String
  L = LineInput(1)
  Debug.Print '
  Reset
End Sub
```

Resume Instruction

Syntax Resume *label*
-or-
Resume Next

Group Error Handling

Description Form 1: Resume execution at *label*.

Form 2: Resume execution at the next statement.

Once an error has occurred, the error handler can use Resume to continue execution. The error handler must use Resume or **Exit** at the end.

Note: This instruction clears the **Err** and sets **Error`$'** to null.

Example

```
'#Language "WWB.NET"
Sub Main
  On Error GoTo X
  Err.Raise 1
  Debug.Print "RESUMING"
  Exit Sub

X:  Debug.Print "Err="; Err
  Resume Next
End Sub
```

Return Instruction

Syntax Return *expr*

Group Flow Control

Description The return instruction causes the **Function** block to exit with the value of the *expr*.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Func(2) ' 6
End Sub
```

```
Function Func(N)
  Return N*3
End Function
```

RGB Function

Syntax RGB(*red*, *green*, *blue*)

Group Miscellaneous

Description Return a color. Some useful color constants are predefined:

- vbBlack - same as RGB(0,0,0)
- vbRed - same as RGB(255,0,0)
- vbGreen - same as RGB(0,255,0)
- vbYellow - same as RGB(255,255,0)
- vbBlue - same as RGB(0,0,255)
- vbMagenta - same as RGB(255,0,255)
- vbCyan - same as RGB(0,255,255)
- vbWhite - same as RGB(255,255,255)

See Also **QBColor( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Hex(RGB(255,0,0)) '"FF0000"
End Sub
```

Righ' Function

Syntax Righ'(', *Len*)

Group String

Description Return the last *Len* chars of '.

Parameter	Description
'	Return the right portion of this string value. '
Len	Return this many chars. If 'is shorter than that then just return '.

See Also **InStr( )**, **InStrRev( )**, **Lef'( )**, **Len( )**, **Mi'( )**, **Replac'( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Righ'("Hello",3) '"llo"
End Sub
```

RmDir Instruction

Syntax RmDir *Nam'*

Group File

Description Remove directory *Nam'*.

Parameter	Description
Nam'	This string value is the path and name of the directory. A path relative to the current directory can be used.

See Also **MkDir**.

Example

```
'#Language "WWB.NET"
Sub Main
  RmDir "C:\WWTEMP"
End Sub
```

Rnd Function

Syntax Rnd([*Num*])

Group Math

Description Return a random number greater than or equal to zero and less than one.

Parameter	Description
Num	See table below.

Num	Description
<0	Return the same number every time, using *Num* as the seed.
>0	Return the next random number in the sequence.
0	Return the most recently generated number.
omitted	Return the next random number in the sequence.

See Also **Randomize**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Rnd() ' 0.??????????????
End Sub
```

Round Function

Syntax Round([*Num*][, *Places*])

Group Math

Description Return the number rounded to the specified number of decimal places.

Parameter	Description
Num	Round this numeric value. '
Places	Round to this number of decimal places. If this is omitted then round to the nearest integer value.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Round(.5)      ' 0
  Debug.Print Round(.500001)  ' 1
  Debug.Print Round(1.499999) ' 1
  Debug.Print Round(1.5)     ' 2
  Debug.Print Round(11.11)    ' 11
  Debug.Print Round(11.11,1)  ' 11.1
End Sub
```

RSe' Function

Syntax RSe'(', *Len*)

Group String

Description Return a string from ' with only the *Len* chars.

Parameter	Description
'	Return the this string value right justified.
Len	Return this many chars. If ' is shorter than that then fill the remainder with spaces.

See Also **LSe'( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print RSe'("Hello",6) '"Hello "
End Sub
```

RTri' Function

Syntax RTri'(')

Group String

Description Return the string with ' s trailing spaces removed.

Parameter	Description
'	Copy this string without the trailing spaces. '

See Also **LTri'( )**, **Tri'( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print "."; RTri'("  x  "); "." '".  x."
End Sub
```

SaveSetting Instruction

Syntax SaveSetting *AppNam'*, *Sectio'*, *Ke'*, *Setting*

Group Settings

Description Save the *Setting* for *Key* in *Section* in project *AppName*. Win16 and Win32s store settings in a .ini file named *AppName*. Win32/Win64 store settings in the registration database under "HKEY_CURRENT_USER\ Software\ VB and VBA Program Settings\ *AppName*\ *Section*\ *Key*". If *AppName* starts with "..\" then "VB and VBA Program Settings\" is omitted.

Parameter	Description
AppNam'	This string value is the name of the project which has this *Section* and *Key*.
Sectio'	This string value is the name of the section of the project settings.
Ke'	This string value is the name of the key in the section of the project settings.
Setting	Set the key to this value. (The value is stored as a string.)

Example

```
'#Language "WWB.NET"
Sub Main
   SaveSetting "MyApp","Font","Size",10
End Sub
```

Second Function

Syntax Second(*dateexpr*)

Group Time/Date

Description Return the second of the minute (0 to 59).

Parameter	Description
dateexpr	Return the second of the minute for this date value. '

See Also **Hour( )**, **Minute( )**, **Time( )**.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print Second(#12:00:01 AM#) ' 1
End Sub
```

Seek Instruction

Syntax Seek'*StreamNum*, *Count*

Group File

Description Position *StreamNum* for input *Count*.

Note: Unicode text files opened with Input mode use character positions, not byte positions.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
Count	For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1.

See Also **Seek( )**.

Example

```
'#Language "WWB.NET"
Sub Main
   FileOpen 1, "XXX", OpenMode.Input
   L = LineInput(1)
   Seek'1, 1 ' rewind to start of file
   Input'1, A
   FileClose'1
   Debug.Print A
End Sub
```

Seek Function

Syntax Seek(*StreamNum*)

Group File

Description Return *StreamNum* current position. For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1.

Note: Unicode text files opened with Input mode use character positions, not byte positions.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

See Also **Seek**.

Example

```
'#Language "WWB.NET"
Sub Main
   FileOpen 1, "XXX", OpenMode.Input
   Debug.Print Seek(1) ' 1
   L = LineInput(1)
   Debug.Print Seek(1)
   FileClose'1
End Sub
```

Select Case Statement

Syntax

```
Select Case expr
[Case caseexpr[, ...] [ : instruction ]
   statements]...
[Case Else [ : instruction ]
   statements]
End Select
```

Group Flow Control

Description Select the appropriate case by comparing the *expr* with each of the *caseexpr*s. Select the Case Else part if no *caseexpr* matches. (If the Case Else is omitted then skip the entire Select...End Select block.) Only the *statement*s from the matched Case up to the next Case are executed.

caseexpr	Description
expr	Execute if equal.
Is < *expr*	Execute if less than.
Is <= *expr*	Execute if less than or equal to.
Is > *expr*	Execute if greater than.
Is >= *expr*	Execute if greater than or equal to.
Is <> *expr*	Execute if not equal to.
expr1 To *expr2*	Execute if greater than or equal to *expr1* and less than or equal to *expr2*.

See Also **If**, **Choose( )**, **IIf( )**.

Example

```
'#Language "WWB.NET"
Sub Main
   S = InputBox("Enter hello, goodbye, dinner or sleep:")
   Select Case UCase(S)
   Case "HELLO", "HI"
      Debug.Print "come in"
   Case "GOODBYE", "BYE"
      Debug.Print "see you later"
   Case "DINNER"
      Debug.Print "Please come in."
      Debug.Print "Dinner will be ready soon."
   Case "SLEEP"
      Debug.Print "Sorry."
      Debug.Print "We are full for the night"
   Case Else
      Debug.Print "What?"
   End Select
End Sub
```

SendKeys Instruction

Syntax SendKeys *Key*[, *Wait*]

Group Miscellaneous

Description Send *Key'* to Windows.

Parameter	Description
Key'	Send the keys in this string value to Windows. (Refer to table below.)
Wait	If this is not zero then the keys are sent before executing the next instruction. If this is omitted or zero then the keys are sent during the following instructions.

Key	Description
+	Shift modifier key: the following key is a shifted key
^	Ctrl modifier key: the following key is a control key
%	Alt modifier key: the following key is an alt key
(keys)	Modifiers apply to all keys
~	Send Enter key
k	Send k Key (k is any single char)
K	Send Shift k Key (K is any capital letter)
{special n}	special key (n is an optional repeat count)
{mouse x,y}	mouse key (x,y is an optional screen position)
{k}	Send k Key (any single char)
{K}	Send Shift k Key (any single char)
{Cancel}	Send Break Key
{Esc}	Send Escape Key
{Escape}	Send Escape Key
{Enter}	Send Enter Key
{Menu}	Send Menu Key (Alt)
{Help}	Send Help Key (?)
{Prtsc}	Send Print Screen Key
{Print}	Send
{Execute}	Send ?
{Tab}	Send
{Pause}	Send Pause Key
{Tab}	Send Tab Key
{BS}	Send Back Space Key
{BkSp}	Send Back Space Key
{BackSpace}	Send Back Space Key
{Del}	Send Delete Key
{Delete}	Send Delete Key
{Ins}	Send Insert Key
{Insert}	Send Insert Key
{Left}	Send Left Arrow Key
{Right}	Send Right Arrow Key
{Up}	Send Up Arrow Key
{Down}	Send Down Arrow Key
{PgUp}	Send Page Up Key
{PgDn}	Send Page Down Key
{Home}	Send Home Key
{End}	Send End Key
{Select}	Send ?
{Clear}	Send Num Pad 5 Key
{Pad0..9}	Send Num Pad 0-9 Keys
{Pad*}	Send Num Pad * Key

{Pad+}	Send Pad + Key
{PadEnter}	Send Num Pad Enter
{Pad.}	Send Num Pad . Key
{Pad-}	Send Num Pad - Key
{Pad/}	Send Num Pad / Key
{F1..24}	Send F1 to F24 Keys

Mouse

Mouse movement and button clicks:

- {Move x,y} - move the mouse to (x,y)
- {ClickLeft x,y} - move the mouse to (x,y) and click the left button. (This is the same as {DownLeft x,y}{UpLeft}.)
- {DoubleClickLeft x,y} - move the mouse to (x,y) and click the left button. (This is NOT the same as {ClickLeft x,y}{ClickLeft}.)
- {DownLeft x,y} - move the mouse to (x,y) and push the left button down.
- {UpLeft x,y} - move the mouse to (x,y) and release the left button.
- {...Middle x,y} - similarly named keys for the middle mouse button.
- {...Right x,y} - similarly named keys for the right mouse button.

The x,y values are screen pixel locations, where (0,0) is in the upper-left corner. In all cases the x,y is optional. If omitted, the previous mouse position is used.

See Also **AppActivate**, **KeyName**, **Shell( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  SendKeys "%S"    ' send Alt-S (Search)
  SendKeys "GoTo~~" ' send G o T o {Enter} {Enter}
End Sub
```

SetAttr Instruction

Syntax SetAttr *Nam'*, *Attrib*

Group File

Description Set the *attribute*s for file *Nam'*. If the file does not exist then a run-time error occurs.

Parameter	Description
Nam'	This string value is the path and name of the file. A path relative to the current directory can be used.
Attrib	Set the file's *attribute*s to this numeric value.

Example

```
'#Language "WWB.NET"
Sub Main
  Attrib = GetAttr("XXX")
  SetAttr "XXX",1 ' readonly
  Debug.Print GetAttr("XXX") ' 1
```

```
  SetAttr "XXX",Attrib
End Sub
```

SetLocale Instruction

Syntax SetLocale *LocaleID*

Group Miscellaneous

Description Set the *LocaleID* for the current thread.

Pocket PC Not supported.

Parameter	Description
LocaleID	Set the current thread's locale to this value.

See Also **GetLocale**.

Example

```
'#Language "WWB.NET"
Sub Main
  SetLocale &H409 ' English, US
End Sub
```

Sgn Function

Syntax Sgn(*Num*)

Group Math

Description Return the sign.

Parameter	Description
Num	Return the sign of this numeric value. Return -1 for negative. Return 0 for zero. Return 1 for positive.

See Also **Abs**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Sgn(9)  ' 1
  Debug.Print Sgn(0)  ' 0
  Debug.Print Sgn(-9) '-1
End Sub
```

Shell Function

Syntax Shell(*Nam*[, *WindowType*])

Group Miscellaneous

Description Execute program *Nam'*. This is the same as using File|Run from the Program Manager. This instruction can run .COM, .EXE, .BAT and .PIF files. If successful, return the task ID.

Pocket PC The *WindowType* parameter is ignored.

Parameter	Description
Nam'	This string value is the path and name of the program to run. Command line arguments follow the program name. (A long file name containing a space must be surrounded by literal double quotes.)
WindowType	This controls how the application's main window is shown. See the table below.

WindowType	Value	Effect
vbHide	0	Hide Window
vbNormalFocus	1, 5, 9	Normal Window
vbMinimizedFocus	2	Minimized Window (default)
vbMaximizedFocus	3	Maximized Window
vbNormalNoFocus	4, 8	Normal Deactivated Window
vbMinimizedNoFocus	6, 7	Minimized Deactivated Window

See Also **AppActivate**, **SendKeys**.

Example

```
'#Language "WWB.NET"
Sub Main
   X = Shell("Calc") ' run the calc program
   AppActivate X
   SendKeys "% R" ' restore calc's main window
   SendKeys "30*2{+}10=",1 '70
End Sub
```

SByte Data Type

Syntax **Dim** v As SByte

Group Data Type

Description An 8 bit signed integer value.

Short Data Type

Syntax **Dim** v As Short

Group Data Type

Description A 16 bit signed integer value.

ShowPopupMenu Function

Syntax ShowPopupMenu(*StrArra'( )*[, *PopupStyle*][, *XPos*, *YPos*])

Group User Input

Description Show a popup menu and return the number of the item selected. The item number is the index of the StrArray selected minus LBound(StrArray). The value -1 is returned in no menu item is selected.

Parameter	Description
StrArra'( )	This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.
PopupMenuStyle	This controls how the popup menu is aligned. Any combination of styles may used together. See the table below.
XPos	When the menu is put up the alignment will be at this window position. If this is omitted then the current mouse position is used.
YPos	When the menu is put up the alignment will be at this window position. If this is omitted then the current mouse position is used.

PopupStyle	Value	Effect
vbPopupLeftTopAlign	0	Align menu left edge at XPos and top at YPos. (default)
vbPopupUseLeftButton	1	User can select menu choices with the left mouse button only.
vbPopupUseRightButton	2	User can select menu choices with the left or right mouse button.
vbPopupRightAlign	4	Align menu with right edge at the XPos.
vbPopupCenterAlign	8	Align menu center at the XPos.
vbPopupVCenterAlign	16	Align menu center at the YPos.
vbPopupBottomAlign	32	Align menu bottom at the YPos.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim Items(0 To 2) As String
  Items(0) = "Item &1"
  Items(1) = "Item &2"
  Items(2) = "Item &3"
  X = ShowPopupMenu(Items) ' show popup menu
  Debug.Print X ' item selected
End Sub
```

Sin Function

Syntax Sin(*Num*)

Group Math

Description Return the sine.

Parameter	Description

Num	Return the sine of this numeric value. This is the number of radians. There are 2*Pi radians in a full circle.

See Also **Atn**, **Cos**, **Tan**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Sin(1) ' 0.8414709848079
End Sub
```

Single Data Type

Syntax **Dim** v As Single

Group Data Type

Description A 32 bit real value.

Spac' Function

Syntax Spac'(*Len*)

Group String

Description Return the string *Len* spaces long.

Parameter	Description
Len	Create a string this many spaces long.

See Also **StrDu'( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print "."; Spac'(3); "." "".  ."
End Sub
```

Split Function

Syntax Split(*Str*, [*Sep*], [*Max*])

Group Miscellaneous

Description Return a string array containing substrings from the original string.

Parameter	Description
Str	Extract substrings from this string value.
Sep	Look for this string value to separate the substrings. (Default: " ")
Max	Create at most this many substrings. (Default -1, which means create as many as are found.)

See Also **Join( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Split("1 2 3")(1) '"2"
End Sub
```

Sqr Function

Syntax Sqr(*Num*)

Group Math

Description Return the square root.

Parameter	Description
Num	Return the square root of this numeric value.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Sqr(9) ' 3
End Sub
```

Static Definition

Syntax Static *vardeclaration* [= *initialvalue*][, ...]

Group Declaration

Description A static variable retains it value between *procedure* calls. Dimension var array (s) using the *dimension*s to establish the minimum and maximum index value for each dimension. If the *dimension*s are omitted then a scalar (single value) variable is defined. A dynamic array is declared using ( ) without any *dimension*s. It must be **ReDim**ensioned before it can be used.

See Also **Dim**, **Option** Base, **Private**, **Public**, **ReDim**.

Example

```
'#Language "WWB.NET"
Sub A
  Static X
  Debug.Print X
  X = "Hello"
End Sub

Sub Main
  A
  A ' prints "Hello"
End Sub
```

Stop Instruction

Syntax Stop

Group Flow Control

Description Pause execution. If execution is resumed then it starts at the next instruction. Use **End** to terminate the *macro* completely.

Example

```
'#Language "WWB.NET"
Sub Main
  For I = 1 To 10
    Debug.Print I
    If I = 3 Then Stop
  Next I
End Sub
```

St' Function

Syntax St'(*Num*)

Group String

Description Return the string representation of *Num*.

Parameter	Description
Num	Return the string representation of this numeric value. Positive values begin with a blank. Negative values begin with a dash '-'.

See Also **CStr( )**, **He'( )**, **Oc'( )**, **Val( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print St'(9*9) ' 81
End Sub
```

StrCom' Function

Syntax StrComp(*Str1*,*Str2*,*Comp*)

Group String

Description Compare two strings.

Parameter	Description
Str1	Compare this string with *Str2*. '
Str2	Compare this string with *Str1*. '
Comp	This numeric value indicates the type of comparison. See Comp table below.

Result	Description
-1	*Str1* is less than *Str2*.
0	*Str1* is equal to *Str2*.
1	*Str1* is greater than *Str2*.

Comp	Value	Effect

vbUseCompareOption	-1	Performs the comparison using the **Option** Compare statement value.
vbBinaryCompare	0	Compares the string's binary data.
vbTextCompare	1	Compares the string's text using the collation rules.
vbDatabaseCompare	2	Microsoft Access only. (Not supported.)

See Also **LCas'( )**, **Option** Compare, **StrCon'( )**, **UCas'( )**.

Example

```
'#Language "WWB.NET"
Sub Main
   Debug.Print StrComp("F","e") ' -1
   Debug.Print StrComp("F","e",1) ' 1
   Debug.Print StrComp("F","f",1) ' 0
End Sub
```

StrCon' Function

Syntax StrCon'(*Str*,*Conv*)

Group String

Description Convert the string.

Parameter	Description
Str	Convert this string value. '
Conv	This numeric value indicates the type of conversion. See conversion table below.

Conv	Value	Effect
vbUpperCase	1	Convert a String to upper case.
vbLowerCase	2	Convert a String to lower case.
vbProperCase	3	Convert a String to proper case. (Not supported.)
vbWide	4	Convert a String to wide. (Only supported for eastern locales.)
vbNarrow	8	Convert a String to narrow. (Only supported for eastern locales.)
vbKatakana	16	Convert a String to Katakana. (Only supported for Japanese locales.)
vbHiragana	32	Convert a String to Hiragana. (Only supported for Japanese locales.)
vbUnicode or vbFromANSIBytes	64	Convert an ANSI (locale dependent) byte array to a Unicode string.
vbANSI	4160	Convert an ANSI (locale dependent) string to a Unicode string.
vbFromUnicode or vbANSIBytes	128	Convert from Unicode to an ANSI (locale dependent) byte array.
vbANSI	4224	Convert from Unicode to an ANSI (locale dependent) string.
vbUTF8	4352	Convert a Unicode string to a UTF-8 string.
vbUTF8Bytes	256	Convert a Unicode string to a UTF-8 byte array.
vbFromUTF8	4608	Convert a UTF-8 string to a Unicode string.
vbFromUTF8Bytes	512	Convert a UTF-8 byte array to a Unicode string.
vbToBytes	1024	Convert a String to a byte array containing the low byte of each char.
vbFromBytes	2048	Convert a byte array to a String by setting the low byte of each char.

Conversion Rules If multiple conversions are specified, the conversions occur in this order:

- vbFromBytes, vbUnicode, vbFromANSI, vbFromANSIBytes, vbFromUTF8 or vbFromUTF8Bytes (choose one, optional)
- vbUpperCase, vbLowerCase, vbWide, vbNarrow, vbKatakana or vbHiragana (choose one or more, optional)
- vbToBytes, vbFromUnicode, vbANSI, vbANSIBytes, vbUTF8 or vbUTF8Bytes (choose one, optional)

See Also **LCas'( )**, **StrComp( )**, **UCas'( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim B(1 To 3) As Byte
  B(1) = 65
  B(2) = 66
  B(3) = 67
  Debug.Print StrCon'(B,vbUnicode) '"ABC"
End Sub
```

StrDu' Function

Syntax StrDu'(*Len*, *Char*)

Group String

Description Return the string *Len* long filled with *Char* or the first char of *Cha'*.

Parameter	Description
Len	Create a string this many chars long.
Char	Fill the string with this char value. If this is a numeric value then use the ASCII char equivalent. If this is a string value use the first char of that string. '

See Also **Spac'( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print StrDu'(4,65)    '"AAAA"
  Debug.Print StrDu'(4,"ABC") '"AAAA"
End Sub
```

String Data Type

Syntax **Dim** v As String

Group Data Type

Description An arbitrary length string value. Some useful string constants are predefined:

- vbNullChar - same as Chr(0)
- vbNullString - not implemented
- vbCrLf - same as Chr(13) & Chr(10)

- vbCr - same as Chr(13)
- vbLf - same as Chr(10)
- vbBack - same as Chr(8)
- vbFormFeed - same as Chr(12)
- vbTab - same as Chr(9)
- vbVerticalTab - same as Chr(11)

StrRevers' Function

Syntax StrRevers'(*S*)

Group String

Description Return the string with the characters in reverse order.

Parameter	Description
S	Return this string with the characters in reverse order.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print StrRevers'("ABC") 'CBA
End Sub
```

Structure Definition

Syntax

```
[ | Private | Public ] _
Structure name
  [ Dim | Private | Public ] elem [([dimension[, ...]])] As type
  [...]
End Structure
```

Group Declaration

Description Define a new *user structure*. Each *elem* defines an element of the type for storing data. *As type* defines the type of data that can be stored. A *user defined structure variable* has a value for each *elem*. Use .*elem* to access individual element values.

Access If no access is specified then **Public** is assumed.

Example

```
'#Language "WWB.NET"
Structure Employee
  Dim FirstName As String
  Dim LastName As String
  Dim Title As String
  Dim Salary As Double
End Structure

Sub Main
```

```
  Dim e As Employee
  e.FirstName = "John"
  e.LastName = "Doe"
  e.Title = "President"
  e.Salary = 100000
  Debug.Print e.FirstName "'John"
  Debug.Print e.LastName  "'Doe"
  Debug.Print e.Title     "'President"
  Debug.Print e.Salary    ' 100000
End Sub
```

Sub Definition

Syntax

```
[ | Private | Public | Friend ] _
Sub name[([param[, ...]])] _
  [Handles var.eventname]
  statements
End Sub
```

Group Declaration

Description User defined subroutine. The subroutine defines a set of *statement*s to be executed when it is called. The values of the calling *arglist* are assigned to the *param*s. A subroutine does not return a result.

Access If no access is specified then **Public** is assumed.

Handles The Sub provides an event handler for the **WithEvents** variable's (*var*) event (*eventname*).

See Also **Declare**, **Function**, **Property**, **WithEvents**.

Example

```
'#Language "WWB.NET"
Sub IdentityArray(A()) ' A() is an array of numbers
  For I = LBound(A) To UBound(A)
    A(I) = I
  Next I
End Sub

Sub CalcArray(A(), B, C) ' A() is an array of numbers
  For I = LBound(A) To UBound(A)
    A(I) = A(I)*B+C
  Next I
End Sub

Sub ShowArray(A()) ' A() is an array of numbers
  For I = LBound(A) To UBound(A)
    Debug.Print "("; I; ")="; A(I)
  Next I
End Sub

Sub Main
  Dim X(1 To 4)
```

```
  IdentityArray X()  ' X(1)=1, X(2)=2, X(3)=3, X(4)=4
  CalcArray X(), 2, 3 ' X(1)=5, X(2)=7, X(3)=9, X(4)=11
  ShowArray X()      ' print X(1), X(2), X(3), X(4)
End Sub
```

SystemTypeName Function

Syntax SystemTypeNam'(*Name*)

Group Variable Info

Description Return a fully qualified Common Language Runtime type name corresponding to the VB type name. Return **Nothing** if the specified *Name* is not a valid VB type name.

Parameter	Description
Name	This is the VB type name.

See Also **TypeName**, **VbTypeName**, **VarType**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim X As Object
  Debug.Print SystemTypeName(X) '"Empty"
  X = 1
  Debug.Print SystemTypeName(X) '"Integer"
End Sub
```

Tan Function

Syntax Tan(*Num*)

Group Math

Description Return the tangent.

Parameter	Description
Num	Return the tangent of this numeric value.

See Also **Atn**, **Cos**, **Sin**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Tan(1) ' 1.5574077246549
End Sub
```

Text Dialog Item Definition

Syntax Text *X*, *Y*, *DX*, *DY*, *Titl*[, *.Field*][, *Options*]

Group User Dialog

Description Define a text item.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Titl'	The value of this string is the title of the text control.
Field	This identifier is the name of the field. The *dialogfunc* receives this name as *string*. If this identifer is omitted then the first two words of the title are used.
Options	This numeric value controls the alignment of the text. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option	Description
0	Text is left aligned.
1	Text is right aligned.
2	Text is centered.

See Also **Begin Dialog**.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  Dialog dlg ' show dialog (wait for ok)
End Sub
```

TextBox Dialog Item Definition

Syntax TextBox *X*, *Y*, *DX*, *DY*, *.Fiel*[, *Options*]

Group User Dialog

Description Define a textbox item.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Field	The value of the text box is accessed via this field.

Options This numeric value controls the type of text box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option	Description
0	Text box allows a single line of text to be entered.
1	Text box allows multiple lines of text can be entered.
2	Locked text box displays multiple lines of text.
-1	Text box allows a hidden password can be entered.

See Also **Begin Dialog**.

Example

```
'#Language "WWB.NET"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,25,180,20,.Tex'
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  dlg.Tex' = "none"
  Dialog dlg ' show dialog (wait for ok)
  Debug.Print dlg.Tex'
End Sub
```

Timer Function

Syntax Timer

Group Time/Date

Description Return the number of seconds past midnight. (This is a real number, accurate to about 1/18th of a second.)

See Also **Date**, **Now**, **Time**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Timer ' example: 45188.13
End Sub
```

TimeSerial Function

Syntax TimeSerial(*Hour*, *Minute*, *Second*)

Group Time/Date

Description Return a **Date** value.

Parameter	Description
Hour	This numeric value is the hour (0 to 23).
Minute	This numeric value is the minute (0 to 59).
Second	This numeric value is the second (0 to 59).

See Also **DateSerial**, **DateValue**, **TimeValue**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print TimeSerial(13,30,0) '1:30:00 PM
End Sub
```

TimeValue Function

Syntax TimeValue(*Dat'*)

Group Time/Date

Description Return the time part of date encoded as a string value.

Parameter	Description
Dat'	Convert this string value to the time part of date it represents.

See Also **DateSerial**, **DateValue**, **TimeSerial**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print TimeValue("1/1/2000 12:00:01 AM")
    '12:00:01 AM
End Sub
```

Throw Instruction/Function

Syntax Throw [*exception*]

Group Throw Handling

Description Throw an exception the caller. Execution of the current procedure is terminated immediately.

Parameter	Description
exception	Throw this exception. The exception expression may be omitted in a Catch block.

See Also **Try**.

Example

```
'#Language "WWB.NET"
Sub Main
  Try
    DoIt
  Catch Ex As Exception
    Debug.Print Ex.ToString() '"error"
  End Try
End Sub

Sub DoIt
```

```
  Throw New Exception("error")
End Sub
```

Tri' Function

Syntax Tri'(')

Group String

Description Return the string with 's leading and trailing spaces removed.

Parameter	Description
'	Copy this string without the leading or trailing spaces. '

See Also **LTri'( )**, **RTri'( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print "."; Tri'("  x  "); "." '".x."
End Sub
```

True Keyword

Group Constant

Description A *conditional expression* is True when its value is non-zero. A function that returns True returns the value -1.

Try Statement

Syntax

```
Try
  statements
[ Catch [exception [As type]] [When condition]
  statements]...
[ Finally
  statements]
End Try
```

Group Error Handling

Description A Try block allows exceptions to be handled programmatically. When an exception occurs in the Try block, each Catch block is check for a matching *condition*. (If the condition expression is omitted, the Catch block is matched.) If a matching Catch block is found, the block's statements are executed. The optional Finally block is always executed regardless of whether an exception occurs or not.

See Also **Throw**, **Using**.

Example

```
'#Language "WWB.NET"
Sub Main
  Try
    DoIt
  Catch Ex As System.Exception
    Debug.Print Ex.ToString() '"error"
  End Try
End Sub

Sub DoIt
  Throw New System.Exception("error")
End Sub
```

TryCast Function

Syntax TryCast(*expr*, *objtype*)

Group Conversion

Description Return *expr*'s type is related to *objtype* type. If it is not, Nothing will be returned.

Parameter	Description
expr	Cast the value of this expression.
objtype	Cast to this type.

See Also **CType**, **DirectCast**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim V As Object
  V = Err
  Debug.Print TypeName(TryCast(V, ErrObject)) ' ErrObject
End Sub
```

TypeName Function

Syntax TypeNam'(*var*)

Group Variable Info

Description Return a string indicating the type of value stored in *var*.

Parameter	Description
var	Return a string indicating the type of value stored in this variable.

Result	Description
Empty	*Object* variable is empty. It has never been assigned a value.
Null	*Object* variable is null.
Boolean	Variable contains a **Boolean** value.
Byte	Variable contains a **Byte** value.
SByte	Variable contains a **SByte** value.

Short	Variable contains an **Short** value.
UShort	Variable contains an **UShort** value.
Integer	Variable contains an **Integer** value.
UInteger	Variable contains an **UInteger** value.
Long	Variable contains a **Long** value.
ULong	Variable contains a **ULong** value.
Decimal	Variable contains a **Decimal** value.
Single	Variable contains a **Single** value.
Double	Variable contains a **Double** value.
Date	Variable contains a **Date** value.
String	Variable contains a **String** value.
Object	Variable contains an **Object** reference that is not Nothing. (An object may return a type name specific to that type of object.)
Nothing	Variable contains an **Object** reference that is Nothing.
Error	Variable contains a error code value.
Unknown	Variable contains a non-ActiveX Automation object reference.
( )	Variable contains an array value. The TypeName of the element followed by ( ).

See Also **SystemTypeName**, **VarType**, **VbTypeName**.

Example

```
'#Language "WWB.NET"
Sub Main
   Dim X As Object
   Debug.Print TypeName(X) '"Empty"
   X = 1
   Debug.Print TypeName(X) '"Integer"
   X = 100000
   Debug.Print TypeName(X) '"Long"
   X = 1.1
   Debug.Print TypeName(X) '"Double"
   X = "A"
   Debug.Print TypeName(X) '"String"
  'X = CreateObject("Word.Basic")
   Debug.Print TypeName(X) '"Object"
   X = Array(0,1,2)
   Debug.Print TypeName(X) '"Object()"
End Sub
```

TypeOf Operator

Syntax TypeOf *expr* **Is** *objtype*

Group Operator

Description Return the **True** if *expr* refers to an object of *objtype*.

See Also **Objects**.

Example

```
'#Language "WWB.NET"
Sub Main
```

```
  Debug.Print TypeOf Err Is ErrObject ' True
End Sub
```

UBound Function

Syntax UBound(*arrayvar*[, *dimension*])

Group Variable Info

Description Return the highest index.

Parameter	Description
arrayvar	Return the highest index for this array variable.
dimension	Return the highest index for this dimension of *arrayvar*. If this is omitted then return the highest index for the first dimension.

See Also **LBound( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim A(3,6)
  Debug.Print UBound(A)   ' 3
  Debug.Print UBound(A,1) ' 3
  Debug.Print UBound(A,2) ' 6
End Sub
```

UCas' Function

Syntax UCas'( )

Group String

Description Return a string from ’where all the lowercase letters have been uppercased.

Parameter	Description
'	Return the string value of this after all chars have been converted to lowercase. '

See Also **LCas'( )**, **StrComp( )**, **StrCon'( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print UCas'("Hello") '"HELLO"
End Sub
```

UInteger Data Type

Syntax **Dim** v As UInteger

Group Data Type

Description A 32 bit unsigned integer value. The number of bits is controlled by the **#Language** setting.

ULong Data Type

Syntax **Dim** v As ULong

Group Data Type

Description A 64 bit unsigned integer value. The number of bits is controlled by the **#Language** setting.

Unlock Instruction

Syntax Unlock *StreamNum*
-or-
Unlock *StreamNum*, *RecordNum*
-or-
Unlock *StreamNum*, [*start*] To *end*

Group File

Description Form 1: Unlock all of *StreamNum*.

Form 2: Unlock a record (or byte) of *StreamNum*.

Form 3: Unlock a range of records (or bytes) of *StreamNum*. If *start* is omitted then unlock starting at the first record (or byte).

Note: For sequential files (Input, Output and Append) unlock always affects the entire file.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
RecordNum	For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1.
start	First record (or byte) in the range.
end	Last record (or byte) in the range.

See Also **Lock**, **Open**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim V As Variant
  FileOpen 1, "SAVE_V.DAT", OpenMode.Binary
  Lock'1
  Get'1, 1, V
  V = "Hello"
```

```
  Put'1, 1, V
  Unlock'1
  FileClose'1
End Sub
```

UShort Data Type

Syntax Dim v As UShort

Group Data Type

Description A 16 bit unsigned integer value.

Val Function

Syntax Val(')

Group String

Description Return the value of the '.

Parameter	Description
'	Return the numeric value for this string value. A string value begins with &O is an octal number. A string value begins with &H is a hex number. Otherwise it is decimal number.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Val("-1000") '-1000
End Sub
```

Using Statement

Syntax

```
Using expr
  statements
End Using
-or-
Using vardeclaration[, ...]
  statements
End Using
```

Group Error Handling

Description A Using block allows resources to be freed automatically. A resource implements the IDisposable interface. When the End Using statement is executed all of the resources are freed.

See Also **Try**.

Example

```
'#Language "WWB.NET"
Sub Main
  Using res As New Resource
    ' ... use res
  End Using ' res.Dispose is called
End Sub
```

VarType Function

Syntax VarType(*var*)

Group Variable Info

Description Return a number indicating the type of value stored in *var*.

Parameter	Description
var	Return a number indicating the type of value stored in this variable.

Result	Value	Description
vbEmpty	0	*Object* variable is Nothing. It has never been assigned a value.
vbShort	2	Variable contains an **Short** value.
vbInteger	3	Variable contains an **Integer** value. The value depends on the **#Language** setting.
vbLong	20	Variable contains a **Long** value. The value depends on the **#Language** setting.
vbSingle	4	Variable contains a **Single** value.
vbDouble	5	Variable contains a **Double** value.
vbCurrency	6	Variable contains a Currency value.
vbDate	7	Variable contains a **Date** value.
vbString	8	Variable contains a **String** value.
vbObject	9	Variable contains an **Object** reference.
vbError	10	Variable contains a error code value.
vbBoolean	11	Variable contains a **Boolean** value.
vbDataObject	13	Variable contains a non-ActiveX Automation object reference.
vbDecimal	14	Variable contains a **Decimal** value.
vbSByte	16	Variable contains a **SByte** value.
vbByte	17	Variable contains a **Byte** value.
vbUShort	18	Variable contains a **UShort** value.
vbUInteger	19	Variable contains a **UInteger** value. The value depends on the **#Language** setting.
vbULong	21	Variable contains a **ULong** value. The value depends on the **#Language** setting.
vbUserDefinedType	36	Variable contains a User Defined **Structure** value.
+vbArray	8192	Variable contains an array value. Use VarType( ) And 255 to get the type of element stored in the array.

See Also **SystemTypeName**, **TypeName**, **VbTypeName**.

Example

```
'#Language "WWB.NET"
Sub Main
```

```
  Dim X As Object
  Debug.Print VarType(X) ' 0
  X = 1
  Debug.Print VarType(X) ' 2
  X = 100000
  Debug.Print VarType(X) ' 3
  X = 1.1
  Debug.Print VarType(X) ' 5
  X = "A"
  Debug.Print VarType(X) ' 8
 'X = CreateObject("Word.Basic")
  Debug.Print VarType(X) ' 9
  X = Array(0,1,2)
  Debug.Print VarType(X) ' 8204 (8192+12)
End Sub
```

VbTypeName Function

Syntax VbTypeNam'(*Name*)

Group Variable Info

Description Return a string indicating VB type name corresponding to the fully qualified Common Language Runtime type name. Return **Nothing** if the specified *Name* is not a valid VB type name.

Parameter	Description
Name	This is Common Language Runtime type name.

See Also **SystemTypeName**, **TypeName**, **VarType**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim X As Object
  Debug.Print VbTypeName(X) '"Empty"
  X = 1
  Debug.Print VbTypeName(X) '"Integer"
End Sub
```

Wait Instruction

Syntax Wait [*Delay*]

Group Miscellaneous

Description Wait for *Delay* seconds.

Parameter	Description
Delay	Wait for this number of seconds. If omitted then wait for 5 seconds. '

Example

```
'#Language "WWB.NET"
Sub Main
  Wait .5 ' wait for one half second
End Sub
```

Weekday Function

Syntax Weekday(*dateexpr*)

Group Time/Date

Description Return the weekday.

- vbSunday (1) - Sunday
- vbMonday (2) - Monday
- vbTuesday (3) - Tuesday
- vbWednesday (4) - Wednesday
- vbThursday (5) - Thursday
- vbFriday (6) - Friday
- vbSaturday (7) - Saturday

Parameter	Description
dateexpr	Return the weekday for this date value. '

See Also **Date( )**, **Day( )**, **Month( )**, **WeekdayName( )**, **Year( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Weekday(#1/1/1900#) ' 2
  Debug.Print Weekday(#1/1/2000#) ' 7
End Sub
```

WeekdayName Function

Syntax WeekdayName(NumZ{day}[, CondZ{abbrev}])

Group Time/Date

Description Return the localized name of the weekday.

Parameter	Description
day	Return the localized name of this weekday. (1-7)
abbrev	If this conditional value is **True** then return the abbreviated form of the weekday name.

See Also **Weekday( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print WeekdayName(1) 'Sunday
```

```
  Debug.Print WeekdayName(Weekday(Now))
End Sub
```

While Statement

Syntax

```
While condexpr
  statements
End While
```

Group Flow Control

Description Execute *statements* while *condexpr* is **True**.

See Also **Do**, **For**, **For Each**, **Exit** While.

Example

```
'#Language "WWB.NET"
Sub Main
  I = 2
  While I < 10
    I = I*2
  End While
  Debug.Print I ' 16
End Sub
```

Win16 Keyword

Group Constant

Description **True** if running in 16 bits. **False** if running in 32 or 64 bits.

Win32 Keyword

Group Constant

Description **True** if running in 32 bits. **False** if running in 16 or 64 bits.

Win64 Keyword

Group Constant

Description **True** if running in 64 bits. **False** if running in 16 or 32 bits.

With Statement

Syntax With *objexpr*
*statement*s
End With

Group Object

Description *Method* and *property* references may be abbreviated inside a With block. Use *.method* or *.property* to access the object specified by the With *objexpr*.

Example
```
'#Language "WWB.NET"
Sub Main
  Dim App As Object
  'App = CreateObject("WinWrap.CppDemoApplication")
  With App
    .Move 20,30 ' move icon to 20,30
  End With
End Sub
```

WithEvents Definition

Syntax [**Dim** | **Private** | **Public**] _
WithEvents *name* As *objtype*[, ...]

Group Declaration

Description Dimensioning a module level variable WithEvents allows the macro to implement event handling **Sub**s. The variable's As type must be a type from a referenced type library (or language extension) which implements events.

See Also **Dim**, **Private**, **Public**, **RaiseEvent**.

Example
```
'#Language "WWB.NET"
Dim WithEvents X As Thing

Sub Main
  'X = New Thing
  X.DoIt ' DoIt method raises DoingIt event
End Sub

Private Sub X_DoingIt Handles X.DoingIt
  Debug.Print "X.DoingIt event"
End Sub
```

Write Instruction

Syntax Write *StreamNum*, *expr*[, ...]

Group File

Description Write's *expr*(s) to *StreamNum*. String values are quoted. Null values are written as #NULL#. Boolean values are written as #FALSE# or #TRUE#. Date values are written as #date#. Error codes are written as #ERROR number#.

See Also **Input**, **LineInput**, **Print**, **PrintLine**, **WriteLine**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim A, B, C
  A = 1
  B = 2
  ' = "Hello"
  FileOpen 1, "XXX", OpenMode.Output
  Write'1, A, B, '
  FileClose'1
End Sub
```

WriteLine Instruction

Syntax WriteLine *StreamNum*, *expr*[, ...]

Group File

Description Write's *expr*(s) to *StreamNum*. String values are quoted. A newline is printed at the end. Null values are written as #NULL#. Boolean values are written as #FALSE# or #TRUE#. Date values are written as #date#. Error codes are written as #ERROR number#.

See Also **Input**, **LineInput**, **Print**, **PrintLine**, **Write**.

Example

```
'#Language "WWB.NET"
Sub Main
  Dim A, B, C
  A = 1
  B = 2
  ' = "Hello"
  FileOpen 1, "XXX", OpenMode.Output
  Write 1, A, B, C
  FileClose 1
End Sub
```

Year Function

Syntax Year(*dateexpr*)

Group Time/Date

Description Return the year.

Parameter	Description
dateexpr	Return the year for this date value. '

See Also **Date( )**, **Day( )**, **Month( )**, **Weekday( )**.

Example

```
'#Language "WWB.NET"
Sub Main
  Debug.Print Year(#1/1/1900#) ' 1900
  Debug.Print Year(#1/1/2000#) ' 2000
End Sub
```

Objects Overview

Each object supports a particular set of *methods* and *properties*. Each method/property has zero or more parameters. Parameters may be optional, in which case the parameter can be specified by using name := value.

- *objexpr*.*method* [*expr*][, ...] [*param* := *expr*][,...]
  Call *method* for *objexpr*.
- *objexpr*.*method*[([*expr*][, ...] [*param* := *expr*][,...])]
  Return the value of *method* for *objexpr*.
- *objexpr*.*property*[([*expr*][, ...] [*param* := *expr*][,...])]
  Return the value of *property* for *objexpr*.
- *objexpr*.*property*[([*expr*][, ...])] = *expr*
  Assign the value of *property* for *objexpr*.
- Set *objexpr*.*property*[([*expr*][, ...])] = *objexpr*
  Set the object reference of *property* for *objexpr*.

Note: *objexpr*!*name* is short hand for *objexpr*.*defaultproperty*("*name*"). Use *objexpr*![name] if name contains any characters that are not allowed in an identifier.

Error List

The following table lists all error codes with the associated error text.

Error	Description
10000	Execution interrupted.
10001	Out of memory.
10008	Invalid '#Uses "module" comment.
10009	Invalid '#Uses module dependency.
10010	Macro is already running.
10011	Can't allocate memory to macro/module.
10012	Macro/module has syntax errors.
10013	Macro/module does not exist.
10014	Another macro is paused and can't continue at this time.
10017	No macro is currently active.
10018	Sub/Function does not exist.
10019	Wrong number of parameters.
10021	Can't allocate large array.
10022	Array is not dimensioned.
10023	Array index out of range.

10024	Array lower bound is larger than upper bound.
10025	Array has a different number of indexes.
10030	User dialog has not been defined.
10031	User pressed cancel.
10032	User dialog item id is out of range.
10033	No UserDialog is currently displayed.
10034	Current UserDialog is inaccessible.
10035	Wrong with, don't GoTo into or out of With blocks.
10040	Module could not be loaded.
10041	Function not found in module.
10048	File not opened with read access.
10049	File not opened with write access.
10050	Record length exceeded.
10051	Could not open file.
10052	File is not open.
10053	Attempt to read past end-of-file.
10054	Expecting a stream number in the range 1 to 511.
10055	Input does not match var type.
10056	Expecting a length in the range 1 to 32767.
10057	Stream number is already open.
10058	File opened in the wrong mode for this operation.
10059	Error occurred during file operation.
10060	Expression has an invalid floating point operation.
10061	Divide by zero.
10062	Overflow.
10063	Expression underflowed minimum representation.
10064	Expression loss of precision in representation.
10069	String value is not a valid number.
10071	Resume can only be used in an On Error handler.
10075	Null value can't be used here.
10080	Type mismatch.
10081	Type mismatch for parameter #1.
10082	Type mismatch for parameter #2.
10083	Type mismatch for parameter #3.
10084	Type mismatch for parameter #4.
10085	Type mismatch for parameter #5.
10086	Type mismatch for parameter #6.
10087	Type mismatch for parameter #7.
10088	Type mismatch for parameter #8.
10089	Type mismatch for parameter #9.
10090	OLE Automation error.
10091	OLE Automation: no such property or method.
10092	OLE Automation: server cannot create object.
10093	OLE Automation: server cannot load file.
10094	OLE Automation: Object var is 'Nothing'.
10095	OLE Automation: server could not be found.
10096	OLE Automation: no object currently active.
10097	OLE Automation: wrong number of parameters.
10098	OLE Automation: bad index.

10099	OLE Automation: no such named parameter.
10100	Directory could not be found.
10101	File could not be killed.
10102	Directory could not be created.
10103	File could not be renamed.
10104	Directory could not be removed.
10105	Drive not found.
10106	Source file could not be opened.
10107	Destination file could not be created.
10108	Source file could not be completely read.
10109	Destination file could not be completely written.
10110	Missing close brace '}'.
10111	Invalid key name.
10112	Missing close paren ')'.
10113	Missing close bracket ']'.
10114	Missing comma ','.
10115	Missing semi-colon ';'.
10116	SendKeys couldn't install the Windows journal playback hook.
10119	String too long (too many keys).
10120	Window could not be found.
10130	DDE is not available.
10131	Too many simultaneous DDE conversations.
10132	Invalid channel number.
10133	DDE operation did not complete in time.
10134	DDE server died.
10135	DDE operation failed.
10140	Can't access the clipboard.
10150	Window style must be in the range from 1 to 9.
10151	Shell failed.
10160	Declare is not implemented.
10200	Basic is halted due to an unrecoverable error condition.
10201	Basic is busy and can't provide the requested service.
10202	Basic call failed.
10203	Handler property: prototype specification is invalid.
10204	Handler is already in use.

WWB-COM: Overview

WWB-COM Use **'#Language "WWB-COM"** for Visual Basic for Applications(TM) compatibility.

Declaration **'#Language**, **'#Reference**, **'#Uses**, **Attribute**, **Class Module**, **Code Module**, **Const**, **Declare**, **Def**type, **Delegate**, **Dim**, **Enum...End Enum**, **Event**, **Function...End Function**, **Object Module**, **Option**, **Private**, **Property...End Property**, **Public**, **ReDim**, **Static**, **Sub...End Sub**, **Type...End Type**, **WithEvents**.

Data Type **Any**, **Boolean**, **Byte**, **Currency**, **Date**, **Decimal**, **Double**, **Huge_**, **Integer**, **Long**, **Object**, **PortInt**, **SByte**, **Single**, **String**, **String*n**, **UHuge_**, **UInteger**, **ULong**, **Variant**, *obj type*, *user dialog*, *user enum*, *user type*.

Assignment **Assign**: (=, +=, -=, *=, /=, \=, ^=, <<=, >>=), **Erase**, **Let**, **LSet**, **RSet**, **Set**.

Flow Control **Call**, **CallByName**, **Do...Loop**, **End**, **Exit**, **For...Next**, **For Each...Next**, **GoTo**, **If...ElseIf...Else...End If**, **MacroCheck**, **MacroCheckThis**, **MacroRun**, **MacroRunThis**, **ModuleLoad**, **ModuleLoadThis**, **RaiseEvent**, **Return**, **Select Case...End Select**, **Stop**, **While...Wend**.

Error Handling **Err**, **Error**, **On Error**, **Resume**.

Conversion **Array**, **CBool**, **CByte**, **CCur**, **CDate**, **CDec**, **CDbl**, **CHuge_**, **CInt**, **CLng**, **CSByte**, **CSng**, **CStr**, **CUHuge_**, **CUInt**, **CULng**, **CVar**, **CVDate**, **CVErr**, **Val**.

Variable Info **IsArray**, **IsDate**, **IsEmpty**, **IsError**, **IsMissing**, **IsNull**, **IsNumeric**, **IsObject**, **LBound**, **TypeName**, **UBound**, **VarType**,

Constant **Empty**, **False**, **Nothing**, **Null**, **True**, **Win16**, **Win32**, **Win64**.

Math **Abs**, **Atn**, **Cos**, **Exp**, **Fix**, **Int**, **Log**, **Randomize**, **Rnd**, **Round**, **Sgn**, **Sin**, **Sqr**, **Tan**.

String **Asc**, **AscB**, **AscW**, **Chr**, **ChrB**, **ChrW**, **Format**, **Hex**, **InStr**, **InStrB**, **InStrRev**, **Join**, **LCase**, **Left**, **LeftB**, **Len**, **LenB**, **LTrim**, **Mid**, **MidB**, **Oct**, **Replace**, **Right**, **RightB**, **RTrim**, **Space**, **Split**, **Str**, **StrComp**, **StrConv**, **StrReverse**, **String**, **Trim**, **UCase**.

Object **CreateObject**, **GetObject**, **Me**, **With...End With**.

Time/Date **Date**, **DateAdd**, **DateDiff**, **DatePart**, **DateSerial**, **DateValue**, **Day**, **Hour**, **Minute**, **Month**, **MonthName**, **Now**, **Second**, **Time**, **Timer**, **TimeSerial**, **TimeValue**, **Weekday**, **WeekdayName**, **Year**.

File **ChDir**, **ChDrive**, **Close**, **CurDir**, **Dir**, **EOF**, **FileAttr**, **FileCopy**, **FileDateTime**, **FileLen**, **FreeFile**, **Get**, **GetAttr**, **Input**, **Input**, **Kill**, **Line Input**, **Loc**, **Lock**, **LOF**, **MkDir**, **Name**, **Open**, **Print**, **Put**, **Reset**, **RmDir**, **Seek**, **Seek**, **SetAttr**, **Unlock**, **Write**,

User Input **Dialog**, **GetFilePath**, **InputBox**, **MsgBox**. **ShowPopupMenu**

User Dialog **Begin Dialog...End Dialog**, **CancelButton**, **CheckBox**, **ComboBox**, **DropListBox**, **GroupBox**, **ListBox**, **MultiListBox**, **OKButton**, **OptionButton**, **OptionGroup**, **Picture**, **PushButton**, **Text**, **TextBox**.

Dialog Function DialogFunc, **DlgControlId**, **DlgCount**, **DlgEnable**, **DlgEnd**, **DlgFocus**, **DlgListBoxArray**, **DlgName**, **DlgNumber**, **DlgSetPicture**, **DlgText**, **DlgType**, **DlgValue**, **DlgVisible**.

DDE **DDEExecute**, **DDEInitiate**, **DDEPoke**, **DDERequest**, **DDETerminate**, **DDETerminateAll**.

Settings **DeleteSetting**, **GetAllSettings**, **GetSetting**, **SaveSetting**

Miscellaneous AboutWinWrapBasic, **AppActivate**, **Assign**, **Attribute**, **Beep**, **CallersLine**, **Choose**, **Clipboard**, **Command**, **Decode64**, **Decode64B**, **Decrypt64**, **Decrypt64B**, **Debug**.Print, **DoEvents**, **Encode64**, **Encode64B**, **Encrypt64**, **Encrypt64B**, **Environ**, **Eval**, **IIf**, **GetLocale**, **KeyName**, **MacroDir**, **QBColor**, **Rem**, **RGB**, **SendKeys**, **SetLocale**, **Shell**, **Wait**.

Operator **Operators**: +, -, ^, *, /, \, Mod, +, -, <<, >>, &, =, <>, <, >, <=, >=, **Like**, **New**, **TypeOf**, Not, And, AndAlso, Or, OrElse, Xor, Eqv, Imp, **Is**, **IsNot**, **AddressOf**.

More:

#Language Special Comment

#Reference Special Comment

#Uses Special Comment

AboutWinWrapBasic Instruction

Abs Function

AddressOf Operator

Any Data Type

AppActivate Instruction

Array Function

Asc Function

Assign Instruction

Assign Operators

Atn Function

Attribute Definintion/Statement

Beep Instruction

Begin Dialog Definition

Boolean Data Type

Byte Data Type

Terms

arglist [| *expr* | *param*:=*expr*][, ...]

A list of zero or more *expr*s that are assigned to the parameters of the *procedure*.

• A positional parameter may be skipped by omitting the expression. Only optional parameters may be skipped.

• Positional parameter assignment is done with *expr*. Each parameter is assigned in turn. By name parameter assignment may follow.

• By name parameter assignment is done with *param*:=*expr*. All following parameters must be assigned by name.

arrayvar A variable that holds an array of values. A *Variant* variable can hold an array. Dynamic arrays can be **ReDim**ensioned.

As [New] type As type
-or-
As New *objtype*

Dim, **Private**, **Public** and **Static** statements may declare variable types using *As type* or As New *objtype*. A variable declared using As New *objtype* is automatically created prior to use, if the variable is **Nothing**.

As type Variable and parameter types, as well as, function and property results may be specified using As type: **Boolean**, **Byte**, **Currency**, **Date**, **Decimal**, **Double**, **Huge_**, **Integer**, **Long**, **Object**, **PortInt**, **SByte**, **Single**, **String**, **String*n**, **UHuge_**, **Variant**, *objtype*, *user delegate*, *user dialog*, *user enum*, *user type*.

attribute A file attribute is zero or more of the following values added together.

Attribute Value Description

vbNormal 0 Normal file.

vbReadOnly 1 Read-only file.

vbHidden 2 Hidden file.

vbSystem 4 System file.

vbVolume 8 Volume label.

vbDirectory 16 MS-DOS directory.

vbArchive 32 File has changes since last backup.

vbAlias 64 File is an alias.

big-endian Multiple byte data values (not strings) are stored with the highest order byte first. For example, the long integer &H01020304 is stored as this sequence of four bytes: &H01, &H02, &H03 and &H04. A Binary or Random file written using **Put** uses *little-endian* format so that it can be read using **Get** on any machine. (Big-endian machines, like the Power-PC, reverse the bytes as they are read by **Get** or written by **Put**.)

bytearray A variable that holds an array of byte values.

caseexpr An expression which specifies a single value or a range. Refer to the **Select Case** statement.

charlist A group of one or more characters enclosed by [ ] as part of **Like** operator's right string expression.

- This list contains single characters and/or character ranges which describe the characters in the list.
- A range of characters is indicated with a hyphen (-) between two characters. The first character must be ordinally less than or equal to the second character.
- Special pattern characters like ?, *, # and [can be matched as literal characters.
- The] character can not be part of charlist, but it can be part of the pattern outside the charlist.

condexpr An expression that returns a numeric result. If the result is zero then the conditional is **False**. If the result is non-zero then the conditional is **True**.

0 'false
-1 'true

X > 20 'true if X is greater than 20
S$ = "hello" 'true if S$ equals "hello"

dateexpr An expression that returns a **date** result. Use #literal-date# to express a date value.

#1/1/2000# ' Jan 1, 2000
Now+7 ' seven days from now
DateSerial(**Year**(**Now**)+1,**Month**(**Now**),**Day**(**Now**))
' one year from now

dialogfunc A dialog function executes while a *user dialog* is visible.

dimension [*lower* To] *upper*

Array dimension. If *lower* is omitted then the lower bound is zero or one depending on the **Option** Base setting. (The lower bound of an array element in a **Type** definition is not affected by the Option Base setting.) *upper* must be at least as big as *lower*.

Dim A(100 To 200) '101 values

Note: For **ReDim** the *lower* and *upper* may be any valid *expression*. Otherwise, *lower* and *upper* must be constant expressions.

dlgvar A dialog variable holds values for fields in the dialog. Dialog variables are declared using **Dim** dlgvar As *user dialog*.

expr An simple or complex expression that returns the appropriate result.

- Simple: *var*, *cond*, *date*, *num*, *str*, *obj*, *field*, *method*, **function** (result) or **property** (result).
- Complex: One or more simple expressions with parenthesises and **Operators**.

field Use .field to access individual fields in a dialog variable.

dlg.LastName$
dlg.ZipCode

initialvalue Initial value for a variable. Use { *expr*, ... } to create an array value.

instruction A single command.

Beep
Debug.**Print** "Hello"
Today = **Date**

Multiple instructions may be used instead of a single instruction by separating the single instructions with colons.

X = 1:**Debug**.**Print** X
If X = 1 Then **Debug**.**Print** "X=";X:**Stop**
Beep ' must resume from **Stop** to get to here

label An identifier that *name*s a statement. Identifiers start with a letter. Following chars may be a letter, an underscore or a digit.

little-endian Multiple byte data values (not strings) are stored with the lowest order byte first. For example, the long integer &H01020304 is stored as this sequence of four bytes: &H04, &H03, &H02 and &H01. A Binary or Random file written using **Put** uses little-endian format so that it can be read using **Get** on any machine. (*Big-endian* machines, like the Power-PC, reverse the bytes as they are read by **Get** or written by **Put**.)

macro A macro is like an application. Execution starts at the macro's Sub **Main**.

method An object provides methods and *properties*. Methods can be called as subs (the return value is ignored), or used as functions (the return value is used).

If the method name contains characters that are not legal in a *name*, surround the method name with [].

App.[Title$]

module A file with **public** symbols that are accessible by other modules/*macros* via the **'#Uses** special comment.

- A module is loaded on demand.
- A **code module** is a code library.
- An **object module** or **class module** implements an object.
- A module may also access other modules with its own **'#Uses** special comments.

name An identifier that names a variable or a user defined *procedure*. Identifiers start with a letter. Following chars may be a letter, an underscore or a digit.

Count
DaysTill2000
Get_Data

num An expression that returns a numeric result. Use &O to express an octal number. Use &H to express a hex number. The interpretation &H and &O is affected by the **'#Language** setting. Specific types of numbers can be indicated by using one of the endings shown in the table below:

Ending **Description**

I or % The number is a **Integer** value. The interpretation of the type is controlled by the **'#Language** setting.

L or & The number is a **Long** value. The interpretation of the type is controlled by the **'#Language** setting.

H The number is a **Huge_** value.

UI The number is a **UInteger** value. The interpretation of the type is controlled by the **'#Language** setting.

UL The number is a **ULong** value. The interpretation of the type is controlled by the **'#Language** setting.

UH The number is a **UHuge_** value.

C The number is a **Currency** value.

D The number is a **Decimal** value.

F or ! The number is a **Single** value.

R or # The number is a **Double** value.

? The number is a **PortInt** value.

@ The number is a **Currency** value. The interpretation of the type is controlled by the **'#Language** setting.

numstr An expression that returns a *numeric* or *string* result.

numvar A variable that holds one numeric value.

objexpr A expression that returns a reference to an object or *module*.

CreateObject("WinWrap.CDemoApplication")

objtype A specific type defined by your application, another application or by an **object module** or **class module**.

objvar A variable that holds a *objexpr* which references an object. Object variables are declared using As *Object* in a **Dim**, **Private** or **Public** statement.

param [[Optional] [| ByVal | ByRef] | ParamArray] *param*[*type*][( )] [*As type*] [= *defaultvalue*]

The *param* receives the value of the associated expression in the **Declare**, **Sub**, **Function** or **Property** call. (See *arglist*.)

• An Optional *param* may be omitted from the call. It may also have a *defaultvalue*. The parameter receives the defaultvalue if a value is not specified by the call. If the defaultvalue is omitted, the parameter is a **Variant** and no value is specified in the call then **IsMissing** will return **True**.

• All parameters following an Optional parameter must also be Optional.

• ParamArray may be used on the final *param*. It must be an array of **Variant** type. It must not follow any Optional parameters. The ParamArray receives all the expressions at the end of the call as an array. If **LBound**(*param*) > **UBound**(*param*) then the ParamArray didn't receive any expressions.

• If the *param* is not ByVal and the expression is merely a variable then the *param* is a reference to that variable (ByRef). (Changing *param* changes the variable.) Otherwise, the parameter variable is local to the *procedure*, so changing its value does not affect the caller.

• Use *param*( ) to specify an array parameter. An array parameter must be referenced and can not be passed by value. The bounds of the parameter array are available via **LBound( )** and **UBound( )**.

precedence When several operators are used in an expression, each operator is evaluated in a predetermined order. Operators are evaluated in this order:

• ^ (power)

- - (negate)
- * (multiply), / (divide)
- \ (integer divide)
- Mod (integer remainder)
- + (add), - (difference)
- << (shift left), >> (shift right)
- & (string concatenate)
- = (equal), <> (not equal), < (less than) > (greater than), <= (less than or equal to), >= (greater than or equal to), **Like**, (string similarity) **New**, (object creation) **TypeOf**, (object type) **Is**, (object equivalence) **IsNot** (object non-equivalence)
- Not (bitwise invert)
- And (bitwise and), AndAlso (short-circuit logical and)
- Or (bitwise or), OrElse (short-circuit logical or)
- Xor (bitwise exclusive-or)
- Eqv (bitwise equivalence)
- Imp (bitwise implication)

Operators shown on the same line are evaluated from left to right.

procedure A **sub**routine, **function** or **property**.

property An object provides *method*s and properties. Properties may be used as values (like a function call) or changed (using assignment syntax).

If the property name contains characters that are not legal in a *name*, surround the property name with [].

App.[Title$]

statement Zero or more *instruction*s. A statement is at least one line long. **Begin Dialog**, **Do**, **For**, **If** (multiline), **Select Case**, **While** and **With** statements are always more than one line long. A single line statement continues on the next line if it ends a line with a space and an underscore ' _'.

```
S$ = "This long string is easier to read, " + _
    "if it is broken across two lines."
Debug.Print S$
```

str An expression that returns a string result.

```
"Hello"
S$
S$ + " Goodbye"
```

S$ & " Goodbye"
Mid$(S$,2)

A string constant may have a 'C' suffix

S$ = "x"C

The 'C' suffix indicates that the string is one character long.

strarray A variable that holds an array of string values.

streamnum An expression that returns a numeric result. Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

strvar A variable that holds one string value.

FirstName$

type Variable and parameter types, as well as, function and property results may be specified using a type character as the last character in their name.

Type char **As Type**

% **Integer**, The interpretation of the type is controlled by the **'#Language** setting.

& **Long**, The interpretation of the type is controlled by the **'#Language** setting.

! **Single**

Double

@ **Currency**, The interpretation of the type is controlled by the **'#Language** setting.

$ **String**

? **PortInt**

user delegate User defined delegates are defined with **Delegate**.

user dialog User defined dialogs are defined with **Begin Dialog**.

user enum User defined enums are defined with **Enum**.

user type User defined types are defined with **Type**.

usertypevar A user defined type variable holds values for elements of the user defined type. User defined types are defined using **Type**.

- Declare with **Dim**, **Private**, **Public** or **Static**.
- Declare as a parameter of **Sub**, **Function** or **Property** definition.

var A variable holds either a string, a numeric value or an array of values depending on its type.

vardeclaration *name*[*type*][([*dimension*[, ...]])][*As [New] type*]

The *name* declares a variable.

variantvar A variant variable can hold any type of value (except **String*n**). or it can hold an array.

AboutWinWrapBasic Instruction

Syntax AboutWinWrapBasic [*Timeout*]

Group Miscellaneous

Description Show the WinWrap Basic about box.

Parameter **Description**

Timeout This numeric value is the maximum number of seconds to show the about box. A value less than or equal to zero displays the about box until the user closes it. If this value is omitted then a three second timeout is used.

Example
```
'#Language "WWB-COM"
Sub Main
  AboutWinWrapBasic
End Sub
```

Abs Function

Syntax Abs(*Num*)

Group Math

Description Return the absolute value.

Parameter **Description**

Num Return the absolute value of this numeric value. If this value is **Null** then **Null** is returned.

See Also **Sgn**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Abs(9)  ' 9
  Debug.Print Abs(0)  ' 0
  Debug.Print Abs(-9) ' 9
End Sub
```

AddressOf Operator

Syntax AddressOf [*expr.*]*name*

Group Operator

Description Return a delegate for the *expr macro/module*'s Sub or Function matching *name*.

A delegate's Sub or Function is called using the delegate's Invoke method.

Note: The AddressOf operator returns an object which holds a weak reference to the macro/module. If no other references to the macro/module exist then the macro/module reference is deleted and using the result of the AddressOf operator causes a run-time error.

Parameter **Description**

expr This is a macro/module reference. If omitted then the current macro/module's reference (Me) is used.

name Return a delegate for this Sub or Function.

See Also **Delegate**.

Example
```
'#Language "WWB-COM"
Delegate Function OpType(ByVal v1 As Variant, ByVal v2 As Variant) As Variant

SubMain
  Debug.Print DoOp(AddressOf Add,1,2) ' 3
  Debug.Print DoOp(AddressOf Subtract,1,2) '-1
End Sub

Function DoOp(ByVal op As OpType, _
        ByVal v1 As Variant, ByVal v2 As Variant) As Variant
  DoOp = op.Invoke(v1,v2)
EndFunction

Function Add(ByVal v1 As Variant, ByVal v2 As Variant) As Variant
  Add = v1+v2
End Function

Function Subtract(ByVal v1 As Variant, ByVal v2 As Variant) As Variant
  Subtract = v1-v2
EndFunction
```

Any Data Type

Syntax **Declare** ...(... v As Any ...)...

Group Data Type

Description Any variable expression (**Declare** only).

AppActivate Instruction

Syntax AppActivate *Title$*
-or-
AppActivate *TaskID*

Group Miscellaneous

Description Form 1: Activate the application top-level window titled *Title$*. If no window by that title exists then the first window with at title that starts with *Title$* is activated. If no window matches then an error occurs.

Form 2: Activate the application top-level window for task *TaskID*. If no window for that task exists then an error occurs.

Parameter **Description**

Title$ The name shown in the title bar of the window.

TaskID This numeric value is the task identifier.

See Also **SendKeys**, **Shell( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  ' make ProgMan the active application
  AppActivate "Program Manager"
End Sub
```

Array Function

Syntax Array([*expr*[, ...]])

Group Conversion

Description Return a variant value array containing the *expr*s.

Example
```
'#Language "WWB-COM"
Sub Main
  X = Array(0,1,4,9)
  Debug.Print X(2) ' 4
End Sub
```

Asc Function

Syntax Asc(*S$*)

Group String

Description Return the ASCII value.

Note: A similar function, AscW, returns the Unicode value. Another similar function, AscB, returns the first byte in *S$*.

Parameter **Description**

S$ Return the ASCII value of the first char in this string value.

See Also **Chr$( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Asc("A") ' 65
End Sub
```

Assign Instruction

Syntax Assign *lhs*, *rhs*[, *Depth*]

Group Miscellaneous

Description Assign the value of the *rhs* to the *lhs*.

Parameter **Description**

lhs This string represents the variable expression to be set.

rhs Evaluate this string value and assign it to the *lhs* expression.

Depth This integer value indicates how deep into the stack to locate the local variables. If Depth = 0 then use the current *procedure*. If this value is omitted then the depth is 0.

VBA This language element is not VBA compatible.

See Also **Eval**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim X As String
  Assign "X", """Hello"""
  Debug.Print X 'Hello
  A
  Debug.Print X 'Bye
End Sub
Sub A
  Dim X As String
  Assign "X", """Welcome"""
  Assign "X", """Bye""", 1
  Debug.Print Eval("X") 'Welcome
```

```
  Debug.Print Eval("X",1) 'Bye
End Sub
```

Assign Operators

Syntax *var* [Op] *expr*

Group Assignment

Description Assign a value to *var*.

Op **Description**

= Assign the value of *expr* to *var*.

+= The same as: *var* = *var* + (*expr*). This language element is not VBA compatible.

-= The same as: *var* = *var* - (*expr*). This language element is not VBA compatible.

*= The same as: *var* = *var* * (*expr*). This language element is not VBA compatible.

/= The same as: *var* = *var* / (*expr*). This language element is not VBA compatible.

\= The same as: *var* = *var* \ (*expr*). This language element is not VBA compatible.

^= The same as: *var* = *var* ^ (*expr*). This language element is not VBA compatible.

<<= The same as: *var* = *var* << (*expr*). This language element is not VBA compatible.

>>= The same as: *var* = *var* >> (*expr*). This language element is not VBA compatible.

&= The same as: *var* = *var* & (*expr*). This language element is not VBA compatible.

Example
```
'#Language "WWB-COM"
Sub Main
  X = 1
  X = X*2
  Debug.Print X ' 2
End Sub
```

Atn Function

Syntax Atn(*Num*)

Group Math

Description Return the arc tangent. This is the number of radians. There are 2*Pi radians in a full circle.

Parameter **Description**

Num Return the arc tangent of this numeric value.

See Also **Cos**, **Sin**, **Tan**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** Atn(1)*4 ' 3.1415926535898
EndSub

Attribute Definintion/Statement

Syntax Attribute attributename = value
Attribute varname.attributename = value
Attribute procname.attributename = value

Group Declaration

Description All attribute definitions and statements are ignored except for:

- Form 1: Module level attribute

Attribute VB_**Name** = "name"
Attribute VB_GlobalNameSpace = bool
Attribute VB_Creatable = bool
Attribute VB_PredeclaredId = bool
Attribute VB_Exposed = bool
Attribute VB_HelpID = int
Attribute VB_Description = "text"

VB_Name - Declares the name of the **class module** or **object module**.
VB_GlobalNameSpace - Declares the class module as a global class. (ignored)
VB_Creatable - Declares the module as creatable (True), non-creatable (False). (ignored)
VB_PredeclaredId - Declares the module as a predeclared identifier (True). (ignored)
VB_Exposed - Declares the module as public (True). (ignored)
VB_HelpID - Declares the module's help context displayed by the object browser.
VB_Description - Declares the module's help text displayed by the object browser.

- Form 2: Macro/Module level variable attribute

Public varname As type
Attribute varname.VB_VarUserMemId = 0
Attribute varname.VB_VarHelpID = int
Attribute varname.VB_VarDescription = "text"

VB_VarUserMemID - Declares **Public** varname as the default property for a **class module** or **object module**.
VB_VarHelpID - Declares the variable's help context displayed by the object browser.
VB_VarDescription - Declares the variable's help text displayed by the object browser.

- Form 3: User defined procedure attribute

[**Sub** | **Function** | **Property** [**Get**|**Let**|**Set**]] procname ...
Attribute procname.VB_UserMemId = 0
Attribute procname.VB_HelpID = int

Attribute procname.VB_Description = "text"
...
End [**Sub** | **Function** | **Property**]

VB_UserMemID - Declares **Property** procname as the default property for a **class module** or **object module**.
VB_HelpID - Declares the procedure's help context displayed by the object browser.
VB_Description - Declares the procedure's help text displayed by the object browser.

HelpFile Each macro/module can define the HelpFile for the object browser:

'#HelpFile "helpfile"

where "helpfile" is a full path to the help file associated with the help text and help context.

Beep Instruction

Syntax Beep

Group Miscellaneous

Description Sound the bell.

Example
```
'#Language "WWB-COM"
Sub Main
  Beep ' beep the bell
End Sub
```

Begin Dialog Definition

Syntax
```
Begin Dialogname [X, Y,] DX, DY[, Title$] _
      [, .dialogfunc]
  User Dialog Item
  [User Dialog Item]...
EndDialog
```

Group User Dialog

Description Define a *user dialog* type to be used later in a **Dim** As *name* statement.

Parameter **Description**

X This numeric value is the distance from the left edge of the screen to the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font. If this is omitted then the dialog will be centered.

Y This numeric value is the distance from the top edge of the screen to the top edge of the dialog box. It is measured in 1/12 ths of the average character width for the dialog's font. If this is omitted then the dialog will be centered.

DX This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.

DY This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.

Title$ This string value is the title of the user dialog. If this is omitted then there is no title.

dialogfunc This is the function name that implements the **DialogFunc** for this *user dialog*. If this is omitted then the UserDialog doesn't have a dialogfunc.

User Dialog Item
One of: **CancelButton**, **CheckBox**, **ComboBox**, **DropListBox**, **GroupBox**, **ListBox**, **MultiListBox**, **OKButton**, **OptionButton**, **OptionGroup**, **PushButton**, **Text**, **TextBox**.

```
Example          '#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  Dialog dlg ' show dialog (wait for ok)
End Sub
```

Boolean Data Type

Syntax **Dim** v As Boolean

Group Data Type

Description A **True** or **False** value.

Byte Data Type

Syntax **Dim** v As Byte

Group Data Type

Description An 8 bit unsigned integer value.

CallByName Instruction

Syntax CallByName(*Obj*,*ProcName*,*CallType*,[*expr*[, ...]])

Group Flow Control

Description Call an *Obj*'s method/property, *ProcName*, by name. Pass the *expr*s to the method/property.

Parameter **Description**

Obj Call the method/property for this object reference.

ProcName This string value is the name of the method/property to be called.

CallType Type of method/property call. See table below.

expr These expressions are passed to the obj's method/property.

CallType Value Effect

vbMethod 1 Call or evaluate the method.

vbGet 2 Evaluate the property's value.

vbLet 4 Assign the property's value.

vbSet 8 Set the property's reference.

Example
```
'#Language "WWB-COM"
Sub Main
  On ErrorResume Next
  CallByName Err, "Raise", vbMethod, 1
  Debug.Print CallByName(Err, "Number", vbGet) ' 1
EndSub
```

CallersLine Function

Syntax CallersLine[(*Depth*)]

Group Miscellaneous

Description Return the caller's line as a text string.

The text format is: "[macroname|subname#linenum] linetext".

Parameter Description

Depth This integer value indicates how deep into the stack to get the caller's line. If Depth = -1 then return the current line. If Depth = 0 then return the calling subroutine's current line, etc.. If Depth is greater than or equal to the call stack depth then a null string is returned. If this value is omitted then the depth is 0.

Example
```
'#Language "WWB-COM"
Sub Main
  A
EndSub
Sub A
  Debug.Print CallersLine '"[(untitled 1)|Main#  2] A"
EndSub
```

Call Instruction

Syntax Call *name*[(*arglist*)]
-or-
name [*arglist*]

Group Flow Control

Description Evaluate the *arglist* and call subroutine (or function) *name* with those values. Sub (or function) *name* must be previously defined by either a **Sub**, **Function** or **Property** definition. If *name* is a function then the result is discarded. If Call is omitted and *name* is a subroutine then the *arglist* must not be enclosed in parens.

See Also **Declare**, **Sub**.

Example
```
'#Language "WWB-COM"
Sub Show(Title$,Value)
  Debug.Print Title$; "="; Value
EndSub

Sub Main
  Call Show("2000/9",2000/9) ' 222.2222222222
  Show "1<2",1<2          'True
End Sub
```

CancelButton Dialog Item Definition

Syntax CancelButton *X*, *Y*, *DX*, *DY*[, .*Field*]

Group User Dialog

Description Define a cancel button item. Pressing the Cancel button from a **Dialog** instruction causes a run-time error. (**Dialog**( ) function call returns 0.)

Parameter **Description**

X This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.

Y This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.

DX This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.

DY This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.

Field This identifier is the name of the field. The *dialogfunc* receives this name as *string*. If this is omitted then the field name is "Cancel".

See Also **Begin Dialog**.

Example
```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,30,"Please push the Cancel button"
    OKButton 40,90,40,20
    CancelButton 110,90,60,20
  EndDialog
  Dim dlg As UserDialog
  Dialog dlg ' show dialog (wait for cancel)
  Debug.Print "Cancel was not pressed"
End Sub
```

CBool Function

Syntax CBool(*expr*)

Group Conversion

Description Convert to a **Boolean** value. Zero converts to **False**, while all other values convert to **True**.

Parameter **Description**

expr Convert a number or string value to a boolean value.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print CBool(-1) 'True
  Debug.Print CBool(0)  'False
  Debug.Print CBool(1)  'True
EndSub
```

CByte Function

Syntax CByte(*expr*)

Group Conversion

Description Convert to an 8 bit unsigned integer **Byte** value.

Parameter **Description**

expr Convert a number or string value to an unsigned byte value.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print CByte(1.6) ' 2
EndSub
```

CCur Function

Syntax CCur(*expr*)

Group Conversion

Description Convert to a **Currency** value.

Parameter **Description**

expr Convert a number or string value to a currency value.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print CCur("1E6") ' 1000000
EndSub
```

CDate Function

Syntax CDate(*expr*)
-or-
CVDate(*expr*)

Group Conversion

Description Convert to a **Date** value.

Parameter **Description**

expr Convert a number or string value to a date value.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print CDate(2) ' 1/1/00
EndSub
```

CDbl Function

Syntax CDbl(*expr*)

Group Conversion

Description Convert to a **Double** precision real.

Parameter **Description**

expr Convert a number or string value to a double precision real.

Example
```
'#Language "WWB-COM"
Sub Main
```

```
  Debug.Print CDbl("1E6") ' 1000000
EndSub
```

CDec Function

Syntax CDec(*expr*)

Group Conversion

Description Convert to a **Decimal** (96 bit scaled real).

Parameter **Description**

expr Convert a number or string value to a 96 bit scaled real.

Example

```
'#Language "WWB-COM"
Sub Main
  Debug.Print CDec("1E16")+0.1 ' 10000000000000000.1
EndSub
```

ChDrive Instruction

Syntax ChDrive *Drive$*

Group File

Description Change the current drive to *Drive$*.

Pocket PC Not supported.

Parameter **Description**

Drive$ This string value is the drive letter.

See Also **ChDir**, **CurDir$( )**.

Example

```
'#Language "WWB-COM"
Sub Main
  ChDrive "B"
  Debug.Print CurDir$() '"B:\"
End Sub
```

ChDir Instruction

Syntax ChDir *Dir$*

Group File

Description Change the current directory to *Dir$*.

Pocket PC Not supported.

Parameter **Description**

Dir$ This string value is the path and name of the directory.

See Also **ChDrive**, **CurDir$( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  ChDir "C:\"
  Debug.Print CurDir$() '"C:\"
End Sub
```

CheckBox Dialog Item Definition

Syntax CheckBox *X*, *Y*, *DX*, *DY*, *Title$*, .*Field*[, *Options*]

Group User Dialog

Description Define a checkbox item.

Parameter **Description**

X This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.

Y This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.

DX This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.

DY This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.

Field The value of the check box is accessed via this field. Unchecked is 0, checked is 1 and grayed is 2.

Options This numeric value controls the type of check box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option **Description**

0 Check box is either check or unchecked.

1 Check box is either check, unchecked or grayed, and it switches between checked and unchecked when clicked.

2 Check box is either check, unchecked or grayed, and it cycles through all three states as the button is clicked.

See Also **Begin Dialog**.

Example
```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
```

```
    CheckBox 10,25,180,15,"&Check box",.Check
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  dlg.Check = 1
  Dialog dlg ' show dialog (wait for ok)
  Debug.Print dlg.Check
EndSub
```

Choose Function

Syntax Choose(*Index*, *expr*[, ...])

Group Flow Control

Description Return the value of the *expr* indicated by *Index*.

Parameter **Description**

Index The numeric value indicates which *expr* to return. If this value is less than one or greater than the number of *expr*s then **Null** is returned.

expr All expressions are evaluated.

See Also **If**, **Select Case**, **IIf( )**.

Example

```
'#Language "WWB-COM"
Sub Main
  Debug.Print Choose(2,"Hi","there") '"there"
EndSub
```

Chr$ Function

Syntax Chr[$](*Num*)

Group String

Description Return a one char string for the ASCII value.

Note: A similar function, ChrW, returns a single char Unicode string. Another similar function, ChrB, returns a single byte ASCII string.

Parameter **Description**

Num Return one char string for this ASCII numeric value.

See Also **Asc( )**.

Example

```
'#Language "WWB-COM"
Sub Main
```

```
  Debug.Print Chr$(48) '"0"
End Sub
```

CHuge_ Function

Syntax CHuge_(*expr*)

Group Conversion

Description Convert to a 64 bit signed integer (**Huge_**). If *expr* is too big (or too small) to fit then an overflow error occurs.

Some versions of Windows do not support CHuge_.

VBA This language element is not VBA compatible.

Parameter **Description**

expr Convert a number or string value to a 64 bit signed integer.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print CHuge_(2^32) ' 4294967296
EndSub
```

CInt Function

Syntax CInt(*expr*)

Group Conversion

Description Convert to an **Integer**. If *expr* is too big (or too small) to fit then an overflow error occurs.

Parameter **Description**

expr Convert a number or string value to an **Integer**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print CInt(1.6) ' 2
End Sub
```

Class_Initialize Sub

Syntax
```
Private Sub Class_Initialize()
  ...
End Sub
```

Group Declaration

Description **Class module** initialization subroutine. Each time a new instance is created for a class module the Class_Initialize sub is called. If Class_Initialize is not defined then no special initialization occurs.

See Also **Code Module**, **Class_Terminate**.

Class Module

Group Declaration

Description A class *module* implements an object.

- Has a set of **Public** *procedure*s accessible from other *macros* and *modules*.
- These public symbols are accessed via an object variable.
- Has an optional set of **Event**s that can be raised.
- Public **Const**s, **Type**s, arrays, fixed length strings are not allowed.
- Has an optional Private Sub **Class_Initialize** which is called when an instance is created.
- Has an optional Private Sub **Class_Terminate** which is called when an instance is destroyed.
- A class module is similar to a **object module** except that no instance is automatically created.
- To create an instance use:

```
Dim Obj As classname
Set Obj = New classname
```

See Also **Code Module**, **Object Module**, **Uses**, **Class_Initialize**, **Class_Terminate**.

Example

```
'A.WWB
'#Language "WWB-COM"
'#Uses "File.CLS"
Sub Main
  Dim File As New File
  File.Attach "C:\AUTOEXEC.BAT"
  Debug.Print File.ReadLine
End Sub

'File.CLS
'File|New Module|Class Module
'Edit|Properties|Name=File
'#Language "WWB-COM"
Option Explicit
Dim FN As Integer
Public Sub Attach(FileName As String)
  FN = FreeFile
  Open FileName For Input As #FN
EndSub
```

```
Public Sub Detach()
  If FN <> 0 Then'Close #FN
  FN = 0
EndSub
Public Function ReadLine() As String
  Line Input '#'FN, ReadLine
End Function

Private Sub Class_Initialize()
  Debug.Print "Class_Initialize"
End Sub

Private Sub Class_Terminate()
  Debug.Print "Class_Terminate"
  Detach
EndSub
```

Class_Terminate Sub

Syntax **Private Sub** Class_Terminate()
...
End Sub

Group Declaration

Description **Class module** termination subroutine. Each time an instance is destroyed for a class module the Class_Terminate sub is called. If Class_Terminate is not defined then no special termination occurs.

See Also **Code Module**, **Class_Initialize**.

Clipboard Instruction/Function

Syntax Clipboard *Text$*
-or-
Clipboard[$][( )]

Group Miscellaneous

Description Form 1: Set the clipboard to *Text$*. This is like the Edit|Copy menu command.

Form 2: Return the text in the clipboard.

Parameter **Description**

Text$ Put this string value into the clipboard.

Example

```
'#Language "WWB-COM"
Sub Main
  Debug.Print Clipboard$()
```

```
  Clipboard "Hello"
  Debug.Print Clipboard$() '"Hello"
End Sub
```

CLng Function

Syntax CLng(*expr*)

Group Conversion

Description Convert to a **Long**. If *expr* is too big (or too small) to fit then an overflow error occurs.

Parameter **Description**

expr Convert a number or string value to a **Long**.

Example

```
'#Language "WWB-COM"
Sub Main
  Debug.Print CLng(1.6) ' 2
End Sub
```

Close Instruction

Syntax Close [[#]*StreamNum*][, ...]

Group File

Description Close *StreamNum*s.

Parameter **Description**

StreamNum Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros. If this is omitted then all open streams for the current *macro*/*module* are closed.

See Also **Open**, **Reset**.

Example

```
'#Language "WWB-COM"
Sub Main
  ' read the first line of XXX and print it
  Open "XXX" ForInput As #1
  Line Input #1, L$
  Debug.Print L$
  Close #1
End Sub
```

Code Module

Group Declaration

Description A Code *module* implements a code library.

- Has a set of **Public** *procedure*s accessible from other *macros* and *modules*.
- These public symbols are accessed directly.
- May be initialized by a Private Sub **Main**.

See Also **Class Module**, **Object Module**, **Uses**, **Main**.

Example
```
'A.WWB
'#Language "WWB-COM"
'#Uses "Module1.BAS"
SubMain
  Debug.Print Value '"Hello"
End Sub

'Module1.BAS
'File|New Module|Code Module
'Edit|Properties|Name=Module1
'#Language "WWB-COM"
Option Explicit
Private mValue As String
Property Get Value() As String
  Value = mValue
EndProperty
'this sub is called when the module is first loaded
Private Sub Main
  mValue = "Hello"
End Sub
```

ComboBox Dialog Item Definition

Syntax ComboBox *X*, *Y*, *DX*, *DY*, *StrArray$( )*, *.Field$*[, *Options*]

Group User Dialog

Description Define a combobox item. Combo boxes combine the functionality of an edit box and a list box.

Parameter **Description**

X This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.

Y This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.

DX This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.

DY This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.

StrArray$( ) This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.

Field$ The value of the combo box is accessed via this field. This is the text in the edit box.

Options This numeric value controls the type of combo box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option **Description**

0 List is not sorted.

2 List is sorted.

See Also **Begin Dialog**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim combos$(3)
  combos$(0) = "Combo 0"
  combos$(1) = "Combo 1"
  combos$(2) = "Combo 2"
  combos$(3) = "Combo 3"
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    ComboBox 10,25,180,60,combos$(),.combo$
    OKButton 80,90,40,20
  EndDialog
  Dim dlg As UserDialog
  dlg.combo$ = "none"
  Dialog dlg ' show dialog (wait for ok)
  Debug.Print dlg.combo$
End Sub
```

Command$ Function

Syntax Command[$]

Group Miscellaneous

Description Contains the value of the **MacroRun** parameters.

See Also **MacroRun**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print "Command line parameter is: """;
  Debug.Print Command$;
  Debug.Print """"
End Sub
```

Const Definition

Syntax [| **Private** | **Public**] _
Const *name*[*type*] [*As Type*] = *expr*[, ...]

Group Declaration

Description Define *name* as the value of *expr*. The *expr* may be refer other constants or built-in functions. If the type of the constants is not specified, the type of *expr* is used. Constants defined outside a **Sub**, **Function** or **Property** block are available in the entire *macro*/*module*.

Private is assumed if neither **Private** or **Public** is specified.

Note: Const statement in a **Sub**, **Function** or **Property** block may not use **Private** or **Public**.

Example
```
'#Language "WWB-COM"
Sub Main
  Const Pi = 4*Atn(1), e = Exp(1)
  Debug.Print Pi ' 3.14159265358979
  Debug.Print e  ' 2.71828182845905
EndSub
```

Cos Function

Syntax Cos(*Num*)

Group Math

Description Return the cosine.

Parameter **Description**

Num Return the cosine of this numeric value. This is the number of radians. There are 2*Pi radians in a full circle.

See Also **Atn**, **Sin**, **Tan**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Cos(1) ' 0.54030230586814
EndSub
```

CreateObject Function

Syntax CreateObject(*Class$*)

Group Object

Description Create a new object of type *Class$*. Use **Set** to assign the returned object to an object variable.

Parameter **Description**

Class$ This string value is the application's registered class name. If this application is not currently active it will be started.

See Also **Objects**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim App As Object
  Set App = CreateObject("WinWrap.CppDemoApplication")
  App.Move 20,30 ' move icon to 20,30
  Set App = Nothing
  App.Quit     ' run-time error (no object)
End Sub
```

CSByte Function

Syntax CSByte(*expr*)

Group Conversion

Description Convert to an 8 bit signed integer **SByte** value.

VBA This language element is not VBA compatible and requires the **#Language** "WWB-COM" setting.

Parameter **Description**

expr Convert a number or string value to a signed byte value.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print CSByte(1.6) ' 2
EndSub
```

CSng Function

Syntax CSng(*expr*)

Group Conversion

Description Convert to a **Single** precision real. If *expr* is too big (or too small) to fit then an overflow error occurs.

Parameter **Description**

expr Convert a number or string value to a single precision real.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** CSng(**Sqr**(2)) ' 1.4142135381699
EndSub

CStr Function

Syntax CStr(*expr*)

Group Conversion

Description Convert to a **String**.

Parameter **Description**

expr Convert a number or string value to a string value using the current locale (**GetLocale**).

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** CStr(**Sqr**(2)) '"1.4142135623731" - US English locale
End Sub

CUHuge_ Function

Syntax CUHuge_(*expr*)

Group Conversion

Description Convert to a 64 bit unsigned integer (**UHuge_**). If *expr* is too big (or too small) to fit then an overflow error occurs.

Some versions of Windows do not support CUHuge_.

VBA This language element is not VBA compatible and requires the **#Language** "WWB-COM" setting.

Parameter **Description**

expr Convert a number or string value to a 64 bit unsigned integer.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** CUHuge_(2^32) ' 4294967296
EndSub

CUInt Function

Syntax CUInt(*expr*)

Group Conversion

Description Convert to an **UInteger**. If *expr* is too big (or too small) to fit then an overflow error occurs.

VBA This language element is not VBA compatible and requires the **#Language** "WWB-COM" setting.

Parameter **Description**

expr Convert a number or string value to an **UInteger**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print CUInt(1.6) ' 2
EndSub
```

CULng Function

Syntax CULng(*expr*)

Group Conversion

Description Convert to a **ULong**. If *expr* is too big (or too small) to fit then an overflow error occurs.

VBA This language element is not VBA compatible and requires the **#Language** "WWB-COM" setting.

Parameter **Description**

expr Convert a number or string value to a **ULong**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print CULng(1.6) ' 2
EndSub
```

CurDir$ Function

Syntax CurDir[$]([*Drive$*])

Group File

Description Return the current directory for *Drive$*.

Pocket PC Not supported.

Parameter **Description**

Drive$ This string value is the drive letter. If this is omitted or null then return the current directory for the current drive.

See Also **ChDir**, **ChDrive**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** CurDir$()
End Sub

Currency Data Type

Syntax **Dim** v As Currency

Group Data Type

Description A 64 bit fixed point real. (A twos complement binary value scaled by 10000.)

CVar Function

Syntax CVar(*expr*)

Group Conversion

Description Convert to a **Variant** value.

Parameter **Description**

expr Convert a number or string value (or object reference) to a variant value.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** CVar(**Sqr**(2)) ' 1.4142135623731
EndSub

CVErr Function

Syntax CVErr(*expr*)

Group Conversion

Description Convert to a **variant** that contains an error code. An error code can't be used in expressions.

Parameter **Description**

expr Convert a number or string value to an error code.

See Also **IsError**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** CVErr(1) ' **Error** 1
EndSub

DateAdd Function

Syntax DateAdd(*interval*, *number*, *dateexpr*)

Group Time/Date

Description Return a **Date** value a number of intervals from another date.

Parameter **Description**

interval This string value indicates which kind of interval to add.

number Add this many intervals. Use a negative value to get an earlier date.

dateexpr Calculate the new date relative to this date value. If this value is **Null** then **Null** is returned.

Interval **Description**

yyyy Year

q Quarter

m Month

y Day of year

d Day

w Weekday

ww Week

h Hour

n Minute

s Second

See Also **DateDiff**, **DatePart**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print DateAdd("yyyy",1,#1/1/2000#) '1/1/2001
EndSub
```

DateDiff Function

Syntax DateDiff(*interval*, *dateexpr1*, *dateexpr2*)

Group Time/Date

Description Return the number of intervals between two dates.

Parameter **Description**

interval This string value indicates which kind of interval to subtract.

dateexpr1 Calculate the from this date value to dateexpr2. If this value is **Null** then **Null** is returned.

dateexpr2 Calculate the from dateexpr1 to this date value. If this value is **Null** then **Null** is returned.

Interval	**Description**
yyyy	Year
q	Quarter
m	Month
y	Day of year
d	Day
w	Weekday
ww	Week
h	Hour
n	Minute
s	Second

See Also **DateAdd**, **DatePart**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print DateDiff("yyyy",#1/1/1990#,#1/1/2000#) ' 10
End Sub
```

Date Function

Syntax Date[$]

Group Time/Date

Description Return today's date as a **Date** value.

See Also **Now**, **Time**, **Timer**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Date ' example: 1/1/1995
EndSub
```

DatePart Function

Syntax DatePart(*interval*, *dateexpr*)

Group Time/Date

Description Return the number from the date corresponding to the interval.

Parameter **Description**

interval This string value indicates which kind of interval to extract.

dateexpr Get the interval from this date value. If this value is **Null** then **Null** is returned.

Interval **Description (return value range)**

yyyy Year (100-9999)

q Quarter (1-4)

m Month (1-12)

y Day of year (1-366)

d Day (1-31)

w Weekday (1-7)

ww Week (1-53)

h Hour (0-23)

n Minute (0-59)

s Second (0-59)

See Also **DateAdd**, **DateDiff**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print DatePart("yyyy",#1/1/2000#) ' 2000
EndSub
```

DateSerial Function

Syntax DateSerial(*Year*, *Month*, *Day*)

Group Time/Date

Description Return a **Date** value.

Parameter **Description**

Year This numeric value is the year (0 to 9999). (0 to 99 are interpreted by the operating system.)

Month This numeric value is the month (1 to 12).

Day This numeric value is the day (1 to 31).

See Also **DateValue**, **TimeSerial**, **TimeValue**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** DateSerial(2000,7,4) '7/4/2000
EndSub

DateValue Function

Syntax DateValue(*Date$*)

Group Time/Date

Description Return the day part of the date encoded as a string.

Parameter **Description**

Date$ Convert this string value to the day part of date it represents.

See Also **DateSerial**, **TimeSerial**, **TimeValue**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** DateValue("1/1/2000 12:00:01 AM")
    '1/1/2000
EndSub

Day Function

Syntax Day(*dateexpr*)

Group Time/Date

Description Return the day of the month (1 to 31).

Parameter **Description**

dateexpr Return the day of the month for this date value. If this value is **Null** then **Null** is returned.

See Also **Date( )**, **Month( )**, **Weekday( )**, **Year( )**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** Day(#1/1/1900#) ' 1
  Debug.**Print** Day(#1/2/1900#) ' 2
EndSub

DDEExecute Instruction

Syntax DDEExecute *ChanNum*, *Command$*[, *Timeout*]

Group DDE

Description Send the DDE Execute *Command$* string via DDE *ChanNum*.

Pocket PC Not supported.

Parameter **Description**

ChanNum This is the channel number returned by the **DDEInitiate** function. Up to 10 channels may be used at one time.

Command$ Send this command value to the server application. The interpretation of this value is defined by the server application.

Timeout The command will generate an error if the number of seconds specified by the timeout is exceeded before the command has completed. The default is five seconds.

Example
```
'#Language "WWB-COM"
Sub Main
  ChanNum = DDEInitiate("PROGMAN","PROGMAN")
  DDEExecute ChanNum,"[CreateGroup(XXX)]"
  DDETerminate ChanNum
End Sub
```

DDEInitiate Function

Syntax DDEInitiate(*App$*, *Topic$*)

Group DDE

Description Initiate a DDE conversation with *App$* using *Topic$*. If the conversation is successfully started then the return value is a channel number that can be used with other DDE instructions and functions.

Pocket PC Not supported.

Parameter **Description**

App$ Locate this server application.

Topic$ This is the server application's topic. The interpretation of this value is defined by the server application.

Example
```
'#Language "WWB-COM"
Sub Main
  ChanNum = DDEInitiate("PROGMAN","PROGMAN")
  DDEExecute ChanNum,"[CreateGroup(XXX)]"
  DDETerminate ChanNum
End Sub
```

DDEPoke Instruction

Syntax DDEPoke *ChanNum*, *Item$*, *Data$*[, *Timeout*]

Group DDE

Description Poke *Data$* to the *Item$* via DDE *ChanNum*.

Pocket PC Not supported.

Parameter **Description**

ChanNum This is the channel number returned by the **DDEInitiate** function. Up to 10 channels may be used at one time.

Item$ This is the server application's item. The interpretation of this value is defined by the server application.

Data$ Send this data value to the server application. The interpretation of this value is defined by the server application.

Timeout The command will generate an error if the number of seconds specified by the timeout is exceeded before the command has completed. The default is five seconds.

Example
```
'#Language "WWB-COM"
Sub Main
  ChanNum = DDEInitiate("PROGMAN","PROGMAN")
  DDEPoke ChanNum,"Group","XXX"
  DDETerminate ChanNum
End Sub
```

DDERequest$ Function

Syntax DDERequest[$](*ChanNum*, *Item$*[, *Timeout*])

Group DDE

Description Request information for *Item$*. If the request is not satisfied then the return value will be a null string.

Pocket PC Not supported.

Parameter **Description**

ChanNum This is the channel number returned by the **DDEInitiate** function. Up to 10 channels may be used at one time.

Item$ This is the server application's item. The interpretation of this value is defined by the server application.

Timeout The command will generate an error if the number of seconds specified by the timeout is exceeded before the command has completed. The default is five seconds.

Example
```
'#Language "WWB-COM"
Sub Main
  ChanNum = DDEInitiate("PROGMAN","PROGMAN")
  Debug.Print DDERequest$(ChanNum,"Groups")
```

```
  DDETerminate ChanNum
End Sub
```

DDETerminateAll Instruction

Syntax DDETerminateAll

Group DDE

Description Terminate all open DDE channels.

Pocket PC Not supported.

Example
```
'#Language "WWB-COM"
Sub Main
  ChanNum = DDEInitiate("PROGMAN","PROGMAN")
  DDEExecute ChanNum,"[CreateGroup(XXX)]"
  DDETerminateAll
EndSub
```

DDETerminate Instruction

Syntax DDETerminate *ChanNum*

Group DDE

Description Terminate DDE *ChanNum*.

Pocket PC Not supported.

Parameter	**Description**
ChanNum	This is the channel number returned by the **DDEInitiate** function. Up to 10 channels may be used at one time.

Example
```
'#Language "WWB-COM"
Sub Main
  ChanNum = DDEInitiate("PROGMAN","PROGMAN")
  DDEExecute ChanNum,"[CreateGroup(XXX)]"
  DDETerminate ChanNum
End Sub
```

Debug Object

Syntax Debug.Clear
-or-
Debug.**Print** [*expr*[; ...][;]]

Group Miscellaneous

Description Form 1: Clear the output window.

Form 2: Print the *expr*(s) to the output window. Use ; to separate expressions. A *num* is it automatically converted to a string before printing (just like **Str$( )**). If the instruction does not end with a ; then a newline is printed at the end.

Example
```
'#Language "WWB-COM"
Sub Main
  X = 4
  Debug.Print "X/2="; X/2 ' 2
  Debug.Print "Start..."; ' don't print a newline
  Debug.Print "Finish" ' print a newline
EndSub
```

Decimal Data Type

Syntax **Dim** v As Decimal

Group Data Type

Description A 96 bit scaled real value. A decimal number is of the form: s*m*10^-p where

- s - sign (+1 or -1)
- m - mantissa, unsigned binary value of 96 bits (0 to 79,228,162,514,264,337,593,543,950,335)
- p - scaling power (0 to +28)

VBA This language element is not VBA compatible and requires the **#Language** "WWB-COM" setting.

Declare Definition

Syntax
```
[ | Private | Public ] _
Declare Sub name Lib "dll name" _
  [Alias "module name"] [([param[, ...]])]
-or-
[ | Private | Public ] _
Declare Function name[type] Lib "dll name" _
  [Alias "module name"] [([param[, ...]])] [As type[()]]
```

Group Declaration

Description Interface to a DLL defined subroutine or function. The values of the calling *arglist* are assigned to the *param*s.

WARNING! Be very careful when declaring DLL subroutines or functions. If you make a mistake and declare the parementers or result incorrectly then Windows might halt. Save any open documents before testing new DLL declarations.

Err.LastDLLError returns the error code for that last DLL call.

Access If no access is specified then **Public** is assumed.

Pocket PC Not supported.

Parameter **Description**

name This is the name of the subroutine or function being defined. If Alias "module name" is omitted then this is the module name, too.

"dll name" This is the DLL file where the module's code is.

"module name" This is the name of the module in the DLL file. If this is #number then it is the ordinal number of the module. If it is omitted then *name* is the module name.
The DLL is searched for the specified module name. If this module exists, it is used. All As String parameters are converted from Unicode to ASCII prior to calling the DLL and from ASCII to Unicode afterwards. (Use "Unicode:module name" to prevent ASCII to Unicode conversion.)
If the module does not exist, one or two other module names are tried:
1) For Windows NT only: The module name with a "W" appended is tried. All As String parameters are passed as Unicode to calling the DLL.
2) For Windows NT and Windows 95: The module name with an "A" appended is tried. All As String parameters are converted from Unicode to ASCII prior to calling the DLL and from ASCII to Unicode afterwards.
If none of these module names is found a run-time error occurs.

params A list of zero or more *params* that are used by the DLL subroutine or function. (Note: A ByVal string's value may be modified by the DLL.)

See Also **Function**, **Sub**, **Call**.

Example
```
'#Language "WWB-COM"
Declare Function GetActiveWindow& Lib "user32" ()
Declare Function GetWindowTextLengthA& Lib "user32" _
  (ByVal hwnd&)
Declare Sub GetWindowTextA Lib "user32" _
  (ByVal hwnd&, ByVal lpsz$, ByVal cbMax&)

Function ActiveWindowTitle$()
  ActiveWindow = GetActiveWindow()
  TitleLen = GetWindowTextLengthA(ActiveWindow)
  Title$ = Space$(TitleLen)
  GetWindowTextA ActiveWindow,Title$,TitleLen+1
  ActiveWindowTitle$ = Title$
End Function

SubMain
  Debug.Print ActiveWindowTitle$()
End Sub
```

Decode64 Function

Syntax Decode64[$](*Data*)
-or-
Decode64B(*Data*)

Group Miscellaneous

Description Return a string using the base 64 decoding algorithm.

Decode64B returns a **Byte** array.

Parameter **Description**

Data Return this string's base 64 decoding.

See Also **Encode64**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Decode64("SGVsbG8gV29ybGQhIQ==") '"Hello World!!"
End Sub
```

Decrypt64 Function

Syntax Decrypt64[$](*Data* [,*Password*])
-or-
Decrypt64B(*Data* [,*Password*])

Group Miscellaneous

Description Return a string using the RC4 stream decryption algorithm.

Decrypt64B returns a **Byte** array.

Parameter **Description**

Data Return this string's decryption. (The string is first decoded using base 64 decoding.)

Password Decrypt using this password.

See Also **Decode64**, **Encode64**, **Encrypt64**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Decrypt64("Y4GFrF+k1YUHwjEzsg==", "abc") '"Hello World!!"
End Sub
```

Def Definition

Syntax Def{Bool|Cur|Date|Dbl|Int|Lng|Obj|Sng|Str|Var} _
letterrange[, ...]

Group Declaration

Description Define untyped variables as:

- DefBool - **Boolean**
- DefByte - **Byte**
- DefCur - **Currency**
- DefDate - **Date**
- DefDec - **Decimal**
- DefDbl - **Double**
- DefInt - **Integer**
- DefLng - **Long**
- DefObj - **Object**
- DefSng - **Single**
- DefStr - **String**
- DefVar - **Variant**

Parameter **Description**

letterrange letter, or letter-letter: A letter is one of A to Z. When letter-letter is used, the first letter must be alphabetically before the second letter. Variable names that begin with a letter in this range default to declared type.

If a variable name begins with a letter not in any letterrange then the variable is a **Variant**. The letterranges are not allowed to overlap.

See Also **Option** Explicit.

Example
```
'#Language "WWB-COM"
DefInt A,C-W,Y' integer
DefBool B    ' boolean
DefStr X     ' string
        ' all others are variant

Sub Main
  B = 1      ' B is an boolean
  Debug.Print B ' True
```

```
  X = "A"      ' X is a string
  Debug.Print X '"A"
  Z = 1        ' Z is a variant (anything)
  Debug.Print Z ' 1
  Z = "Z"
  Debug.Print Z '"Z"
End Sub
```

Delegate Definition

Syntax [| **Private** | **Public**] _
Delegate **Sub** *name* [([*param*[, ...]])]
-or-
[| **Private** | **Public**] _
Delegate **Function** *name*[*type*] _
[([*param*[, ...]])] [*As type*[()]]

Group Declaration

Description Define a new *user delegate* (subroutine or function pointer type).

A delegate's Sub or Function is called using the delegate's Invoke method.

Access If no access is specified then **Public** is assumed.

Parameter **Description**

name This is the name of the delegate being defined.

params A list of zero or more *params* that are used by the delegate's subroutine or function.

See Also **AddressOf**.

Example

```
'#Language "WWB-COM"
Delegate Function OpType(ByVal v1 As Variant, ByVal v2 As Variant) As Variant

SubMain
  Debug.Print DoOp(AddressOf Add,1,2) ' 3
  Debug.Print DoOp(AddressOf Subtract,1,2) '-1
EndSub

Function DoOp(ByVal op As OpType, _
        ByVal v1 As Variant, ByVal v2 As Variant) As Variant
  DoOp = op.Invoke(v1,v2)
EndFunction

Function Add(ByVal v1 As Variant, ByVal v2 As Variant) As Variant
  Add = v1+v2
End Function
```

```
Function Subtract(ByVal v1 As Variant, ByVal v2 As Variant) As Variant
  Subtract = v1-v2
EndFunction
```

DeleteSetting Instruction

Syntax DeleteSetting *AppName$*, *Section$*[, *Key$*]

Group Settings

Description Delete the settings for *Key* in *Section* in project *AppName*. Win16 and Win32s store settings in a .ini file named *AppName*. Win32/Win64 store settings in the registration database under "HKEY_CURRENT_USER\ Software\ VB and VBA Program Settings\ *AppName*\ *Section*\ *Key*". If *AppName* starts with "..\" then "VB and VBA Program Settings\" is omitted.

Parameter **Description**

AppName$ This string value is the name of the project which has this *Section* and *Key*.

Section$ This string value is the name of the section of the project settings.

Key$ This string value is the name of the key in the section of the project settings. If this is omitted then delete the entire section.

Example

```
'#Language "WWB-COM"
Sub Main
  SaveSetting "MyApp","Font","Size",10
  DeleteSetting "MyApp","Font","Size"
End Sub
```

DialogFunc Prototype

Syntax **Function***dialogfunc*(*DlgItem$*, *Action%*, *SuppValue?*) _
As **Boolean**
Select Case Action%
Case 1 ' **Dialog** box initialization
...
Case 2 ' Value changing or button pressed
...
Case 3 ' **TextBox** or **ComboBox** text changed
...
Case 4 ' Focus changed
...
Case 5 ' Idle
...
Case 6 ' **Function** key
...
End Select
End Function

Group Dialog Function

Description A *dialogfunc* implements the dynamic dialog capabilities.

Parameter **Description**

DlgItem This string value is the name of the user dialog item's *field*.

Action This numeric value indicates what action the dialog function is being asked to do.

SuppValue This numeric value provides additional information for some actions.

Action **Description**

1 Dialog box initialization. *DlgItem* is a null string. *SuppValue* is the dialog's window handle. Set *dialogfunc* = **True** to terminate the dialog.

2 **CheckBox**, **DropListBox**, **ListBox**, **MultiListBox** or **OptionGroup**: *DlgItem*'s value has changed. *SuppValue* is the new value.
CancelButton, **OKButton** or **PushButton**: *DlgItem*'s button was pushed. *SuppValue* is meaningless. Set *dialogfunc* = **True** to prevent the dialog from closing.

3 **ComboBox** or **TextBox**: *DlgItem*'s text changed and losing focus. *SuppValue* is the number of characters.

4 Item *DlgItem* is gaining focus. *SuppValue* is the item that is losing focus. (The first item is 0, second is 1, etc.)

5 Idle processing. *DlgItem* is a null string. *SuppValue* is zero. Set *dialogfunc* = **True** to continue receiving idle actions. The idle action is called as often as possible. Use

Wait .1
to reduce the number of idle calls to 10 per second.

6 Function key (F1-F24) was pressed. *DlgItem* has the focus. *SuppValue* is the function key number and the shift/control/alt key state.
Regular function keys range from 1 to 24.
Shift function keys have &H100 added.
Control function keys have &H200 added.
Alt function keys have &H400 added.
(Alt-F4 closes the dialog and is never passed to the Dialog Function.)

See Also **Begin Dialog**.

Example
```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
    PushButton 110,90,60,20,"&Hello"
  EndDialog
  Dim dlg As UserDialog
  Debug.PrintDialog(dlg)
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue?) As Boolean
  Debug.Print "Action="; Action%
  If Action% <> 1 And Action% <> 5 Then
    Debug.Print DlgItem$'; "="""; DlgText(DlgItem$); """"
  End If
  Debug.Print "SuppValue="; SuppValue?
  Select Case Action%
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
    If DlgItem$ = "Hello" Then
      MsgBox "Hello"
      DialogFunc = True 'do not exit the dialog
    EndIf
  Case 4 ' Focus changed
    Debug.Print "DlgFocus="""; DlgFocus(); """"
  Case 6 ' Function key
    If SuppValue? And &H100 Then Debug.Print "Shift-";
    If SuppValue? And &H200 Then Debug.Print "Ctrl-";
    If SuppValue? And &H400 Then Debug.Print "Alt-";
    Debug.Print "F" & (SuppValue And &HFF)
```

```
  EndSelect
End Function
```

Dialog Instruction/Function

Syntax Dialog *dialogvar*[, *default*]
-or-
Dialog(*dialogvar*[, *default*])

Group User Input

Description Display the dialog associated with *dialogvar*. The initial values of the dialog fields are provided by *dialogvar*. If the **OK button** or any **push button** is pressed then the fields in dialog are copied to the *dialogvar*. The Dialog( ) function returns a value indicating which button was pressed. (See the result table below.)

Parameter **Description**

dlgvar This variable that holds the values of the fields in a dialog. Use .*field* to access individual fields in a dialog variable.

default This numeric value indicates which button is the default button. (Pressing the Enter key on a non-button pushes the default button.) Use -2 to indicate that there is no default button. Other possible values are shown in the result table below. If this value is omitted then the first **PushButton**, **OKButton** or **CancelButton** is the default button.

Result **Description**

-1 **OK button** was pressed.

0 **Cancel button** was pressed.

>0 Nth **push button** was pressed.

See Also **Begin Dialog**.

Example

```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  Dialog dlg ' show dialog (wait for ok)
End Sub
```

Dim Definition

Syntax Dim [**WithEvents**] *vardeclaration*[, ...]

Group Declaration

Description Dimension var array(s) using the *dimension*s to establish the minimum and maximum index value for each dimension. If the *dimension*s are omitted then a scalar (single value) variable is defined. A dynamic array is declared using ( ) without any *dimension*s. It must be **ReDim**ensioned before it can be used.

See Also **Begin Dialog**, **Dialog**, **Option** Base, **Private**, **Public**, **ReDim**, **Static**, **WithEvents**.

Example
```
'#Language "WWB-COM"
Sub DoIt(Size)
  Dim C0, C1(), C2(2,3)
  ReDim C1(Size+1) ' dynamic array
  C0 = 1
  C1(0) = 2
  C2(0,0) = 3
  Debug.Print C0; C1(0); C2(0,0) ' 1 2 3
EndSub

Sub Main
  DoIt 1
End Sub
```

Dir$ Function

Syntax Dir[$]([*Pattern$*][, *AttribMask*])

Group File

Description Scan a directory for the first file matching *Pattern$*.

Parameter **Description**

Pattern$ This string value is the path and name of the file search pattern. If this is omitted then continue scanning with the previous pattern. Each *macro* has its own independent search. A path relative to the current directory can be used.

AttribMask This numeric value controls which files are found. A file with an *attribute* that matches will be found.

See Also **GetAttr( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  F$ = Dir$("*.*")
  While F$ <> ""
    Debug.Print F$
    F$ = Dir$()
  Wend
EndSub
```

DlgControlId Function

Syntax DlgControlId(*DlgItem*)

Group Dialog Function

Description Return the *field*'s window id.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter **Description**

DlgItem If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name.

Example
```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
    PushButton 110,90,60,20,"&Hello"
  EndDialog
  Dim dlg As UserDialog
  Debug.PrintDialog(dlg)
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue?) As Boolean
  Debug.Print "Action="; Action%
  Select Case Action%
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
    If DlgItem$ = "Hello" Then
      DialogFunc = True 'do not exit the dialog
    EndIf
  Case 4 ' Focus changed
    Debug.Print "DlgFocus="""; DlgFocus(); """"
    Debug.Print "DlgControlId("; DlgItem$; ")=";
    Debug.Print DlgControlId(DlgItem$)
  End Select
End Function
```

DlgCount Function

Syntax DlgCount()

Group Dialog Function

Description Return the number of dialog items in the dialog.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Example
```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
  EndDialog
  Dim dlg As UserDialog
  Dialog dlg
EndSub

Function DialogFunc(DlgItem$, Action%, SuppValue?) As Boolean
  Debug.Print "Action="; Action%
  Select Case Action%
  Case 1 ' Dialog box initialization
    Beep
    Debug.Print "DlgCount="; DlgCount() ' 3
  End Select
End Function
```

DlgEnable Instruction/Function

Syntax DlgEnable *DlgItem*[, *Enable*]
-or-
DlgEnable(*DlgItem*)

Group Dialog Function

Description Instruction: Enable or disable *DlgItem*.

Function: Return **True** if *DlgItem* is enabled.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter	Description
DlgItem	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name. **Note:** Use -1 to enable or disable all the dialog items at once.
Enable	It this numeric value is **True** then enable *DlgItem*. Otherwise, disable it. If this omitted then toggle it.

```
Example          '#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
    PushButton 110,90,60,20,"&Disable"
  EndDialog
  Dim dlg As UserDialog
  Debug.PrintDialog(dlg)
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue?) As Boolean
  Debug.Print "Action="; Action%
  Select Case Action%
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
    Select Case DlgItem$
    Case "Disable"
      DlgText DlgItem$,"&Enable"
      DlgEnable "Text",False
      DialogFunc = True 'do not exit the dialog
    Case "Enable"
      DlgText DlgItem$,"&Disable"
      DlgEnable "Text",True
      DialogFunc = True 'do not exit the dialog
    EndSelect
  End Select
EndFunction
```

DlgEnd Instruction

Syntax DlgEnd *ReturnCode*

Group Dialog Function

Description Set the return code for the **Dialog** Function and close the user dialog.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter **Description**

ReturnCode Return this numeric value.

```
Example          '#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 210,120,.DialogFunc
```

```
    Text 10,10,190,15,"Please push the Close button"
    OKButton 30,90,60,20
    CheckBox 120,90,60,20,"&Close",.CheckBox1
  EndDialog
  Dim dlg As UserDialog
  Debug.PrintDialog(dlg)
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue?) As Boolean
  Debug.Print "Action="; Action%
  Select Case Action%
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
    Select Case DlgItem$
    Case "CheckBox1"
      DlgEnd 1000
    EndSelect
  End Select
EndFunction
```

DlgFocus Instruction/Function

Syntax DlgFocus *DlgItem*
-or-
DlgFocus[$]()

Group Dialog Function

Description Instruction: Move the focus to this *DlgItem*.

Function: Return the *field* name which has the focus as a string.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter **Description**

DlgItem If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name.

Example

```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
    PushButton 110,90,60,20,"&Hello"
  EndDialog
```

```
  Dim dlg As UserDialog
  Debug.PrintDialog(dlg)
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue?) As Boolean
  Debug.Print "Action="; Action%
  Select Case Action%
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
    If DlgItem$ = "Hello" Then
      MsgBox "Hello"
      DialogFunc = True 'do not exit the dialog
    EndIf
  Case 4 ' Focus changed
    Debug.Print "DlgFocus="""; DlgFocus(); """"
  EndSelect
End Function
```

DlgListBoxArray Instruction/Function

Syntax DlgListBoxArray *DlgItem*, *StrArray$( )*
-or-
DlgListBoxArray(*DlgItem*[, *StrArray$( )*])

Group Dialog Function

Description Instruction: Set the list entries for *DlgItem*.

Function: Return the number entries in *DlgItem*'s list.

This instruction/function must be called directly or indirectly from a *dialogfunc*. The *DlgItem* should refer to a **ComboBox**, **DropListBox**, **ListBox** or **MultiListBox**.

Parameter **Description**

DlgItem If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name.

StrArray$( ) Set the list entries of *DlgItem*. This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.

Example

```
'#Language "WWB-COM"
Dim lists$()

SubMain
  ReDim lists$(0)
  lists$(0) = "List 0"
```

```
  Begin Dialog UserDialog 200,119,.DialogFunc
    Text 10,7,180,14,"Please push the OK button"
    ListBox 10,21,180,63,lists(),.list
    OKButton 30,91,40,21
    PushButton 110,91,60,21,"&Change"
  EndDialog
  Dim dlg As UserDialog
  dlg.list = 2
  Dialog dlg ' show dialog (wait for ok)
  Debug.Print dlg.list
EndSub

Function DialogFunc(DlgItem$, Action%, SuppValue?) As Boolean
  Select Case Action%
  Case 2 ' Value changing or button pressed
    If DlgItem$ = "Change" Then
      Dim N As Integer
      N = UBound(lists$)+1
      ReDim Preserve lists$(N)
      lists$(N) = "List " & N
      DlgListBoxArray "list",lists$()
      DialogFunc = True 'do not exit the dialog
    EndIf
  End Select
EndFunction
```

DlgName Function

Syntax DlgName[$](*DlgItem*)

Group Dialog Function

Description Return the *field* name of the *DlgItem* number.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter **Description**

DlgItem This numeric value is the dialog item number. The first item is 0, second is 1, etc.

Example

```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
  EndDialog
  Dim dlg As UserDialog
```

```
  Dialog dlg
EndSub

Function DialogFunc(DlgItem$, Action%, SuppValue?) As Boolean
  Debug.Print "Action="; Action%
  Select Case Action%
  Case 1 ' Dialog box initialization
    Beep
    For I = 0 To DlgCount()-1
      Debug.Print I; DlgName(I)
    Next I
  End Select
EndFunction
```

DlgNumber Function

Syntax DlgNumber(*DlgItem$*)

Group Dialog Function

Description Return the number of the *DlgItem$*. The first item is 0, second is 1, etc.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter **Description**

DlgItem$ This string value is the dialog item's *field* name.

Example

```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
  EndDialog
  Dim dlg As UserDialog
  Dialog dlg
EndSub

Function DialogFunc(DlgItem$, Action%, SuppValue?) As Boolean
  Debug.Print "Action="; Action%
  Select Case Action%
  Case 1 ' Dialog box initialization
    Beep
  Case 4 ' Focus changed
    Debug.Print DlgItem$; "="; DlgNumber(DlgItem$)
  EndSelect
End Function
```

DlgSetPicture Instruction

Syntax DlgSetPicture *DlgItem*, *FileName*, *Type*

Group Dialog Function

Description Instruction: Set the file name for *DlgItem*.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter **Description**

DlgItem If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name.

FileName Set the file name of *DlgItem* to this string value.

Type This numeric value indicates the type of bitmap used. See below.

Type **Effect**

0 *FileName* is the name of the bitmap file. If the file does not exist then "(missing picture)" is displayed.

3 The clipboard's bitmap is displayed. If the clipboard does not contain a bitmap then "(missing picture)" is displayed.

16 Same a 0, but instead of displaying "(missing picture)" a run-time error occurs.

19 Same a 3, but instead of displaying "(missing picture)" a run-time error occurs.

Example

```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Picture 10,10,180,75,"",0,.Picture
    OKButton 30,90,60,20
    PushButton 110,90,60,20,"&View"
  EndDialog
  Dim dlg As UserDialog
  Debug.PrintDialog(dlg)
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue?) As Boolean
  Debug.Print "Action="; Action%
  Select Case Action%
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
    Select Case DlgItem$
    Case "View"
      FileName = GetFilePath("Bitmap","BMP")
      DlgSetPicture "Picture",FileName,0
```

```
      DialogFunc = True 'do not exit the dialog
    EndSelect
  End Select
EndFunction
```

DlgText Instruction/Function

Syntax DlgText *DlgItem*, *Text*
-or-
DlgText[$](*DlgItem*)

Group Dialog Function

Description Instruction: Set the text for *DlgItem*.

Function: Return the text from *DlgItem*.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter **Description**

DlgItem If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name. **Note:** Use -1 to access the dialog's title.

Text Set the text of *DlgItem* to this string value.

Example

```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
    PushButton 110,90,60,20,"&Now"
  End Dialog
  Dim dlg As UserDialog
  Debug.PrintDialog(dlg)
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue?) As Boolean
  Debug.Print "Action="; Action%
  Select Case Action%
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
    Select Case DlgItem$
    Case "Now"
      DlgText "Text",CStr(Now)
      DialogFunc = True 'do not exit the dialog
```

```
    EndSelect
  End Select
EndFunction
```

DlgType Function

Syntax DlgType[$](*DlgItem*)

Group Dialog Function

Description Return a string value indicating the type of the *DlgItem*. One of: "**CancelButton**", "**CheckBox**", "**ComboBox**", "**DropListBox**", "**GroupBox**", "**ListBox**", "**MultiListBox**", "**OKButton**", "**OptionButton**", "**OptionGroup**", "**PushButton**", "**Text**", "**TextBox**".

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter **Description**

DlgItem If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name.

Example

```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
  EndDialog
  Dim dlg As UserDialog
  Dialog dlg
EndSub

Function DialogFunc(DlgItem$, Action%, SuppValue?) As Boolean
  Debug.Print "Action="; Action%
  Select Case Action%
  Case 1 ' Dialog box initialization
    Beep
    For I = 0 To DlgCount()-1
      Debug.Print I; DlgType(I)
    Next I
  End Select
EndFunction
```

DlgValue Instruction/Function

Syntax DlgValue *DlgItem*, *Value*
-or-
DlgValue(*DlgItem*)

Group Dialog Function

Description Instruction: Set the numeric value(s) *DlgItem*.

Function: Return the numeric value(s) for *DlgItem*. (A MultiListBox user dialog item returns an array.)

This instruction/function must be called directly or indirectly from a *dialogfunc*. The *DlgItem* should refer to a **CheckBox**, **ComboBox**, **DropListBox**, **ListBox**, **MultiListBox** or **OptionGroup**.

Parameter **Description**

DlgItem If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name.

Value Set the text of *DlgItem* to this numeric value. (A MultiListBox user dialog item uses an array.)

Example
```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 150,147,.DialogFunc
    GroupBox 10,7,130,77,"Direction",.Field1
    PushButton 100,28,30,21,"&Up"
    PushButton 100,56,30,21,"&Dn"
    OptionGroup .Direction
      OptionButton 20,21,80,14,"&North",.North
      OptionButton 20,35,80,14,"&South",.South
      OptionButton 20,49,80,14,"&East",.East
      OptionButton 20,63,80,14,"&West",.West
    OKButton 10,91,130,21
    CancelButton 10,119,130,21
  EndDialog
  Dim dlg As UserDialog
  Dialog dlg
  MsgBox "Direction=" & dlg.Direction
EndSub

Function DialogFunc(DlgItem$, Action%, SuppValue?) As Boolean
  Select Case Action%
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
    Select Case DlgItem$
    Case "Up"
```

```
      DlgValue "Direction",0
      DialogFunc = True 'do not exit the dialog
    Case "Dn"
      DlgValue "Direction",1
      DialogFunc = True 'do not exit the dialog
    EndSelect
  End Select
EndFunction
```

DlgVisible Instruction/Function

Syntax DlgVisible *DlgItem*[, *Visible*]
-or-
DlgVisible(*DlgItem*)

Group Dialog Function

Description Instruction: Show or hide *DlgItem*.

Function: Return **True** if *DlgItem* is visible.

This instruction/function must be called directly or indirectly from a *dialogfunc*.

Parameter **Description**

DlgItem If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's *field* name.

Enable It this numeric value is **True** then show *DlgItem*. Otherwise, hide it. If this omitted then toggle it.

Example

```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120,.DialogFunc
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,40,180,15,.Text
    OKButton 30,90,60,20
    PushButton 110,90,60,20,"&Hide"
  EndDialog
  Dim dlg As UserDialog
  Debug.PrintDialog(dlg)
End Sub

Function DialogFunc(DlgItem$, Action%, SuppValue?) As Boolean
  Debug.Print "Action="; Action%
  Select Case Action%
  Case 1 ' Dialog box initialization
    Beep
  Case 2 ' Value changing or button pressed
```

```
    Select Case DlgItem$
    Case "Hide"
      DlgText DlgItem$,"&Show"
      DlgVisible "Text",False
      DialogFunc = True 'do not exit the dialog
    Case "Show"
      DlgText DlgItem$,"&Hide"
      DlgVisible "Text",True
      DialogFunc = True 'do not exit the dialog
    EndSelect
  End Select
EndFunction
```

DoEvents Instruction

Syntax DoEvents

Group Miscellaneous

Description This instruction allows other applications to process events.

Example

```
'#Language "WWB-COM"
Sub Main
  DoEvents ' let other apps work
End Sub
```

Do Statement

Syntax

```
Do
  statements
Loop
-or-
Do {Until|While} condexpr
  statements
Loop
-or-
Do
  statements
Loop {Until|While} condexpr
```

Group Flow Control

Description Form 1: Do *statements* forever. The loop can be exited by using **Exit** or **Goto**.

Form 2: Check for loop termination before executing the loop the first time.

Form 3: Execute the loop once and then check for loop termination.

Loop Termination:

- Until *condexpr*: Do *statement*s until *condexpr* is **True**.
- While *condexpr*: Do *statement*s while *condexpr* is **True**.

See Also **For**, **For Each**, **Exit** Do, **While**.

Example
```
'#Language "WWB-COM"
Sub Main
  I = 2
  Do
    I = I*2
  Loop Until I > 10
  Debug.Print I ' 16
End Sub
```

Double Data Type

Syntax **Dim** v As Double

Group Data Type

Description A 64 bit real value.

DropListBox Dialog Item Definition

Syntax DropListBox *X*, *Y*, *DX*, *DY*, *StrArray$( )*, .*Field*[, *Options*]

Group User Dialog

Description Define a drop-down listbox item.

Parameter **Description**

X This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.

Y This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.

DX This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.

DY This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.

StrArray$( ) This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.

Field The value of the drop-down list box is accessed via this field. It is the index of the *StrArray$( )* var.

Options This numeric value controls the type of drop-down list box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option **Description**

0 Text box is not editable and list is not sorted.

1 Text box is editable and list is not sorted.

2 Text box is not editable and list is sorted.

3 Text box is editable and list is sorted.

See Also **Begin Dialog**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim lists$(3)
  lists$(0) = "List 0"
  lists$(1) = "List 1"
  lists$(2) = "List 2"
  lists$(3) = "List 3"
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    DropListBox 10,25,180,60,lists$(),.list1
    DropListBox 10,50,180,60,lists$(),.list2,1
    OKButton 80,90,40,20
  EndDialog
  Dim dlg As UserDialog
  dlg.list1 = 2      ' list1 is a numeric field
  dlg.list2 = "xxx"   ' list2 is a string field
  Dialog dlg ' show dialog (wait for ok)
  Debug.Print lists$(dlg.list1)
  Debug.Print dlg.list2
End Sub
```

Empty Keyword

Group Constant

Description A *variantvar* that does not have any value.

Encode64 Function

Syntax Encode64[$](*Data*)
-or-
Encode64B[$](*Data*)

Group Miscellaneous

Description Return a string using the base 64 encoding algorithm.

Parameter **Description**

Data Return this string's base 64 encoding.

See Also **Decode64**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Encode64("Hello World!!") '"SGVsbG8gV29ybGQhIQ=="
End Sub
```

Encrypt64 Function

Syntax Encrypt64[$](*Data* [,*Password*])

Group Miscellaneous

Description Return a string using the RC4 stream encryption algorithm. (The string is also encoded using base 64 encoding.)

Parameter **Description**

Data Return this string's encryption.

Password Encrypt using this password.

See Also **Decode64**, **Decrypt64**, **Encode64**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Encrypt64("Hello World!!", "abc") '"Y4GFrF+k1YUHwjEzsg=="
End Sub
```

End Instruction

Syntax End

Group Flow Control

Description The end instruction causes the *macro* to terminate immediately. If the macro was run by another macro using the **MacroRun** instruction then that macro continues on the instruction following the **MacroRun**.

Example
```
'#Language "WWB-COM"
Sub DoSub
  L$ = UCase$(InputBox$("Enter End:"))
  If L$ = "END" Then End
  Debug.Print "End was not entered."
End Sub
```

```
Sub Main
  Debug.Print "Before DoSub"
  DoSub
  Debug.Print "After DoSub"
End Sub
```

Enum Definition

Syntax [| **Private** | **Public**] _
Enum *name*
  elem [= *value*]
 [...]
End Enum

Group Declaration

Description Define a new *user enum*. Each *elem* defines an element of the enum. If *value* is given then that is the element's value. The value can be any constant integer expression. If *value* is omitted then the element's value is one more than the previous element's value. If there is no previous element then zero is used.

Access If no access is specified then **Public** is assumed.

Example

```
'#Language "WWB-COM"
Enum Days
  Monday
  Tuesday
  Wednesday
  Thursday
  Friday
  Saturday
  Sunday
End Enum

SubMain
  Dim D As Days
  For D = Monday To Friday
    Debug.Print D ' 0 through 4
  Next D
End Sub
```

Environ Function

Syntax Environ[$](*Index*)
-or-
Environ[$](*Name*)

Group Miscellaneous

Description Return an environment string.

Pocket PC Not supported.

Parameter **Description**

Index Return this environment string's value. If there is no environment string at this index a null string is returned. Indexes start at one.

Name Return this environment string's value. If the environment string can't be found a null string is returned.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Environ("Path")
EndSub
```

EOF Function

Syntax EOF(*StreamNum*)

Group File

Description Return **True** if *StreamNum* is at the end of the file.

Parameter **Description**

StreamNum Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

Example
```
'#Language "WWB-COM"
Sub Main
  Open "XXX" For Input As #1
   While Not EOF(1)
     Line Input #1, L
     Debug.Print L
  Wend
  'Close #1
End Sub
```

Erase Instruction

Syntax Erase *arrayvar*[, ...]
-or-
Erase *usertypevar.elem*[, ...]

Group Assignment

Description Reset *arrayvar* or *user defined type* array element to zero. (Dynamic arrays are reset to undimensioned arrays.) String arrays values are set to a null string. *arrayvar* must be declared as an array.

- Declare with **Dim**, **Private**, **Public** or **Static**.
- Declare as a parameter of **Sub**, **Function** or **Property** definition.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim X%(2)
  X%(1) = 1
  Erase X%
  Debug.Print X%(1) ' 0
EndSub
```

Err Object

Syntax Err

Group Error Handling

Description Set Err to zero to clear the last error event. Err in an expression returns the last error code. Add vbObjectError to your error number in ActiveX Automation objects. Use Err.Raise or **Error** to trigger an error event.

Err[.Number]

This is the error code for the last error event. Set it to zero (or use Err.Clear) to clear the last error condition. Use **Error** or Err.Raise to trigger an error event. This is the default property.

Err.Description

This string is the description of the last error event.

Err.Source

This string is the error source file name of the last error event.

Err.HelpFile

This string is the help file name of the last error event.

Err.HelpContext

This number is the help context id of the last error event.

Err.Clear

Clear the last error event.

Err.Raise [Number:=]*errorcode* _
[, [Source:=]*source*] _
[, [Description:=]*errordesc*] _
[, [HelpFile:=]*helpfile*] _
[, [HelpContext:=]*context*]

Raise an error event.

Err.LastDLLError

Returns the error code for the last DLL call (see **Declare**).

Example
```
'#Language "WWB-COM"
Sub Main
  On Error GoTo Problem
  Err = 1 ' set to error #1 (handler not triggered)
  ExitSub

  Problem: ' error handler
  Error Err ' halt macro with message
End Sub
```

Error Instruction/Function

Syntax Error *ErrorCode*
-or-
Error[$]([*ErrorCode*])

Group Error Handling

Description Instruction: Signal error *ErrorCode*. This triggers error handling just like a real error. The current *procedure*'s error handler is activated, unless it is already active or there isn't one. In that case the calling *procedure*'s error handler is tried. (Use **Err**.Raise to provide complete error information.)

Function: The Error( ) function returns the error text string.

Parameter	**Description**
ErrorCode	This is the error number.

Example
```
'#Language "WWB-COM"
Sub Main
  On Error GoTo Problem
  Error 1 ' simulate error #1
  Exit Sub
```

```
Problem: ' error handler
  Debug.Print "Err.Description="; Err.Description
  Resume Next
EndSub
```

Eval Function

Syntax Eval(*Expr*[, *Depth*])

Group Miscellaneous

Description Return the value of the string expression as evaluated.

Parameter **Description**

Expr Evaluate this string value.

Depth This integer value indicates how deep into the stack to locate the local variables. If Depth = 0 then use the current *procedure*. If this value is omitted then the depth is 0.

See Also **Assign**.

Example

```
'#Language "WWB-COM"
Sub Main
  Dim X As String
  X = "Hello"
  Debug.Print Eval("X") 'Hello
  A
EndSub
Sub A
  Dim X As String
  X = "Bye"
  Debug.Print Eval("X") 'Bye
  Debug.Print Eval("X",1) 'Hello
EndSub
```

Event Definition

Syntax [| **Public**] _
Event *name*[([*param*[, ...]])]

Group Declaration

Description User defined event. The event defines a sub that can be defined using **WithEvents**.. The values of the calling *arglist* are assigned to the *param*s.

Access If no access is specified then **Public** is assumed.

See Also **RaiseEvent**.

Example
```
' Class1
'#Language "WWB-COM"
Event Changing(ByVal OldValue As String, ByVal NewValue As String)

Private Value_ As String

Property Get Value As String
  Value = Value_
EndProperty

Property Let Value(ByVal NewValue As String)
  RaiseEvent Changing(Value_, NewValue)
  Value_ = NewValue
EndProperty

'#Uses "Class1.cls"

Dim WithEvents c1 As Class1

Sub Main
  Set c1 = New Class1
  c1.Value = "Hello"
  c1.Value = "Goodbye"
End Sub

Sub c1_Changing(ByVal OldValue As String, ByVal NewValue As String)
  Debug.Print "OldValue=""" & OldValue & """, NewValue=""" & NewValue & """"
EndSub
```

Exit Instruction

Syntax Exit {All|Do|For|Function|Property|Sub|While}

Group Flow Control

Description The exit instruction causes the *macro* to continue with out doing some or all of the remaining instructions.

Exit	**Description**
All	Exit all *macro*s.
Do	Exit the **Do** loop.
For	Exit the **For** of **For Each** loop.
Function	Exit the **Function** block. Note: This instruction clears the **Err** and sets **Error`$'** to null.
Property	Exit the **Property** block. Note: This instruction clears the **Err** and sets **Error`$'** to null.

Sub Exit the **Sub** block. Note: This instruction clears the **Err** and sets **Error`$'** to null.

While Exit the **While** loop.

```
Example          '#Language "WWB-COM"
Sub Main
  L$ = InputBox$("Enter Do, For, While, Sub or All:")
  Debug.Print "Before DoSub"
  DoSub UCase$(L$)
  Debug.Print "After DoSub"
EndSub

Sub DoSub(L$)
  Do
    If L$ = "DO" Then Exit Do
    I = I+1
  Loop While I < 10
  If I = 0 Then Debug.Print "Do was entered"

  For I = 1 To 10
    If L$ = "FOR" Then Exit For
  Next I
  If I = 1 Then Debug.Print "For was entered"

  I = 10
  While I > 0
    If L$ = "WHILE" Then Exit While
    I = I-1
  Wend
  If I = 10 Then Debug.Print "While was entered"

  If L$ = "SUB" Then Exit Sub
  Debug.Print "Sub was not entered."
  If L$ = "ALL" Then Exit All
  Debug.Print "All was not entered."
End Sub
```

Exp Function

Syntax Exp(*Num*)

Group Math

Description Return the exponential.

Parameter **Description**

Num Return e raised to the power of this numeric value. The value e is approximately 2.718282.

See Also **Log**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Exp(1) ' 2.718281828459
EndSub
```

False Keyword

Group Constant

Description A *condexpr* is false when its value is zero. A function that returns False returns the value 0.

FileAttr Function

Syntax FileAttr(*StreamNum*, *ReturnValue*)

Group File

Description Return *StreamNum*'s open mode or file handle.

Parameter **Description**

StreamNum Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

ReturnValue 1 - return the mode used to open the file: 1=Input, 2=Output, 4=Random, 8=Append, 32=Binary
2 - return the file handle

See Also **Open**.

Example
```
'#Language "WWB-COM"
Sub Main
  Open "XXX" For Output As #1
  Debug.Print FileAttr(1,1) ' 2
 'Close #1
EndSub
```

FileCopy Instruction

Syntax FileCopy *FromName$*, *ToName$*

Group File

Description Copy a file.

Parameter **Description**

FromName$ This string value is the path and name of the source file. A path relative to the current directory can be used.

ToName$ This string value is the path and name of the destination file. A path relative to the current directory can be used.

Example
```
'#Language "WWB-COM"
Sub Main
  FileCopy "C:\AUTOEXEC.BAT","C:\AUTOEXEC.BAK"
End Sub
```

FileDateTime Function

Syntax FileDateTime(*Name$*)

Group File

Description Return the date and time file *Name$* was last changed as a **Date** value. If the file does not exist then a run-time error occurs.

Parameter **Description**

Name$ This string value is the path and name of the file. A path relative to the current directory can be used.

Example
```
'#Language "WWB-COM"
Sub Main
  F$ = Dir$("*.*")
  While F$ <> ""
    Debug.Print F$; " "; FileDateTime(F$)
    F$ = Dir$()
  Wend
EndSub
```

FileLen Function

Syntax FileLen(*Name$*)

Group File

Description Return the length of file *Name$*. If the file does not exist then a run-time error occurs.

Parameter **Description**

Name$ This string value is the path and name of the file. A path relative to the current directory can be used.

Example
```
'#Language "WWB-COM"
Sub Main
  F$ = Dir$("*.*")
  While F$ <> ""
    Debug.Print F$; " "; FileLen(F$)
    F$ = Dir$()
```

```
  Wend
EndSub
```

Fix Function

Syntax Fix(*Num*)

Group Math

Description Return the integer value.

Parameter **Description**

Num Return the integer portion of this numeric value. The number is truncated. Positive numbers return the next lower integer. Negative numbers return the next higher integer. If this value is **Null** then **Null** is returned.

See Also **Int**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Fix(9.9)  ' 9
  Debug.Print Fix(0)    ' 0
  Debug.Print Fix(-9.9) '-9
End Sub
```

For Each Statement

Syntax
```
For Each var In items
  statements
Next [var]
```

Group Flow Control

Description Execute *statements* for each item in *items*.

Parameter **Description**

var This is the iteration variable.

items This is the collection of items to be done.

See Also **Do**, **For**, **Exit** For, **While**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim Document As Object
  For Each Document In App.Documents
    Debug.Print Document.Title
  Next Document
End Sub
```

Format$ Function

Syntax Format[$](*expr*[, *form$*], [*firstday*], _
[*firstweek*])

Group String

Description Return the formatted string representation of *expr*.

Parameter **Description**

expr Return the formatted string representation of this numeric value.

form Format *expr* using to this string value. If this is omitted then return the *expr* as a string.

firstday Format using this day as the first day of the week. If this is omitted then the vbSunday is used.

firstweek Format using this week as the first week of the year. If this is omitted then the vbFirstJan1 is used.

firstday **Value Description**

vbUseSystemFirstDay 0 Use the systems first day of the week.

vbSunday 1 Sunday (default)

vbMonday 2 Monday

vbTuesday 3 Tuesday

vbWednesday 4 Wednesday

vbThursday 5 Thursday

vbFriday 6 Friday

vbSaturday 7 Saturday

firstweek **Value Description**

vbUseSystem 0 Use the systems first week of the year.

vbFirstJan1 1 The week that January 1 occurs in. This is the default value.

2 vbFirstFourDays The first week that has at least four days in the year.

3 vbFirstFullWeek The first week that entirely in the year.

See Also **Predefined Date Format**, **Predefined Number Format**, **User defined Date Format**, **User defined Number Format**, **User defined Text Format**.

Format Predefined Date

Description The following predefined date formats may be used with the **Format** function. Predefined formats may not be combined with user defined formats or other predefined formats.

Form	Description
General Date	Same as **user defined date format** "c"
Long Date	Same as **user defined date format** "dddddd"
Medium Date	**Not supported at this time.**
Short Date	Same as **user defined date format** "ddddd"
Long Time	Same as **user defined date format** "ttttt"
Medium Time	Same as **user defined date format** "hh:mm AMPM"
Short Time	Same as **user defined date format** "hh:mm"

Format Predefined Number

Description The following predefined number formats may be used with the **Format** function. Predefined formats may not be combined with user defined formats or other predefined formats.

Form	Description
General Number	Return number as is.
Currency	Same as **user defined number format** "$#,##0.00;($#,##0.00)" **Not locale dependent at this time.**
Fixed	Same as **user defined number format** "0.00".
Standard	Same as **user defined number format** "#,##0.00".
Percent	Same as **user defined number format** "0.00%".
Scientific	Same as **user defined number format** "0.00E+00".
Yes/No	Return "No" if zero, else return "Yes".
True/False	Return "True" if zero, else return "False".
On/Off	Return "On" if zero, else return "Off".

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Format$(2.145,"Standard") ' 2.15
EndSub
```

Format User Defined Date

Description The following date formats may be used with the **Format** function. Date formats may be combined to create the user defined date format. User defined date formats may not be combined with other user defined formats or predefined formats.

Parameter Description

: insert localized time separator

/ insert localized date separator

c insert ddddd ttttt, insert date only if t=0, insert time only if d=0

d insert day number without leading zero

dd insert day number with leading zero

ddd insert abbreviated day name

dddd insert full day name

ddddd insert date according to Short Date format

dddddd insert date according to Long Date format

w insert day of week number

ww insert week of year number

m insert month number without leading zero
insert minute number without leading zero (if follows h or hh)

mm insert month number with leading zero
insert minute number with leading zero (if follows h or hh)

mmm insert abbreviated month name

mmmm insert full month name

q insert quarter number

y insert day of year number

yy insert year number (two digits)

yyyy insert year number (four digits, no leading zeros)

h insert hour number without leading zero

hh insert hour number with leading zero

n insert minute number without leading zero

nn insert minute number with leading zero

s insert second number without leading zero

ss insert second number with leading zero

ttttt insert time according to time format

AM/PM use 12 hour clock and insert AM (hours 0 to 11) and PM (12 to 23)

am/pm use 12 hour clock and insert am (hours 0 to 11) and pm (12 to 23)

A/P use 12 hour clock and insert A (hours 0 to 11) and P (12 to 23)

a/p use 12 hour clock and insert a (hours 0 to 11) and p (12 to 23)

AMPM use 12 hour clock and insert localized AM/PM strings

\c insert character c

"text" insert literal text

Example

Format User Defined Number

Description The following number formats may be used with the **Format** function. Number formats may be combined to create the user defined number format. User defined number formats may not be combined with other user defined formats or predefined formats.

User defined number formats can contain up to four sections separated by ';':

- form - format for non-negative expr, '-'format for negative expr, empty and null expr return ""
- form;negform - negform: format for negative expr
- form;negform;zeroform - zeroform: format for zero expr
- form;negform;zeroform;nullform - nullform: format for null expr

Parameter **Description**

digit, don't include leading/trailing zero digits (all the digits left of decimal point are returned)
eg. Format(19,"###") returns "19"
eg. Format(19,"#") returns "19"

0 digit, include leading/trailing zero digits
eg. Format(19,"000") returns "019"
eg. Format(19,"0") returns "19"

. decimal, insert localized decimal point
eg. Format(19.9,"###.00") returns "19.90"
eg. Format(19.9,"###.##") returns "19.9"

, thousands, insert localized thousand separator every 3 digits
"xxx," or "xxx,." mean divide expr by 1000 prior to formatting
two adjacent commas ",," means divide expr by 1000 again
eg. Format(1900000,"0,,") returns "2"
eg. Format(1900000,"0,,.0") returns "1.9"

% percent, insert %, multiply expr by 100 prior to formatting

: insert localized time separator

/ insert localized date separator

E+ e+ E- e- use exponential notation, insert E (or e) and the signed exponent
eg. Format(1000,"0.00E+00") returns "1.00E+03"
eg. Format(.001,"0.00E+00") returns "1.00E-03"

- + $ ( ) space insert literal char
eg. Format(10,"$#") returns "$10"

\c insert character c
eg. Format(19,"\####\#") returns "#19#"

"text" insert literal text
eg. Format(19,"""##""###""##""") returns "##19##"

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print Format**$(2.145,"#.00") ' 2.15
EndSub

Format User Defined Text

Description The following text formats may be used with the **Format** function. Text formats may be combined to create the user defined text format. User defined text formats may not be combined with other user defined formats or predefined formats.

User defined text formats can contain one or two sections separated by ';':

- form - format for all strings
- form;nullform - nullform: format for empty and null strings

Parameter **Description**

@ char placeholder, insert char or space

& char placeholder, insert char or nothing

< all chars lowercase

> all chars uppercase

! fill placeholder from left-to-right (default is right-to-left)

\c insert character c

"text" insert literal text

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print Format**("123","ab@c") '" ab1c23"
  Debug.**PrintFormat**("123","!ab@c") '" ab3c"
End Sub

For Statement

Syntax For *Num* = *First* To *Last* [Step *Inc*]
statements
Next [*Num*]

Group Flow Control

Description Execute *statements* while *Num* is in the range *First* to *Last*.

Parameter **Description**

Num This is the iteration variable.

First Set *Num* to this value initially.

Last Continue looping while *Num* is in the range. See *Step* below.

Step If this numeric value is greater than zero then the for loop continues as long as *Num* is less than or equal to *Last*. If this numeric value is less than zero then the for loop continues as long as *Num* is greater than or equal to *Last*. If this is omitted then one is used.

See Also **Do**, **For Each**, **Exit** For, **While**.

Example
```
'#Language "WWB-COM"
Sub Main
  For I = 1 To 2000 Step 100
    Debug.Print I; I+I; I*I
  Next I
End Sub
```

FreeFile Function

Syntax FreeFile[( )]

Group File

Description Return the next unused shared stream number (greater than or equal to 256). Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print FreeFile ' 256
  FN = FreeFile
  Open "XXX" For Output As #FN
  Debug.Print FreeFile ' 257
 'Close #FN
  Debug.Print FreeFile ' 256
EndSub
```

Friend Keyword

Group Declaration

Description Friend **Function**s, **Property**s and **Sub**s in a *module* are available in all other *macro*s/modules that access it. Friends are not accessible via **Object** variables.

Function Definition

Syntax [| **Private** | **Public** | **Friend**] _
[Default] _
Function *name*[*type*][([*param*[, ...]])] _
  [*As type*[()]]
  *statement*s
End Function

Group Declaration

Description User defined function. The function defines a set of *statement*s to be executed when it is called. The values of the calling *arglist* are assigned to the *param*s. Assigning to *name*[*type*] sets the value of the function result.

Access If no access is specified then **Public** is assumed.

See Also **Declare**, **Property**, **Sub**.

Example
```
'#Language "WWB-COM"
Function Power(X,Y)
  P = 1
  For I = 1 To Y
    P = P*X
  Next I
  Power = P
End Function

SubMain
  Debug.Print Power(2,8) ' 256
End Sub
```

GetAllSettings Function

Syntax GetAllSettings(*AppName$*, *Section$*)

Group Settings

Description Get all of *Section*'s settings in project *AppName*. Settings are returned in a **Variant**. **Empty** is returned if there are no keys in the section. Otherwise, the Variant contains a two dimension

array: (I,0) is the key and (I,1) is the setting. Win16 and Win32s store settings in a .ini file named *AppName*. Win32/Win64 store settings in the registration database under "HKEY_CURRENT_USER\ Software\ VB and VBA Program Settings\ *AppName*\ *Section*\ *Key*". If *AppName* starts with "..\" then "VB and VBA Program Settings\" is omitted.

Parameter **Description**

AppName$ This string value is the name of the project which has this *Section* and *Key*.

Section$ This string value is the name of the section of the project settings.

Example
```
'#Language "WWB-COM"
Sub Main
  SaveSetting "MyApp","Font","Size",10
  SaveSetting "MyApp","Font","Name","Courier"
  Settings = GetAllSettings("MyApp","Font")
  For I = LBound(Settings) To UBound(Settings)
    Debug.Print Settings(I,0); "="; Settings(I,1)
  Next I
  DeleteSetting "MyApp","Font"
End Sub
```

GetAttr Function

Syntax GetAttr(*Name$*)

Group File

Description Return the *attribute*s for file *Name$*. If the file does not exist then a run-time error occurs.

Parameter **Description**

Name$ This string value is the path and name of the file. A path relative to the current directory can be used.

Example
```
'#Language "WWB-COM"
Sub Main
  F$ = Dir$("*.*")
  While F$ <> ""
    Debug.Print F$; " "; GetAttr(F$)
    F$ = Dir$()
  Wend
EndSub
```

GetConnection

This is a specialized method similar to the EmbeddedProc object that could be used for getting the connection object to Microsoft Dynamics. The Dynamics client must be installed on the same machine. Use the ERP.BeginTrans and ERP.CommitTrans before and after the use of this method. This exposes the business functions but the programmer must know how to use them.

Syntax: oValue = SYS.GetConnection(vSource)

oValue (Object) An object declaredConn as Axapta3 or Object, if you choose to use late binding. It will return the ADO, RDO, OLEDB, ERP, or Web connection object

vSource (Variant) data source name (DSN) or the data source number

Example:

```
Dim oConn As Axapta3

Dim oRecord As IAxaptaRecord


Set oConn = SYS.GetConnection("Dynamics")


Set oRecord = oConn.CreateRecord("INVENTJOURNALTABLE")

If oRecord Is Nothing Then

  App.MsgBox "Error: Axapta Record object failed."

  Exit Sub

End If

App.MsgBox "Record company: " & oRecord.company
```

GetFilePath$ Function

Syntax GetFilePath[$]([*DefName$*], [*DefExt$*], [*DefDir$*], _
[*Title$*], [*Option*])

Group User Input

Description Put up a dialog box and get a file path from the user. The returned string is a complete path and file name. If the cancel button is pressed then a null string is returned.

Parameter **Description**

DefName$* Set the initial File Name in the to this string value. If this is omitted then *.*DefExt$ is used.

DefExt$ Initially show files whose extension matches this string value. (Multiple extensions can be specified by using ";" as the separator.) If this is omitted then * is used.

A "filter" may be specified using "description|*.ext|". Multiple descriptions can be used by repeating this format. (e.g. "Bitmap files|*.bmp|PNG files|*.png|JPEG files|*.jpg")

DefDir$ This string value is the initial directory. If this is omitted then the current directory is used.

Title$ This string value is the title of the dialog. If this is omitted then "Get File Path" is used.

Option This numeric value determines the file selection options. If this is omitted then zero is used. See table below.

Option **Effect**

0 Only allow the user to select a file that exists.

1 Confirm creation when the user selects a file that does not exist.

2 Allow the user to select any file whether it exists or not.

3 Confirm overwrite when the user selects a file that exists.

+4 Selecting a different directory changes the application's current directory.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print GetFilePath$()
EndSub
```

Get Instruction

Syntax Get [#]*StreamNum*, [*RecordNum*], *var*

Group File

Description Get a variable's value from *StreamNum*.

Parameter **Description**

StreamNum Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

RecordNum For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1. If this is omitted then the current position (or record number) is used.

var This variable value is read from the file. For a fixed length variable (like **Long**) the number of bytes required to restore the variable are read. For a **Variant** variable two bytes are read which describe its type and then the variable value is read accordingly. For a *user type* variable each field is read in sequence. For an array variable each element is read in sequence. For a dynamic array variable the number of dimensions and range of each dimension is read prior to reading the array values. All binary data values are read from the file in *little-endian* format.

Note: When reading a string (or a dynamic array) from a Binary mode file the length (or array dimension) information is not read. The current string length determines how much string data is read. The current array dimension determines how may array elements are read.

See Also **Open**, **Put**.

Example '#Language "WWB-COM"
Sub Main
  Dim V As **Variant**
  Open "SAVE_V.DAT" **For** Binary Access Read As #1
  Get #1, , V
 '**Close** #1
End Sub

GetLocale Function

Syntax GetLocale

Group Miscellaneous

Description Get the locale ID for the current thread.

See Also **SetLocale**.

Example '#Language "WWB-COM"
Sub Main
  SetLocale &H409 ' English, US
  Debug.**PrintHex**(GetLocale) '"409"
End Sub

GetObject Function

Syntax GetObject([*File$*][, *Class$*])

Group Object

Description Get an existing object of type *Class$* from *File$*. Use **Set** to assign the returned object to an object variable.

Pocket PC Not supported.

Parameter **Description**

File$* This is the file where the object resides. If this is omitted then the currently active object for *Class$ is returned.

Class$ This string value is the application's registered class name. If this application is not currently active it will be started. If this is omitted then the application associated with the file's extension will be started.

Example '#Language "WWB-COM"
Sub Main
  Dim App As **Object**
  Set App = GetObject(,"WinWrap.CppDemoApplication")
  App.Move 20,30 ' move icon to 20,30
  Set App = **Nothing**

```
  App.Quit     ' run-time error (no object)
End Sub
```

GetSetting Function

Syntax GetSetting[$](*AppName$*, *Section$*, *Key$*[, *Default$*])

Group Settings

Description Get the setting for *Key* in *Section* in project *AppName*. Win16 and Win32s store settings in a .ini file named *AppName*. Win32/Win64 store settings in the registration database under "HKEY_CURRENT_USER\ Software\ VB and VBA Program Settings\ *AppName*\ *Section*\ *Key*". If *AppName* starts with "..\" then "VB and VBA Program Settings\" is omitted.

Parameter **Description**

AppName$ This string value is the name of the project which has this *Section* and *Key*.

Section$ This string value is the name of the section of the project settings.

Key$ This string value is the name of the key in the section of the project settings.

Default$ Return this string value if no setting has been saved. If this is omitted then a null string is used.

Example

```
'#Language "WWB-COM"
Sub Main
  SaveSetting "MyApp","Font","Size",10
  Debug.Print GetSetting("MyApp","Font","Size") ' 10
End Sub
```

Goto Instruction

Syntax GoTo *label*

Group Flow Control

Description Go to the *label* and continue execution from there. Only *label*s in the current user defined *procedure* are accessible.

Example

```
'#Language "WWB-COM"
Sub Main
  X = 2
Loop:
  X = X*X
  If X < 100 Then GoTo Loop
  Debug.Print X ' 256
End Sub
```

GroupBox Dialog Item Definition

Syntax GroupBox *X*, *Y*, *DX*, *DY*, *Title$*[, .*Field*]

Group User Dialog

Description Define a groupbox item.

Parameter **Description**

X This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.

Y This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.

DX This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.

DY This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.

Title$ This string value is the title of the group box.

Field This identifier is the name of the field. The *dialogfunc* receives this name as *string*. If this identifer is omitted then the first two words of the title are used.

See Also **Begin Dialog**.

Example
```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    GroupBox 10,25,180,60,"Group box"
    OKButton 80,90,40,20
    End Dialog
  Dim dlg As UserDialog
  Dialog dlg ' show dialog (wait for ok)
End Sub
```

Hex$ Function

Syntax Hex[$](*Num*)

Group String

Description Return a hex string.

Parameter **Description**

Num Return a hex encoded string for this numeric value.

See Also **Oct$( )**, **Str$( )**, **Val( )**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** Hex$(15) 'F
End Sub

Hour Function

Syntax Hour(*dateexpr*)

Group Time/Date

Description Return the hour of the day (0 to 23).

Parameter **Description**

dateexpr Return the hour of the day for this date value. If this value is **Null** then **Null** is returned.

See Also **Minute( )**, **Second( )**, **Time( )**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** Hour(#12:00:01 AM#) ' 0
EndSub

Huge_ Data Type

Syntax **Dim** v As Huge_

Group Data Type

Description A 64 bit signed integer value.

Some versions of Windows do not support Huge_.

VBA This language element is not VBA compatible.

If Statement

Syntax If *condexpr* Then [*instruction*] [Else *instruction*]
-or-
If *condexpr* Then
  *statement*s
[ElseIf *condexpr* Then
  *statement*s]...
[Else
  *statement*s]
End If

Group Flow Control

Description Form 1: Single line if statement. Execute the *instruction* following the Then if *condexpr* is **True**. Otherwise, execute the *instruction* following the Else. The Else portion is optional.

Form 2: The multiple line if is useful for complex ifs. Each if *condexpr* is checked in turn. The first **True** one causes the following *statement*s to be executed. If all are **False** then the Else's *statement*s are executed. The ElseIf and Else portions are optional.

Form 3: If *objexpr*'s type is the same type or a type descended from *objtype* the Then portion is executed.

See Also **Select Case**, **Choose( )**, **IIf( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  S = InputBox("Enter hello, goodbye, dinner or sleep:")
  S = UCase(S)
  If S = "HELLO" Then Debug.Print "come in"
  If S = "GOODBYE" Then Debug.Print "see you later"
  If S = "DINNER" Then
    Debug.Print "Please come in."
    Debug.Print "Dinner will be ready soon."
  ElseIf S = "SLEEP" Then
    Debug.Print "Sorry."
    Debug.Print "We are full for the night"
  End If
End Sub
```

IIf Function

Syntax IIf(*condexpr*, *TruePart*, *FalsePart*)

Group Miscellaneous

Description Return the value of the parameter indicated by *condexpr*. Both *TruePart* and *FalsePart* are evaluated.

Parameter **Description**

condexpr If this value is **True** then return *TruePart*. Otherwise, return *FalsePart*.

TruePart Return this value if *condexpr* is **True**.

FalsePart Return this value if *condexpr* is **False**.

See Also **If**, **Select Case**, **Choose( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print IIf(1 > 0,"True","False") '"True"
End Sub
```

InputBox$ Function

Syntax InputBox[$](*Prompt$*[, *Title$*][, *Default$*][, *XPos*, *YPos*])

Group User Input

Description Display an input box where the user can enter a line of text. Pressing the OK button returns the string entered. Pressing the Cancel button returns a null string.

Parameter **Description**

Prompt$ Use this string value as the prompt in the input box.

Title$ Use this string value as the title of the input box. If this is omitted then the input box does not have a title.

Default$ Use this string value as the initial value in the input box. If this is omitted then the initial value is blank.

XPos When the dialog is put up the left edge will be at this screen position. If this is omitted then the dialog will be centered.

YPos When the dialog is put up the top edge will be at this screen position. If this is omitted then the dialog will be centered.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim L$
  L$ = InputBox$("Enter some text:", _
      "Input Box Example","asdf")
  Debug.Print L$
End Sub
```

Input$ Function

Syntax Input[$](*N*, *StreamNum*)

Group File

Description Return *N* chars from *StreamNum*.

Parameter **Description**

N Read this many chars. If fewer than that many chars are left before the end of file then a run-time error occurs.

StreamNum Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim L, T$
```

```
  Open "XXX" For Input As #1
  L = LOF(1)
  T$ = Input$(L,1)
  Close #1
  Debug.Print T$;
End Sub
```

Input Instruction

Syntax Input [#]*StreamNum*, *var*[, ...]

Group File

Description Get input from *StreamNum* and assign it to *var*s. Input values are comma delimited. Leading and trailing spaces are ignored. If the first char (following the leading spaces) is a quote (") then the string is terminated by an ending quote. Special values #NULL#, #FALSE#, #TRUE#, #date# and #ERROR number# are converted to their appropriate value and data type.

See Also **Line Input**, **Print**, **Write**.

Example

```
'#Language "WWB-COM"
Sub Main
  Open "XXX" For Input As #1
  Input #1, A, B, C$
  Debug.Print A; B; C$
 'Close #1
End Sub
```

InStr Function

Syntax InStr([*Index*,]*S1$*, *S2$*)

Group String

Description Return the index where *S2$* first matches *S1$*. If no match is found return 0.

Note: A similar function, InStrB, returns the byte index instead.

Parameter **Description**

Index Start searching for *S2$* at this index in *S1$*. If this is omitted then start searching from the beginning of *S1$*.

S1$* Search for *S2$ in this string value. If this value is **Null** then **Null** is returned.

S2$* Search *S1$ for this string value. If this value is **Null** then **Null** is returned.

See Also **InStrRev( )**, **Left$( )**, **Len( )**, **Mid$( )**, **Replace$( )**, **Right$( )**.

Example

```
'#Language "WWB-COM"
Sub Main
```

```
  Debug.Print InStr("Hello","l") ' 3
EndSub
```

InStrRev Function

Syntax InStrRev(*S1$*, *S2$*[, *Index*])

Group String

Description Return the index where *S2$* last matches *S1$*. If no match is found return 0.

Parameter **Description**

S1$* Search for *S2$ in this string value. If this value is **Null** then **Null** is returned.

S2$* Search *S1$ for this string value. If this value is **Null** then **Null** is returned.

Index Start searching for *S2$* ending at this index in *S1$*. If this is omitted then start searching from the end of *S1$*.

See Also **Left$( )**, **Len( )**, **Mid$( )**, **Replace$( )**, **Right$( )**.

Example

```
'#Language "WWB-COM"
Sub Main
  Debug.Print InStrRev("Hello","l") ' 4
EndSub
```

Integer Data Type

Syntax **Dim** v As Integer

Group Data Type

Description A 16 bit signed integer value.

Int Function

Syntax Int(*Num*)

Group Math

Description Return the integer value.

Parameter **Description**

Num Return the largest integer which is less than or equal to this numeric value. If this value is **Null** then **Null** is returned.

See Also **Fix**.

Example

```
'#Language "WWB-COM"
Sub Main
```

```
  Debug.Print Int(9.9)  ' 9
  Debug.Print Int(0)    ' 0
  Debug.Print Int(-9.9) '-10
EndSub
```

IsArray Function

Syntax IsArray(*expr*)

Group Variable Info

Description Return the **True** if *expr* is an array of values.

Parameter **Description**

expr A array variable or a variant var can contain multiple of values.

See Also **TypeName**, **VarType**.

Example

```
'#Language "WWB-COM"
Sub Main
  Dim X As Variant, Y(2) As Integer
  Debug.Print IsArray(X) 'False
  X = Array(1,4,9)
  Debug.Print IsArray(X) 'True
  X = Y
  Debug.Print IsArray(X) 'True
EndSub
```

IsDate Function

Syntax IsDate(*expr*)

Group Variable Info

Description Return the **True** if *expr* is a valid date.

Parameter **Description**

expr A variant expression to test for a valid date.

See Also **TypeName**, **VarType**.

Example

```
'#Language "WWB-COM"
Sub Main
  Dim X As Variant
  X = 1
  Debug.Print IsDate(X) 'False
  X = Now
  Debug.Print IsDate(X) 'True
End Sub
```

IsEmpty Function

Syntax IsEmpty(*expr*)

Group Variable Info

Description Return the **True** if *expr* is **Empty**.

Parameter **Description**

expr A expression is **Empty** if it has never been assign a value.

See Also **TypeName**, **VarType**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim X As Variant
  Debug.Print IsEmpty(X) 'True
  X = 0
  Debug.Print IsEmpty(X) 'False
  X = Empty
  Debug.Print IsEmpty(X) 'True
End Sub
```

IsError Function

Syntax IsError(*expr*)

Group Variable Info

Description Return the **True** if *expr* is an error code.

Parameter **Description**

expr A variant expression to test for an error code value.

See Also **TypeName**, **VarType**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim X As Variant
  Debug.Print IsError(X) 'False
  X = CVErr(1)
  Debug.Print IsError(X) 'True
EndSub
```

IsMissing Function

Syntax IsMissing(*expr*)

Group Variable Info

Description Return the **True** if Optional parameter *expr* does not have a defaultvalue and it did not get a value. An Optional parameter may be omitted in the **Sub**, **Function** or **Property** call.

Parameter **Description**

expr Return **True** if this variant parameter's argument expression was not specified in the **Sub**, **Function** or **Property** call.

Example
```
'#Language "WWB-COM"
Sub Main
  Opt          'IsMissing(A)=True
  Opt "Hi"     'IsMissing(A)=False
  Many         'No args
  Many 1,"Hello"  'A(0)=1 A(1)=Hello
  OptBye       '"Bye"
  OptBye "No"    '"No"
End Sub

Sub Opt(Optional A)
  Debug.Print "IsMissing(A)="; IsMissing(A)
End Sub

Sub Many(ParamArray A())
  If LBound(A) > UBound(A) Then
    Debug.Print "No args"
  Else
    For I = LBound(A) To UBound(A)
    Debug.Print "A(" & I & ")=" & A(I) & " ";
    Next I
    Debug.Print
  EndIf
End Sub

Sub OptBye(Optional A As String = "Bye")
  Debug.Print A
End Sub
```

IsNot Operator

Syntax *expr* IsNot *expr*

Group Operator

Description Return the **False** if both *expr*s refer to the same object.

VBA This language element is not VBA compatible and requires the **#Language** "WWB-COM" setting.

See Also **Objects**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim X As Object
  Dim Y As Object
  Debug.Print X IsNot Y ' False
End Sub
```

IsNull Function

Syntax IsNull(*expr*)

Group Variable Info

Description Return the **True** if *expr* is **Null**.

Parameter **Description**

expr A variant expression to test for **Null**.

See Also **IsNull**, **TypeName**, **VarType**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim X As Variant
  Debug.Print IsEmpty(X) 'True
  Debug.Print IsNull(X)  'False
  X = 1
  Debug.Print IsNull(X)  'False
  X = "1"
  Debug.Print IsNull(X)  'False
  X = Null
  Debug.Print IsNull(X)  'True
  X = X*2
  Debug.Print IsNull(X)  'True
End Sub
```

IsNumeric Function

Syntax IsNumeric(*expr*)

Group Variable Info

Description Return the **True** if *expr* is a numeric value.

Parameter **Description**

expr A variant expression is a numeric value if it is *numeric* or string value that represents a number.

See Also **TypeName**, **VarType**.

```
Example         '#Language "WWB-COM"
Sub Main
  Dim X As Variant
  X = 1
  Debug.Print IsNumeric(X) 'True
  X = "1"
  Debug.Print IsNumeric(X) 'True
  X = "A"
  Debug.Print IsNumeric(X) 'False
EndSub
```

IsObject Function

Syntax IsObject(*expr*)

Group Variable Info

Description Return the **True** if *expr* contains an object reference.

Parameter **Description**

var If *expr* is an object reference return **True**.

See Also **TypeName**, **VarType**.

```
Example         '#Language "WWB-COM"
Sub Main
  Dim X As Variant
  X = 1
  Debug.Print IsObject(X) 'False
  X = "1"
  Debug.Print IsObject(X) 'False
  Set X = Nothing
  Debug.Print IsObject(X) 'True
End Sub
```

Is Operator

Syntax *expr* Is *expr*
-or-
expr **IsNot** *expr*

Group Operator

Description Return the **True** if both *expr*s refer to the same object.

The IsNot operator inverts the result.

See Also **Objects**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim X As Object
  Dim Y As Object
  Debug.Print X Is Y ' True
  Debug.Print X IsNot Y ' False
End Sub
```

Join Function

Syntax Join(*StrArray*, [*Sep*])

Group Miscellaneous

Description Return a string by concatenating all the values in the array with Sep in between each one.

Parameter **Description**

StrArray Concatenate values from this array.

Sep Use this string value to separate the values. (Default: " ")

See Also **Split( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Join(Array(1,2,3)) '"1 2 3"
EndSub
```

KeyName Function

Syntax KeyName(*Key*)

Group Miscellaneous

Description Return the key name for a key number. This is the name used by **SendKeys**.

Parameter **Description**

Key Key number.

See Also **SendKeys**.

Example
```
'#Language "WWB-COM"
Sub Main
```

```
  Debug.Print KeyName(&H270)  "^{F1}"
EndSub
```

Kill Instruction

Syntax Kill *Name$*

Group File

Description Delete the file named by *Name$*.

Parameter **Description**

Name$ This string value is the path and name of the file. A path relative to the current directory can be used.

Example

```
'#Language "WWB-COM"
Sub Main
  Kill "XXX"
End Sub
```

#Language Special Comment

Syntax '#Language "WWB-COM"

Group Declaration

Description Selects enhanced Visual Basic for Applications(TM) compatibility.

Including this special comment enables these enhancements:

- Data Types: **Decimal**, **SByte**, **UHuge_**, **UInteger** and **ULong**.
- Instructions: **Return**
- Conversion Functions: **CSByte**, **CUHuge_**, **CUInt** and **CULng**.
- **Operators**: AndAlso, **IsNot** and OrElse.

If this special comment is not included then the macro/module is parsed using '#Language "WWB-COM" rules without the enhancements.

Language reference by **group**:

- *Declaration*, *Data Type*, *Assignment*
- *Flow Control*, *Error Handling*
- *Conversion*, *Variable Info*, *Constant*
- *Math*, *String*, *Object*, *Time/Date*, *File*
- *User Input*, *User Dialog*, *Dialog Function*

- *DDE, Settings, Miscellaneous*
- *Operator*

Version Available for WinWrap Basic version 9.1 or higher.

LBound Function

Syntax LBound(*arrayvar*[, *dimension*])

Group Variable Info

Description Return the lowest index.

Parameter **Description**

arrayvar Return the lowest index for this array variable.

dimension Return the lowest index for this dimension of *arrayvar*. If this is omitted then return the lowest index for the first dimension.

See Also **UBound( )**.

```
Example         '#Language "WWB-COM"
Sub Main
  Dim A(-1 To 3,2 To 6)
  Debug.Print LBound(A)   '-1
  Debug.Print LBound(A,1) '-1
  Debug.Print LBound(A,2) ' 2
End Sub
```

LCase$ Function

Syntax LCase[$](*S$*)

Group String

Description Return a string from *S$* where all the uppercase letters have been lowercased.

Parameter **Description**

S$ Return the string value of this after all chars have been converted to lowercase. If this value is **Null** then **Null** is returned.

See Also **StrComp( )**, **StrConv$( )**, **UCase$( )**.

```
Example         '#Language "WWB-COM"
Sub Main
  Debug.Print LCase$("Hello") '"hello"
EndSub
```

Left$ Function

Syntax Left[$](*S$*, *Len*)

Group String

Description Return a string from *S$* with only the *Len* chars.

Note: A similar function, LeftB, returns the first *Len* bytes.

Parameter **Description**

S$ Return the left portion of this string value. If this value is **Null** then **Null** is returned.

Len Return this many chars. If *S$* is shorter than that then just return *S$*.

See Also **InStr( )**, **InStrRev( )**, **Len( )**, **Mid$( )**, **Replace$( )**, **Right$( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Left$("Hello",2) '"He"
EndSub
```

Len Function

Syntax Len(*S$*)
-or-
Len(*usertypevar*)

Group String

Description Return the number of characters in *S$*.

Note: A similar function, LenB, returns the number of bytes in the string. For a *usertypevar*, LenB returns the number of bytes of memory occupied by the variable's data.

Parameter **Description**

S$ Return the number of chars in this string value. If this value is **Null** then **Null** is returned.

usertypevar Return the number of bytes required to store this user type structure variable. If the user type has any dynamic **String** and **Variant** elements the length returned may not be as big as the actual number of bytes required.

See Also **InStr( )**, **InStrRev( )**, **Left$( )**, **Mid$( )**, **Replace$( )**, **Right$( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Len("Hello") ' 5
EndSub
```

Like Operator

Syntax *str1* Like *str2*

Group Operator

Description Return the **True** if *str1* matches pattern *str2*. The pattern in *str2* is one or more of the special character sequences shown in the following table.

Char(s) **Description**

? Match any single character.

* Match zero or more characters.

Match a single digit (0-9).

[*charlist*] Match any char in the list.

[!*charlist*] Match any char not in the list.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print "abcdfgcdefg" Like "" ' False
  Debug.Print "abcdfgcdefg" Like "a*g" ' True
  Debug.Print "abcdfgcdefg" Like "a*cde*g" ' True
  Debug.Print "abcdfgcdefg" Like "a*cd*cd*g" ' True
  Debug.Print "abcdfgcdefg" Like "a*cd*cd*g" ' True
  Debug.Print "00aa" Like "####" ' False
  Debug.Print "00aa" Like "????" ' True
  Debug.Print "00aa" Like "##??" ' True
  Debug.Print "00aa" Like "*##*" ' True
  Debug.Print "hk" Like "hk*" ' True
End Sub
```

Line Input Instruction

Syntax **Line Input** [#]*StreamNum*, *S$*

Group File

Description Get a line of input from *StreamNum* and assign it to *S$*.

See Also **Input**, **Print**, **Write**.

Example
```
'#Language "WWB-COM"
Sub Main
  Open "XXX" For Input As #1
  Line Input #1, S$
  Debug.Print S$
```

```
  Close #1
End Sub
```

ListBox Dialog Item Definition

Syntax ListBox *X*, *Y*, *DX*, *DY*, *StrArray$( )*, .*Field*[, *Options*]

Group User Dialog

Description Define a listbox item.

Parameter **Description**

X This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.

Y This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.

DX This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.

DY This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.

StrArray$( ) This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.

Field The value of the list box is accessed via this field. It is the index of the *StrArray$( )* var.

Options This numeric value controls the type of list box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option **Description**

0 List is not sorted.

1 List is not sorted and horizontally scrollable.

2 List is sorted.

3 List is sorted and horizontally scrollable.

See Also **Begin Dialog**, **MultiListBox**.

Example

```
'#Language "WWB-COM"
Sub Main
  Dim lists$(3)
  lists$(0) = "List 0"
  lists$(1) = "List 1"
  lists$(2) = "List 2"
  lists$(3) = "List 3"
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
```

```
    ListBox 10,25,180,60,lists$(),.list
    OKButton 80,90,40,20
  EndDialog
  Dim dlg As UserDialog
  dlg.list = 2
  Dialog dlg ' show dialog (wait for ok)
  Debug.Print dlg.list
End Sub
```

Loc Function

Syntax Loc(*StreamNum*)

Group File

Description Return *StreamNum* file position. For Random mode files this is the current record number minus one. For Binary mode files it is the current byte position minus one. Otherwise, it is the current byte position minus one divided by 128. The first position in the file is 0.

Parameter **Description**

StreamNum Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

Example

```
'#Language "WWB-COM"
Sub Main
  Open "XXX" For Input As #1
  L = Loc(1)
 'Close #1
  Debug.Print L ' 0
End Sub
```

Lock Instruction

Syntax Lock *StreamNum*
-or-
Lock *StreamNum*, *RecordNum*
-or-
Lock *StreamNum*, [*start*] To *end*

Group File

Description Form 1: Lock all of *StreamNum*.

Form 2: Lock a record (or byte) of *StreamNum*.

Form 3: Lock a range of records (or bytes) of *StreamNum*. If *start* is omitted then lock starting at the first record (or byte).

Note: Be sure to **Unlock** for each Lock instruction.
Note: For sequential files (Input, Output and Append) lock always affects the entire file.

Parameter **Description**

StreamNum Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

RecordNum For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1.

start First record (or byte) in the range.

end Last record (or byte) in the range.

See Also **Open**, **Unlock**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim V As Variant
  Open "SAVE_V.DAT" For Binary As #1
  Lock #1
  Get #1, 1, V
  V = "Hello"
  Put #1, 1, V
  Unlock #1
 'Close #1
End Sub
```

LOF Function

Syntax LOF(*StreamNum*)

Group File

Description Return *StreamNum* file length (in bytes).

Parameter **Description**

StreamNum Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

Example
```
'#Language "WWB-COM"
Sub Main
  Open "XXX" For Input As #1
  L = LOF(1)
 'Close #1
  Debug.Print L
EndSub
```

Log Function

Syntax Log(*Num*)

Group Math

Description Return the natural logarithm.

Parameter **Description**

Num Return the natural logarithm of this numeric value. The value e is approximately 2.718282.

See Also **Exp**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Log(1) ' 0
End Sub
```

LSet Instruction

Syntax LSet *strvar* = *str*
-or-
LSet *usertypevar1* = *usertypevar2*

Group Assignment

Description Form 1: Assign the value of *str* to *strvar*. Shorten *str* by removing trailing chars (or extend with blanks). The previous length *strvar* is maintained.

Form 2: Assign the value of *usertypevar2* to *usertypevar1*. If *usertypevar2* is longer than *usertypevar1* then only copy as much as *usertypevar1* can handle.

See Also **RSet**.

Example
```
'#Language "WWB-COM"
Sub Main
  S$ = "123"
  LSet S$ = "A"
  Debug.Print "."; S$; "." '".A  ."
EndSub
```

LTrim$ Function

Syntax LTrim[$](*S$*)

Group String

Description Return the string with *S$*'s leading spaces removed.

Parameter **Description**

S$ Copy this string without the leading spaces. If this value is **Null** then **Null** is returned.

See Also **RTrim$( )**, **Trim$( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print "."; LTrim$("  x  "); "." '".x  ."
End Sub
```

MacroCheck Function

Syntax MacroCheck(*MacroName$*)

Group Flow Control

Description Check the syntax of a *macro*/*module*. Does not execute the macro/module. Returns an Err object if there is a syntax error, otherwise Nothing is returned.

Parameter **Description**

MacroName$ Check the macro named by this string value.

See Also **MacroCheckThis**, **MacroRun**, **MacroRunThis**, **ModuleLoad**, **ModuleLoadThis**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim E As ErrObject
  Set E = MacroCheck("Demo")
  If Not E IsNothing Then Debug.Print E.Description
EndSub
```

MacroCheckThis Function

Syntax MacroCheckThis(*MacroCode$*)

Group Flow Control

Description Check the syntax the *macro*/*module* code in *MacroCode*. The macro/module code is not executed. Returns an Err object if there is a syntax error, otherwise Nothing is returned.

Parameter **Description**

MacroName$ Check the macro code in this string value.

See Also **MacroCheck**, **MacroRun**, **MacroRunThis**, **ModuleLoad**, **ModuleLoadThis**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim E As ErrObject
  Set E = MacroCheckThis("bad macro")
```

```
  If Not E Is Nothing Then Debug.Print E.Description
End Sub
```

MacroDir$ Function

Syntax MacroDir[$]

Group Flow Control

Description Return the directory of the current macro. A run-time error occurs if the current macro has never been saved.

See Also **MacroRun**.

Example

```
'#Language "WWB-COM"
Sub Main
  ' open the file called Data that is in the
  ' same directory as the macro
  Open MacroDir & "\Data" ForInput As #1
  Line Input #1, S$
  Debug.Print S$
 'Close #1
End Sub
```

MacroRun Instruction

Syntax MacroRun *MacroName$*[, *Command$*]

Group Flow Control

Description Play a *macro*. Execution will continue at the following statement after the macro has completed.

Parameter **Description**

MacroName$ Run the macro named by this string value.

*Command$* Pass this string value as the macro's **Command$** value.

See Also **Command$**, **MacroCheck**, **MacroCheckThis**, **MacroDir$**, **MacroRunThis**, **ModuleLoad**, **ModuleLoadThis**.

Example

```
'#Language "WWB-COM"
Sub Main
  Debug.Print "Before Demo"
  MacroRun "Demo"
  Debug.Print "After Demo"
End Sub
```

MacroRunThis Instruction

Syntax MacroRunThis *MacroCode$*

Group Flow Control

Description Play the *macro* code in *MacroCode*. Execution will continue at the following statement after the macro code has completed. The macro code can be either a single line or a complete macro.

Parameter **Description**

MacroName$ Run the macro code in this string value.

See Also **Command$**, **MacroCheck**, **MacroCheckThis**, **MacroDir$**, **MacroRun**, **ModuleLoad**, **ModuleLoadThis**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print "Before Demo"
  MacroRunThis "MsgBox ""Hello"""
  Debug.Print "After Demo"
End Sub
```

Main Sub

Syntax
```
Sub Main()
  ...
End Sub
-or-
Private Sub Main()
  ...
End Sub
```

Group Declaration

Description Form 1: Each *macro* must define Sub Main. A macro is a "program". Running a macro starts the Sub Main and continues to execute until the subroutine finishes.

Form 2: A code *module* may define a Private Sub Main. This Sub Main is the **code module** initialization subroutine. If Main is not defined then no special initialization occurs.

See Also **Code Module**.

Me Object

Syntax Me

Group Object

Description Me references the current macro/module. It can be used like any other *object variable*, except that it's reference can't be changed.

See Also **Set**.

Example
```
'#Language "WWB-COM"
Sub Main
  DoIt
  Me.DoIt ' calls the same sub
EndSub
Sub DoIt
  MsgBox "Hello"
End Sub
```

Mid$ Function/Assignment

Syntax Mid[$](*S$*, *Index*[, *Len*])
-or-
Mid[$](*strvar*, *Index*[, *Len*]) = *S$*

Group String

Description Function: Return the substring of *S$* starting at *Index* for *Len* chars.

Instruction: Assign *S$* to the substring in *strvar* starting at *Index* for *Len* chars.

Note: A similar function, MidB, returns the *Len* bytes starting a byte *Index*.

Parameter **Description (Mid Function)**

S$ Copy chars from this string value. If this value is **Null** then **Null** is returned.

Index Start copying chars starting at this index value. If the string is not that long then return a null string.

Len Copy this many chars. If the *S$* does not have that many chars starting at *Index* then copy the remainder of *S$*.

Parameter **Description (Mid Assignment)**

strvar Change part of this string.

Index Change *strvar* starting at this index value. If the string is not that long then it is not changed.

Len The number of chars copied is smallest of: the value of *Len*, the length of *S$* and the remaining length of *strvar*. (If this value is omitted then the number of chars copied is the smallest of: the length of *S$* and the remaining length of *strvar*.)

S$ Copy chars from this string value.

See Also **InStr( )**, **Left$( )**, **Len( )**, **Replace$( )**, **Right$( )**.

Example '#Language "WWB-COM"
Sub Main
  S$ = "Hello There"
  Mid$(S$,7) = "??????????"
  Debug.**Print** S$ '"Hello ?????"
  Debug.**Print** Mid$("Hello",2,1) '"e"
EndSub

Minute Function

Syntax Minute(*dateexpr*)

Group Time/Date

Description Return the minute of the hour (0 to 59).

Parameter **Description**

dateexpr Return the minute of the hour for this date value. If this value is **Null** then **Null** is returned.

See Also **Hour( )**, **Second( )**, **Time( )**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** Minute(#12:00:01 AM#) ' 0
EndSub

MkDir Instruction

Syntax MkDir *Name$*

Group File

Description Make directory *Name$*.

Parameter **Description**

Name$ This string value is the path and name of the directory. A path relative to the current directory can be used.

See Also **RmDir**.

Example '#Language "WWB-COM"
Sub Main
  MkDir "C:\WWTEMP"
End Sub

ModuleLoad Function

Syntax ModuleLoad(*ModuleName$*, *CreateNew*)

Group Flow Control

Description Load a *module*. Does not execute the module. Macro's can not be loaded. Returns an object if successful, otherwise Nothing is returned.

Parameter **Description**

ModuleName$ Load the module named by this string value.

CreateNew Return a new instance if True. Otherwise return the default instance. A class module does not have a default instance. A code module can not have a new instance.

See Also **MacroCheck**, **MacroCheckThis**, **MacroRun**, **MacroRunThis**, **ModuleLoadThis**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim Obj As Object
  Set Obj = ModuleLoad("Demo")
  Obj.DoIt ' call Demo's DoIt method
End Sub
```

ModuleLoadThis Function

Syntax ModuleLoadThis(*ModuleCode$*, *CreateNew*)

Group Flow Control

Description Load *ModuleCode* as a *module*. Does not execute the module. Macro's can not be loaded. Returns an object if successful, otherwise Nothing is returned.

Parameter **Description**

ModuleCode$ Load the module code in this string value.

CreateNew Return a new instance if True. Otherwise return the default instance. A class module does not have a default instance. A code module can not have a new instance.

See Also **MacroCheck**, **MacroCheckThis**, **MacroRun**, **MacroRunThis**, **ModuleLoad**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim Obj As Object
  Set Obj = ModuleLoadThis("Sub DoIt" & vbCrLf & "End Sub", False)
  Obj.DoIt ' call Demo's DoIt method
End Sub
```

Month Function

Syntax Month(*dateexpr*)

Group Time/Date

Description Return the month of the year (1 to 12).

Parameter **Description**

dateexpr Return the month of the year for this date value. If this value is **Null** then **Null** is returned.

See Also **Date( )**, **Day( )**, **MonthName( )**, **Weekday( )**, **Year( )**.

```
Example         '#Language "WWB-COM"
Sub Main
  Debug.Print Month(#1/1/1900#) ' 1
  Debug.Print Month(#2/1/1900#) ' 2
EndSub
```

MonthName Function

Syntax MonthName(NumZ{month}[, CondZ{abbrev}])

Group Time/Date

Description Return the localized name of the month.

Parameter **Description**

month Return the localized name of this month. (1-12)

abbrev If this conditional value is **True** then return the abbreviated form of the month name.

See Also **Month( )**.

```
Example         '#Language "WWB-COM"
Sub Main
  Debug.Print MonthName(1) 'January
  Debug.Print MonthName(Month(Now))
EndSub
```

MsgBox Instruction/Function

Syntax MsgBox *Message$*[, *Type*][, *Title$*]
-or-
MsgBox(*Message$*[, *Type*][, *Title$*])

Group User Input

Description Show a message box titled *Title$*. *Type* controls what the message box looks like (choose one value from each category). Use MsgBox( ) if you need to know what button was pressed. The result indicates which button was pressed.

Result **Value Button Pressed**

vbOK 1 OK button

vbCancel 2 Cancel button

vbAbort 3 Abort button

vbRetry 4 Retry button

vbIgnore 5 Ignore button

vbYes 6 Yes button

vbNo 7 No button

Parameter Description

Message$ This string value is the text that is shown in the message box.

Type This numeric value controls the type of message box. Choose one value from each of the following tables.

Title$ This string value is the title of the message box.

Button Value Effect

vbOkOnly 0 OK button

vbOkCancel 1 OK and Cancel buttons

vbAbortRetryIgnore 2 Abort, Retry, Ignore buttons

vbYesNoCancel 3 Yes, No, Cancel buttons

vbYesNo 4 Yes and No buttons

vbRetryCancel 5 Retry and Cancel buttons

Icon Value Effect

0 No icon

vbCritical 16 Stop icon

vbQuestion 32 Question icon

vbExclamation 48 Attention icon

vbInformation 64 Information icon

Default Value Effect

vbDefaultButton1 0 First button

vbDefaultButton2 256 Second button

vbDefaultButton3 512 Third button

Mode Value Effect

vbApplicationModal 0 Application modal

vbSystemModal 4096 System modal

vbMsgBoxSetForeground &h10000 Show message box in front of all other windows

Example

```
'#Language "WWB-COM"
Sub Main
  MsgBox "Please press OK button"
  If MsgBox("Please press OK button",vbOkCancel) = vbOK Then
    Debug.Print "OK was pressed"
  Else
  Debug.Print "Cancel was pressed"
  End If
EndSub
```

MultiListBox Dialog Item Definition

Syntax MultiListBox *X*, *Y*, *DX*, *DY*, *StrArray$( )*, .*Field*[, *Options*]

Group User Dialog

Description Define a multiple selection listbox item.

Parameter Description

X This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.

Y This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.

DX This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.

DY This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.

StrArray$( ) This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.

Field The values of the list box are accessed via this field. It is the index of the *StrArray$( )* var.

Options This numeric value controls the type of list box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option Description

0 List is not sorted.

1 List is not sorted and horizontally scrollable.

2 List is sorted.

3 List is sorted and horizontally scrollable.

See Also **Begin Dialog**, **ListBox**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim lists$(3)
  lists$(0) = "List 0"
  lists$(1) = "List 1"
  lists$(2) = "List 2"
  lists$(3) = "List 3"
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    MultiListBox 10,25,180,60,lists$(),.list
    OKButton 80,90,40,20
  EndDialog
  Dim dlg As UserDialog
  dlg.list = Array(0,2)
  Dialog dlg ' show dialog (wait for ok)
  Dim i As Integer
  For i = LBound(dlg.list) To UBound(dlg.list)
    Debug.Print dlg.list(i);
  Next i
  Debug.Print
EndSub
```

Name Instruction

Syntax Name *OldName$* As *NewName$*

Group File

Description Rename file *OldName$* as *NewName$*.

Parameter **Description**

OldName$ This string value is the path and name of the file. A path relative to the current directory can be used.

NewName$ This is the new file name (and path). A path relative to the current directory can be used.

Example
```
'#Language "WWB-COM"
Sub Main
  Name "AUTOEXEC.BAK" As "AUTOEXEC.SAV"
End Sub
```

New Operator

Syntax New *objtype*

Group Operator

Description Returns a new instance of *objtype*.

Parameter **Description**

objtype This is the new object's type.

See Also **Objects**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim obj As Object
  Set obj = New Dictionary
End Sub
```

Nothing Keyword

Group Constant

Description An *objexpr* that does not refer to any object.

Now Function

Syntax Now

Group Time/Date

Description Return the current date and time as a **Date** value.

See Also **Date**, **Time**, **Timer**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Now ' example: 1/1/1995 10:05:32 AM
EndSub
```

Null Keyword

Group Constant

Description A *variant expression* that is null. A null value propagates through an expression causing the entire expression to be Null. Attempting to use a Null value as a string or numeric argument causes a run-time error. A Null value prints as "#NULL#".

See Also **IsNull**.

Example
```
'#Language "WWB-COM"
Sub Main
  X = Null
  Debug.Print X = Null  '#NULL#
```

```
  Debug.Print IsNull(X) 'True
End Sub
```

Object Data Type

Syntax **Dim** v As Object

Group Data Type

Description An object reference value. (see **Objects**) An object reference may also appear to have a data value, see table below:

Data	Description
get-reference	Use in any *object expression*.
set-reference	Use **Set** to change the reference.
get-value	Use the reference's default property value.
assign-value	Use **Assign** (or **Let**) to change the reference's default property value.

Object_Initialize Sub

Syntax **Private Sub Object**_Initialize()

```
  ...
EndSub
```

Group Declaration

Description Object module initialization subroutine. Each time a new instance is created for a Object module the Object_Initialize sub is called. If Object_Initialize is not defined then no special initialization occurs.

Note: Object_Initialize is also called for the instance that is automatically created.

See Also **Object Module**, **Object_Terminate**.

Oct$ Function

Syntax Oct[$](*Num*)

Group String

Description Return a octal string.

Parameter	Description
Num	Return an octal encoded string for this numeric value.

See Also **Hex$( )**, **Str$( )**, **Val( )**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** Oct$(15) '17
End Sub

On Error Instruction

Syntax On **Error** GoTo 0
-or-
On **Error** GoTo *label*
-or-
On **ErrorResume** Next

Group Error Handling

Description Form 1: Disable the error handler (default).

Form 2: Send error conditions to an error handler.

Form 3: Error conditions continue execution at the next statement.

On Error sets or disables the error handler. Each user defined *procedure* has its own error handler. The default is to terminate the *macro* on any error. The **Err** object's properties are set whenever an error occurs. Once an error has occurred and the error handler is executing any further errors will terminate the macro, unless the **Err** object has been cleared.

Note: This instruction clears the **Err** and sets **Error`$'** to null.

Example '#Language "WWB-COM"
Sub Main
  On **ErrorResume** Next
  Err.Raise 1
  Debug.**Print** "RESUMING, **Err**="; **Err**
  On **Error** GoTo X
  Err.Raise 1
  Exit Sub

X: **Debug**.**Print** "**Err**="; **Err**
  Err.Clear
  Debug.**Print** "**Err**="; **Err**
  Resume Next
EndSub

OKButton Dialog Item Definition

Syntax OKButton *X*, *Y*, *DX*, *DY*[, .*Field*]

Group User Dialog

Description Define an OK button item. Pressing the OK button updates the *dlgvar* field values and closes the dialog. (**Dialog**( ) function call returns -1.)

Parameter **Description**

X This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.

Y This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.

DX This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.

DY This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.

Field This identifier is the name of the field. The *dialogfunc* receives this name as *string*. If this is omitted then the field name is "OK".

See Also **Begin Dialog**.

Example
```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,30,"Please push the OK button"
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  Dialog dlg ' show dialog (wait for ok)
End Sub
```

Open Instruction

Syntax Open *Name$***For** mode [Access access] [lock] As _
[#]*StreamNum* [**Len** = *RecordLen*]

Group File

Description Open file *Name$* for mode as *StreamNum*.

Reading Unicode Files To read a Unicode (UTF-16 or UTF-8) file, just open using Input mode. All the input is converted automatically.

Writing Unicode Files To write a Unicode (UTF-16 or UTF-8) file, open using Output or Append mode and write the appropriate Byte Order Mark (BOM). All the printed output is converted automatically.

To create a Unicode (UTF-16) text file use:

```
Open FileName$ For Output As #fn
Print #fn, vbUTF16BOM; ' first char is the UTF-16 Byte Order Mark
```

... ' everything automatically converted to UTF-16
Close #fn

To create a Unicode (UTF-8) text file use:

Open FileName$ **For** Output As #fn
Print #fn, vbUTF8BOM; ' first three chars are the UTF-8 **Byte** Order Mark
... ' everything automatically converted to UTF-8
Close #fn

Pocket PC Access and lock ignored.

Parameter **Description**

Name$ This string value is the path and name of the file. A path relative to the current directory can be used.

mode May be Input, Output, Append, Binary or Random.

access May be Read, Write or Read Write. (Only allowed for Binary and Random modes.)

lock May be Shared, Lock Read, Lock Write or Lock Read Write.

StreamNum Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

RecordLen This numeric value is the record length for Random mode files. Other file modes ignore this value.

See Also **Close**, **FileAttr**, **FreeFile**, **Reset**.

Example
```
'#Language "WWB-COM"
Sub Main
  Open "XXX" For Output As #1
  Print #1, "1,2,""Hello"""
  Close #1
EndSub
```

Operators

Syntax ^ Not * / \ Mod + - << >> & < <= > >= = <> **IsIsNot**
And AndAlso Or OrElse Xor Eqv Imp

Description These operators are available for numbers n1 and n2 or strings s1 and s2. If any value in an expression is **Null** then the expression's value is **Null**. The order of operator evaluation is controlled by operator *precedence*.

Operator **Description**

- *n1* Negate *n1*.

n1 ^ *n2* Raise *n1* to the power of *n2*.

n1 * *n2* Multiply *n1* by *n2*.

n1 / *n2* Divide *n1* by *n2*.

n1 \ *n2* Divide the integer value of *n1* by the integer value of *n2*.

n1 Mod *n2* Remainder of the integer value of *n1* after dividing by the integer value of *n2*.

n1 + *n2* Add *n1* to *n2*.

s1 + *s2* Concatenate *s1* with *s2*.

n1 - *n2* Difference of *n1* and *n2*.

n1 << *n2* Shift *n1* by *n2* bits to the left. The number of bits to shift is indicated by the lowest bits of *n2*. This language element is not VBA compatible.

n1 >> *n2* Shift *n1* by *n2* bits to the right. The number of bits to shift is indicated by the lowest bits of *n2*. (For signed numbers the sign bit is propagated to the right.) This language element is not VBA compatible.

s1 & *s2* Concatenate *s1* with *s2*.

n1 < *n2* Return **True** if *n1* is less than *n2*.

n1 <= *n2* Return **True** if *n1* is less than or equal to *n2*.

n1 > *n2* Return **True** if *n1* is greater than *n2*.

n1 >= *n2* Return **True** if *n1* is greater than or equal to *n2*.

n1 = *n2* Return **True** if *n1* is equal to *n2*.

n1 <> *n2* Return **True** if *n1* is not equal to *n2*.

s1 < *s2* Return **True** if *s1* is less than *s2*.

s1 <= *s2* Return **True** if *s1* is less than or equal to *s2*.

s1 > *s2* Return **True** if *s1* is greater than *s2*.

s1 >= *s2* Return **True** if *s1* is greater than or equal to *s2*.

s1 = *s2* Return **True** if *s1* is equal to *s2*.

s1 <> *s2* Return **True** if *s1* is not equal to *s2*.

Not *n1* Bitwise invert the integer value of *n1*. Only Not **True** is **False**.

n1 And *n2* Bitwise and the integer value of *n1* with the integer value *n2*.

n1 AndAlso *n2* Logical and of *n1* with the *n2*. If *n1* is **False**, *n2* is not evaluated. This language element is not VBA compatible and requires the **#Language** "WWB-COM" setting.

n1 Or *n2* Bitwise or the integer value of *n1* with the integer value *n2*.

n1 OrElse *n2* Logical or of *n1* with the *n2*. If *n1* is **True**, *n2* is not evaluated. This language element is not VBA compatible and requires the **#Language** "WWB-COM" setting.

n1 Xor *n2* Bitwise exclusive-or the integer value of *n1* with the integer value *n2*.

n1 Eqv *n2* Bitwise equivalence the integer value of *n1* with the integer value *n2* (same as Not (n1 Xor n2)).

n1 Imp *n2* Bitwise implicate the integer value of *n1* with the integer value *n2* (same as (Not n1) Or n2).

```
Example          '#Language "WWB-COM"
Sub Main
  N1 = 10
  N2 = 3
  S1$ = "asdfg"
  S2$ = "hjkl"
  Debug.Print -N1      '-10
  Debug.Print N1 ^ N2   ' 1000
  Debug.Print Not N1    '-11
  Debug.Print N1 * N2   ' 30
  Debug.Print N1 / N2   ' 3.3333333333333
  Debug.Print N1 \ N2   ' 3
  Debug.Print N1 Mod N2  ' 1
  Debug.Print N1 + N2   ' 13
  Debug.Print S1$ + S2$  '"asdfghjkl"
  Debug.Print N1 - N2   ' 7
  Debug.Print N1 << N2  ' 80
  Debug.Print N1 >> N2  ' 1
  Debug.Print N1 & N2   '"103"
  Debug.Print N1 < N2   'False
  Debug.Print N1 <= N2  'False
  Debug.Print N1 > N2   'True
  Debug.Print N1 >= N2  'True
  Debug.Print N1 = N2   'False
  Debug.Print N1 <> N2  'True
  Debug.Print S1$ < S2$  'True
  Debug.Print S1$ <= S2$ 'True
  Debug.Print S1$ > S2$  'False
  Debug.Print S1$ >= S2$ 'False
  Debug.Print S1$ = S2$  'False
  Debug.Print S1$ <> S2$ 'True
  Debug.Print N1 And N2    ' 2
  Debug.Print N1 AndAlso N2 ' True
  Debug.Print N1 Or N2     ' 11
  Debug.Print N1 OrElse N2  ' True
  Debug.Print N1 Xor N2    ' 9
  Debug.Print N1 Eqv N2    ' -10
  Debug.Print N1 Imp N2    ' -9
End Sub
```

OptionButton Dialog Item Definition

Syntax OptionButton *X*, *Y*, *DX*, *DY*, *Title$*[, .*Field*]

Group User Dialog

Description Define an option button item.

Parameter **Description**

X This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.

Y This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.

DX This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.

DY This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.

Title$ The value of this string is the title of the option button.

See Also **Begin Dialog**, **OptionGroup**.

Example
```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    OptionGroup .options
      OptionButton 10,30,180,15,"Option &0"
      OptionButton 10,45,180,15,"Option &1"
      OptionButton 10,60,180,15,"Option &2"
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  dlg.options = 2
  Dialog dlg ' show dialog (wait for ok)
  Debug.Print dlg.options
EndSub
```

Option Definition

Syntax Option Base [0|1]
-or-
Option Compare [Binary | **Text**]
-or-
Option Explicit
-or-
Option **Private** Module

Group Declaration

Description Option Base: Set the default base index for array declarations. Affects **Dim**, **Static**, **Private**, **Public** and **ReDim**. Does not affect **Array**, ParamArray or arrays declare in a **Type**. Option Base 0 is the default.

Option Compare: Set the default comparison mode for <, <=, =, >, >=, <>, **Like** and **StrComp**.

- Option Compare Binary - compare string text using binary data (default)
- Option Compare Text - compare string text using the collation rules

Option Explicit: Require all variables to be declared prior to use. Variables are declared using **Dim**, **Private**, **Public**, **Static** or as a parameter of **Sub**, **Function** or **Property** blocks.

Option Private: Public symbols defined by the module are only accessible from the same project.

Example
```
'#Language "WWB-COM"
Option Base 1
Option Explicit

SubMain
  Dim A
    Dim C(2) ' same as Dim C(1 To 2)
    Dim D(0 To 2)
  A = 1
  B = 2 ' B has not been declared
End Sub
```

OptionGroup Dialog Item Definition

Syntax OptionGroup .*Field*
OptionButton *X*, *Y*, *DX*, *DY*, *Title$*[, .*Field*]
OptionButton *X*, *Y*, *DX*, *DY*, *Title$*[, .*Field*]
...

Group User Dialog

Description Define a optiongroup and option button items.

Parameter **Description**

Field The value of the option group is accessed via this field. This first option button is 0, the second is 1, etc.

X This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.

Y This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.

DX This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.

DY This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.

Title$ The value of this string is the title of the option button.

See Also **Begin Dialog**, **OptionButton**.

Example
```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    OptionGroup .options
      OptionButton 10,30,180,15,"Option &0"
      OptionButton 10,45,180,15,"Option &1"
      OptionButton 10,60,180,15,"Option &2"
    OKButton 80,90,40,20
  End Dialog
  Dim dlg As UserDialog
  dlg.options = 2
  Dialog dlg ' show dialog (wait for ok)
  Debug.Print dlg.options
EndSub
```

Picture Dialog Item Definition

Syntax Picture *X*, *Y*, *DX*, *DY*, *FileName$*, *Type*[, .*Field*]

Group User Dialog

Description Define a picture item. The bitmap is automatically sized to fit the item's entire area.

Parameter **Description**

X This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.

Y This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.

DX This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.

DY This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.

FileName$ The value of this string is the .BMP file shown in the picture control.

Type This numeric value indicates the type of bitmap used. See below.

Field This identifier is the name of the field. The *dialogfunc* receives this name as *string*. If this identifer is omitted then the first two words of the title are used.

Type **Effect**

0 *FileName* is the name of the bitmap file. If the file does not exist then "(missing picture)" is displayed.

3 The clipboard's bitmap is displayed. Not supported.

+16 Instead of displaying "(missing picture)" a run-time error occurs.

See Also **Begin Dialog**.

Example
```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120
    Picture 10,10,180,75,"SAMPLE.BMP",0
    OKButton 80,90,40,20
  EndDialog
  Dim dlg As UserDialog
  Dialog dlg ' show dialog (wait for ok)
End Sub
```

PortInt Data Type

Syntax **Dim** v As PortInt

Group Data Type

Description A portable integer value.

- For Win16: A 16 bit signed integer value.
- For Win32: A 32 bit signed integer value.
- For Win64: A 64 bit signed integer value.

Print Instruction

Syntax Print #*StreamNum*, [*expr*[; ...][;]]

Group File

Description Print the *expr*(s) to *StreamNum*. Use ; to separate expressions. A *num* is it automatically converted to a string before printing (just like **Str$( )**). If the instruction does not end with a ; then a newline is printed at the end.

See Also **Input**, **Line Input**, **Write**.

Example
```
'#Language "WWB-COM"
Sub Main
  A = 1
  B = 2
  C$ = "Hello"
  Open "XXX" For Output As #1
  Print #1, A; ","; B; ","""; C$; """"
 'Close #1
End Sub
```

Private Definition

Syntax Private [**WithEvents**] *vardeclaration*[, ...]

Group Declaration

Description Create arrays (or simple variables) which are available to the entire *macro*/*module*, but not other macros/modules. Dimension var array(s) using the *dimension*s to establish the minimum and maximum index value for each dimension. If the *dimension*s are omitted then a scalar (single value) variable is defined. A dynamic array is declared using ( ) without any *dimension*s. It must be **ReDim**ensioned before it can be used. The Private statement must be placed outside of **Sub**, **Function** or **Property** blocks.

See Also **Dim**, **Option** Base, **Public**, **ReDim**, **Static**, **WithEvents**.

Example
```
'#Language "WWB-COM"
Private A0, A1(1), A2(1,1)

Sub Init
  A0 = 1
  A1(0) = 2
  A2(0,0) = 3
End Sub

SubMain
  Init
  Debug.Print A0; A1(0); A2(0,0) ' 1 2 3
EndSub
```

Private Keyword

Group Declaration

Description **Private Const**s, **Declare**s, **Function**s, **Property**s, **Sub**s and **Type**s are only available in the current *macro*/*module*.

Property Definition

Syntax [| **Private** | **Public** | **Friend**] _
[Default] _
Property **Get***name*[*type*][([*param*[, ...]])] [*As type*[()]]
  *statement*s
End Property
-or-
[| **Private** | **Public** | **Friend**] _
[Default] _
Property [**Let**|**Set**] *name*[([*param*[, ...]])]
  *statement*s
End Property

Group Declaration

Description User defined property. The property defines a set of *statement*s to be executed when its value is used or changed. A property acts like a variable, except that getting its value calls Property Get and changing its value calls Property Let (or Property Set). Property Get and Property Let with the same *name* define a property that holds a value. Property Get and Property Set with the same *name* define a property that holds an object reference. The values of the calling *arglist* are assigned to the *param*s. (For Property Let and Property Set the last parameter is the value on the right hand side of the assignment operator.)

Access If no access is specified then **Public** is assumed.

See Also **Function**, **Sub**.

Example
```
'#Language "WWB-COM"
Dim X_Value

Property Get X()
  X = X_Value
End Property

Property Let X(NewValue)
  If Not IsNull(NewValue) Then X_Value = NewValue
End Property

Sub Main
  X = "Hello"
  Debug.Print X '"Hello"
```

```
  X = Null
  Debug.Print X
End Sub
```

Public Definition

Syntax Public [**WithEvents**] *vardeclaration*[, ...]

Group Declaration

Description Create arrays (or simple variables) which are available to the entire *macro*/*module* and other macros/modules. Dimension var array(s) using the *dimension*s to establish the minimum and maximum index value for each dimension. If the *dimension*s are omitted then a scalar (single value) variable is defined. A dynamic array is declared using ( ) without any *dimension*s. It must be **ReDim**ensioned before it can be used. The Public statement must be placed outside of **Sub**, **Function** or **Property** blocks.

See Also **Dim**, **Option** Base, **Private**, **ReDim**, **Static**, **WithEvents**.

Example

```
'#Language "WWB-COM"
Public A0, A1(1), A2(1,1)

Sub Init
  A0 = 1
  A1(0) = 2
  A2(0,0) = 3
End Sub

SubMain
  Init
  Debug.Print A0; A1(0); A2(0,0) ' 1 2 3
EndSub
```

PushButton Dialog Item Definition

Syntax PushButton *X*, *Y*, *DX*, *DY*, *Title$*[, .*Field*]

Group User Dialog

Description Define a push button item. Pressing the push button updates the *dlgvar* field values and closes the dialog. (**Dialog**( ) function call returns the push button's ordinal number in the dialog. The first push button returns 1.)

Parameter **Description**

X This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.

Y This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.

DX This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.

DY This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.

Title$ The value of this string is the title of the push button control.

Field This identifier is the name of the field. The *dialogfunc* receives this name as *string*. If this identifer is omitted then the first two words of the title are used.

See Also **Begin Dialog**.

Example
```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,30,"Please push the DoIt button"
    OKButton 40,90,40,20
    PushButton 110,90,60,20,"&Do It"
  EndDialog
  Dim dlg As UserDialog
  Debug.PrintDialog(dlg)
End Sub
```

Put Instruction

Syntax Put [#]*StreamNum*, [*RecordNum*], *var*

Group File

Description Write a variable's value to *StreamNum*.

Parameter **Description**

StreamNum Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

RecordNum For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1. If this is omitted then the current position (or record number) is used.

var This variable value is written to the file. For a fixed length variable (like **Long**) the number of bytes required to store the variable are written. For a **Variant** variable two bytes which describe its type are written and then the variable value is written accordingly. For a *user type* variable each field is written in sequence. For an array variable each element is written in sequence. For a dynamic array variable the number of dimensions and range of each dimension is written prior to writing the array values. All binary data values are written to the file in *little-endian* format.

Note: When a writing string (or a dynamic array) to a Binary mode file the string length (or array dimension) information is not written. Only the string data or array elements are written.

See Also **Get**, **Open**.

Example '#Language "WWB-COM"
Sub Main
  Dim V As **Variant**
  Open "SAVE_V.DAT" **For** Binary Access **Write** As #1
  Put #1, , V
 '**Close** #1
End Sub

QBColor Function

Syntax QBColor(*num*)

Group Miscellaneous

Description Return the appropriate color defined by Quick Basic.

num **color**

0 black

1 blue

2 green

3 cyan

4 red

5 magenta

6 yellow

7 white

8 gray

9 light blue

10 light green

11 light cyan

12 light red

13 light magenta

14 light yellow

15 bright white

See Also **RGB( )**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print Hex**(QBColor(1)) '"800000"

```
  Debug.PrintHex(QBColor(7))  '"C0C0C0"
  Debug.Print Hex(QBColor(8))  '"808080"
  Debug.PrintHex(QBColor(9))  '"FF0000"
  Debug.Print Hex(QBColor(10)) '"FF00"
  Debug.Print Hex(QBColor(12)) '"FF"
  Debug.Print Hex(QBColor(15)) '"FFFFFF"
EndSub
```

RaiseEvent Instruction

Syntax RaiseEvent *name*[(*arglist*)]

Group Flow Control

Description Raise an **Event**. Evaluate the *arglist* and call *name* with those values.

See Also **Event**, **WithEvents**.

Example

```
'Class1.cls
'#Language "WWB-COM"
Event Changing(ByVal OldValue As String, ByVal NewValue As String)

Private Value_ As String

Property Get Value As String
  Value = Value_
EndProperty

Property Let Value(ByVal NewValue As String)
  RaiseEvent Changing(Value_, NewValue)
  Value_ = NewValue
EndProperty

'Macro.bas
'#Uses "Class1.cls"
'#Language "WWB-COM"

Dim WithEvents c1 As Class1

SubMain
  Set c1 = New Class1
  c1.Value = "Hello"
  c1.Value = "Goodbye"
End Sub

Sub c1_Changing(ByVal OldValue As String, ByVal NewValue As String)
```

```
  Debug.Print "OldValue=""" & OldValue & """, NewValue=""" & NewValue & """"
End Sub
```

Randomize Instruction

Syntax Randomize [*Seed*]

Group Math

Description Randomize the random number generator.

Parameter **Description**

Seed This numeric value sets the initial seed for the random number generator. If this value is omitted then the current time is used as the seed.

See Also **Rnd( )**.

Example

```
'#Language "WWB-COM"
Sub Main
  Randomize
  Debug.Print Rnd ' 0.??????????????
End Sub
```

ReDim Instruction

Syntax ReDim [Preserve] *vardeclaration* [*As type*][, ...]

Group Declaration

Description Redimension a dynamic *arrayvar* or *user defined type* array element. Use Preserve to keep the array values. Otherwise, the array values will all be reset. When using preserve only the last index of the array may change, but the number of indexes may not. (A one-dimensional array can't be redimensioned as a two-dimensional array.)

See Also **Dim**, **Option** Base, **Private**, **Public**, **Static**.

Example

```
'#Language "WWB-COM"
Sub Main
  Dim X()
  ReDim X(3)
  Debug.Print UBound(X) ' 3
  ReDim X(200)
  Debug.Print UBound(X) ' 200
End Sub
```

#Reference Special Comment

Syntax '#Reference {uuid}#vermajor.verminor#lcid#[path[#name]]

Description The Reference comment indicates that the current *macro*/*module* references the COM type library identified. Reference comment lines must be the first lines in the macro/module (following the global **Attribute**s). Reference comments are in reverse priority (from lowest to highest). The IDE does not display the reference comments.

Parameter **Description**

uuid Type library's universally unique identifier.

vermajor Type library's major version number.

verminor Type library's minor version number.

lcid Type library's locale identifier.

path Type library's path.

name Type library's name.

Example '#Reference {00025E01-0000-0000-C000-000000000046}#4.0#0#C:\PROGRAM FILES\COMMON FILES\MICROSOFT SHARED\DAO\DAO350.DLL#Microsoft DAO 3.5 **Object** Library

Rem Instruction

Syntax Rem ...
-or-
'...

Group Miscellaneous

Description Both forms are comments. The Rem form is an instruction. The ' form can be used at the end of any line. All text from either ' or Rem to the end of the line is part of the comment. That text is not executed.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print "Hello" ' prints to the output window
  Rem the macro terminates at Main's End Sub
End Sub
```

Replace$ Function

Syntax Replace[$](*S$*, *Pat$*, *Rep$*, [*Index*], [*Count*])

Group String

Description Replace *Pat$* with *Rep$* in *S$*.

Parameter **Description**

S$ This string value is searched. Replacements are made in the string returned by Replace.

Pat$ This string value is the pattern to look for.

Rep$ This string value is the replacement.

Index This numeric value is the starting index in *S$*. Replace(S,Pat,Rep,N) is equivalent to Replace(Mid (S,N),Pat,Rep). If this is omitted use 1.

Count This numeric value is the maximum number of replacements that will be done. If this is omitted use -1 (which means replace all occurrences).

See Also **InStr( )**, **InStrRev( )**, **Left$( )**, **Len( )**, **Mid$( )**, **Right$( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Replace$("abcabc","b","B")   '"aBcaBc"
  Debug.Print Replace$("abcabc","b","B",,1) '"aBcabc"
  Debug.Print Replace$("abcabc","b","B",3)  '"caBc"
  Debug.Print Replace$("abcabc","b","B",9)  ""
End Sub
```

Reset Instruction

Syntax Reset

Group File

Description Close all open streams for the current *macro/module*.

See Also **Close**, **Open**.

Example
```
'#Language "WWB-COM"
Sub Main
  ' read the first line of XXX and print it
  Open "XXX" ForInput As #1
  Line Input #1, L$
  Debug.Print L$
  Reset
End Sub
```

Resume Instruction

Syntax Resume *label*
-or-
Resume Next

Group Error Handling

Description Form 1: Resume execution at *label*.

Form 2: Resume execution at the next statement.

Once an error has occurred, the error handler can use Resume to continue execution. The error handler must use Resume or **Exit** at the end.

Note: This instruction clears the **Err** and sets **Error`$'** to null.

```
Example          '#Language "WWB-COM"
Sub Main
  On Error GoTo X
  Err.Raise 1
  Debug.Print "RESUMING"
  ExitSub

X:  Debug.Print "Err="; Err
  Resume Next
EndSub
```

Return Instruction

Syntax Return *expr*

Group Flow Control

Description The return instruction causes the **Function** block to exit with the value of the *expr*.

VBA This language element is not VBA compatible and requires the **#Language** "WWB-COM" setting.

```
Example          '#Language "WWB-COM"
Sub Main
  Debug.Print Func(2) ' 6
EndSub

Function Func(N)
  Return N*3
End Function
```

RGB Function

Syntax RGB(*red*, *green*, *blue*)

Group Miscellaneous

Description Return a color. Some useful color constants are predefined:

- vbBlack - same as RGB(0,0,0)
- vbRed - same as RGB(255,0,0)
- vbGreen - same as RGB(0,255,0)

- vbYellow - same as RGB(255,255,0)
- vbBlue - same as RGB(0,0,255)
- vbMagenta - same as RGB(255,0,255)
- vbCyan - same as RGB(0,255,255)
- vbWhite - same as RGB(255,255,255)

See Also **QBColor( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Hex(RGB(255,0,0)) '"FF0000"
EndSub
```

Right$ Function

Syntax Right[$](*S$*, *Len*)

Group String

Description Return the last *Len* chars of *S$*.

Note: A similar function, RightB, returns the last *Len* bytes.

Parameter **Description**

S$ Return the right portion of this string value. If this value is **Null** then **Null** is returned.

Len Return this many chars. If *S$* is shorter than that then just return *S$*.

See Also **InStr( )**, **InStrRev( )**, **Left$( )**, **Len( )**, **Mid$( )**, **Replace$( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Right$("Hello",3) '"llo"
EndSub
```

RmDir Instruction

Syntax RmDir *Name$*

Group File

Description Remove directory *Name$*.

Parameter **Description**

Name$ This string value is the path and name of the directory. A path relative to the current directory can be used.

See Also **MkDir**.

Example
```
'#Language "WWB-COM"
Sub Main
  RmDir "C:\WWTEMP"
End Sub
```

Rnd Function

Syntax Rnd([*Num*])

Group Math

Description Return a random number greater than or equal to zero and less than one.

Parameter **Description**

Num See table below.

Num **Description**

<0 Return the same number every time, using *Num* as the seed.

>0 Return the next random number in the sequence.

0 Return the most recently generated number.

omitted Return the next random number in the sequence.

See Also **Randomize**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Rnd() ' 0.??????????????
EndSub
```

Round Function

Syntax Round([*Num*][, *Places*])

Group Math

Description Return the number rounded to the specified number of decimal places.

Parameter **Description**

Num Round this numeric value. If this value is **Null** then **Null** is returned.

Places Round to this number of decimal places. If this is omitted then round to the nearest integer value.

```
Example         '#Language "WWB-COM"
Sub Main
  Debug.Print Round(.5)     ' 0
  Debug.Print Round(.500001) ' 1
  Debug.Print Round(1.499999) ' 1
  Debug.Print Round(1.5)    ' 2
  Debug.Print Round(11.11)   ' 11
  Debug.Print Round(11.11,1) ' 11.1
End Sub
```

RSet Instruction

Syntax RSet *strvar* = *str*

Group Assignment

Description Assign the value of *str* to *strvar*. Shorten *str* by removing trailing chars (or extend with leading blanks). The previous length *strvar* is maintained.

See Also **LSet**.

```
Example         '#Language "WWB-COM"
Sub Main
  S$ = "123"
  RSet S$ = "A"
  Debug.Print "."; S$; "." '". A."
EndSub
```

RTrim$ Function

Syntax RTrim[$](*S$*)

Group String

Description Return the string with *S$*'s trailing spaces removed.

Parameter **Description**

S$ Copy this string without the trailing spaces. If this value is **Null** then **Null** is returned.

See Also **LTrim$( )**, **Trim$( )**.

```
Example         '#Language "WWB-COM"
Sub Main
  Debug.Print "."; RTrim$("  x  "); "." '". x."
End Sub
```

SaveSetting Instruction

Syntax SaveSetting *AppName$*, *Section$*, *Key$*, *Setting*

Group Settings

Description Save the *Setting* for *Key* in *Section* in project *AppName*. Win16 and Win32s store settings in a .ini file named *AppName*. Win32/Win64 store settings in the registration database under "HKEY_CURRENT_USER\ Software\ VB and VBA Program Settings\ *AppName*\ *Section*\ *Key*". If *AppName* starts with "..\" then "VB and VBA Program Settings\" is omitted.

Parameter **Description**

AppName$ This string value is the name of the project which has this *Section* and *Key*.

Section$ This string value is the name of the section of the project settings.

Key$ This string value is the name of the key in the section of the project settings.

Setting Set the key to this value. (The value is stored as a string.)

Example
```
'#Language "WWB-COM"
Sub Main
  SaveSetting "MyApp","Font","Size",10
End Sub
```

Second Function

Syntax Second(*dateexpr*)

Group Time/Date

Description Return the second of the minute (0 to 59).

Parameter **Description**

dateexpr Return the second of the minute for this date value. If this value is **Null** then **Null** is returned.

See Also **Hour( )**, **Minute( )**, **Time( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Second(#12:00:01 AM#) ' 1
EndSub
```

Seek Function

Syntax Seek(*StreamNum*)

Group File

Description Return *StreamNum* current position. For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1.

Note: Unicode text files opened with Input mode use character positions, not byte positions.

Parameter **Description**

StreamNum Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

See Also **Seek**.

Example
```
'#Language "WWB-COM"
Sub Main
  Open "XXX" For Input As #1
  Debug.Print Seek(1) ' 1
  Line Input #1, L$
  Debug.Print Seek(1)
 'Close #1
End Sub
```

Seek Instruction

Syntax Seek [#]*StreamNum*, *Count*

Group File

Description Position *StreamNum* for input *Count*.

Note: Unicode text files opened with Input mode use character positions, not byte positions.

Parameter **Description**

StreamNum Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

Count For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1.

See Also **Seek( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Open "XXX" For Input As #1
  Line Input #1, L$
  Seek #1, 1 ' rewind to start of file
  Input #1, A
 'Close #1
  Debug.Print A
EndSub
```

Select Case Statement

Syntax **Select** Case *expr*
[Case *caseexpr*[, ...] [: *instruction*]
statements]...
[Case Else [: *instruction*]
statements]
EndSelect

Group Flow Control

Description Select the appropriate case by comparing the *expr* with each of the *caseexpr*s. Select the Case Else part if no *caseexpr* matches. (If the Case Else is omitted then skip the entire Select...End Select block.) Only the *statements* from the matched Case up to the next Case are executed.

caseexpr **Description**

expr Execute if equal.

Is < *expr* Execute if less than.

Is <= *expr* Execute if less than or equal to.

Is > *expr* Execute if greater than.

Is >= *expr* Execute if greater than or equal to.

Is <> *expr* Execute if not equal to.

expr1 To *expr2* Execute if greater than or equal to *expr1* and less than or equal to *expr2*.

See Also **If**, **Choose( )**, **IIf( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  S = InputBox("Enter hello, goodbye, dinner or sleep:")
  Select Case UCase(S)
  Case "HELLO", "HI"
    Debug.Print "come in"
  Case "GOODBYE", "BYE"
    Debug.Print "see you later"
  Case "DINNER"
    Debug.Print "Please come in."
    Debug.Print "Dinner will be ready soon."
  Case "SLEEP"
    Debug.Print "Sorry."
    Debug.Print "We are full for the night"
  Case Else
    Debug.Print "What?"
```

```
  End Select
EndSub
```

SendKeys Instruction

Syntax SendKeys *Keys$*[, *Wait*]

Group Miscellaneous

Description Send *Keys$* to Windows.

Parameter **Description**

Keys$ Send the keys in this string value to Windows. (Refer to table below.)

Wait If this is not zero then the keys are sent before executing the next instruction. If this is omitted or zero then the keys are sent during the following instructions.

Key **Description**

+ Shift modifier key: the following key is a shifted key

^ Ctrl modifier key: the following key is a control key

% Alt modifier key: the following key is an alt key

(keys) Modifiers apply to all keys

~ Send Enter key

k Send k Key (k is any single char)

K Send Shift k Key (K is any capital letter)

{special n} special key (n is an optional repeat count)

{mouse x,y} mouse key (x,y is an optional screen position)

{k} Send k Key (any single char)

{K} Send Shift k Key (any single char)

{Cancel} Send Break Key

{Esc} Send Escape Key

{Escape} Send Escape Key

{Enter} Send Enter Key

{Menu} Send Menu Key (Alt)

{Help} Send Help Key (?)

{Prtsc} Send Print Screen Key

{Print} Send

{Execute} Send ?

{Tab} Send

{Pause} Send Pause Key

{Tab} Send Tab Key

{BS} Send Back Space Key

{BkSp} Send Back Space Key

{BackSpace} Send Back Space Key

{Del} Send Delete Key

{Delete} Send Delete Key

{Ins} Send Insert Key

{Insert} Send Insert Key

{Left} Send Left Arrow Key

{Right} Send Right Arrow Key

{Up} Send Up Arrow Key

{Down} Send Down Arrow Key

{PgUp} Send Page Up Key

{PgDn} Send Page Down Key

{Home} Send Home Key

{End} Send End Key

{Select} Send ?

{Clear} Send Num Pad 5 Key

{Pad0..9} Send Num Pad 0-9 Keys

{Pad*} Send Num Pad * Key

{Pad+} Send Pad + Key

{PadEnter} Send Num Pad Enter

{Pad.} Send Num Pad . Key

{Pad-} Send Num Pad - Key

{Pad/} Send Num Pad / Key

{F1..24} Send F1 to F24 Keys

Mouse Mouse movement and button clicks:

- {Move x,y} - move the mouse to (x,y)
- {ClickLeft x,y} - move the mouse to (x,y) and click the left button. (This is the same as {DownLeft x,y} {UpLeft}.)
- {DoubleClickLeft x,y} - move the mouse to (x,y) and click the left button. (This is NOT the same as {Click-Left x,y}{ClickLeft}.)
- {DownLeft x,y} - move the mouse to (x,y) and push the left button down.
- {UpLeft x,y} - move the mouse to (x,y) and release the left button.
- {...Middle x,y} - similarly named keys for the middle mouse button.
- {...Right x,y} - similarly named keys for the right mouse button.

The x,y values are screen pixel locations, where (0,0) is in the upper-left corner. In all cases the x,y is optional. If omitted, the previous mouse position is used.

See Also **AppActivate**, **KeyName**, **Shell( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  SendKeys "%S"    ' send Alt-S (Search)
  SendKeys "GoTo~~" ' send G o T o {Enter} {Enter}
EndSub
```

SetAttr Instruction

Syntax SetAttr *Name$*, *Attrib*

Group File

Description Set the *attribute*s for file *Name$*. If the file does not exist then a run-time error occurs.

Parameter **Description**

Name$ This string value is the path and name of the file. A path relative to the current directory can be used.

Attrib Set the file's *attribute*s to this numeric value.

Example
```
'#Language "WWB-COM"
Sub Main
  Attrib = GetAttr("XXX")
  SetAttr "XXX",1 ' readonly
  Debug.Print GetAttr("XXX") ' 1
  SetAttr "XXX",Attrib
EndSub
```

Set Instruction

Syntax Set *objvar* = *objexpr*
-or-
Set *objvar* = **New***objtype*

Group Assignment

Description Form 1: Set *objvar*'s object reference to the object reference of *objexpr*.

Form 2: Set *objvar*'s object reference to the a new instance of *objtype*.

The Set instruction is how object references are assigned.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim App As Object
  Set App = CreateObject("WinWrap.CppDemoApplication")
  App.Move 20,30 ' move icon to 20,30
  Set App = Nothing
  App.Quit    ' run-time error (no object)
EndSub
```

SetLocale Instruction

Syntax SetLocale *LocaleID*

Group Miscellaneous

Description Set the *LocaleID* for the current thread.

Pocket PC Not supported.

Parameter **Description**

LocaleID Set the current thread's locale to this value.

See Also **GetLocale**.

Example
```
'#Language "WWB-COM"
Sub Main
  SetLocale &H409 ' English, US
End Sub
```

Shell Function

Syntax Shell(*Name$*[, *WindowType*])

Group Miscellaneous

Description Execute program *Name$*. This is the same as using File|Run from the Program Manager. This instruction can run .COM, .EXE, .BAT and .PIF files. If successful, return the task ID.

Pocket PC The *WindowType* parameter is ignored.

Parameter **Description**

Name$ This string value is the path and name of the program to run. Command line arguments follow the program name. (A long file name containing a space must be surrounded by literal double quotes.)

WindowType This controls how the application's main window is shown. See the table below.

WindowType **Value Effect**

vbHide 0 Hide Window

vbNormalFocus 1, 5, 9 Normal Window

vbMinimizedFocus

2 Minimized Window (default)

vbMaximizedFocus

3 Maximized Window

vbNormalNoFocus

4, 8 Normal Deactivated Window

vbMinimizedNoFocus

6, 7 Minimized Deactivated Window

See Also **AppActivate**, **SendKeys**.

Example
```
'#Language "WWB-COM"
Sub Main
  X = Shell("Calc") ' run the calc program
  AppActivate X
  SendKeys "% R" ' restore calc's main window
  SendKeys "30*2{+}10=",1 '70
EndSub
```

ShowPopupMenu Function

Syntax ShowPopupMenu(*StrArray$( )*[, *PopupStyle*][, *XPos*, *YPos*])

Group User Input

Description Show a popup menu and return the number of the item selected. The item number is the index of the StrArray selected minus LBound(StrArray). The value -1 is returned in no menu item is selected.

Parameter **Description**

StrArray$( ) This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.

PopupMenuStyle This controls how the popup menu is aligned. Any combination of styles may used together. See the table below.

XPos When the menu is put up the alignment will be at this window position. If this is omitted then the current mouse position is used.

YPos When the menu is put up the alignment will be at this window position. If this is omitted then the current mouse position is used.

PopupStyle Value Effect

vbPopupLeftTopAlign 0 Align menu left edge at XPos and top at YPos. (default)

vbPopupUseLeftButton 1 User can select menu choices with the left mouse button only.

vbPopupUseRightButton 2 User can select menu choices with the left or right mouse button.

vbPopupRightAlign 4 Align menu with right edge at the XPos.

vbPopupCenterAlign 8 Align menu center at the XPos.

vbPopupVCenterAlign 16 Align menu center at the YPos.

vbPopupBottomAlign 32 Align menu bottom at the YPos.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim Items(0 To 2) As String
  Items(0) = "Item &1"
  Items(1) = "Item &2"
  Items(2) = "Item &3"
  X = ShowPopupMenu(Items) ' show popup menu
  Debug.Print X ' item selected
End Sub
```

Sin Function

Syntax Sin(*Num*)

Group Math

Description Return the sine.

Parameter Description

Num Return the sine of this numeric value. This is the number of radians. There are 2*Pi radians in a full circle.

See Also **Atn**, **Cos**, **Tan**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** Sin(1) ' 0.8414709848079
EndSub

Space$ Function

Syntax Space[$](*Len*)

Group String

Description Return the string *Len* spaces long.

Parameter **Description**

Len Create a string this many spaces long.

See Also **String$( )**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** "."; Space$(3); "." '". ."
EndSub

Split Function

Syntax Split(*Str*, [*Sep*], [*Max*])

Group Miscellaneous

Description Return a string array containing substrings from the original string.

Parameter **Description**

Str Extract substrings from this string value.

Sep Look for this string value to separate the substrings. (Default: " ")

Max Create at most this many substrings. (Default -1, which means create as many as are found.)

See Also **Join( )**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** Split("1 2 3")(1) '"2"
EndSub

Sqr Function

Syntax Sqr(*Num*)

Group Math

Description Return the square root.

Parameter **Description**

Num Return the square root of this numeric value.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Sqr(9) ' 3
End Sub
```

Static Definition

Syntax Static *vardeclaration*[, ...]

Group Declaration

Description A static variable retains it value between *procedure* calls. Dimension var array(s) using the *dimension*s to establish the minimum and maximum index value for each dimension. If the *dimension*s are omitted then a scalar (single value) variable is defined. A dynamic array is declared using ( ) without any *dimension*s. It must be **ReDim**ensioned before it can be used.

See Also **Dim**, **Option** Base, **Private**, **Public**, **ReDim**.

Example
```
'#Language "WWB-COM"
Sub A
  Static X
  Debug.Print X
  X = "Hello"
EndSub

Sub Main
  A
  A ' prints "Hello"
EndSub
```

StrComp$ Function

Syntax StrComp(*Str1*,*Str2*,*Comp*)

Group String

Description Compare two strings.

Parameter **Description**

Str1 Compare this string with *Str2*. If this value is **Null** then **Null** is returned.

Str2 Compare this string with *Str1*. If this value is **Null** then **Null** is returned.

Comp This numeric value indicates the type of comparison. See Comp table below.

Result **Description**

-1 *Str1* is less than *Str2*.

0 *Str1* is equal to *Str2*.

1 *Str1* is greater than *Str2*.

Null *Str1* or *Str2* is **Null**.

Comp **Value Effect**

vbUseCompareOption -1 Performs the comparison using the **Option** Compare statement value.

vbBinaryCompare 0 Compares the string's binary data.

vbTextCompare 1 Compares the string's text using the collation rules.

vbDatabaseCompare 2 Microsoft Access only. (Not supported.)

See Also **LCase$( )**, **Option** Compare, **StrConv$( )**, **UCase$( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print StrComp("F","e") ' -1
  Debug.Print StrComp("F","e",1) ' 1
  Debug.Print StrComp("F","f",1) ' 0
EndSub
```

StrConv$ Function

Syntax StrConv[$](*Str*,*Conv*)

Group String

Description Convert the string.

Parameter **Description**

Str Convert this string value. If this value is **Null** then **Null** is returned.

Conv This numeric value indicates the type of conversion. See conversion table below.

Conv **Value Effect**

vbUpperCase 1 Convert a String to upper case.

vbLowerCase 2 Convert a String to lower case.

vbProperCase 3 Convert a String to proper case. (Not supported.)

vbWide 4 Convert a String to wide. (Only supported for eastern locales.)

vbNarrow 8 Convert a String to narrow. (Only supported for eastern locales.)

vbKatakana 16 Convert a String to Katakana. (Only supported for Japanese locales.)

vbHiragana 32 Convert a String to Hiragana. (Only supported for Japanese locales.)

vbUnicode or vbFromANSIBytes 64 Convert an ANSI (locale dependent) byte array to a Unicode string.

vbANSI 4160 Convert an ANSI (locale dependent) string to a Unicode string.

vbFromUnicode or vbANSIBytes 128 Convert from Unicode to an ANSI (locale dependent) byte array.

vbANSI 4224 Convert from Unicode to an ANSI (locale dependent) string.

vbUTF8 4352 Convert a Unicode string to a UTF-8 string.

vbUTF8Bytes 256 Convert a Unicode string to a UTF-8 byte array.

vbFromUTF8 4608 Convert a UTF-8 string to a Unicode string.

vbFromUTF8Bytes 512 Convert a UTF-8 byte array to a Unicode string.

vbToBytes 1024 Convert a String to a byte array containing the low byte of each char.

vbFromBytes 2048 Convert a byte array to a String by setting the low byte of each char.

Conversion Rules If multiple conversions are specified, the conversions occur in this order:

- vbFromBytes, vbUnicode, vbFromANSI, vbFromANSIBytes, vbFromUTF8 or vbFromUTF8Bytes (choose one, optional)
- vbUpperCase, vbLowerCase, vbWide, vbNarrow, vbKatakana or vbHiragana (choose one or more, optional)
- vbToBytes, vbFromUnicode, vbANSI, vbANSIBytes, vbUTF8 or vbUTF8Bytes (choose one, optional)

See Also **LCase$( )**, **StrComp( )**, **UCase$( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim B(1 To 3) As Byte
  B(1) = 65
  B(2) = 66
  B(3) = 67
  Debug.Print StrConv$(B,vbUnicode) '"ABC"
EndSub
```

Str$ Function

Syntax Str[$](*Num*)

Group String

Description Return the string representation of *Num*.

Parameter **Description**

Num Return the string representation of this numeric value. Positive values begin with a blank. Negative values begin with a dash '-'.

See Also **CStr( )**, **Hex$( )**, **Oct$( )**, **Val( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Str$(9*9) ' 81
EndSub
```

StrReverse$ Function

Syntax StrReverse[$](*S*)

Group String

Description Return the string with the characters in reverse order.

Parameter **Description**

S Return this string with the characters in reverse order.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print StrReverse$("ABC") 'CBA
EndSub
```

Sub Definition

Syntax
```
[ | Private | Public | Friend ] _
Sub name[([param[, ...]])]
  statements
End Sub
```

Group Declaration

Description User defined subroutine. The subroutine defines a set of *statement*s to be executed when it is called. The values of the calling *arglist* are assigned to the *param*s. A subroutine does not return a result.

Access If no access is specified then **Public** is assumed.

See Also **Declare**, **Function**, **Property**.

Example
```
'#Language "WWB-COM"
Sub IdentityArray(A()) ' A() is an array of numbers
  For I = LBound(A) To UBound(A)
    A(I) = I
  Next I
End Sub
```

```
Sub CalcArray(A(), B, C) ' A() is an array of numbers
  For I = LBound(A) To UBound(A)
    A(I) = A(I)*B+C
  Next I
End Sub

Sub ShowArray(A()) ' A() is an array of numbers
  For I = LBound(A) To UBound(A)
    Debug.Print "("; I; ")="; A(I)
  Next I
End Sub

Sub Main
  Dim X(1 To 4)
  IdentityArray X()   ' X(1)=1, X(2)=2, X(3)=3, X(4)=4
  CalcArray X(), 2, 3 ' X(1)=5, X(2)=7, X(3)=9, X(4)=11
  ShowArray X()       ' print X(1), X(2), X(3), X(4)
End Sub
```

String$ Function

Syntax String[$](*Len*, *Char*)

Group String

Description Return the string *Len* long filled with *Char* or the first char of *Char$*.

Parameter **Description**

Len Create a string this many chars long.

Char Fill the string with this char value. If this is a numeric value then use the ASCII char equivalent. If this is a string value use the first char of that string. If this value is **Null** then **Null** is returned.

See Also **Space$( )**.

Example

```
'#Language "WWB-COM"
Sub Main
  Debug.Print String$(4,65)   '"AAAA"
  Debug.Print String$(4,"ABC") '"AAAA"
End Sub
```

Tan Function

Syntax Tan(*Num*)

Group Math

Description Return the tangent.

Parameter **Description**

Num Return the tangent of this numeric value.

See Also **Atn**, **Cos**, **Sin**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Tan(1) ' 1.5574077246549
EndSub
```

TextBox Dialog Item Definition

Syntax TextBox *X*, *Y*, *DX*, *DY*, .*Field$*[, *Options*]

Group User Dialog

Description Define a textbox item.

Parameter **Description**

X This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.

Y This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.

DX This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.

DY This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.

Field The value of the text box is accessed via this field.

Options This numeric value controls the type of text box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option **Description**

0 Text box allows a single line of text to be entered.

1 Text box allows multiple lines of text can be entered.

2 Locked text box displays multiple lines of text.

-1 Text box allows a hidden password can be entered.

See Also **Begin Dialog**.

Example
```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    TextBox 10,25,180,20,.Text$
```

```
    OKButton 80,90,40,20
  EndDialog
  Dim dlg As UserDialog
  dlg.Text$ = "none"
  Dialog dlg ' show dialog (wait for ok)
  Debug.Print dlg.Text$
EndSub
```

Text Dialog Item Definition

Syntax Text *X*, *Y*, *DX*, *DY*, *Title$*[, .*Field*][, *Options*]

Group User Dialog

Description Define a text item.

Parameter **Description**

X This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.

Y This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.

DX This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.

DY This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.

Title$ The value of this string is the title of the text control.

Field This identifier is the name of the field. The *dialogfunc* receives this name as *string*. If this identifer is omitted then the first two words of the title are used.

Options This numeric value controls the alignment of the text. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option **Description**

0 Text is left aligned.

1 Text is right aligned.

2 Text is centered.

See Also **Begin Dialog**.

Example

```
'#Language "WWB-COM"
Sub Main
  Begin Dialog UserDialog 200,120
    Text 10,10,180,15,"Please push the OK button"
    OKButton 80,90,40,20
  End Dialog
```

```
  Dim dlg As UserDialog
  Dialog dlg ' show dialog (wait for ok)
End Sub
```

Time Function

Syntax Time[$]

Group Time/Date

Description Return the current time as a **Date** value.

See Also **Date**, **Now**, **Timer**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Time ' example: 09:45:00 am
EndSub
```

Timer Function

Syntax Timer

Group Time/Date

Description Return the number of seconds past midnight. (This is a real number, accurate to about 1/18th of a second.)

See Also **Date**, **Now**, **Time**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Timer ' example: 45188.13
EndSub
```

TimeSerial Function

Syntax TimeSerial(*Hour*, *Minute*, *Second*)

Group Time/Date

Description Return a **Date** value.

Parameter **Description**

Hour This numeric value is the hour (0 to 23).

Minute This numeric value is the minute (0 to 59).

Second This numeric value is the second (0 to 59).

See Also **DateSerial**, **DateValue**, **TimeValue**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** TimeSerial(13,30,0) '1:30:00 PM
EndSub

TimeValue Function

Syntax TimeValue(*Date$*)

Group Time/Date

Description Return the time part of date encoded as a string value.

Parameter **Description**

Date$ Convert this string value to the time part of date it represents.

See Also **DateSerial**, **DateValue**, **TimeSerial**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** TimeValue("1/1/2000 12:00:01 AM")
    '12:00:01 AM
EndSub

Trim$ Function

Syntax Trim[$](*S$*)

Group String

Description Return the string with *S$*'s leading and trailing spaces removed.

Parameter **Description**

S$ Copy this string without the leading or trailing spaces. If this value is **Null** then **Null** is returned.

See Also **LTrim$( )**, **RTrim$( )**.

Example '#Language "WWB-COM"
Sub Main
  Debug.**Print** "."; Trim$(" x "); "." '".x."
End Sub

Type Definition

Syntax [| **Private** | **Public**] _
Type *name*
elem [([*dimension*[, ...]])] *As [New] type*
[...]
End Type

Group Declaration

Description Define a new *user type*. Each *elem* defines an element of the type for storing data. *As [New] type* defines the type of data that can be stored. A *user defined type variable* has a value for each *elem*. Use .*elem* to access individual element values.

Access If no access is specified then **Public** is assumed.

Example
```
'#Language "WWB-COM"
Type Employee
  FirstName As String
  LastName As String
  Title As String
  Salary As Double
End Type

SubMain
  Dim e As Employee
  e.FirstName = "John"
  e.LastName = "Doe"
  e.Title = "President"
  e.Salary = 100000
  Debug.Print e.FirstName '"John"
  Debug.Print e.LastName  '"Doe"
  Debug.Print e.Title     '"President"
  Debug.Print e.Salary    ' 100000
EndSub
```

TypeName Function

Syntax TypeName[$](*var*)

Group Variable Info

Description Return a string indicating the type of value stored in *var*.

Parameter **Description**

var Return a string indicating the type of value stored in this variable.

Result **Description**

Empty *Variant* variable is empty. It has never been assigned a value.

Null *Variant* variable is null.

Boolean Variable contains a **Boolean** value.

Byte Variable contains a **Byte** value.

SByte Variable contains a **SByte** value.

Short Variable contains an **Short** value.

UShort Variable contains an **UShort** value.

Integer Variable contains an **Integer** value.

UInteger Variable contains an **UInteger** value.

Long Variable contains a **Long** value.

ULong Variable contains a **ULong** value.

Huge_ Variable contains a **Huge_** value.

UHuge_ Variable contains a **UHuge_** value.

Decimal Variable contains a **Decimal** value.

Single Variable contains a **Single** value.

Double Variable contains a **Double** value.

Currency Variable contains a **Currency** value.

Date Variable contains a **Date** value.

String Variable contains a **String** value.

Object Variable contains an **Object** reference that is not Nothing. (An object may return a type name specific to that type of object.)

Nothing Variable contains an **Object** reference that is Nothing.

Error Variable contains a error code value.

Variant Variable contains a variant value. (Only used for arrays of variants.)

Unknown Variable contains a non-ActiveX Automation object reference.

( ) Variable contains an array value. The TypeName of the element followed by ( ).

See Also **VarType**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim X As Variant
  Debug.Print TypeName(X) '"Empty"
```

```
  X = 1
  Debug.Print TypeName(X) '"Integer"
  X = 100000
  Debug.Print TypeName(X) '"Long"
  X = 1.1
  Debug.Print TypeName(X) '"Double"
  X = "A"
  Debug.Print TypeName(X) '"String"
  Set X = CreateObject("Word.Basic")
  Debug.Print TypeName(X) '"Object"
  X = Array(0,1,2)
  Debug.Print TypeName(X) '"Variant()"
End Sub
```

TypeOf Operator

Syntax TypeOf *expr***Is** *objtype*

Group Operator

Description Return the **True** if *expr* refers to an object of *objtype*.

See Also **Objects**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print TypeOf Err Is ErrObject ' True
End Sub
```

UBound Function

Syntax UBound(*arrayvar*[, *dimension*])

Group Variable Info

Description Return the highest index.

Parameter **Description**

arrayvar Return the highest index for this array variable.

dimension Return the highest index for this dimension of *arrayvar*. If this is omitted then return the highest index for the first dimension.

See Also **LBound( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim A(3,6)
  Debug.Print UBound(A)   ' 3
```

```
  Debug.Print UBound(A,1) ' 3
  Debug.Print UBound(A,2) ' 6
EndSub
```

UCase$ Function

Syntax UCase[$](*S$*)

Group String

Description Return a string from *S$* where all the lowercase letters have been uppercased.

Parameter **Description**

S$ Return the string value of this after all chars have been converted to lowercase. If this value is **Null** then **Null** is returned.

See Also **LCase$( )**, **StrComp( )**, **StrConv$( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print UCase$("Hello") '"HELLO"
EndSub
```

UInteger Data Type

Syntax **Dim** v As UInteger

Group Data Type

Description A 16 bit unsigned integer value. The number of bits is controlled by the **#Language** setting.

VBA This language element is not VBA compatible and requires the **#Language** "WWB-COM" setting.

ULong Data Type

Syntax **Dim** v As ULong

Group Data Type

Description A 32 bit unsigned integer value. The number of bits is controlled by the **#Language** setting.

VBA This language element is not VBA compatible and requires the **#Language** "WWB-COM" setting.

Unlock Instruction

Syntax Unlock *StreamNum*
-or-
Unlock *StreamNum*, *RecordNum*
-or-
Unlock *StreamNum*, [*start*] To *end*

Group File

Description Form 1: Unlock all of *StreamNum*.

Form 2: Unlock a record (or byte) of *StreamNum*.

Form 3: Unlock a range of records (or bytes) of *StreamNum*. If *start* is omitted then unlock starting at the first record (or byte).

Note: For sequential files (Input, Output and Append) unlock always affects the entire file.

Parameter **Description**

StreamNum Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

RecordNum For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1.

start First record (or byte) in the range.

end Last record (or byte) in the range.

See Also **Lock**, **Open**.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim V As Variant
  Open "SAVE_V.DAT" For Binary As #1
  Lock #1
  Get #1, 1, V
  V = "Hello"
  Put #1, 1, V
  Unlock #1
 'Close #1
EndSub
```

#Uses Special Comment

Syntax '#Uses "*module*" [Only:[**Win16**|**Win32**|Win64]] ...
-or-
'$Include: "*module*"

Group Declaration

Description The Uses comment indicates that the current *macro*/*module* uses public and friend symbols from the *module*. The Only option indicates that the module is only loaded for that Windows platform.

Parameter **Description**

module Public and Friend symbols from this module are accessible. If the module name is a relative path then the path is relative to the macro/module containing the Uses comment. For example, if module "A:\B\C\D.BAS" has this uses comment:
'#Uses "E.BAS"
then it uses "A:\B\C\E.BAS".

See Also **Class Module**, **Code Module**, **Object Module**.

Example
```
'Macro A.WWB
'#Language "WWB-COM"
'#Uses "B.BAS"
SubMain
  Debug.Print BFunc$("Hello") '"HELLO"
End Sub

'Module B.BAS
'#Language "WWB-COM"
Public Function BFunc$(S$)
  BFunc$ = UCase(S$)
End Function
```

Val Function

Syntax Val(*S$*)

Group String

Description Return the value of the *S$*.

Parameter **Description**

S$ Return the numeric value for this string value. A string value begins with &O is an octal number. A string value begins with &H is a hex number. Otherwise it is decimal number.

Example
```
'#Language "WWB-COM"
Sub Main
```

```
  Debug.Print Val("-1000") '-1000
EndSub
```

Variant Data Type

Syntax **Dim** v As Variant

Group Data Type

Description An empty, numeric, currency, date, string, object, error code, null or array value.

VarType Function

Syntax VarType(*var*)

Group Variable Info

Description Return a number indicating the type of value stored in *var*.

Parameter **Description**

var Return a number indicating the type of value stored in this variable.

Result **Value Description**

vbEmpty 0 *Variant* variable is Empty. It has never been assigned a value.

vbNull 1 *Variant* variable is null.

vbInteger 2 Variable contains an **Integer** value. The value depends on the **#Language** setting.

vbLong 3 Variable contains a **Long** value. The value depends on the **#Language** setting.

vbSingle 4 Variable contains a **Single** value.

vbDouble 5 Variable contains a **Double** value.

vbCurrency 6 Variable contains a **Currency** value.

vbDate 7 Variable contains a **Date** value.

vbString 8 Variable contains a **String** value.

vbObject 9 Variable contains an **Object** reference. If the object reference supports a default property the VarType of the default property's value is returned instead of vbObject.

vbError 10 Variable contains a error code value.

vbBoolean 11 Variable contains a **Boolean** value.

vbVariant 12 Variable contains a variant value. (Only used for arrays of variants.)

vbDataObject 13 Variable contains a non-ActiveX Automation object reference.

vbDecimal 14 Variable contains a **Decimal** value.

vbSByte 16 Variable contains a **SByte** value.

vbByte 17 Variable contains a **Byte** value.

vbUInteger 18 Variable contains a **UInteger** value. The value depends on the **#Language** setting.

vbULong 19 Variable contains a **ULong** value. The value depends on the **#Language** setting.

vbHuge_ 20 Variable contains a **Huge_** value.

vbUHuge_ 21 Variable contains a **UHuge_** value.

vbUserDefinedType 36 Variable contains a User Defined **Type** value.

+vbArray 8192 Variable contains an array value. Use VarType( ) And 255 to get the type of element stored in the array.

See Also **TypeName**.

Example '#Language "WWB-COM"

```
Sub Main
  Dim X As Variant
  Debug.Print VarType(X) ' 0
  X = 1
  Debug.Print VarType(X) ' 2
  X = 100000
  Debug.Print VarType(X) ' 3
  X = 1.1
  Debug.Print VarType(X) ' 5
  X = "A"
  Debug.Print VarType(X) ' 8
  Set X = CreateObject("Word.Basic")
  Debug.Print VarType(X) ' 9
  X = Array(0,1,2)
  Debug.Print VarType(X) ' 8204 (8192+12)
End Sub
```

Wait Instruction

Syntax Wait [*Delay*]

Group Miscellaneous

Description Wait for *Delay* seconds.

Parameter **Description**

Delay Wait for this number of seconds. If omitted then wait for 5 seconds. If this value is **Null** then **Null** is returned.

Example '#Language "WWB-COM"

```
Sub Main
```

```
  Wait .5 ' wait for one half second
End Sub
```

Weekday Function

Syntax Weekday(*dateexpr*)

Group Time/Date

Description Return the weekday.

- vbSunday (1) - Sunday
- vbMonday (2) - Monday
- vbTuesday (3) - Tuesday
- vbWednesday (4) - Wednesday
- vbThursday (5) - Thursday
- vbFriday (6) - Friday
- vbSaturday (7) - Saturday

Parameter **Description**

dateexpr Return the weekday for this date value. If this value is **Null** then **Null** is returned.

See Also **Date( )**, **Day( )**, **Month( )**, **WeekdayName( )**, **Year( )**.

Example

```
'#Language "WWB-COM"
Sub Main
  Debug.Print Weekday(#1/1/1900#) ' 2
  Debug.Print Weekday(#1/1/2000#) ' 7
EndSub
```

WeekdayName Function

Syntax WeekdayName(NumZ{day}[, CondZ{abbrev}])

Group Time/Date

Description Return the localized name of the weekday.

Parameter **Description**

day Return the localized name of this weekday. (1-7)

abbrev If this conditional value is **True** then return the abbreviated form of the weekday name.

See Also **Weekday( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print WeekdayName(1) 'Sunday
  Debug.Print WeekdayName(Weekday(Now))
EndSub
```

While Statement

Syntax
```
While condexpr
  statements
Wend
```

Group Flow Control

Description Execute *statements* while *condexpr* is **True**.

See Also **Do**, **For**, **For Each**, **Exit** While.

Example
```
'#Language "WWB-COM"
Sub Main
  I = 2
  While I < 10
    I = I*2
  Wend
  Debug.Print I ' 16
EndSub
```

Win16 Keyword

Group Constant

Description **True** if running in 16 bits. **False** if running in 32 or 64 bits.

Win32 Keyword

Group Constant

Description **True** if running in 32 bits. **False** if running in 16 or 64 bits.

Win64 Keyword

Group Constant

Description **True** if running in 64 bits. **False** if running in 16 or 32 bits.

WithEvents Definition

Syntax [**Dim** | **Private** | **Public**] _
WithEvents *name* As *objtype*[, ...]

Group Declaration

Description Dimensioning a module level variable WithEvents allows the macro to implement event handling **Sub**s. The variable's As type must be a type from a referenced type library (or language extension) which implements events.

See Also **Dim**, **Private**, **Public**, **RaiseEvent**.

Example
```
'#Language "WWB-COM"
Dim WithEvents X As Thing

SubMain
  Set X = New Thing
  X.DoIt ' DoIt method raises DoingIt event
End Sub

Private Sub X_DoingIt
  Debug.Print "X.DoingIt event"
End Sub
```

With Statement

Syntax
```
With objexpr
  statements
End With
```

Group Object

Description *Method* and *property* references may be abbreviated inside a With block. Use .*method* or .*property* to access the object specified by the With *objexpr*.

Example
```
'#Language "WWB-COM"
Sub Main
  Dim App As Object
  Set App = CreateObject("WinWrap.CppDemoApplication")
  With App
    .Move 20,30 ' move icon to 20,30
  End With
EndSub
```

Write Instruction

Syntax Write #*StreamNum*, *expr*[, ...]

Group File

Description Write's *expr*(s) to *StreamNum*. String values are quoted. Null values are written as #NULL#. Boolean values are written as #FALSE# or #TRUE#. Date values are written as #date#. Error codes are written as #ERROR number#.

See Also **Input**, **Line Input**, **Print**.

Example
```
'#Language "WWB-COM"
Sub Main
  A = 1
  B = 2
  C$ = "Hello"
  Open "XXX" For Output As #1
  Write #1, A, B, C$
 'Close #1
End Sub
```

WWB-COM_Topic

Syntax Abs(*Num*)

Group Math

Description Return the absolute value.

Parameter	Description
Num	Return the absolute value of this numeric value. '

See Also **Sgn**.

Example
```
'#Language "WWB.NET"
Sub Main
  Debug.Print Abs(9)  ' 9
  Debug.Print Abs(0)  ' 0
  Debug.Print Abs(-9) ' 9
End Sub
```

WWB-COM: Overview

WWB-COM Use **'#Language "WWB-COM"** for Visual Basic for Applications(TM) compatibility.

Declaration **'#Language**, **'#Reference**, **'#Uses**, **Attribute**, **Class Module**, **Code Module**, **Const**, **Declare**, **Def**type, **Delegate**, **Dim**, **Enum...End Enum**, **Event**, **Function...End Function**, **Object Module**, **Option**, **Private**, **Property...End Property**, **Public**, **ReDim**, **Static**, **Sub...End Sub**, **Type...End Type**, **WithEvents**.

Data Type **Any**, **Boolean**, **Byte**, **Currency**, **Date**, **Decimal**, **Double**, **Huge_**, **Integer**, **Long**, **Object**, **PortInt**, **SByte**, **Single**, **String**, **String*n**, **UHuge_**, **UInteger**, **ULong**, **Variant**, *obj type*, *user dialog*, *user enum*, *user type*.

Assignment **Assign**: (=, +=, -=, *=, /=, \=, ^=, <<=, >>=), **Erase**, **Let**, **LSet**, **RSet**, **Set**.

Flow Control **Call**, **CallByName**, **Do...Loop**, **End**, **Exit**, **For...Next**, **For Each...Next**, **GoTo**, **If...ElseIf...Else...End If**, **MacroCheck**, **MacroCheckThis**, **MacroRun**, **MacroRunThis**, **ModuleLoad**, **ModuleLoadThis**, **RaiseEvent**, **Return**, **Select Case...End Select**, **Stop**, **While...Wend**.

Error Handling **Err**, **Error**, **On Error**, **Resume**.

Conversion **Array**, **CBool**, **CByte**, **CCur**, **CDate**, **CDec**, **CDbl**, **CHuge_**, **CInt**, **CLng**, **CSByte**, **CSng**, **CStr**, **CUHuge_**, **CUInt**, **CULng**, **CVar**, **CVDate**, **CVErr**, **Val**.

Variable Info **IsArray**, **IsDate**, **IsEmpty**, **IsError**, **IsMissing**, **IsNull**, **IsNumeric**, **IsObject**, **LBound**, **TypeName**, **UBound**, **VarType**,

Constant **Empty**, **False**, **Nothing**, **Null**, **True**, **Win16**, **Win32**, **Win64**.

Math **Abs**, **Atn**, **Cos**, **Exp**, **Fix**, **Int**, **Log**, **Randomize**, **Rnd**, **Round**, **Sgn**, **Sin**, **Sqr**, **Tan**.

String **Asc**, **AscB**, **AscW**, **Chr**, **ChrB**, **ChrW**, **Format**, **Hex**, **InStr**, **InStrB**, **InStrRev**, **Join**, **LCase**, **Left**, **LeftB**, **Len**, **LenB**, **LTrim**, **Mid**, **MidB**, **Oct**, **Replace**, **Right**, **RightB**, **RTrim**, **Space**, **Split**, **Str**, **StrComp**, **StrConv**, **StrReverse**, **String**, **Trim**, **UCase**.

Object **CreateObject**, **GetObject**, **Me**, **With...End With**.

Time/Date **Date**, **DateAdd**, **DateDiff**, **DatePart**, **DateSerial**, **DateValue**, **Day**, **Hour**, **Minute**, **Month**, **MonthName**, **Now**, **Second**, **Time**, **Timer**, **TimeSerial**, **TimeValue**, **Weekday**, **WeekdayName**, **Year**.

File **ChDir**, **ChDrive**, **Close**, **CurDir**, **Dir**, **EOF**, **FileAttr**, **FileCopy**, **FileDateTime**, **FileLen**, **FreeFile**, **Get**, **GetAttr**, **Input**, **Input**, **Kill**, **Line Input**, **Loc**, **Lock**, **LOF**, **MkDir**, **Name**, **Open**, **Print**, **Put**, **Reset**, **RmDir**, **Seek**, **Seek**, **SetAttr**, **Unlock**, **Write**,

User Input **Dialog**, **GetFilePath**, **InputBox**, **MsgBox**. **ShowPopupMenu**

User Dialog **Begin Dialog...End Dialog**, **CancelButton**, **CheckBox**, **ComboBox**, **DropListBox**, **GroupBox**, **ListBox**, **MultiListBox**, **OKButton**, **OptionButton**, **OptionGroup**, **Picture**, **PushButton**, **Text**, **TextBox**.

Dialog Function **DialogFunc**, **DlgControlId**, **DlgCount**, **DlgEnable**, **DlgEnd**, **DlgFocus**, **DlgListBoxArray**, **DlgName**, **DlgNumber**, **DlgSetPicture**, **DlgText**, **DlgType**, **DlgValue**, **DlgVisible**.

DDE **DDEExecute**, **DDEInitiate**, **DDEPoke**, **DDERequest**, **DDETerminate**, **DDETerminateAll**.

Settings **DeleteSetting**, **GetAllSettings**, **GetSetting**, **SaveSetting**

Miscellaneous AboutWinWrapBasic, **AppActivate**, **Assign**, **Attribute**, **Beep**, **CallersLine**, **Choose**, **Clipboard**, **Command**, **Decode64**, **Decode64B**, **Decrypt64**, **Decrypt64B**, **Debug**.Print, **DoEvents**, **Encode64**, **Encode64B**, **Encrypt64**, **Encrypt64B**, **Environ**, **Eval**, **IIf**, **GetLocale**, **KeyName**, **MacroDir**, **QBColor**, **Rem**, **RGB**, **SendKeys**, **SetLocale**, **Shell**, **Wait**.

Operator Operators: +, -, ^, *, /, \, Mod, +, -, <<, >>, &, =, <>, <, >, <=, >=, **Like**, **New**, **TypeOf**, Not, And, AndAlso, Or, OrElse, Xor, Eqv, Imp, **Is**, **IsNot**, **AddressOf**.

More:

Year Function

Syntax Year(*dateexpr*)

Group Time/Date

Description Return the year.

Parameter **Description**

dateexpr Return the year for this date value. If this value is **Null** then **Null** is returned.

See Also **Date( )**, **Day( )**, **Month( )**, **Weekday( )**.

Example
```
'#Language "WWB-COM"
Sub Main
  Debug.Print Year(#1/1/1900#) ' 1900
  Debug.Print Year(#1/1/2000#) ' 2000
EndSub
```

AboutWinWrapBasic Instruction

Syntax AboutWinWrapBasic [*Timeout*]

Group Miscellaneous

Description Show the WinWrap Basic about box.

Parameter **Description**

Timeout This numeric value is the maximum number of seconds to show the about box. A value less than or equal to zero displays the about box until the user closes it. If this value is omitted then a three second timeout is used.

Example
```
'#Language "WWB-COM"
Sub Main
```

```
    AboutWinWrapBasic
End Sub
```