

Core

List of types:

[Bit](#)
[DateTime](#)
[Exception](#)
[Intent](#)
[Keywords](#)
[LayoutValues](#)
[Notification](#)
[RemoteViews](#)
[Service](#)
[String](#)
[StringBuilder](#)
[Timer](#)

Bit

Bit is a predefined object containing bitwise related methods.

Example:

```
Dim flags As Int  
flags = Bit.Or(100, 200)
```

Events:

None

Members:

- ⇒ [And](#) (N1 As Int, N2 As Int) As Int
- ⇒ [Not](#) (N As Int) As Int
- ⇒ [Or](#) (N1 As Int, N2 As Int) As Int
- ⇒ [ParseInt](#) (Value As String, Radix As Int) As Int
- ⇒ [ShiftLeft](#) (N As Int, Shift As Int) As Int
- ⇒ [ShiftRight](#) (N As Int, Shift As Int) As Int
- ⇒ [ToBinaryString](#) (N As Int) As String
- ⇒ [ToHexString](#) (N As Int) As String
- ⇒ [ToOctalString](#) (N As Int) As String
- ⇒ [UnsignedShiftRight](#) (N As Int, Shift As Int) As Int
- ⇒ [Xor](#) (N1 As Int, N2 As Int) As Int

Members description:

⇒ [And](#) (N1 As Int, N2 As Int) As Int

Returns the bitwise AND of the two values.

⇒ [Not](#) (N As Int) As Int

Returns the bitwise complement of the given value.

⇒ [Or](#) (N1 As Int, N2 As Int) As Int

Returns the bitwise OR of the two values.

⇒ **ParseInt** (Value As String, Radix As Int) As Int

Parses Value as an integer using the specified radix.
Radix - Should be between 2 to 36.

⇒ **ShiftLeft** (N As Int, Shift As Int) As Int

Shifts N left.
Shift - Number of positions to shift.

⇒ **ShiftRight** (N As Int, Shift As Int) As Int

Shifts N right.
Keeps the original value sign
Shift - Number of positions to shift.

⇒ **ToBinaryString** (N As Int) As String

Returns a string representation of N in base 2.

⇒ **ToHexString** (N As Int) As String

Returns a string representation of N in base 16.

⇒ **ToOctalString** (N As Int) As String

Returns a string representation of N in base 8.

⇒ **UnsignedShiftRight** (N As Int, Shift As Int) As Int

Shifts N right.
Shifts zeroes in the leftmost positions.
Shift - Number of positions to shift.

⇒ **Xor** (N1 As Int, N2 As Int) As Int

Returns the bitwise XOR of the two values.

DateTime

Date and time related methods.

DateTime is a predefined object. You should not declare it yourself.

Date and time values are stored as ticks. Ticks are the number of milliseconds since January 1, 1970.

This value is too large to be stored in an Int variable. It should only be stored in a Long variable.

The methods `DateTime.Date` and `DateTime.Time` convert the ticks value to a string.

You can get the current time with `DateTime.Now`.

Example:

```
Dim now As Long
now = DateTime.Now
Msgbox("The date is: " & DateTime.Date(now) & CRLF & _
    "The time is: " & DateTime.Time(now), "")
```

Events:

None

Members:

⇒ **Add** (Ticks As Long, Years As Int, Months As Int, Days As Int) As Long

⇒ **Date** (Ticks As Long) As String

⇒ **DateFormat** As String

⇒ **DateParse** (Date As String) As Long

⇒ **GetDayOfMonth** (Ticks As Long) As Int

⇒ **GetDayOfWeek** (Ticks As Long) As Int

⇒ **GetDayOfYear** (Ticks As Long) As Int

⇒ **GetHour** (Ticks As Long) As Int

⇒ **GetMinute** (Ticks As Long) As Int

⇒ **GetMonth** (Ticks As Long) As Int

⇒ [GetSecond](#) (Ticks As Long) As Int

⇒ [GetYear](#) (Ticks As Long) As Int

📖 [Now](#) As Long [read only]

🔗 [TicksPerDay](#) As Long

🔗 [TicksPerHour](#) As Long

🔗 [TicksPerMinute](#) As Long

🔗 [TicksPerSecond](#) As Long

⇒ [Time](#) (Ticks As Long) As String

📖 [TimeFormat](#) As String

⇒ [TimeParse](#) (Time As String) As Long

Members description:

⇒ [Add](#) (Ticks As Long, Years As Int, Months As Int, Days As Int) As Long

Returns a ticks value which is the result of adding the specified time spans to the given ticks value. Pass negative values if you want to subtract the values.

Example:

```
Dim Tomorrow As Long
Tomorrow = DateTime.Add(DateTime.Now, 0, 0, 1)
Log("Tomorrow date is: " & DateTime.Date(Tomorrow))
```

⇒ [Date](#) (Ticks As Long) As String

Returns a string representation of the date (which is stored as ticks).

The date format can be set with the DateFormat keyword.

Example:

```
Log("Today is: " & DateTime.Date(DateTime.Now))
```

📖 [DateFormat](#) As String

Gets or sets the format used to parse date strings.

See this page for the supported patterns: [formats](#).

The default pattern is MM/dd/yyyy (04/23/2002 for example).

⇒ [DateParse](#) (Date As String) As Long

Parses the given date string and returns its ticks representation.

An exception will be thrown if parsing fails.

Example:

```
Dim SomeTime As Long
SomeTime = DateTime.DateParse("02/23/2007")
```

⇒ [GetDayOfMonth](#) (Ticks As Long) As Int

Returns the day of month component from the ticks value.

Values are between 1 to 31.

⇒ [GetDayOfWeek](#) (Ticks As Long) As Int

Returns the day of week component from the ticks value.

Values are between 1 to 7, where 1 means Sunday.

You can use the [AHLocale](#) library if you need to change the first day.

⇒ [GetDayOfYear](#) (Ticks As Long) As Int

Returns the day of year component from the ticks value.

Values are between 1 to 366.

⇒ [GetHour](#) (Ticks As Long) As Int

Returns the hour of day component from the ticks value.

Values are between 0 to 23.

⇒ [GetMinute](#) (Ticks As Long) As Int

Returns the minutes within a hour component from the ticks value.

Values are between 0 to 59.

⇒ [GetMonth](#) (Ticks As Long) As Int

Returns the month of year component from the ticks value.
Values are between 1 to 12.

GetSecond (Ticks As Long) As Int

Returns the seconds within a minute component from the ticks value.
Values are between 0 to 59.

GetYear (Ticks As Long) As Int

Returns the year component from the ticks value.

Now As Long [read only]

Gets the current time as ticks (number of milliseconds since January 1, 1970).

TicksPerDay As Long

TicksPerHour As Long

TicksPerMinute As Long

TicksPerSecond As Long

Time (Ticks As Long) As String

Returns a string representation of the time (which is stored as ticks).
The time format can be set with the `TimeFormat` keyword.

Example:

```
Log("The time now is: " & DateTime.Time(DateTime.Now))
```

TimeFormat As String

Gets or sets the format used to parse time strings.
See this page for the supported patterns: [formats](#).
The default pattern is HH:mm:ss (23:45:12 for example).

TimeParse (Time As String) As Long

Parses the given time string and returns its ticks representation.
Note that the returned value date will be today.

Exception

Holds a thrown exception.
You can access the last thrown exception by calling `LastException`.
For example:

```
Try
Dim in As InputStream
in = File.OpenInput(File.DirInternal, "SomeMissingFile.txt")
'...
Catch
Log>LastException.Message)
End Try
If in.Initialized Then in.Close
```

Events:

None

Members:

[IsInitialized](#) As Boolean

[Message](#) As String [read only]

Members description:

[IsInitialized](#) As Boolean

[Message](#) As String [read only]

Intent

Intent objects are messages which you can send to the OS in order to do some external action.
The Intent object should be sent with `StartActivity` keyword.
See this [page](#) for a list of standard constants.
Example, launch YouTube application:

```
Dim Intent1 As Intent
Intent1.Initialize(Intent1.ACTION_MAIN, "")
Intent1.SetComponent("com.google.android.youtube/.HomeActivity")
StartActivity(Intent1)
```

Events:

None

Members:

 [Action](#) As String

 [ACTION_APPWIDGET_UPDATE](#) As String

 [ACTION_CALL](#) As String

 [ACTION_EDIT](#) As String

 [ACTION_MAIN](#) As String

 [ACTION_PICK](#) As String

 [ACTION_SEND](#) As String

 [ACTION_VIEW](#) As String

 [AddCategory](#) (Category As String)

 [ExtrasToString](#) As String

 [Flags](#) As Int

 [GetData](#)

 [GetExtra](#) (Name As String) As Object

 [HasExtra](#) (Name As String) As Boolean

 [Initialize](#) (Action As String, Uri As String)

 [Initialize2](#) (Uri As String, Flags As Int)

 [IsInitialized](#) As Boolean

 [PutExtra](#) (Name As String, Value As Object)

 [SetComponent](#) (Component As String)

 [SetType](#) (Type As String)

 [WrapAsIntentChooser](#) (Title As String)

Members description:

 [Action](#) As String

Gets or sets the Intent action.

 [ACTION_APPWIDGET_UPDATE](#) As String

 [ACTION_CALL](#) As String


 [ACTION_EDIT](#) As String

 [ACTION_MAIN](#) As String

 [ACTION_PICK](#) As String

 [ACTION_SEND](#) As String

 [ACTION_VIEW](#) As String

 [AddCategory](#) (Category As String)

Adds a category describing the intent required operation.

➤ ExtrasToString As String

Returns a string containing the extra items. This is useful for debugging.

📄 Flags As Int

Gets or sets the flags component.

➤ GetData

Retrieves the data component as a string.

➤ GetExtra (Name As String) As Object

Returns the item value with the given key.

➤ HasExtra (Name As String) As Boolean

Tests whether an item with the given key exists.

➤ Initialize (Action As String, Uri As String)

Initializes the object using the given Action and data Uri. Action can be one of the action constants or any other string. Pass an empty string if a Uri is not required.

➤ Initialize2 (Uri As String, Flags As Int)

Initializes the object by parsing the Uri.
Flags - Additional integer value. Pass 0 if it is not required.

Example:

```
Dim Intent1 As Intent
Intent1.Initialize2("http://www.basic4ppc.com", 0)
StartActivity(Intent1)
```

➤ IsInitialized As Boolean

➤ PutExtra (Name As String, Value As Object)

Adds extra data to the intent.

➤ SetComponent (Component As String)

Explicitly sets the component that will handle this intent.

➤ SetType (Type As String)

Sets the MIME type.

Example:

```
Intent1.SetType("plain/text")
```

➤ WrapAsIntentChooser (Title As String)

Wraps the intent in another "chooser" intent. A dialog will be displayed to the user with the available services that can act on the intent. `WrapAsIntentChooser` should be the last method called before sending the intent.

Keywords

These are the internal keywords.

Events:

None

Members:

➤ [Abs](#) (Number As Double) As Double

➤ [ACos](#) (Value As Double) As Double

➤ [ACosD](#) (Value As Double) As Double

➤ [Array](#)

➤ [Asc](#) (Char As Char) As Int

➤ [ASin](#) (Value As Double) As Double

➤ [ASinD](#) (Value As Double) As Double

➤ [ATan](#) (Value As Double) As Double

- ⇒ [ATanD](#) (Value As Double) As Double
- ⇒ [BytesToString](#) (Data() As Byte, StartOffset As Int, Length As Int, CharSet As String) As String
- ⇒ [CallSub](#) (Component As Object, Sub As String) As String
- ⇒ [CallSub2](#) (Component As Object, Sub As String, Argument As Object) As String
- ⇒ [CallSub3](#) (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object) As String
- ⇒ [CancelScheduledService](#) (Service As Object)
- ⇒ [Catch](#)
- ⇒ [cE](#) As Double
- ⇒ [Ceil](#) (Number As Double) As Double
- ⇒ [CharsToString](#) (Chars() As Char, StartOffset As Int, Length As Int) As String
- ⇒ [Chr](#) (UnicodeValue As Int) As Char
- ⇒ [ConfigureHomeWidget](#) (LayoutFile As String, EventName As String, UpdateIntervalMinutes As Int, WidgetName As String) As android.widget.RemoteViews
- ⇒ [Continue](#)
- ⇒ [Cos](#) (Radians As Double) As Double
- ⇒ [CosD](#) (Degrees As Double) As Double
- ⇒ [cPI](#) As Double
- ⇒ [CRLF](#) As String
- ⇒ [Density](#) As Float
- ⇒ [Dim](#)
- ⇒ [DipToCurrent](#) (Length As Int) As Int
- ⇒ [DoEvents](#)
- ⇒ [Exit](#)
- ⇒ [ExitApplication](#)
- ⇒ [False](#) As Boolean
- ⇒ [File](#) As File
- ⇒ [Floor](#) (Number As Double) As Double
- ⇒ [For](#)
- ⇒ [GetDeviceLayoutValues](#) As LayoutValues
- ⇒ [GetType](#) (object As Object) As String
- ⇒ [If](#)
- ⇒ [InputList](#) (Items As List, Title As String, CheckedItem As Int) As Int
- ⇒ [InputMap](#) (Items As Map, Title As String)
- ⇒ [InputMultiList](#) (Items As List, Title As String) As List
- ⇒ [Is](#)
- ⇒ [IsBackgroundTaskRunning](#) (ContainerObject As Object, TaskId As Int) As Boolean
- ⇒ [IsNumber](#) (Text As String) As Boolean
- ⇒ [IsPaused](#) (Component As Object) As Boolean

- ⇒ [LastException](#) As Exception
- ⇒ [LoadBitmap](#) (Dir As String, FileName As String) As Bitmap
- ⇒ [LoadBitmapSample](#) (Dir As String, FileName As String, MaxWidth As Int, MaxHeight As Int) As Bitmap
- ⇒ [Log](#) (Message As String)
- ⇒ [Logarithm](#) (Number As Double, Base As Double) As Double
- ⇒ [Max](#) (Number1 As Double, Number2 As Double) As Double
- ⇒ [Min](#) (Number1 As Double, Number2 As Double) As Double
- ⇒ [Msgbox](#) (Message As String, Title As String)
- ⇒ [Msgbox2](#) (Message As String, Title As String, Positive As String, Cancel As String, Negative As String, Icon As android.graphics.Bitmap) As Int
- ⇒ [Not](#) (Value As Boolean) As Boolean
- ⇒ [Null](#) As Object
- ⇒ [NumberFormat](#) (Number As Double, MinimumIntegers As Int, MaximumFractions As Int) As String
- ⇒ [NumberFormat2](#) (Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean) As String
- ⇒ [PerXToCurrent](#) (Percentage As Float) As Int
- ⇒ [PerYToCurrent](#) (Percentage As Float) As Int
- ⇒ [Power](#) (Base As Double, Exponent As Double) As Double
- ⇒ [ProgressDialogHide](#)
- ⇒ [ProgressDialogShow](#) (Text As String)
- ⇒ [ProgressDialogShow2](#) (Text As String, Cancelable As Boolean)
- ⇒ [QUOTE](#) As String
- ⇒ [Regex](#) As Regex
- ⇒ [Return](#)
- ⇒ [Rnd](#) (Min As Int, Max As Int) As Int
- ⇒ [RndSeed](#) (Seed As Long)
- ⇒ [Round](#) (Number As Double) As Long
- ⇒ [Round2](#) (Number As Double, DecimalPlaces As Int) As Double
- ⇒ [Select](#)
- ⇒ [Sender](#) As Object
- ⇒ [Sin](#) (Radians As Double) As Double
- ⇒ [SinD](#) (Degrees As Double) As Double
- ⇒ [Sqrt](#) (Value As Double) As Double
- ⇒ [StartActivity](#) (Activity As Object)
- ⇒ [StartService](#) (Service As Object)
- ⇒ [StartServiceAt](#) (Service As Object, Time As Long, DuringSleep As Boolean)
- ⇒ [StopService](#) (Service As Object)
- ⇒ [Sub](#)

⇒ [TAB](#) As String

⇒ [Tan](#) (Radians As Double) As Double

⇒ [TanD](#) (Degrees As Double) As Double

⇒ [ToastMessageShow](#) (Message As String, LongDuration As Boolean)

⇒ [True](#) As Boolean

⇒ [Try](#)

⇒ [Type](#)

⇒ [Until](#)

⇒ [While](#)

Members description:

⇒ [Abs](#) (Number As Double) As Double

Returns the absolute value.

⇒ [ACos](#) (Value As Double) As Double

Returns the angle measured with radians.

⇒ [ACosD](#) (Value As Double) As Double

Returns the angle measured with degrees.

⇒ [Array](#)

Creates a single dimension array of the specified type.

The syntax is: Array As type (list of values).

Example:

```
Dim Days() As String
Days = Array As String("Sunday", "Monday", ...)
```

⇒ [Asc](#) (Char As Char) As Int

Returns the unicode code point of the given character or first character in string.

⇒ [ASin](#) (Value As Double) As Double

Returns the angle measured with radians.

⇒ [ASinD](#) (Value As Double) As Double

Returns the angle measured with degrees.

⇒ [ATan](#) (Value As Double) As Double

Returns the angle measured with radians.

⇒ [ATanD](#) (Value As Double) As Double

Returns the angle measured with degrees.

⇒ [BytesToString](#) (Data() As Byte, StartOffset As Int, Length As Int, CharSet As String) As String

Decodes the given bytes arrays as a string.

Data - The bytes array.

StartOffset - The first byte to read.

Length - Number of bytes to read.

CharSet - The name of the character set.

Example:

```
Dim s As String
s = BytesToString(Buffer, 0, Buffer.Length, "UTF-8")
```

⇒ [CallSub](#) (Component As Object, Sub As String) As String

Calls the given sub. CallSub can be used to call a sub which belongs to a different module.

However the sub will only be called if the other module is not paused. In that case an empty string will be returned.

You can use IsPaused to test whether a module is paused.

This means that one activity cannot call a sub of a different activity. As the other activity will be paused for sure.

CallSub allows an activity to call a service sub or a service to call an activity sub.

Note that it is not possible to call subs of code modules.

CallSub can also be used to call subs in the current module. Pass an empty string as the component in that case.

Example:

`CallSub(Main, "RefreshData")`

CallSub2 (Component As Object, Sub As String, Argument As Object) As String

Similar to CallSub. Calls a sub with a single argument.

CallSub3 (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object) As String

Similar to CallSub. Calls a sub with two arguments.

CancelScheduledService (Service As Object)

Cancels previously scheduled tasks for this service.

Catch

Any exception thrown inside a try block will be caught in the catch block.

Call LastException to get the caught exception.

Syntax:

Try

...

Catch

...

End Try

e As Double

e (natural logarithm base) constant.

Ceil (Number As Double) As Double

Returns the smallest double that is greater or equal to the specified number and is equal to an integer.

CharsToString (Chars() As Char, StartOffset As Int, Length As Int) As String

Creates a new String by copying the characters from the array.

Copying starts from StartOffset and the number of characters copied equals to Length.

Chr (UnicodeValue As Int) As Char

Returns the character that is represented by the given unicode value.

ConfigureHomeWidget (LayoutFile As String, EventName As String, UpdateIntervalMinutes As Int, WidgetName As String) As android.widget.RemoteViews

Creates a RemoteViews object based on the layout file. The compiler will generate the required XML files based on the parameters.

See the widgets tutorial for more information: [Widgets tutorial](#).

Note that all parameters must be strings or numbers as they are read by the compiler.

LayoutFile - The widget layout file.

EventName - Sets the subs that will handle the RemoteViews events.

UpdateIntervalMinutes - Sets the update interval. Pass 0 to disable automatic updates. Minimum value is 30 (or 0).

WidgetName - The name of the widget as appears in the widgets list.

Continue

Stops executing the current iteration and continues with the next one.

Cos (Radians As Double) As Double

Calculated the trigonometric cosine function. Angle measured in radians.

CosD (Degrees As Double) As Double

Calculated the trigonometric cosine function. Angle measured in degrees.

ePI As Double

PI constant.

CRLF As String

New line character. The value of Chr(10).

Density As Float

Returns the device scale, which is DPI / 160.

(DPI stands for dots per inch).

Dim

Declares a variable.

Syntax:

Declare a single variable:

Dim variable name [As type]

The default type is String.

Declare multiple variables. All variables will be of the specified type.

Dim variable1, variable2, ..., [As type]

Note that the shorthand syntax only applies to Dim keyword.

Declare an array:

Dim variable(Rank1, Rank2, ...) [As type]

Example: `Dim Days(7) As String`

The actual rank can be omitted for zero length arrays.

☞ **DipToCurrent (Length As Int) As Int**

Scales the value, which represents a specific length on a default density device (Density = 1.0), to the current device.

For example, the following code will set the width value of this button to be the same physical size on all devices.

```
Button1.Width = DipToCurrent(100)
```

Note that a shorthand syntax for this method is available. Any number followed by the string 'dip' will be converted in the same manner (no spaces are allowed between the number and 'dip').

So the previous code is equivalent to:

```
Button1.Width = 100dip 'dip -> density independent pixel'
```

☞ **DoEvents**

Processes waiting messages in the messages queue.

DoEvents can be called inside lengthy loops to allow the program to update its UI.

Other waiting events will not be handled by DoEvents.

☞ **Exit**

Exits the most inner loop.

☞ **ExitApplication**

Immediately ends the application and stops the process.

Most applications should not use this method and prefer Activity.Finish which lets the OS decide when the process is killed.

🔗 **False As Boolean**

🔗 **File As File**

Files related methods.

☞ **Floor (Number As Double) As Double**

Returns the largest double that is smaller or equal to the specified number and is equal to an integer.

☞ **For**

Syntax:

```
For variable = value1 To value2 [Step interval]
```

```
...
```

```
Next
```

Example:

```
For i = 1 To 10
```

```
Log(i) 'Will print 1 to 10 (inclusive).
```

```
Next
```

If the iterator variable was not declared before it will be of type Int.

☞ **GetDeviceLayoutValues As LayoutValues**

Returns the device LayoutValues.

Example:

```
Log(GetDeviceLayoutValues)
```

☞ **GetType (object As Object) As String**

Returns a string representing the object's java type.

☞ **If**

Single line:

```
If condition Then true-statement [Else false-statement]
```

Multiline:

```
If condition Then
```

```
statement
```

```
Else If condition Then
```

```
statement
```

```
...
Else
statement
End If
```

⇒ **InputDialog (Items As List, Title As String, CheckedItem As Int) As Int**

Shows a modal dialog with a list of items and radio buttons. Pressing on an item will close the dialog. Returns the index of the selected item or DialogResponse.Cancel if the user pressed on the back key. List - Items to display. Title - Dialog title. CheckedItem - The index of the item that will first be selected. Pass -1 if no item should be preselected.

⇒ **InputMap (Items As Map, Title As String)**

Shows a modal dialog with a list of items and checkboxes. The user can select multiple items. The dialog is closed by pressing on the "Ok" button. The items displayed are the map keys. Items with a value of True will be checked. When the user checks or unchecks an item, the related item value gets updated. Items - A map object with the items as keys and their checked state as values.

Example:

```
Dim m As Map
m.Initialize
m.Put("Item #1", True)
m.Put("Item #2", False)
m.Put("Item #3", False)
m.Put("Item #4", True)
InputMap(m, "Check items")
```

⇒ **InputMultiList (Items As List, Title As String) As List**

Shows a modal dialog with a list of items and checkboxes. The user can select multiple items. The dialog is closed by pressing on the "Ok" button. Returns a list with the indices of the selected items. The list is sorted. Returns an empty list if the user has pressed on the back key.

⇒ **Is**

Tests whether the object is of the given type.

Example:

```
For i = 0 To Activity.NumberOfViews - 1
If Activity.GetView(i) Is Button Then
Dim b As Button
b = Activity.GetView(i)
b.Color = Colors.Blue
End If
Next
```

⇒ **IsBackgroundTaskRunning (ContainerObject As Object, TaskId As Int) As Boolean**

Tests whether a background task, submitted by the container object and with the specified id, is running.

⇒ **IsNumber (Text As String) As Boolean**

Tests whether the specified string can be safely parsed as a number.

⇒ **IsPaused (Component As Object) As Boolean**

Tests whether the given component is paused. Will also return true for components that were not started yet.

Example:

```
If IsPaused(Main) = False Then CallSub(Main, "RefreshData")
```

⇒ **LastException As Exception**

Returns the last exception that was caught (if such exists).

⇒ **LoadBitmap (Dir As String, FileName As String) As Bitmap**

Loads the bitmap.

Note that the Android file system is case sensitive.

You should consider using LoadBitmapSample if the image size is large.

The actual file size is not relevant as images are usually stored compressed.

Example:

```
Activity.SetBackgroundImage(LoadBitmap(File.DirAssets, "SomeFile.jpg"))
```

⇒ **LoadBitmapSample (Dir As String, FileName As String, MaxWidth As Int, MaxHeight As Int) As Bitmap**

Loads the bitmap.

The decoder will subsample the bitmap if MaxWidth or MaxHeight are smaller than the bitmap dimensions.

This can save a lot of memory when loading large images.

Example:

```
Activity.SetBackgroundImage(LoadBitmapSample(File.DirAssets, "SomeFile.jpg", Activity.Width,
```

Activity.Height))

Log (Message As String)

Logs a message. The log can be viewed in the LogCat tab.

Logarithm (Number As Double, Base As Double) As Double

Max (Number1 As Double, Number2 As Double) As Double

Returns the larger number between the two numbers.

Min (Number1 As Double, Number2 As Double) As Double

Returns the smaller number between the two numbers.

Msgbox (Message As String, Title As String)

Shows a modal message box with the specified message and title. The dialog will show one OK button.

Example:

```
Msgbox("Hello world", "This is the title")
```

Msgbox2 (Message As String, Title As String, Positive As String, Cancel As String, Negative As String, Icon As android.graphics.Bitmap) As Int

Shows a modal message box with the specified message and title.

Message - The dialog message.

Title - The dialog title.

Positive - The text to show for the "positive" button. Pass "" if you don't want to show the button.

Cancel - The text to show for the "cancel" button. Pass "" if you don't want to show the button.

Negative - The text to show for the "negative" button. Pass "" if you don't want to show the button.

Icon - A bitmap that will be drawn near the title. Pass Null if you don't want to show an icon.

Returns one of the DialogResponse values.

Example:

```
Dim result As Int
result = Msgbox2("This is the message", "This is the title", "Good", "", "Bad", LoadBitmap
(File.DirAssets, "smiley.gif"))
If result = DialogResponse.Positive Then ...
```

Not (Value As Boolean) As Boolean

Inverts the value of the given boolean.

Null As Object

NumberFormat (Number As Double, MinimumIntegers As Int, MaximumFractions As Int) As String

Converts the specified number to a string.

The string will include at least Minimum Integers and at most Maximum Fractions digits.

Example:

```
Log(NumberFormat(12345.6789, 0, 2)) '"12,345.68"'
Log(NumberFormat(1, 3, 0)) '"001"'
```

NumberFormat2 (Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean) As String

Converts the specified number to a string.

The string will include at least Minimum Integers, at most Maximum Fractions digits and at least Minimum Fractions digits.

GroupingUsed - Determines whether to group every three integers.

Example:

```
Log(NumberFormat2(12345.67, 0, 3, 3, false)) '"12345.670"'
```

PerXToCurrent (Percentage As Float) As Int

Returns the actual size of the given percentage of the activity width.

Example:

```
Button1.Width = PerXToCurrent(50) 'Button1.Width = 50% * Activity.Width
```

A shorthand syntax for this method is available. Any number followed by the string '%x' will be converted in the same manner (no spaces are allowed between the number and '%x').

So the previous code is equivalent to:

```
Button1.Width = 50%x
```

PerYToCurrent (Percentage As Float) As Int

Returns the actual size of the given percentage of the activity height.

Example:

```
Button1.Height = PerYToCurrent(50) 'Button1.Height = 50% * Activity.Height
```

A shorthand syntax for this method is available. Any number followed by the string '%y' will be converted in the same manner (no spaces are allowed between the number and '%y').

So the previous code is equivalent to:
Button1.Height = 50%

Power (Base As Double, Exponent As Double) As Double

Returns the Base value raised to the Exponent power.

ProgressDialogHide

Hides a visible progress dialog. Does not do anything if no progress dialog is visible.

ProgressDialogShow (Text As String)

Shows a dialog with a circular spinning bar and the specified text.
Unlike MsgBox and InputList methods, the code will not block.
You should call ProgressDialogHide to remove the dialog.
The dialog will also be removed if the user presses on the Back key.

ProgressDialogShow2 (Text As String, Cancelable As Boolean)

Shows a dialog with a circular spinning bar and the specified text.
Unlike MsgBox and InputList methods, the code will not block.
You should call ProgressDialogHide to remove the dialog.
Cancelable - Whether the user can dismiss the dialog by pressing on the Back key.

QUOTE As String

Quote character. The value of Chr(34).

Regex As Regex

Regular expressions related methods.

Return

Returns from the current sub and optionally returns the given value.
Syntax: Return [value]

Rnd (Min As Int, Max As Int) As Int

Returns a random integer between Min (inclusive) and Max (exclusive).

RndSeed (Seed As Long)

Sets the random seed value.
This method can be used for debugging as it allows you to get the same results each time.

Round (Number As Double) As Long

Returns the closest long number to the given number.

Round2 (Number As Double, DecimalPlaces As Int) As Double

Rounds the given number and leaves up to the specified number of fractional digits.

Select

Compares a single value to multiple values.

Example:

```
Dim value As Int
value = 7
Select value
    Case 1
        Log("One")
    Case 2, 4, 6, 8
        Log("Even")
    Case 3, 5, 7, 9
        Log("Odd larger than one")
    Case Else
        Log("Larger than 9")
End Select
```

Sender As Object

Returns the object that raised the event.

Only valid while inside the event sub.

Example:

```
Sub Button_Click
Dim b As Button
b = Sender
b.Text = "I've been clicked"
End Sub
```

Sin (Radians As Double) As Double

Calculated the trigonometric sine function. Angle measured in radians.

SinD (Degrees As Double) As Double

Calculated the trigonometric sine function. Angle measured in degrees.

Sqrt (Value As Double) As Double

Returns the positive square root.

StartActivity (Activity As Object)

Starts an activity or brings it to front if it already exists.

The target activity will be started once the program is free to process its message queue.

Activity can be a string with the target activity name or it can be the actual activity.

After this call the current activity will be paused and the target activity will be resumed.

This method can also be used to send Intents objects to the system.

Note that you should usually not call StartActivity from a Service.

Example: StartActivity (Activity2)

StartService (Service As Object)

Starts the given service. The service will be first created if it was not started before.

The target service will be started once the program is free to process its message queue.

Note that you cannot show a MsgBox after this call and before the service starts.

Service - The service module or the service name.

Example:

```
StartService(SQLService)
```

StartServiceAt (Service As Object, Time As Long, DuringSleep As Boolean)

Schedules the given service to start at the given time.

Service - The service module or service name. Pass an empty string when calling from a service module that schedules itself.

Time - The time to start the service. If this time has already past the service will be started now.

DuringSleep - Whether to start the service when the device is sleeping. If set to false and the device is sleeping

at the specified time, the service will be started when the device wakes up.

StartServiceAt can be used to schedule a repeating task. You should call it under Service_Start to schedule the next task.

This call cancels previous scheduled tasks (for the same service).

Example:

```
StartServiceAt(SQLService, DateTime.Now + 30 * 1000, false) 'will start after 30 seconds.
```

StopService (Service As Object)

Stops the given service. Service_Destroy will be called. Call StartService afterwards will first create the service.

Service - The service module or service name. Pass empty string to stop the current service (from the service module).

Example:

```
StopService(SQLService)
```

Sub

Declares a sub with the parameters and return type.

Syntax: Sub name [(list of parameters)] [As return-type]

Parameters include name and type.

The lengths of arrays dimensions should not be included.

Example:

```
Sub MySub (FirstName As String, LastName As String, Age As Int, OtherValues() As Double) As Boolean
```

```
...
```

```
End Sub
```

In this example OtherValues is a single dimension array.

The return type declaration is different than other declarations as the array parenthesis follow the type and not the name (which does not exist in this case).

TAB As String

Tab character.

Tan (Radians As Double) As Double

Calculated the trigonometric tangent function. Angle measured in radians.

TanD (Degrees As Double) As Double

Calculated the trigonometric tangent function. Angle measured in degrees.

ToastMessageShow (Message As String, LongDuration As Boolean)

Shows a quick little message that goes out automatically.

Message - The text message to show.

LongDuration - If true then shows the message for a long period, otherwise shows the message for a short period.

True As Boolean

Try

Any exception thrown inside a try block will be caught in the catch block.
Call LastException to get the caught exception.

Syntax:

```
Try
...
Catch
...
End Try
```

Type

Declares a structure.

Can only be used inside sub Globals or sub Process_Globals.

Syntax:

Type type-name (field1, field2, ...)

Fields include name and type.

Example:

```
Type MyType (Name As String, Items(10) As Int)
Dim a, b As MyType
a.Initialize
a.Items(2) = 123
```

Until

Loops until the condition is true.

Syntax:

Do Until condition

```
...
Loop
```

While

Loops while the condition is true.

Syntax:

Do While condition

```
...
Loop
```

LayoutValues

Holds values related to the display.

You can get the values of the current display by calling GetDeviceLayoutValues.

For example:

```
Dim lv As LayoutValues
lv = GetDeviceLayoutValues
Log(lv) 'will print the values to the log
Activity.LoadLayout and Panel.LoadLayout return a LayoutValues object with the values of the
chosen layout variant.
```

Events:

None

Members:

 [Height](#) As Int

 [Scale](#) As Float

 [toString](#) As String

 [Width](#) As Int

Members description:

 [Height](#) As Int

The display height (pixels).

 [Scale](#) As Float

The device scale value which is equal to 'dots per inch' / 160. For most devices the value will be 1.0 or 1.5 (high resolution).

 [toString](#) As String

The display width (pixels).

Notification

A status bar notification. The user can open the notifications screen and press on the notification.

Pressing on the notification will start an activity as set by the notification object.

Notifications are usually used by services as services are not expected to directly start activities.

The notification must have an icon and its "info" must be set.

Example:

```
Dim n As Notification
n.Initialize
n.Icon = "icon"
n.SetInfo("This is the title", "and this is the body.", Main) 'Change Main to "" if this
code is in the main module.
n.Notify(1)
```

Permissions:

android.permission.VIBRATE

Events:

None

Members:

 [AutoCancel](#) As Boolean [write only]

 [Cancel](#) (Id As Int)

 [Icon](#) As String [write only]

 [Initialize](#)

 [Insistent](#) As Boolean [write only]


 [IsInitialized](#) As Boolean

 [Light](#) As Boolean [write only]

 [Notify](#) (Id As Int)

 [Number](#) As Int

 [OnGoingEvent](#) As Boolean [write only]

 [SetInfo](#) (Title As String, Body As String, Activity As Object)


 [Sound](#) As Boolean [write only]

 [Vibrate](#) As Boolean [write only]

Members description:

 [AutoCancel](#) As Boolean [write only]

Sets whether the notification will be canceled automatically when the user clicks on it.

 [Cancel](#) (Id As Int)

Cancels the notification with the given Id.

 [Icon](#) As String [write only]

Sets the icon displayed.

The icon value is the name of the image file without the extension. **The name is case sensitive.**

The image file should be manually copied to the following folder: source folder\Objects\res\drawable.

You can use "icon" to use the application icon (which is also located in this folder):

```
n.Icon = "icon"
```

 [Initialize](#)

Initializes the notification. By default the notification plays a sound, shows a light and vibrates the phone.

 [Insistent](#) As Boolean [write only]

Sets whether the sound will play repeatedly until the user opens the notifications screen.

IsInitialized As Boolean

Light As Boolean [write only]

Sets whether the notification will show a light.

Example:

```
n.Light = False
```

Notify (Id As Int)

Displays the notification.

Id - The notification id. This id can be used to later update this notification (by calling Notify again with the same Id), or to cancel the notification.

Number As Int

Gets or sets a number that will be displayed on the icon. This is useful to represent multiple events in a single notification.

OnGoingEvent As Boolean [write only]

Sets whether this notification is an "ongoing event". The notification will be displayed in the ongoing section and it will not be cleared.

SetInfo (Title As String, Body As String, Activity As Object)

Sets the message text and action.

Title - The message title.

Body - The message body.

Activity - The activity to start when the user presses on the notification.

Pass an empty string to start the current activity (when calling from an activity module).

Example:

```
n.SetInfo("Some title", "Some text", Main)
```

Sound As Boolean [write only]

Sets whether the notification will play a sound.

Example:

```
n.Sound = False
```

Vibrate As Boolean [write only]

Sets whether the notification will vibrate.

Example:

```
n.Vibrate = False
```

RemoteViews

RemoteViews allows indirect access to a home screen widget.

See this tutorial for more information [Widgets tutorial](#).

Events:

RequestUpdate

Disabled

Members:

 [HandleWidgetEvents](#) (StartingIntent As android.content.Intent) As Boolean

 [SetImage](#) (ImageViewName As String, Image As android.graphics.Bitmap)

 [SetProgress](#) (ProgressBarName As String, Progress As Int)

 [SetText](#) (ViewName As String, Text As String)

 [SetTextColor](#) (ViewName As String, Color As Int)

 [SetTextSize](#) (ViewName As String, Size As Float)

 [SetVisible](#) (ViewName As String, Visible As Boolean)

 [UpdateWidget](#)

Members description:

 [HandleWidgetEvents](#) (StartingIntent As android.content.Intent) As Boolean

Checks if the intent starting this service was sent from the widget and raises events based on the intent.

Returns True if an event was raised.

⇒ **setImage (ImageViewName As String, Image As android.graphics.Bitmap)**

Sets the image of the given ImageView.

```
Example:rv.SetImage("ImageView1", LoadBitmap(File.DirAssets, "1.jpg"))
```

⇒ **setProgress (ProgressBarName As String, Progress As Int)**

Sets the progress value of the given ProgressBar. Value should be between 0 to 100.

```
Example:rv.SetProgress("ProgressBar1", 50)
```

⇒ **setText (ViewName As String, Text As String)**

Sets the text of the given view.

```
Example:rv.SetText("Label1", "New text")
```

⇒ **setTextColor (ViewName As String, Color As Int)**

Sets the text color of the given button or label.

```
Example:rv.SetTextColor("Label1", Colors.Red)
```

⇒ **setTextSize (ViewName As String, Size As Float)**

Sets the text size of the given button or label.

```
Example:rv.SetTextSize("Label1", 20)
```

⇒ **setVisible (ViewName As String, Visible As Boolean)**

Sets the visibility of the given view.

```
Example:rv.SetVisible("Button1", False)
```

⇒ **updateWidget**

Updates the widget with the changes done. This method is also responsible for configuring the events.

Service

Each Service module includes a Service object.

This object is used to bring the service in and out of the foreground state.

See the Services tutorial for more information.

Events:

None

Members:

⇒ [StartForeground](#) (Id As Int, Notification As android.app.Notification)

⇒ [StopForeground](#) (Id As Int)

Members description:

⇒ **StartForeground (Id As Int, Notification As android.app.Notification)**

Brings the current service to the foreground state and displays the given notification.

Id - The notification Id (see the notification object documentation).

Notification - The notification that will be displayed.

⇒ **StopForeground (Id As Int)**

Takes the current service out of the foreground state and cancels the notification with the given Id.

String

Strings are immutable in Basic4android, which means that you can change the value of a string variable but you cannot change the text stored in a string object.

No methods like SubString, Trim and ToLowerCase return a new string, **they do not change the value of the current string.**

Typical usage:

```
Dim s As String
s = "some text"
s = s.Replace("a", "b")
```

You can use StringBuilder if you need a mutable string.

Note that string literals are also string objects:

```
Log(" some text ".Trim)
```

Events:

None

Members:

- ⇒ [CharAt](#) (Index [As Int](#)) [As Char](#)
- ⇒ [CompareTo](#) (Other [As String](#)) [As Int](#)
- ⇒ [Contains](#) (SearchFor [As String](#)) [As Boolean](#)
- ⇒ [EndsWith](#) (Suffix [As String](#)) [As Boolean](#)
- ⇒ [EqualsIgnoreCase](#) (other [As String](#)) [As Boolean](#)
- ⇒ [GetBytes](#) (Charset [As String](#)) [As Byte\(\)](#)
- ⇒ [IndexOf](#) (SearchFor [As String](#)) [As Int](#)
- ⇒ [IndexOf2](#) (SearchFor [As String](#), Index [As Int](#)) [As Int](#)
- ⇒ [LastIndexOf](#) (SearchFor [As String](#)) [As Int](#)
- ⇒ [LastIndexOf2](#) (SearchFor [As String](#), Index [As Int](#)) [As Int](#)
- ⇒ [Length](#) [As Int](#)
- ⇒ [Replace](#) (Target [As String](#), Replacement [As String](#)) [As String](#)
- ⇒ [StartsWith](#) (Prefix [As String](#)) [As Boolean](#)
- ⇒ [Substring](#) (BeginIndex [As Int](#)) [As String](#)
- ⇒ [Substring2](#) (BeginIndex [As Int](#), EndIndex [As Int](#)) [As String](#)
- ⇒ [ToLowerCase](#) [As String](#)
- ⇒ [ToUpperCase](#) [As String](#)
- ⇒ [Trim](#) [As String](#)

Members description:

⇒ [CharAt](#) (Index [As Int](#)) [As Char](#)

Returns the character at the given index.

⇒ [CompareTo](#) (Other [As String](#)) [As Int](#)

Lexicographically compares the two strings.

Returns a value less than 0 if the current string precedes Other.

Returns 0 if both strings are equal.

Returns a value larger than 0 if the current string comes after Other.

Note that upper case characters precede lower case characters.

Examples:

```
"abc".CompareTo("da") ' < 0  
"abc".CompareTo("Abc") ' > 0  
"abc".CompareTo("abca") ' < 0
```

⇒ [Contains](#) (SearchFor [As String](#)) [As Boolean](#)

Tests whether the string contains the given string parameter.

⇒ [EndsWith](#) (Suffix [As String](#)) [As Boolean](#)

Returns true if this string ends with the given Suffix.

⇒ [EqualsIgnoreCase](#) (other [As String](#)) [As Boolean](#)

Returns true if both strings are equal ignoring their case.

⇒ [GetBytes](#) (Charset [As String](#)) [As Byte\(\)](#)

Encodes the string into a new array of bytes.

Example:

```
Dim Data() As Byte  
Data = "Some string".GetBytes("UTF8")
```

⇒ [IndexOf](#) (SearchFor [As String](#)) [As Int](#)

Returns the index of the first occurrence of SearchFor string in the string.
Returns -1 if SearchFor was not found.

⇒ IndexOf2 (SearchFor As String, Index As Int) As Int

Returns the index of the first occurrence of SearchFor string in the string.
Starts searching from the given Index.
Returns -1 if SearchFor was not found.

⇒ LastIndexOf (SearchFor As String) As Int

Returns the index of the first occurrence of SearchFor string in the string.
The search starts at the end of the string and advances to the beginning.

⇒ LastIndexOf2 (SearchFor As String, Index As Int) As Int

Returns the index of the first occurrence of SearchFor string in the string.
The search starts at the given index and advances to the beginning.

⇒ Length As Int

Returns the length of this string.

⇒ Replace (Target As String, Replacement As String) As String

Returns a new string resulting from the replacement of all the occurrences of Target with Replacement.

⇒ StartsWith (Prefix As String) As Boolean

Returns true if this string starts with the given Prefix.

⇒ SubString (BeginIndex As Int) As String

Returns a new string which is a substring of the original string.
The new string will include the character at BeginIndex and will extend to the end of the string.

Example:

```
"012345".SubString(2) 'returns "2345"
```

⇒ SubString2 (BeginIndex As Int, EndIndex As Int) As String

Returns a new string which is a substring of the original string.
The new string will include the character at BeginIndex and will extend to the character at EndIndex, not including the last character.

Example:

```
"012345".SubString2(2, 4) 'returns "23"
```

⇒ ToLowerCase As String

Returns a new string which is the result of lower casing this string.

⇒ ToUpperCase As String

Returns a new string which is the result of upper casing this string.

⇒ Trim As String

Returns a copy of the original string without any leading or trailing white spaces.

StringBuilder

StringBuilder is a mutable string, unlike regular strings which are immutable.
StringBuilder is especially useful when you need to concatenate many strings.
The following code demonstrates the performance boosting of StringBuilder:

```
Dim start As Long
start = DateTime.Now
'Regular string
Dim s As String
For i = 1 To 5000
    s = s & i
Next
Log(DateTime.Now - start)
'StringBuilder
start = DateTime.Now
Dim sb As StringBuilder
sb.Initialize
For i = 1 To 5000
    sb.Append(i)
Next
Log(DateTime.Now - start)
```

Tested on a real device, the first 'for loop' took about 20 seconds and the second took less than tenth of a second.

The reason is that the code: `s = s & i` creates a new string each iteration (strings are immutable).
The method `StringBuilder.ToString` converts the object to a string.

Events:

None

Members:

 [Append](#) (Text As String) As StringBuilder

 [Initialize](#)

 [Insert](#) (Offset As Int, Text As String) As StringBuilder

 [IsInitialized](#) As Boolean

 [Length](#) As Int [read only]

 [Remove](#) (StartOffset As Int, EndOffset As Int) As StringBuilder

 [ToString](#) As String

Members description:

 **Append (Text As String) As StringBuilder**


Appends the specified text at the end.
Returns the same object, so you can chain methods.
Example:

```
sb.Append("First line").Append(CRLF).Append("Second line")
```

 **Initialize**

Initializes the object.
Example:

```
Dim sb As StringBuilder  
sb.Initialize  
sb.Append("The value is: ").Append(SomeOtherVariable).Append(CRLF)
```

 **Insert (Offset As Int, Text As String) As StringBuilder**

Inserts the specified text at the specified offset.


 **IsInitialized As Boolean**

 **Length As Int [read only]**

Returns the number of characters.

 **Remove (StartOffset As Int, EndOffset As Int) As StringBuilder**

Removes the specified characters.
StartOffset - The first character to remove.
EndOffset - The ending index. This character will not be removed.

 **ToString As String**

Converts the object to a string.

Timer

A Timer object generates ticks events at specified intervals.

Using a timer is a good alternative to a long loop, as it allows the UI thread to handle other events and messages.

Note that the timer events will not fire while the UI thread is busy running other code (unless you call `DoEvents` keyword).

The timer `Enabled` property is set to `False` by default. To make it start working you should change it to `True`.

Timer events will not fire when the activity is paused, or if a blocking dialog (like `Msgbox`) is visible.

Timers should be declared in Sub Process_Globals. Otherwise you may get multiple timers running when the activity is recreated.

It is also important to disable the timer when the activity is pausing and then enable it when it resumes. This will save CPU and battery.

Events:

Tick

Members:

 [Enabled](#) As Boolean

 [Initialize](#) (EventName As String, Interval As Long)

[Interval As Long](#)

Members description:

[Enabled As Boolean](#)

Gets or sets whether the timer is enabled (ticking).

[Initialize \(EventName As String, Interval As Long\)](#)

Initializes the timer with the event sub prefix and the specified interval (measured in milliseconds).

IMPORTANT: this object should be declared in Sub Process_Globals.

Example:

```
Timer1.Initialize("Timer1", 1000)
Timer1.Enabled = True
```

```
Sub Timer1_Tick
'Handle tick events
End Sub
```

[Interval As Long](#)

Gets or sets the interval between tick events, measured in milliseconds.

[Top](#)