

1. BrightScript	4
1.1 Language Reference	4
1.1.1 Variables, Literals, and Types	5
1.1.2 Operators	8
1.1.3 Objects and Interfaces	11
1.1.4 XML Support	13
1.1.5 Garbage Collection	16
1.1.6 Events	16
1.1.7 Threading Model	17
1.1.8 Scope	17
1.1.9 Intrinsic Objects	18
1.1.10 Program Statements	18
1.1.11 Built-In Functions	
1.1.12 Core Library Extension	31
1.1.13 BrightScript Debug Console	31
1.1.14 BrightScript Versions	34
1.1.15 Reserved Words	35
1.1.16 Example Script	35
1.2 Object Reference	40
1.2.1 Global Functions	43
1.2.2 BrightScript Core Objects	50
1.2.2.1 roArray	50
1.2.2.2 roAssociativeArray	52
1.2.2.3 roBoolean	55
1.2.2.4 roByteArray	55
1.2.2.5 roDouble, roIntrinsicDouble	58
1.2.2.6 roFunction	58
1.2.2.7 roInt, roFloat, roString	59
1.2.2.8 roJRE	63
1.2.2.9 roList	63
1.2.2.10 roMessagePort	66
1.2.2.11 roRegex	68
1.2.2.12 roXMLElement	69
1.2.2.13 roXMLList	72
1.2.3 Presentation and Widget Objects	75
1.2.3.1 roAudioConfiguration	76
1.2.3.2 roAudioOutput	77
1.2.3.3 roAudioPlayer	78
1.2.3.4 roAudioPlayerMx	83
1.2.3.5 roAudioEventMx	86
1.2.3.6 roCanvasWidget	87
1.2.3.7 roClockWidget	90
1.2.3.8 roHdmiInputChanged, roHdmiOutputChanged	92
1.2.3.9 roHtmlWidget	93
1.2.3.10 roHtmlWidgetEvent	103
1.2.3.11 roImageBuffer	104
1.2.3.12 roImagePlayer	104
1.2.3.13 roImageWidget	109
1.2.3.14 roRectangle	113
1.2.3.15 roStreamQueue	115
1.2.3.16 roTextField	117
1.2.3.17 roTextWidget	119
1.2.3.18 roTextWidgetEvent	123
1.2.3.19 roTouchScreen	123
1.2.3.20 roTouchEvent, roTouchCalibrationEvent	127
1.2.3.21 roVideoEvent, roAudioEvent	128
1.2.3.22 roVideoInput	130
1.2.3.23 roVideoMode	132
1.2.3.24 roVideoPlayer	142
1.2.4 File Objects	156
1.2.4.1 roAppendFile	156
1.2.4.2 roCreateFile	157
1.2.4.3 roReadFile	159
1.2.4.4 roReadWriteFile	160
1.2.5 Hashing and Storage Objects	162
1.2.5.1 roBlockCipher	163
1.2.5.2 roBrightPackage	164
1.2.5.3 roDiskErrorEvent	166
1.2.5.4 roDiskMonitor	167
1.2.5.5 roHashGenerator	168
1.2.5.6 roPassKey	169
1.2.5.7 roRegistry	170
1.2.5.8 roRegistrySection	171
1.2.5.9 roSqliteDatabase	172
1.2.5.10 roSqliteEvent	174

1.2.5.11	roSqliteStatement	174
1.2.5.12	roStorageAttached, roStorageDetached	177
1.2.5.13	roStorageHotplug	178
1.2.5.14	roStorageInfo	180
1.2.5.15	roVirtualMemory	182
1.2.6	Content Management Objects	182
1.2.6.1	roAssetCollection	183
1.2.6.2	roAssetFetcher	184
1.2.6.3	roAssetFetcherEvent	187
1.2.6.4	roAssetFetcherProgressEvent	191
1.2.6.5	roAssetPool	192
1.2.6.6	roAssetPoolFiles	193
1.2.6.7	roAssetRealizer	194
1.2.6.8	roAssetRealizerEvent	196
1.2.6.9	roSyncSpec	197
1.2.7	Networking Objects	199
1.2.7.1	roDatagramReceiver	200
1.2.7.2	roDatagramSender	201
1.2.7.3	roDatagramSocket	202
1.2.7.4	roDatagramEvent	203
1.2.7.5	roHttpServer	204
1.2.7.6	roHttpEvent	207
1.2.7.7	roKeyStore	209
1.2.7.8	roMediaServer	210
1.2.7.9	roMediaStreamer	212
1.2.7.10	roMediaStreamerEvent	214
1.2.7.11	roMimeStream	214
1.2.7.12	roMimeStreamEvent	215
1.2.7.13	roNetworkAdvertisement	216
1.2.7.14	roNetworkConfiguration	216
1.2.7.15	roNetworkAttached	230
1.2.7.16	roNetworkDetached	231
1.2.7.17	roNetworkDiscovery	232
1.2.7.18	roNetworkHotplug	234
1.2.7.19	roNetworkStatistics	234
1.2.7.20	roPtp	235
1.2.7.21	roPtpEvent	236
1.2.7.22	roRssArticle	237
1.2.7.23	roRssParser	238
1.2.7.24	roRtspStream	239
1.2.7.25	roSnmAgent	240
1.2.7.26	roSnmEvent	241
1.2.7.27	roStreamByteEvent	242
1.2.7.28	roStreamConnectResultEvent	242
1.2.7.29	roStreamEndEvent	243
1.2.7.30	roStreamLineEvent	243
1.2.7.31	roSyncManager	244
1.2.7.32	roSyncManagerEvent	246
1.2.7.33	roTCPServer	247
1.2.7.34	roTCPConnectEvent	248
1.2.7.35	roUPnPActionResult	248
1.2.7.36	roUPnPController	249
1.2.7.37	roUPnPDevice	251
1.2.7.38	roUPnPSearchEvent	252
1.2.7.39	roUPnPService	253
1.2.7.40	roUPnPServiceEvent	254
1.2.7.41	roTCPStream	255
1.2.7.42	roUrlTransfer	256
1.2.7.43	roUrlEvent	264
1.2.8	Input/Output Objects	268
1.2.8.1	roBtManager	268
1.2.8.2	roBtClientManager	271
1.2.8.3	roBtClientManagerEvent	273
1.2.8.4	roBtClient	273
1.2.8.5	roBtClientEvent	274
1.2.8.6	roCecInterface	275
1.2.8.7	roCecRxFrameEvent	276
1.2.8.8	roCecTxCompleteEvent	277
1.2.8.9	roChannelManager	278
1.2.8.10	roControlPort	282
1.2.8.11	roControlUp, roControlDown	288
1.2.8.12	roGpioButton	289
1.2.8.13	roGpioControlPort	289
1.2.8.14	roIRReceiver	290
1.2.8.15	roIRDownEvent, roIRRepeatEvent, roIRUpEvent	291

1.2.8.16 roIRTransmitter	292
1.2.8.17 roIRTransmitCompleteEvent	293
1.2.8.18 roIRRemote	294
1.2.8.19 roIRRemotePress	295
1.2.8.20 roKeyboard	296
1.2.8.21 roKeyboardPress	297
1.2.8.22 roSequenceMatcher	298
1.2.8.23 roSequenceMatchEvent	299
1.2.8.24 roSerialPort	300
1.2.9 System Objects	302
1.2.9.1 roDeviceCustomization	303
1.2.9.2 roDeviceInfo	304
1.2.9.3 roResourceManager	308
1.2.9.4 roSystemLog	309
1.2.10 Date and Time Objects	310
1.2.10.1 roDateTime	311
1.2.10.2 roNetworkTimeEvent	313
1.2.10.3 roSystemTime	314
1.2.10.4 roTimer	317
1.2.10.5 roTimerEvent	319
1.2.10.6 roTimeSpan	320
1.2.11 Legacy Objects	321
1.2.11.1 roRtspStreamEvent	321
1.2.11.2 roSyncPool	322
1.2.11.3 roSyncPoolEvent	325
1.2.11.4 roSyncPoolFiles	325
1.2.11.5 roSyncPoolProgressEvent	326

BrightScript

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)



BrightScript Refe...ual (ver 7.0).pdf

Note

If you're having trouble viewing the above file, make sure you are accessing this site via HTTPS (e.g. <https://docs.brightsign.biz>).

BrightScript is a powerful scripting language for building media and networked applications for embedded devices. This language features integrated support for a lightweight library of BrightScript objects, which are used to expose the API of the platform (device) that is running BrightScript. The BrightScript language connects generalized script functionality with underlying components for networking, media playback, UI screens, and interactive interfaces; BrightScript is optimized for generating user-friendly applications with minimal programmer effort.

The BrightScript section is divided into two categories:

- **Language Reference:** Outlines the characteristics of the BrightScript language, such as syntax, operators, statements, types, core library, etc.
- **Object Reference:** Provides a directory of publicly available objects, interfaces, and methods that comprise the BrightScript API.

Language Reference

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

The following are some general characteristics of BrightScript, as compared to other common scripting languages:

- BrightScript is not case sensitive.
- Statement syntax is similar to Python, Basic, Ruby, and Lua (and dissimilar to C).
- Like JavaScript and Lua, objects and named data-entry structures are associative arrays.
- BrightScript supports dynamic typing (like JavaScript) and declared types (like C and Java).
- Similar to .Net and Java, BrightScript uses "interfaces" and "components" (i.e. objects).

BrightScript code is compiled into bytecode that is run by an interpreter. The compilation step occurs every time a script is loaded and run. Similar to JavaScript, there is no separate compilation step that results in a saved binary file.

BrightScript and its component architecture are written in 100% C for speed, efficiency, and portability. Since many embedded processors do not have floating-point units, BrightScript makes extensive use of the "integer" type. Unlike some languages (including JavaScript), BrightScript only uses floating point numbers when necessary.

Variables, Literals, and Types

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

ON THIS PAGE

- [Identifiers](#)
- [Types](#)
- [Type Declaration Characters](#)
- [Literals \(Constants\)](#)
- [Array Literals](#)
- [Associative Array Literals](#)
- [Invalid Object Return](#)
- [Numbers](#)
 - [Dynamic Typing](#)
 - [Type Conversion](#)
 - [Type Conversion and Accuracy](#)

Identifiers

Identifiers are names of variables, functions, and labels. They also apply to BrightScript object methods (i.e. functions) and interfaces (which appear after a "." Dot Operator). Identifiers have the following rules:

- Must start with an alphabetic character (a-z).
- May consist of alphabetic characters, numbers, or the underscore symbol ("_").
- Are not case sensitive.
- May be of any length.
- May not be a **reserved word**.
- **(variables only)** May end with an optional type declaration (" \$" for a string, "% for an integer, "!" for a float, "# for a double).

Examples

```
a
boy5
super_man$
42%
```

Types

BrightScript supports both dynamic typing and declared types. This means that every value has a type determined at runtime, but variables can also be instructed to always contain a value of a specified type. If a value is assigned to a variable that has a specified type, the type of the value will be converted to the variable type if possible. If conversion is impossible, a runtime error will occur.

A variable that does not end in a type declaration may change its type dynamically. For example, the statement `a=4` will create an integer, while a following statement specifying that `a="hello"` will change the type of the variable `a` to a string.

BrightScript supports the following types:

- **Boolean:** True or False
- **Integer:** A 32-bit signed integer number
- **Float:** The smallest floating point number format supported by either the hardware or software
- **Double:** The largest floating point number format supported by either the hardware or software. Although Double is an intrinsically understood type, it is implemented internally with the *roIntrinsicDouble* object.
- **String:** A sequence of ASCII (not UTF-8) characters. BrightScript uses two intrinsic string states:
 - **Constant strings:** A statement such as `s="astring"` will create an intrinsic constant string.
 - **roString instances:** Once a string is used in an expression, it becomes an *roString* instance. For example, the statement `s = s + "bstring"` will cause the intrinsic string `s` to convert to an *roString* instance. If this is followed by the statement `s2 = s`, the `s2` value will be a reference to `s`, not a copy of it. The behavior of reference counting strings is new to BrightScript version 3.0.
- **Object:** A reference to a BrightScript object (i.e. a native component). Note that the `type()` function will not return "Object" but the type of object instead (e.g. *roList*, *roVideoPlayer*). Also note that there is no separate type for intrinsic BrightScript Objects. All intrinsic BrightScript Objects are built on the *roAssociativeArray* object type.
- **Interface:** An interface in a BrightScript Object. If a "." Dot Operator is used on an interface type, the member must be static (since there is no object context).
- **Invalid:** A type that can have only one value: `Invalid`. This type is returned in various instances when no other type is valid (for example, when indexing an array that has never been sent). It can also be assigned to a variable with the statement `var = invalid`.

The following are examples of different types. The `?` statement is a shortcut for `print`, while the `type()` function returns a string that identifies the type of the passed expression.

```
BrightScript> ?type(1)
Integer

BrightScript> ?type(1.0)
Float

BrightScript> ?type("hello")
String

BrightScript> ?type(CreateObject("roList"))
roList

BrightScript> ?type(1%)
Integer

BrightScript> b!=1
BrightScript> ?type(b!)
Float

BrightScript> c$="hello"
BrightScript> ?type(c$)
String

BrightScript> d="hello again"
BrightScript> ?type(d)
String

BrightScript> d=1
BrightScript> ?type(d)
Integer

BrightScript> d=1.0
BrightScript> ?type(d)
Float

BrightScript> e=invalid
BrightScript> ?type(e)
Invalid
```

Type Declaration Characters

A type declaration may be used at the end of a variable or literal to fix its type. Variables with the same identifier but separate types are separate variables: For example, defining `a$` and `a%` would create two independent variables.

Character	Type	Examples
\$	String	A\$, ZZ\$
%	Integer	A1%, SUM%
!	Single-Precision (Float)	B!, N1!
#	Double-Precision (Double)	A#, 1/3#, 2#

Literals (Constants)

The following are valid literal types:

- Type Boolean: Either `True` or `False`
- Type Invalid: `Invalid` only
- Type String: A string in quotes (e.g. `"This is a string"`)
- Type Integer: An integer in hex (e.g. `HFF`) or decimal (e.g. `255`) format
- Type Float: A number with a decimal (e.g. `2.01`), in scientific notation (e.g. `1.23456E+30`), or with a Float type designator (e.g. `2!`)
- Type Double: A number in scientific notation containing a double-precision exponent symbol (e.g. `1.23456789D-12`) or with a Double type declaration (e.g. `2.3#`)
- Type Function: Similar to variable formatting (e.g. `MyFunction`)
- Type Integer: `LINE_NUM` – The current source line number

Array Literals

The `[]` Array Operator can be used to declare an array. It can contain literals (constants) or expressions.

```
x = 5
Myarray = []
Myarray = [ 1, 2, 3]
Myarray = [ x+5, true, 1<>2, ["a","b"]]
```

Associative Array Literals

The `{}` Associative Array Operator can be used to define an associative array. It can contain literals (constants) or expressions.

```
aa = { }
aa = {key1:"value", key2: 55, key3: 5+3 }
```

Arrays and associative arrays can also be defined with the following format:

```
aa = {
  Myfunc1: aFunction
  Myvall : "the value"
}
```

Invalid Object Return

Many methods (i.e. functions) that return objects can also return `Invalid` (for example, in cases where there is no object to return). In these cases, the variable accepting the result must be dynamically typed since it may be assigned either type.

The following code will return a type mismatch: `a$` is a string that has a string type declaration, and thus it cannot contain `Invalid`.

```
l = []
a$ = l.pop()
```

Numbers

Dynamic Typing

The following rules determine how integers, doubles, and floats are dynamically typed:

1. If a constant contains 10 or more digits, or if `D` is used in the exponent, the number is Double. Adding a `#` type declaration also forces a constant to be a Double.
2. If the number is not double precision and it contains a decimal point, the number is a Float. Expressing a number in scientific notation using the `E` exponent also forces a constant to be a Float.
3. If neither of the above conditions is true for a constant, the number is an Integer.

Type Conversion

When operations are performed on one or two numbers, the result must be typed as an Integer, Float, or Double. When an addition (+), subtraction (-), or multiplication (*) operation is performed, the result will have the same degree of precision as the most precise operand: For example, multiplying an Integer by a Double will return a number that is a Double.

Only when both operands are Integers will the result be an Integer number. If the result of two Integer operands is outside the 32-bit range, the operation and return will be carried out with Doubles.

Division (/) operates using the same rules as above, except that it can never be carried out at the Integer level: When both operators are Integers, the operation and return will be carried out with Floats.

Comparison operations (e.g. <, >, =) will convert the numbers to the same type before they are compared. The less precise type will always be converted to the more precise type.

Type Conversion and Accuracy

When a Float or Double number is converted to the Integer type, it is *rounded down*. The largest integer that is not greater than the number is used. This also happens when the `INT` function is called on a number.

When a Double number is converted to the Float type, it is *4/5 rounded*. The least significant digit is rounded up if the fractional part is ≥ 5 (otherwise, it is left unchanged).

When a Float number is converted to the Double type, only the seven most significant digits will be accurate.

Operators

ON THIS PAGE

- [Logical and Bitwise Operators](#)
- [Dot Operator](#)
 - [Associative Arrays](#)
- [Array and Function-Call Operators](#)
 - [Array Dimensions](#)
- [Equals Operator](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

Operations in the innermost level of parentheses are performed first. Evaluation then proceeds according to the precedence in the following table. Operations on the same precedence are left-associative, except for exponentiation, which is right-associative.

Description	Symbol(s)
Function Calls or Parentheses	()
Array Operators	., []
Exponentiation	^
Negation	-, +
Multiplication, Division, Modulus	*, /, MOD
Addition, Subtraction	+, -
Comparison	<, >, =, <>, <=, >=
Logical Negation	NOT
Logical Conjunction	AND
Logical OR	OR

String Operators: The following operators work with strings: <, >, =, <>, <=, >=, +

Function References: The = and <> operators work on variables that contain function references and function literals.

Logical and Bitwise Operators

The AND, OR, and NOT operators are used for logical (Boolean) comparisons if the arguments for these operators are Boolean:

```
a = 20
b = 20
c = 20
if a = c and not(b > 40) then print "success"
```

On the other hand, if the arguments for these operators are numeric, they will perform bitwise operations:

```
x = 1 and 2 ' x is zero
y = true and false ' y is false
```

When the AND or OR operator is used for a logical operation, only the necessary amount of the expression is executed. For example, the first statement below will print "True", while the second statement will cause a runtime error (because "invalid" is not a valid operand for OR):

```
print true or invalid
print false or invalid
```

Dot Operator

The "." Dot Operator can be used on any BrightScript object. It also has special meaning when used on an *roAssociativeArray* object, as well as *roXMLElement* and *roXMLList* objects. When used on a BrightScript object, it refers to an interface or method associated with that object. In the following example, *IfInt* refers to the interface and *SetInt()* refers to a method that is part of that interface:

```
i = CreateObject("roInt")
i.IfInt.SetInt(5)
i.SetInt(5)
```

Every object method is part of an interface. However, specifying the interface with the "." Dot Operator is optional. If the interface is omitted, as in the third line of the above example, each interface that is part of the object will be searched for the specified member. If there is a naming conflict (i.e. a method with the same name appears in two interfaces), then the interface should be specified.

Associative Arrays

When the "." Dot Operator is used on an Associative Array, it is the same as calling the `Lookup()` or `AddReplace()` methods, which are member functions of the *roAssociativeArray* object:

```
aa = {}  
aa.newkey = "the value"  
print aa.newkey
```

Note that the parameters of the "." Dot Operator are set at compile time; unlike the `Lookup()` and `AddReplace()` methods, they are not dynamic.

The "." Dot Operator is always case insensitive: For example, the statement `aa.NewKey=55` will create the entry "newkey" in the associative array. To generate case-sensitive keys, instantiate an *roAssociativeArray* object and use the `SetModeCaseSensitive()` method.

Array and Function-Call Operators

The `[]` operator is used to access an array (i.e. any BrightScript object that has an *ifArray* interface, such as *roArray* and *roList* objects). It can also be used to access an associative array. The `[]` operator takes expressions that are evaluated at runtime, while the "." Dot Operator takes identifiers at compile time.

The `()` operator can be used to call a function. When used on a function literal (or variable containing a function reference), that function will be called.

The following code snippet demonstrates the use of both array and function-call operators.

```
aa = CreateObject("roAssociativeArray")  
aa["newkey"] = "the value"  
print aa["newkey"]  
  
array = CreateObject("roArray", 10, true)  
array[2] = "two"  
print array[2]  
  
fivevar = five  
print fivevar()  
  
array[1] = fivevar  
print array[1]() ' print 5  
  
function five() As Integer  
    return 5  
end function
```

Array Dimensions

Arrays in BrightScript are one dimensional. Multi-dimensional arrays are implemented as arrays of arrays. The `[]` operator will automatically map multi-dimensionality. For example, the following two fetching expressions are the same:

```
dim array[5,5,5]
item = array[1][2][3]
item = array[1,2,3]
```

If a multi-dimensional array grows beyond its hint size, the new entries are not automatically set to *roArray*.

Equals Operator

The = operator is used for both assignment and comparison:

```
a = 5
If a = 5 then print "a is 5"
```

Unlike the C language, BrightScript does not support use of the = assignment operator inside an expression. This is meant to eliminate a common class of bugs caused by confusion between assignment and comparison.

When assignment occurs, intrinsic types are copied, while BrightScript objects are reference counted.

Objects and Interfaces

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

ON THIS PAGE

- [BrightScript Objects](#)
- [Wrapper Objects](#)
- [Interfaces](#)
- [Statement and Interface Integration](#)
 - [PRINT](#)
 - [WAIT](#)
 - [Expression Parsing](#)
 - [Array Operator](#)
 - [Member Access Operator](#)

BrightScript Objects

Though BrightScript operates independently of its object architecture and library, they are both required for programming BrightScript applications. The API of a BrightSign platform is exposed to BrightScript as a library objects: Platforms must register a new BrightScript object to expose some part of its API.

BrightScript objects are written in C (or a compatible language such as C++), and are robust against version changes: Scripts are generally backwards compatible with objects that have undergone revisions.

BrightScript objects keep a reference count; they delete themselves when the reference count reaches zero.

Wrapper Objects

All intrinsic BrightScript types (Boolean, Integer, Float, Double, String, and Invalid) have object equivalents. If one of these intrinsic types is passed to a function that expects an object, the appropriate wrapper object will be created, assigned the correct value, and passed to the function (this is sometimes referred to as "autoboxing"): This allows, for example, *roArray* objects to store values (e.g. integers and strings) as well as objects.

Any expression that expects one of the above types will work with the corresponding wrapper object as well: *roBoolean*, *roInt*, *roFloat*, *roDouble*, *roString*.

The following examples illustrate how wrapper objects work:

```
Print 5.tostr()+"th"
Print "5".toint()+5

-5.tostr()      'This will cause an error. Instead, use the following:
(-5).tostr()

if type(5.tostr())<> "String" then stop
if (-5).tostr()<>"-5" then stop
if (1+2).tostr()<>"3" then stop
i=-55
if i.tostr()<>"-55" then stop
if 100%.tostr()<>"100" then stop
if (-100%).tostr()<>"-100" then stop
y%=10
if y%.tostr()<>"10" then stop

if "5".toint()<>5 or type("5".toint())<>"Integer" then stop
if "5".toFloat()<>5.0 or type("5".toFloat())<>"Float" then stop
fs="-1.1"
if fs.toFloat()<>-1.1 or fs.toint()<>-1 then stop

if "01234567".left(3)<>"012" then stop
if "01234567".right(4)<>"4567" then stop
if "01234567".mid(3)<>"34567" then stop
if "01234567".mid(3,1)<>"3" then stop
if "01234567".instr("56")<>5 then stop
if "01234567".instr(6,"56")<>-1 then stop
if "01234567".instr(0,"0")<>0 then stop
```

Interfaces

Interfaces in BrightScript operate similarly to Java or Microsoft COM: An interface is a known set of member functions that implement a set of logic. In some ways, an interface is similar to a virtual base class in C++; any script or program that is compatible with C can use an object interface without regards to the type of object it belongs to: For example, the *roSerialPort* object, which controls the standard serial interface, implements three interfaces: *ifSerialControl*, *ifStreamReceive*, and *ifStreamSend*. Since the print statement sends its output to any object that has an *ifStreamSend* interface, it works with the *roSerialPort* object, as well as any other object with the *ifStreamSend* interface.

Statement and Interface Integration

Some BrightScript statements have integrated functionality with interfaces. This section describes how to use statements with interfaces.

PRINT

Using the PRINT statement in the following format will print into an object that has an *ifStreamSend* interface, including the *roTextField* and *roSerialPort* objects:

```
port = CreateObject("roSerialPort",0,115200)
print port, "string"
```

If the expression being printed evaluates to an object with an *ifEnum* interface, the PRINT statement will print every item that can be enumerated.

In addition to printing the values of intrinsic types, the PRINT statement can also be used to print any object that exposes one of the following interfaces: *ifString*, *ifInt*, *ifFloat*.

WAIT

The WAIT statement can work in conjunction with any object that has an *ifMessagePort* interface.

Expression Parsing

Any expression that expects a certain type of variable—including Integer, Float, Double, Boolean, or String—can accept an object with an interface equivalent of that type: *ifInt*, *ifFloat*, *ifDouble*, *ifBoolean*, *ifString*.

Array Operator

The [] array operator works with any object that has an *ifArray* or *ifAssociativeArray* interface, including arrays, associative arrays, and lists.

Member Access Operator

The member access operator (i.e. **Dot Operator**) works with any object that has an *ifAssociativeArray* interface. It also works with any object when used to call a member function (i.e. method). It also has special meaning when used on an *roXMLElement* or *roXMLList* object.

XML Support

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

ON THIS PAGE

- [Dot Operator](#)
- [Attribute Operator](#)
- [Examples](#)
 - [Flickr code clip](#)

BrightScript provides XML support with two BrightScript objects and a set of dedicated language features:

- **roXMLElement**: This object provides support for parsing, generating, and containing XML.
- **roXMLList**: This object is used to contain a list of *roXMLElement* instances.

Dot Operator

The "." **Dot Operator** has the following features when used with XML objects:

- When used with an *roXMLElement* instance, the "." Dot Operator returns an *roXMLList* instance of the child tags that match the dot operand. If no tags match the operand, an empty list is returned.
- When applied to an *roXMLList* instance, the "." Dot Operator aggregates the results of performing the above operation on each *roXMLElement* in the list.
- When applied to XML, which is technically case sensitive, the "." Dot Operator is still case insensitive. If you wish to perform a case-sensitive XML operation, use the member functions of the *roXMLElement* / *roXMLList* objects.

Attribute Operator

The "@" Attribute Operator can be used with an *roXMLElement* instance to return a named attribute. Though XML is case sensitive, the Attribute Operator is always case insensitive. If the Attribute Operator is used with an *roXMLList* instance, it will only return a value if that list contains exactly one element.

Examples

```
<?xml version="1.0" encoding="utf-8" ?>
<rsp stat="ok">
  <photos page="1" pages="5" perpage="100" total="500">
    <photo id="3131875696" owner="21963906@N06" secret="f248c84625" server="3125"
      farm="4" title="VNY 16R" ispublic="1" isfriend="0" isfamily="0" />
    <photo id="3131137552" owner="8979045@N07" secret="b22cfde7c4" server="3078"
      farm="4" title="hoot" ispublic="1" isfriend="0" isfamily="0" />
  </photos>
</rsp>
```

```
<photo id="3131040291" owner="27651538@N06" secret="ae25ff3942" server="3286"
farm="4" title="172 • 365 :: Someone once told me..." ispublic="1" isfriend="0"
/>
</photos>
</rsp>
```

Given the XML in the above *example.xml*/file, then the following code will return an *roXMLList* instance with three entries:

```
rsp=CreateObject("roXMLElement")
rsp.Parse(ReadAsciiFile("example.xml"))

? rsp.photos.photo
```

The following will return an *roXMLElement* reference to the first photo (id="3131875696"):

```
? rsp.photos.photo[0]
```

The following will return an *roXMLList* reference containing the <photos> tag:

```
? rsp.photos
```

The following will return the string "100":

```
rsp.photos@perpage
```

You can use the *roXMLElement.GetText()* method to return an element's text: For example, if the variable <booklist> contains the element <book lang=eng>The Dawn of Man</book>, then the following code will print the string "The Dawn of Man".

```
Print booklist.book.gettext()
```

Alternatively, using the Attribute Operator will print the string "eng".

```
print booklist.book@lang
```

Flickr code clip

```
REM
REM Interestingness
REM pass an (optional) page of value 1 - 5 to get 100 photos
REM starting at 0/100/200/300/400
REM
REM returns a list of "Interestingness" photos with 100 entries
REM

Function GetInterestingnessPhotoList(http as Object, page=1 As Integer) As Object

    print "page=" ;page

    http.SetUrl("http://api.flickr.com/services/rest/?method=flickr.interestingness.
getList&api_key=YOURKEYGOESHERE&page="+mid(str(page),2))

    xml=http.GetToString()

    rsp=CreateObject("roXMLElement")
    if not rsp.Parse(xml) then stop

    return helperPhotoListFromXML(http, rsp.photos.photo) 'rsp.GetBody().Peek().GetBody()

End Function

Function helperPhotoListFromXML(http As Object, xmllist As Object, owner=invalid As dynamic)
As Object

    photolist=CreateObject("roList")
    for each photo in xmllist
        photolist.Push(newPhotoFromXML(http, photo, owner))
    end for
    return photolist

End Function

REM
REM newPhotoFromXML
REM
REM Takes an roXMLElement Object that is an <photo> ... </photo>
REM Returns an brs object of type Photo
REM photo.GetTitle()
REM photo.GetID()
REM photo.GetURL()
REM photo.GetOwner()
REM

Function newPhotoFromXML(http As Object, xml As Object, owner As dynamic) As Object
    photo = CreateObject("roAssociativeArray")
```

```

    photo.http=http
    photo.xml=xml
    photo.owner=owner
    photo.GetTitle=function():return m.xml@title:end function
    photo.GetID=function():return m.xml@id:end function
    photo.GetOwner=pGetOwner
    photo.GetURL=pGetURL
    return photo
End Function

Function pGetOwner() As String
    if m.owner<>invalid return m.owner
    return m.xml@owner
End Function

Function pGetURL() As String
    a=m.xml.GetAttributes()
    url="http://farm"+a.farm+".static.flickr.com/"+a.server+"/"+a.id+"_"+a.secret+".jpg"
    return url
End Function

```

Garbage Collection

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

BrightScript automatically frees strings when they are no longer used, and it will free objects when their reference count goes to zero. This is carried out at the time the object or string is no longer used; there is no background garbage collection task. The result is a predictable garbage-collection process, with no unexpected stalls in execution.

Objects may enter a state of circular reference counting: Objects that reference each other will never reach a reference count of zero and will need to be freed manually using the `RunGarbageCollector()` method. This method is useful when destroying old presentation data structures and creating a new presentation.

Events

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

Events in BrightScript center around an event loop and the *roMessagePort* object. Most BrightScript objects can post to a message port in the form of an event object: For example, the *roTimer* object posts events of the type *roTimerEvent* when configured intervals are reached.

The following script sets the destination message port using the `SetPort()` method, waits for an event in the form of an *roGpioButton* object, and then processes the event.

```

print "BrightSign Button-LED Test Running"
p = CreateObject("roMessagePort")
gpio = CreateObject("roGpioControlPort")
gpio.SetPort(p)

while true

```



```

msg=wait(0, p)
if type(msg)="roGpioButton" then
    btnn = msg.GetInt()
    if btnn <=5 then
        gpio.SetOutputState(btnn+17,1)
        print "Button Pressed: ";btnn
        sleep(500)
        gpio.SetOutputState(btnn+17,0)
    end if
end if

REM ignore buttons pressed while flashing led above
while p.GetMessage()<>invalid
    end while
end while

```

Note that lines 6-7 can be replaced using the following (and substituting `end while` with `end for`):

```

For each msg in p

```

Threading Model

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

BrightScript runs in a single thread. In general, BrightScript object calls are synchronous if they return quickly, and asynchronous if they take a substantial amount of time to complete. For example, methods belonging to the `roArray` object are all synchronous, while the `Play()` method that is part of the `roVideoPlayer` object will return immediately (it is asynchronous). As a video plays, the `roVideoPlayer` object will post messages to the message port, indicating such events as “media playback finished” or “frame x reached”.

The object implementer decides whether a BrightScript object should launch a background thread to perform a synchronous operation. Sometimes, an object will feature synchronous and asynchronous versions of the same method.

This threading model ensures that the script writer does not have to deal with mutexes and other synchronization objects. The script is always single threaded, and the message port is polled or waited on to receive events into the thread. On the other hand, those implementing BrightScript objects have to consider threading issues: For example, the `roList` and `roMessagePort` objects are thread-safe internally, allowing them to be used by multiple threads.

Scope

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

BrightScript uses the following scoping rules:

- Global variables are not supported; however, there is a single hard-coded global variable (“global”) that is an interface to the [global BrightScript object](#), which contains all global library functions.
- Functions declared with the `Function` statement are global in scope; however, if the function is anonymous, it will still be local in scope.
- Local variables exist within the function scope. If a function calls another function, that new function has its own scope.
- Labels exist within the function scope.

- Block statements such as `For / End For` and `While / End While` do not create a separate scope.

Intrinsic Objects

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

In general, this manual uses the term “object” to refer to “BrightScript components”, which are C or C++ components with interfaces and member functions that BrightScript uses directly. With the exception of some core objects (*roArray*, *roAssociativeArray*, *roInt*, *roMessagePort*, etc.), BrightScript objects are platform specific.

You can create intrinsic objects in BrightScript, but these objects are not BrightScript components. There is currently no way to create a BrightScript component in BrightScript or to create intrinsic objects that have interfaces (intrinsic objects can only contain member functions, properties, and other objects).

A BrightScript object is simply an *roAssociativeArray*. When a member function is called from an associative array, a “this” pointer is set to “m”, and “m” is accessible inside the Function code to access object keys. A “constructor” in BrightScript is simply a normal function at a global scope that creates an *roAssociativeArray* instance and fills in its member functions and properties

See the “snake” game in the appendix for examples of creating intrinsic objects.

Program Statements

ON THIS PAGE

- [Statement Syntax](#)
- [LIBRARY](#)
- [DIM](#)
- [Assignment \(“=”\)](#)
- [END](#)
- [STOP](#)
- [GOTO](#)
- [RETURN](#)
- [PRINT](#)
 - [\[@location\]](#)
 - [TAB \(expression\)](#)
 - [POS\(x\)](#)
- [FOR / END FOR](#)
- [FOR EACH IN / END FOR](#)
- [WHILE / EXIT WHILE](#)
- [IF / THEN / ELSE](#)
- [Block IF / ELSEIF / THEN / ENDIF](#)
- [Function\(\) As Type / End Function](#)
 - ["M" Identifier](#)
 - [Anonymous Functions](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

BrightScript supports the following statement types (note that BrightScript is not case sensitive). The syntax of each statement is documented in more detail later in this chapter.

- `Library`
- `Dim`
- `= (assignment)`

- End
- Stop
- Goto
- Rem (or')
- print
- For / To / End For / Step / Exit For
- For Each / In / End For / Exit For
- While / End While / Exit While
- Function / End Function / As / Return

Example

```
Function Main() As Void

    dim cavemen[10]

    cavemen.push("fred")
    cavemen.push("barney")
    cavemen.push("wilma")
    cavemen.push("betty")

    for each caveman in cavemen
        print caveman
    end for

End Function
```

Statement Syntax

Each line may contain a single statement. However, a colon (:) may be used to separate multiple statements on a single line.

Example

```
myname = "fred"
if myname="fred" then yourname = "barney":print yourname
```

LIBRARY

LIBRARY Filename.brs

The LIBRARY statement allows you to include your own BrightScript libraries (.brs files), which can then be utilized by your script. The LIBRARY statement(s) must occur at the beginning of a script, before any other statements, functions, operators, etc.

The system locates a library by searching the directory containing the current script, as well as the `SYS:/script-lib/` directory. Note that the `Run()` function does not currently change the path of a LIBRARY statement to that of the called script (i.e. the system will continue searching the directory of the caller script). On the other hand, running a script directly from the BrightSign shell does modify the library search path to that of the called script.

The first statement will include a library in the same folder as the script, while the second will include a library in a sub-folder.

```
LIBRARY "myBSL1.brs"
LIBRARY "new_lib/myBSL2.brs"
```

The following statement will include the *bslCore.brs* library, which has some useful BrightScript features, from the `SYS:/script-lib/` directory.

```
LIBRARY "v30/bslCore.brs"
```

DIM

DIM Name (dim1, dim2, ..., dimK)

The DIM ("dimension") statement provides a shortcut for creating *roArray* objects. It sets the variable Name to type "roArray". It can create arrays of arrays as needed for multi-dimensionality. The dimension passed to DIM is the index of the maximum entry to be allocated (i.e. the array initial size = dimension+1), though the array will be resized larger automatically if needed.

The following two lines create identical arrays.

```
Dim array[5]
array = CreateObject("roArray", 6, true)
```

Note

The expression $x[a,b]$ is equivalent to $x[a][b]$.

The following script demonstrates useful operations on a DIM array.

```
Dim c[5, 4, 6]

For x = 1 To 5
  For y = 1 To 4
    For z = 1 To 6
      c[x, y, z] = k
      k = k + 1
    End for
  End for
End for

k=0
For x = 1 To 5
  For y = 1 To 4
    For z = 1 To 6
      If c[x, y, z] <> k Then print "error" : Stop
      k = k + 1
    End for
  End for
End for
```

Assignment ("=")

variable = expression

The assignment statement ("=") assigns a variable to a new value.

In each of the following lines, the variable on the left side of the equals operator is assigned the value of the constant or expression on the right side of the equals operator.

```
a$="a rose is a rose"
b1=1.23
x=2.23
x=x-b1
```

END

The `END` statement terminates script execution normally.

STOP

The `STOP` statement interrupts script execution, returns a "STOP" error, and invokes the debugger. Use the `cont` command at the debugger prompt to continue execution of the script or the `step` command to execute a single step in the script.

GOTO

`GOTO label`

The `GOTO` statement transfers program control to the line number specified by `label`. The `GOTO label` statement results in a branching operation. A `label` is an identifier terminated with a colon on a line that contains no other statements or expressions.

Example

```
mylabel:
print "Hello World"
goto mylabel
```

RETURN

`RETURN expression`

The `RETURN` statement returns from a function back to its caller. If the function is not type `Void`, `RETURN` can also return a value to the caller.

PRINT

`PRINT [#output_object], [@location], item list`

The `PRINT` statement prints an item or list of items in the console. The item(s) may be strings, integers, floats, variables, or expressions. An object with an *ifInt*, *ifFloat*, or *ifString* interface may also be printed. If the `output_object` is specified, this statement will print to an object with an *ifStreamSend* interface.

If the statement is printing a list of items, the items must be separated with semicolons or commas. If semicolons are used, spaces are not inserted between printed items; if commas are used, the cursor will automatically advance to the next print zone before printing the next item.

Positive numbers and zero are printed with a leading space (without a plus sign). Spaces are not inserted before or after strings.

Example

```
x = 5 : print 25; " is equal to"; x ^2
' prints "25 is equal to 25"
```

Example

```
a$ = "string"
print a$;a$,a$;" ";a$
'prints "stringstring      string string"
```

Each print zone in the following example is 16 characters wide. The cursor moves to the next print zone each time a comma is encountered.

```
> print "zone 1","zone 2","zone 3","zone 4"
'prints "zone 1      zone 2      zone 3      zone 4"
```

Example

```
print "print statement #1 ";
print "print statement #2"
'prints "print statement #1 print statement #2"
```

In some cases, semicolons can be dropped. For example, the following statement is legal:

```
Print "this is a five "5"!!"
```

A trailing semicolon overrides the cursor-return so that the next `PRINT` statement begins where the last left off. If no trailing punctuation is used with a `PRINT` statement, the cursor drops to the beginning of the next line.

[@location]

If the console you are printing to has the *ifTextField* interface, you can use the `@` character to specify where printing will begin.

Example

```
print #m.text_field,@width*(height/2-1)+(width-len(msg$))/2,msg$;
```

Whenever you use `PRINT @` on the bottom line of the display, an automatic line-feed causes all displayed lines to move up one line. To prevent this from happening, use a trailing semicolon at the end of the statement.

TAB (expression)

This statement moves the cursor to the specified position on the current line (modulo the width of the console if the `TAB` position is greater than the console width).

Example

```
print tab(5)"tabbed 5";tab(25)"tabbed 25"
```

Note the following about the `TAB` statement:

- The `TAB` statement may be used several times in a `PRINT` list.
- No punctuation is required after a `TAB` statement.
- Numerical expressions may be used to specify a `TAB` position.
- The `TAB` statement cannot be used to move the cursor to the left.
- If the cursor is beyond the specified position, the `TAB` statement is ignored.

POS(x)

This statement returns an integer that indicates the current cursor position from 0 to the maximum width of the window. This statement requires a dummy argument in the form of any numeric expression.

```
print tab(40) pos(0) 'prints 40 at position 40

print "these" tab(pos(0)+5)"words" tab(pos(0)+5)"are";
print tab(pos(0)+5)"evenly" tab(pos(0)+5)"spaced"
```

FOR / END FOR

```
FOR counter_variable = initial_value TO final_value STEP increment / END FOR
```

The `FOR` statement creates an iterative loop that allows a sequence of program statements to be executed a specified number of times.

The `initial_value`, `final_value`, and `increment` can be any expression. The first time the `FOR` statement is executed, these three variables are evaluated and their values are saved; changing the variables during the loop will have no effect on the operation of the loop. However, the `counter_variable` must not be changed, or the loop will not operate normally. The first time the `FOR` statement is executed, the counter is set to both the value and type of the `initial_value`.

At the beginning of each loop, the value of the `counter_variable` is compared with the `final_value`. If the value of the `counter_variable` is greater than the `final_value`, the loop will complete and execution will continue with the statement following the `END FOR` statement. If, on the other hand, the counter has not yet exceeded the `final_value`, control passes to the first statement after the `FOR` statement. If `increment` is a negative number, the loop will complete when the value of the `counter_variable` is less than the `final_value`.

When program flow reaches the `END FOR` statement, the counter is incremented by the specified `increment` amount (or decremented if `increment` is a negative value). If the `STEP [increment]` language is not included in the `FOR` statement, the `increment` defaults to 1.

Use `EXIT FOR` to exit a `FOR` block prematurely.

The following script decrements `i` at the beginning of each loop until it is less than 1.

```
for i=10 to 1 step -1
    print i
end for
```

FOR EACH IN / END FOR

```
FOR EACH item IN object / END FOR
```

The `FOR EACH` statement can iterate through a set of items in any object that has an *IEnumerator* interface (i.e. an enumerator). The `FOR` block is terminated with the `END FOR` statement. Objects that are ordered intrinsically (such as *roList*) are enumerated in order, while objects that have no intrinsic order (such as *roAssociativeArray*) are enumerated in apparent random order. It is possible to delete entries as they are enumerated.

Use `EXIT FOR` to exit a `FOR` block prematurely.

The following objects can be enumerated: *roList*, *roArray*, *roAssociativeArray*, *roMessagePort*.

The following script iterates over an associative array in random order, prints each key/value pair, then deletes it.

```
aa={joe: 10, fred: 11, sue:9}
For each n in aa
    Print n;aa[n]
    aa.delete[n]
end for
```

WHILE / EXIT WHILE

```
WHILE expression / EXIT WHILE
```

A `WHILE` loop executes until the specified expression is false. Use the `EXIT WHILE` statement to exit a `WHILE` block prematurely.

```
k=0
while k<>0
    k=1
    Print "loop once"
end while

while true
    Print "loop once"
    Exit while
End while
```

IF / THEN / ELSE

IF expression THEN statements [ELSE statements]

This is the single-line form of the IF THEN ELSE statement; see the next section for more details about the block form of the IF THEN ELSE statement.

The IF statement instructs the interpreter to test the following expression. If the expression is True, control will proceed to the statements immediately following the expression. If the expression is False, control will jump to either the matching ELSE statement (if there is one) or to the next program line after the block.

Example

```
if x>127 then print "out of range" : end
```

THEN is optional in the above and similar statements. However, THEN is sometimes required to eliminate ambiguity, as in the following example

```
if y=m then m=0 'won't work without THEN
```

Block IF / ELSEIF / THEN / ENDIF

The block (i.e. multi-line) form of IF / THEN / ELSE has the following syntax:

```
If BooleanExpression [ Then ]
  [ Block ]
  [ ElseIfStatement+ ]
  [ ElseStatement ]
End If

ElseIfStatement ::=
  ElseIf BooleanExpression [ Then ]
  [ Block ]

ElseStatement ::=
  Else
  [ Block ]
```

Example

```
vp_msg_loop:
  msg=wait(tiut, p)
  if type(msg)="rovideoevent" then
    if debug then print "video event";msg.getint()
    if lm=0 and msg.getint() = meden then
      if debug then print "videofinished"
      retcode=5
      return
    endif
  else if type(msg)="rogpiobutton" then
    if debug then print "button press";msg
    if esc0 and msg=b0 then retcode=1:return
    if esc1 and msg=b1 then retcode=2:return
    if esc2 and msg=b2 then retcode=3:return
    if esc3 and msg=b3 then retcode=4:return
  else if type(msg)=" Invalid" then
    if debug then print "timeout"
```



```
        retcode=6
        return
    endif

    goto vp_msg_loop
```

Function() As Type / End Function

Function name(parameter As Type, ...) As Type

Each function has its own scope.

A function is declared using the `Function()` statement. The parentheses may contain one or more optional parameters; parameters can also have default values and expressions.

The type of each parameter may be declared. The return type of the function may also be declared. If a parameter type or return type is not declared, it is Dynamic by default. Intrinsic types are passed by value (and a copy is made), while objects are passed by reference. The `Sub` statement can be used instead of `Function` as a shortcut for creating a function with return type `Void`.

A parameter can be one of the following types:

- Integer
- Float
- Double
- String
- Object
- Dynamic

The function return can be one of the following types:

- Void
- Integer
- Float
- Double
- String
- Object
- Dynamic

"M" Identifier

If a function is called from an associative array, then the local variable `m` is set to the associative array in which the function is stored. If the function is not called from an associative array, then its `m` variable is set to an associative array that is global to the module and persists across calls.

The `m` identifier should only be used for the purpose stated above: We do not recommend using `m` as a general-purpose identifier.

Example

```
sub main()
    obj={
        add: add
        a: 5
        b: 10
    }

    obj.add()
    print obj.result
end sub

function add() As void
    m.result=m.a+m.b
end function
```

Anonymous Functions

A function without a name declaration is considered anonymous.

The following is a simple anonymous function declaration:

```
myfunc=function (a, b)
    Return a+b
end function

print myfunc(1,2)
```

Anonymous functions can also be used with associative-array literals:

```
q = {

    starring : function(o, e)
        str = e.GetBody()
        print "Starring: " + str
        toks = box(str).tokenize(",")
        for each act in toks
            actx = box(act).trim()
            if actx <> "" then
                print "Actor: [" + actx + "]"
                o.actors.Push(actx)
            endif
        end for
        return 0
    end function
}

q.starring(myobj, myxml)
```

Built-In Functions

ON THIS PAGE

- [Type\(\)](#)
- [GetGlobalAA\(\)](#)
- [Rnd\(\)](#)
- [Box\(\)](#)
- [Run\(\)](#)
- [Eval\(\)](#)
- [GetLastRunCompileError\(\)](#)
- [GetLastRunRuntimeError\(\)](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

BrightScript features a set of built-in, module-scope, intrinsic functions. A number of file I/O, string, mathematics, and system functions are also available via the *roGlobal* object.

Type()

```
Type(a As Variable) As String
```

This function returns the type of the passed variable and/or object.

GetGlobalAA()

```
GetGlobalAA() As Object
```

This function fetches the global associative array for the current script.

Rnd()

```
Rnd(range As Integer) As Integer
```

```
Rnd(0) As Float
```

If passed a positive, non-zero integer, this function returns a pseudo-random integer between 1 and the argument value. The range includes the argument value: For example, calling `Rnd(55)` will return a pseudo-random integer greater than 0 and less than 56.

If the argument is 0, this function returns a pseudo-random Float value between 0 and 1.

Note

The `Rnd()` functions utilize a pseudo-random seed number that is generated internally and not accessible to the user.

Box()

```
Box(type As Dynamic) As Object
```

This function returns an object version of the specified intrinsic type. Objects will be passed through.

Example

```
b = box("string")
b = box(b) ' b does not change
```

Run()

```
Run(file_name As String, [optional_arg As Dynamic, ...]) As Dynamic
```

```
Run(file_names As roArray, [optional_arg As Dynamic, ...]) As Dynamic
```

This function runs one or more scripts from the current script. You may append optional arguments, which will be passed to the `Main()` function of the script(s). The called script may also return arguments to the caller script.

If a string file name is passed, the function will compile and run the corresponding file. If an array of files is passed, the function will compile each file, link them together, and run them.

Example

```
Sub Main()
    Run("test.brs")
    BreakIfRunError(LINE_NUM)
    Print Run("test2.brs", "arg 1", "arg 2")
    if Run(["file1.brs", "file2.brs"]) <> 4 then stop
    BreakIfRunError(LINE_NUM)
    stop
End Sub

Sub BreakIfRunError(ln)
    el=GetLastRunCompileError()
```

```

    if el=invalid then
        el=GetLastRunRuntimeError()
        if el=&hFC or el=&hE2 then return
        'FC==ERR_NORMAL_END, E2=ERR_VALUE_RETURN
        print "Runtime Error (line ";ln;"): ";el
        stop
    else
        print "compile error (line ";ln;)"
        for each e in el
            for each i in e
                print i;"": ";e[i]
            end for
        end for
    stop
end if
End Sub

```

Eval()

Eval(code_snippet As String) As Dynamic

This function runs the passed code snippet in the context of the current function. The function compiles the snippet, then executes the byte-code. If the code compiles and runs successfully, it will return zero. If the code compiles successfully, but encounters a runtime error, it will return an integer indicating the error code (using the same codes as the `GetLastRunRuntimeError()` function). If compilation fails, it will return an *roList* object; the *roList* structure is identical to that of the `GetLastRunCompileError()` function.

The `Eval()` function can be useful in two cases:

- When you need to dynamically generate code at runtime.
- When you need to execute a statement that could result in a runtime error, but you don't want code execution to stop.

Example

```
PRINT Eval("1/0") 'Returns a divide by zero error.
```

GetLastRunCompileError()

GetLastRunCompileError() As roList

This function returns an *roList* object containing compile errors (or Invalid if no errors occurred). Each *roList* entry is an *roAssociativeArray* object containing the following keys:

- **ERRSTR:** The compile error type (as String)
- **FILESPEC:** The file URI of the script containing the error (as String)
- **ERRNO:** The error number (as Integer)
- **LINENO:** The line number where the error occurs (as Integer)

The following are possible **ERRNO** values:

Error Code		Description	Expanded Description
&hBF	191	ERR_NW	ENDWHILE statement occurs without statement.
&hBE	190	ERR_MISSING_ENDWHILE	WHILE statement occurs without END WHILE statement.
&hBC	188	ERR_MISSING_ENDIF	End of script reached without finding an ENDIF statement.
&hBB	187	ERR_NOLN	No line number found.
&hBA	186	ERR_LNSEQ	Line number sequence error.
&hB9	185	ERR_LOADFILE	Error loading file.
&hB8	184	ERR_NOMATCH	MATCH statement does not match.

&hB7	183	ERR_UNEXPECTED_EOF	Unexpected end of string encountered during string compilation.
&hB6	182	ERR_FOR_NEXT_MISMATCH	Variable on NEXT does not match FOR .
&hB5	181	ERR_NO_BLOCK_END	
&hB4	180	ERR_LABELTWICE	Label defined more than once.
&hB3	179	ERR_UNTERMED_STRING	Literal string does not have end quote.
&hB2	178	ERR_FUN_NOT_EXPECTED	
&hB1	177	ERR_TOO_MANY_CONST	
&hB0	176	ERR_TOO_MANY_VAR	
&hAF	175	ERR_EXIT_WHILE_NOT_IN_WHILE	
&hAE	174	ERR_INTERNAL_LIMIT_EXCEEDED	
&hAD	173	ERR_SUB_DEFINED_TWICE	
&hAC	172	ERR_NOMAIN	
&hAB	171	ERR_FOREACH_INDEX_TM	
&hAA	170	ERR_RET_CANNOT_HAVE_VALUE	
&hA9	169	ERR_RET_MUST_HAVE_VALUE	
&hA8	168	ERR_FUN_MUST_HAVE_RET_TYPE	
&hA7	167	ERR_INVALID_TYPE	
&hA6	166	ERR_NOLONGER	Feature no longer supported.
&hA5	165	ERR_EXIT_FOR_NOT_IN_FOR	
&hA4	164	ERR_MISSING_INITIALIZER	
&hA3	163	ERR_IF_TOO_LARGE	
&hA2	162	ERR_RO_NOT_FOUND	
&hA1	161	ERR_TOO_MANY_LABELS	
&hA0	160	ERR_VAR_CANNOT_BE_SUBNAME	
&h9F	159	ERR_INVALID_CONST_NAME	
&h9E	158	ERR_CONST_FOLDING	

GetLastRunRuntimeError()

GetLastRunRuntimeError() As Integer

This function returns the error code that resulted from the last Run() function.

These codes indicate a normal result:

Error Code		Description	Expanded Description
&hFF	255	ERR_OKAY	
&hFC	252	ERR_NORMAL_END	Execution ended normally, but with termination (e.g. END, shell "exit", window closed).
&hE2	226	ERR_VALUE_RETURN	Return executed with value returned on the stack.
&hE0	224	ERR_NO_VALUE_RETURN	Return executed without value returned on the stack.

The following codes indicate runtime errors:

Error Code	Description	Expanded Description
------------	-------------	----------------------

&hFE	254	ERR_INTERNAL	Unexpected condition occurred.
&hFD	253	ERR_UNDEFINED_OPCODE	Opcode could not be handled.
&hFB	251	ERR_UNDEFINED_OP	Expression operator could not be handled.
&hFA	250	ERR_MISSING_PARN	
&hF9	249	ERR_STACK_UNDER	No value to pop off the stack.
&hF8	248	ERR_BREAK	<code>scriptBreak()</code> function called.
&hF7	247	ERR_STOP	STOP statement executed.
&hF6	246	ERR_RO0	bscNewComponent failed because object class not found.
&hF5	245	ERR_R01	BrightScript member function call does not have right number of parameters.
&hF4	244	ERR_RO2	BrightScript member function not found in object or interface.
&hF3	243	ERR_RO3	BrightScript interface not a member of the object.
&hF2	242	ERR_TOO_MANY_PARAM	Too many function parameters to handle.
&hF1	241	ERR_WRONG_NUM_PARAM	Number of function parameters incorrect.
&hF0	240	ERR_RVIG	Function returns a value, but is ignored.
&hEF	239	ERR_NOTPRINTABLE	Value not printable.
&hEE	238	ERR_NOTWAITABLE	<code>WAIT</code> statement cannot be applied to object because object does not have an <i>rolMessagePort</i> interface.
&hED	237	ERR_MUST_BE_STATIC	Interface calls from <code>rotINTERFACE</code> type must be static.
&hEC	236	ERR_RO4	"." Dot Operator used on object that does not contain legal object or interface reference.
&hEB	235	ERR_NOTYPEOP	Operation attempted on two typeless operands.
&hE9	233	ERR_USE_OF_UNINIT_VAR	Uninitialized variable used illegally.
&hE8	232	ERR_TM2	Non-numeric index applied to array.
&hE7	231	ERR_ARRAYNOTDIMMED	
&hE6	230	ERR_USE_OF_UNINIT_BRSUBREF	Reference to uninitialized SUB.
&hE5	229	ERR_MUST_HAVE_RETURN	
&hE4	228	ERR_INVALID_LVALUE	Left side of the expression is invalid.
&hE3	227	ERR_INVALID_NUM_ARRAY_IDX	Number of array indexes is invalid.
&hE1	225	ERR_UNICODE_NOT_SUPPORTED	
&hE0	224	ERR_NOTFUNOPABLE	
&hDF	223	ERR_STACK_OVERFLOW	
&h20	32	ERR_CN	Continue (<code>cont</code> or <code>c</code>) not allowed.
&h1C	28	ERR_STRINGTOLONG	
&h1A	26	ERR_OS	String space has run out.
&h18	24	ERR_TM	A Type Mismatch (string /number operation mismatch) has occurred.
&h14	20	ERR_DIV_ZERO	
&h12	18	ERR_DD	Attempted to re-dimension array.
&h10	16	ERR_BS	Array subscript out of bounds.

&h0E	14	ERR_MISSING_LN	
&h0C	12	ERR_OUTOFMEM	
&h08	8	ERR_FC	Invalid parameter passed to function /array (e.g. a negative matrix dim or square root).
&h06	6	ERR_OD	Out of data (READ).
&h04	4	ERR_RG	Return without Gosub.
&h02	2	ERR_SYNTAX	
&h00	0	ERR_NF	Next without For.

Core Library Extension

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

There are a number of built-in functions that are not part of the BrightScript Core Library. You can use the **LIBRARY** statement to include this subset of functions:

```
LIBRARY "v30/bslCore.brs"
```

bslBrightScriptErrorCodes() As roAssociativeArray

Returns an associative array of name/value pairs corresponding to BrightScript error codes and their descriptions.

bslGeneralConstraints() As roAssociativeArray

Returns an associative array of name/value pairs corresponding to system constants.

bslUniversalControlEventCodes() As roAssociativeArray

Returns an associative array of name/value pairs corresponding to the remote key code constraints.

AsciiToHex(ascii As String) As String

Returns a hex-formatted version of the passed ASCII string.

HexToAscii(hex As String) As String

Returns an ASCII-formatted version of the passed hex string.

HexToInteger(hex As String) As Integer

Returns the integer value of the passed hex string.

BrightScript Debug Console

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

If, while a script is running, a runtime error occurs or a `STOP` statement is encountered, the BrightSign application will enter the BrightScript debug console. The debug console can be accessed from a terminal program using a null-modem cable connected to the serial, GPIO, or VGA port (depending on the player model). Networked players can also be accessed via [Telnet](#) or [SSH](#). You can also access the debug console at bootup using the BrightSign Shell:

1. Power off the device.
2. Hold the **SVC** button and power on the device.
3. Wait until the `brightsign>` prompt appears in the terminal (which indicates that you are in the BrightSign Shell). You can now release the **SVC** button.
4. Enter `script` at the prompt while in the BrightSign Shell. This will take you to the BrightScript debug console.

Note

The above instructions apply to Series 4 (XTx44, XDx34) and Series 3 players (XTx43, XDx33, HDx23, LS423). To access the debug console on earlier player models, power on the device and wait at least 5 seconds after the power LED (**pwr**) lights up. Then, use a paperclip or pen to press and hold the **SVC** button on the side of the player until the `brightsign>` prompt appears in the terminal.

The console scope is set to the function that was running when a runtime error or `STOP` statement occurred. While in the console, you can type in any BrightScript statement; it will then be compiled and executed in the current context.

In most cases, the debug console is the default device for the `PRINT` statement.

BrightScript Console Commands

The following console commands are currently available in the BrightScript debug console:

Command	Description
<code>bt</code>	Print a backtrace of call-function context frames.
<code>classes</code>	List all public classes.
<code>cont</code> or <code>c</code>	Continue script execution.
<code>counts</code>	List count of BrightScript Component instances.
<code>da</code>	Show disassembly and bytecode for this function.
<code>down</code> or <code>d</code>	Move one position down the function context chain.
<code>exit</code>	Exit the debug shell.
<code>gc</code>	Run the garbage collector and show collection statistics.
<code>hash</code>	Print the internal hash-table histograms.
<code>last</code>	Show the last line that executed.
<code>method <class></code>	List methods provided by specified class.
<code>method <class>.<interface></code>	List methods provided by the specified interface or class.
<code>list</code>	List the current source of the current function.
<code>ld</code>	Show line data (source records)
<code>next</code>	Show the next line to execute.
<code>bsc</code>	List all allocated BrightScript Component instances.
<code>stats</code>	Show statistics.
<code>step</code> or <code>s</code>	Step one program statement.
<code>t</code>	Step one statement and show each executed opcode.
<code>up</code> or <code>u</code>	Move one function up the context chain.
<code>var</code>	Display local variables and their types/values.

print or p or ?

Print variable value or expression.

BrightSign Shell Commands

The following commands are currently available in the BrightSign Shell:

Command	Description
?	Display the list of available commands.
help	Provide more information about a command.
clear	Clear the terminal window.
exit	Exit the shell.
dir <directory>	List the contents of a storage directory.
readperf	Measure the performance of file reading.
writeperf	Measure the performance of file writing.
rwperf	Measure simultaneous file read/write performance.
audioplay	Play an audio snippet.
script	Start the BrightScript debug console.
mfg	Perform a manufacturing test.
ps	List tasks.
reboot	Reboot the system.
shutdown	Shut down the system.
date	Display the date and time.
date <YYYY/MM/DD hh:mm:ss>	Set the date and time.
timezone	Display the time zone.
timezone list	Display a list of accepted time zones.
timezone <zone>	Set the time zone.
chdir <path> or cd <path>	Change the current directory.
delete <filename>	Delete files or or directories recursively.
version	Display software versions.
registry	Access the device registry.
uptime	Display device uptime.
writebootfile	Write a boot file to flash.
erasebootfile	Erase a boot file in flash.
id	Display device identification information.
file <filename>	Determine a file type.
hash	Cryptographically hash a file.
factoryreset	Reset the system to factory defaults.
format <drive> [<filesystem>]	Format a drive using one of the following file systems: "vfat", "ext3", "ext4".
eject <drive>	Eject a drive.
log	Display the system log.

unsetrtc	Clear the real time clock (RTC). Note that this action does not modify the system clock.
probe	Probe a file.
analyze	Analyze a file.
ptpstatus	Display the PTP status.
usblist	Display a list of discovered USB devices.
ifconfig	Configure the network interface. Type <code>help ifconfig</code> for a full list of configuration options.
wifiscan	Scan for WiFi networks.
httpgetperf <url> [<destination_file>]	Measure the download performance from the specified URL. If a destination file is not specified, the contents are discarded.
nslookup	Look up the hostname in the DNS.
videoplay	Play a video snippet.
imageplay	Display an image.
videomode	Change the current video mode .
edid hdmi	Read the EDID from the attached monitor.
powersave	Set the VGA power saving mode.

BrightScript Versions

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

BrightScript Version Matrix			
January 9, 2009			
	HD20000 1.3 Branch	HD2000 2.0 Branch	Compact Main Line
Snapshot Date	1/7/2008	7/16/2008	1/9/2009
Defxxx, on, gosub, clear, random, data, read, restore, err, errl, let, clear, line numbers	X	X	
Intrinsic Arrays	X	X	
Compiler		X	X
AA & dot Op & m reference		X	X
Sub/Functions		X	X
ifEnum & For Each		X	X
For/Next Does Not Always Execute At Least Once		X	X
Exit For		X	X
Invalid Type. Errors that used to be Int Zero are now Invalid. Added rolInvalid; Invalid Autoboxing			X
Array's use roArray; Added ifArray			X
Uinit Var Usage No Longer Allowed			X
Sub can have "As" (like Function)			X

roXML Element & XML Ops dot and @			X
Type() Change: Now matches declaration names (eg. Integer not roINT32)			X
Added roBoolean			X
Added dynamic Type; Type now optional on Sub/Functions			X
And/Or Don't Eval un-needed Terms			X
Sub/Fun Default Parameter Values (e.g. Sub (x=5 As Integer))			X
AA declaration Op { }			X
Array Declaration Op []			X
Change Array Op from () to []			X
Anonymous Functions			X
Added Circ. Ref. Garbage Collector			X
Add Eval(), Run(), and Box()			X

Reserved Words

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

AND	ENDSUB	LINE_NUM	RND
CREATEOBJECT	ENDWHILE	M*	STEP
DIM	EXIT	NEXT	STOP
EACH	EXITWHILE	NOT	SUB
EACH	FALSE	OBJFUN	TAB
ELSE	FOR	OR	THEN
END	FUNCTION	POS	TO
ENDFOR	GOTO	PRINT	TRUE
ENDFUNCTION	IF	REM	TYPE
ENDIF	INVALID	RETURN	WHILE

* Although M is not strictly a reserved word, it should not be used as an identifier outside of its **intended purpose**.

Example Script

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

The following code uses GPIO buttons 1, 2, 3, 4 for controls. It will work on any BrightSign model that has a video output and a GPIO port.

```
REM
REM The game of Snake
```

```
REM demonstrates BrightScript programming concepts
REM June 22, 2008
```

```
REM
REM Every BrightScript program must have a single Main()
REM
```

```
Sub Main()
```

```
    game_board=newGameBoard()
```

```
    While true
```

```
        game_board.SetSnake(newSnake(game_board.StartX(), game_board.StartY()))
```

```
        game_board.Draw()
```

```
        game_board.EventLoop()
```

```
        if game_board.GameOver() then ExitWhile
```

```
    End While
```

```
End Sub
```

```
REM *****
REM *****
REM *****
REM ***** GAME BOARD OBJECT *****
REM *****
REM *****
REM *****
```

```
REM
```

```
REM An example BrightScript constructor. "newGameBoard()" is regular Function of module scope
```

```
REM BrightScript Objects are "dynamic" and created at runtime. They have no "class".
```

```
REM The object container is a BrightScript Component of type roAssociativeArray (AA).
```

```
REM The AA is used to hold member data and member functions.
```

```
REM
```

```
Function newGameBoard() As Object
```

```
    game_board=CreateObject("roAssociativeArray") ' Create a BrightScript Component of type/class roAssociativeArray
```

```
    game_board.Init=gbInit ' Add an entry to the AA of type roFunction with value gbDraw (a sub defined in this module)
```

```
    game_board.Draw=gbDraw
```

```
    game_board.SetSnake=gbSetSnake
```

```
    game_board.EventLoop=gbEventLoop
```

```
    game_board.GameOver=gbGameOver
```

```
    game_board.StartX=gbStartX
```

```
    game_board.StartY=gbStartY
```

```
    game_board.Init() ' Call the Init member function (which is gbInit)
```

```
    return game_board
```

```
End Function
```

```
REM
```

```
REM gbInit() is a member function of the game_board BrightScript Object.
```

```
REM When it is called, the "this" pointer "m" is set to the appropriate instance by
```

```
REM the BrightScript bytecode interpreter
```

```
REM
```

```
Function gbInit() As Void
```

```
    REM
```

```
    REM button presses go to this message port
```

```
    REM
```

```
    m.buttons = CreateObject("roMessagePort")
```

```
    m.gpio = CreateObject("roGpioControlPort")
```

```

m.gpio.SetPort(m.buttons)

REM
REM determine optimal size and position for the snake gameboard
REM
CELLWID=16      ' each cell on game in pixels width
CELLHI=16       ' each cell in pix height
MAXWIDE=30     ' max width (in cells) of game board
MAXHI=30       ' max height (in cells) of game board
vidmode=CreateObject("roVideoMode")
w=cint(vidmode.GetResX()/CELLWID)
if w>MAXWIDE then w = MAXWIDE
h=cint(vidmode.GetResY()/CELLHI)
if h>MAXHI then h=MAXHI

xpix = cint((vidmode.GetResX() - w*CELLWID)/2)      ' center game board on screen
ypix = cint((vidmode.GetResY() - h*CELLHI)/2)      ' center game board on screen

REM
REM Create Text Field with square char cell size
REM
meta=CreateObject("roAssociativeArray")
meta.AddReplace("CharWidth",CELLWID)
meta.AddReplace("CharHeight",CELLHI)
meta.AddReplace("BackgroundColor",&H202020)  'very dark grey
meta.AddReplace("TextColor",&H00FF00)      ' Green
m.text_field=CreateObject("roTextField",xpix,ypix,w,h,meta)
if type(m.text_field)<>"roTextField" then
    print "unable to create roTextField 1"
    stop
endif
End Function

REM
REM As Object refers to type BrightScript Component
REM m the "this" pointer
REM
Sub gbSetSnake(snake As Object)
    m.snake=snake
End Sub

Function gbStartX() As Integer
    return cint(m.text_field.GetWidth()/2)
End Function

Function gbStartY() As Integer
    return cint(m.text_field.GetHeight()/2)
End Function

Function gbEventLoop() As Void

    tick_count=0

    while true
        msg=wait(250, m.buttons)      ' wait for a button, or 250ms (1/4 a second) timeout
        if type(msg)="roGpioButton" then
            if msg.GetInt()=1 m.snake.TurnNorth()
            if msg.GetInt()=2 m.snake.TurnSouth()
            if msg.GetInt()=3 m.snake.TurnEast()
            if msg.GetInt()=4 m.snake.TurnWest()
        else 'here if time out happened, move snake forward
            tick_count=tick_count+1
            if tick_count=6 then
                tick_count=0
            end if
        end if
    end while
end Function

```

```

        if m.snake.MakeLonger(m.text_field) then return
    else
        if m.snake.MoveForward(m.text_field) then return
    endif
endif
end while

End Function

Sub gbDraw()
    REM
    REM given a roTextField Object in "m.text_field", draw a box around its edge
    REM

    solid=191 ' use asc("**") if graphics not enabled
    m.text_field.Cls()

    for w=0 to m.text_field.GetWidth()-1
        print #m.text_field,@w,chr(solid);
        print #m.text_field,@m.text_field.GetWidth()*(m.text_field.GetHeight()-1)+w,chr
(solid);
    end for

    for h=1 to m.text_field.GetHeight()-2
        print #m.text_field,@h*m.text_field.GetWidth(),chr(solid);
        print #m.text_field,@h*m.text_field.GetWidth()+m.text_field.GetWidth()-1,chr(solid);
    end for

    m.snake.Draw(m.text_field)

End Sub

Function gbGameOver() As Boolean
    msg$= " G A M E   O V E R "
    msg0$=" "
    width = m.text_field.GetWidth()
    height = m.text_field.GetHeight()

    while true
        print #m.text_field,@width*(height/2-1)+(width-len(msg$))/2,msg$;
        sleep(300)
        print #m.text_field,@width*(height/2-1)+(width-len(msg$))/2,msg0$;
        sleep(150)
        REM GetMessage returns the message object, or an int 0 if no message available
        If m.buttons.GetMessage() <> invalid Then Return False
    endwhile

End Function

REM *****
REM *****
REM *****
REM ***** SNAKE OBJECT *****
REM *****
REM *****
REM *****

REM
REM construct a new snake BrightScript object
REM
Function newSnake(x As Integer, y As Integer) As Object

' Create AA BrightScript Component; the container for a "BrightScript Object"
snake=CreateObject("roAssociativeArray")
snake.Draw=snkDraw

```

```

snake.TurnNorth=snkTurnNorth
snake.TurnSouth=snkTurnSouth
snake.TurnEast=snkTurnEast
snake.TurnWest=snkTurnWest
snake.MoveForward=snkMoveForward
snake.MakeLonger=snkMakeLonger
snake.AddSegment=snkAddSegment
snake.EraseEndBit=snkEraseEndBit

REM
REM a "snake" is a list of line segments
REM a line segment is an roAssociativeArray that contains a length and direction (given by
the x,y delta needed to move as it is drawn)
REM

snake.seg_list = CreateObject("roList")
snake.AddSegment(1,0,3)

REM
REM The X,Y pos is the position of the head of the snake
REM
snake.snake_X=x
snake.snake_Y=y
snake.body=191 ' use asc("*") if graphics not enabled.
snake.dx=1 ' default snake direction / move offset
snake.dy=0 ' default snake direction / move offset

return snake

End Function

Sub snkDraw(text_field As Object)
x=m.snake_X
y=m.snake_Y
for each seg in m.seg_list
xdelta=seg.xDelta
ydelta=seg.yDelta
for j=1 to seg.Len
text_field.SetCursorPos(x, y)
text_field.SendByte(m.body)
x=x+xdelta
y=y+ydelta
end for
end for
End Sub

Sub snkEraseEndBit(text_field As Object)
x=m.snake_X
y=m.snake_Y
for each seg in m.seg_list
x=x+seg.Len*seg.xDelta
y=y+seg.Len*seg.yDelta
end for

text_field.SetCursorPos(x, y)
text_field.SendByte(32) ' 32 is ascii space, could use asc(" ")

End Sub

Function snkMoveForward(text_field As Object)As Boolean
m.EraseEndBit(text_field)
tail=m.seg_list.GetTail()
REM
REM the following shows how you can use an AA's member functions to perform the same

```

```

    REM functions the BrightScript . operator does behind the scenes for you (when used on an
    AA).
    REM there is not point to this longer method other than illustration
    REM
    len=tail.Lookup("Len")          ' same as len = tail.Len (or tail.len, BrightScript
syntax is not case sensitive)
    len = len-1
    if len=0 then
        m.seg_list.RemoveTail()
    else
        tail.AddReplace("Len",len) ' same as tail.Len=len
    endif

    return m.MakeLonger(text_field)

End Function

Function snkMakeLonger(text_field As Object) As Boolean
    m.snake_X=m.snake_X+m.dx
    m.snake_Y=m.snake_Y+m.dy
    text_field.SetCursorPos(m.snake_X, m.snake_Y)
    if text_field.GetValue()=m.body then return true
    text_field.SendByte(m.body)
    head = m.seg_list.GetHead()
    head.Len=head.Len+1
    return false
End Function

Sub snkAddSegment(dx As Integer, dy As Integer, len as Integer)

    aa=CreateObject("roAssociativeArray")
    aa.AddReplace("xDelta",-dx) ' line segments draw from head to tail
    aa.AddReplace("yDelta",-dy)
    aa.AddReplace("Len",len)
    m.seg_list.AddHead(aa)

End Sub

Sub snkTurnNorth()
    if m.dx<>0 or m.dy<>-1 then m.dx=0:m.dy=-1:m.AddSegment(m.dx, m.dy, 0) 'north
End Sub

Sub snkTurnSouth()
    if m.dx<>0 or m.dy<>1 then m.dx=0:m.dy=1:m.AddSegment(m.dx, m.dy, 0) 'south
End Sub

Sub snkTurnEast()
    if m.dx<>-1 or m.dy<>0 then m.dx=-1:m.dy=0:m.AddSegment(m.dx, m.dy, 0) 'east
End Sub

Sub snkTurnWest()
    if m.dx<>1 or m.dy<>0 then m.dx=1:m.dy=0:m.AddSegment(m.dx, m.dy, 0) 'west
End Sub

```

Object Reference

ON THIS PAGE

- [Interfaces and Methods](#)
- [Classes](#)
- [Object and Class Name Syntax](#)
- [Zones](#)

- [Event Loops](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

BrightSign players use a standardized library of BrightScript objects to expose functionality for software development. To publish a new API for interacting with BrightSign hardware, we create a new BrightScript object.

The pages in this section provide definitions for objects that can be used in BrightScript. A brief description, a list of interfaces, and the member functions of the interfaces are provided for each object class. While most BrightScript objects have self-contained pages, some objects are grouped on the same page if they are closely related or depend on one another for functionality.

Here is a sample of objects that are used frequently when creating applications in BrightScript:

<i>roVideoMode</i>	Configures video output and interacts with displays using CEC/EDID.
<i>roRectangle</i>	Used to define zones/widgets on the screen. This object is passed to many other objects to define their screen area, including <i>roVideoPlayer</i> , <i>roImagePlayer</i> , <i>roImageWidget</i> , <i>roHtmlWidget</i> , <i>roClockWidget</i> , and <i>roCanvasWidget</i> .
<i>roVideoPlayer</i>	Plays video files, streams, and HDMI input.
<i>roImagePlayer</i>	Displays images.
<i>roHtmlWidget</i>	Displays local or remote HTML content using the Chromium rendering engine.
<i>roNetworkConfiguration</i>	Used to configure Ethernet, WiFi, and local network parameters.
<i>roDeviceInfo</i>	Used to retrieve a wide array of system information, including model type, device serial number, and firmware version.

Interfaces and Methods

Every BrightScript object consists of one or more "interfaces." An interface consists of one or more "methods." For example, the *roVideoPlayer* object has several interfaces, including *ifMessagePort*. The interface *ifMessagePort* has one method: `SetPort()`.

The abstract interface *ifMessagePort* is exposed and implemented by both the *roControlPort* and the *roVideoPlayer* objects. Once the `SetPort()` method is called, these objects will send their events to the supplied message port. This is discussed more in the Event Loops section below.

Example

```
p = CreateObject("roMessagePort")
video = CreateObject("roVideoPlayer")
gpio = CreateObject("roControlPort", "BrightSign")
gpio.SetPort(p)
video.SetPort(p)
```

The above syntax makes use of a shortcut provided by the language: The interface name is optional, unless it is needed to resolve name conflicts. For example, the following two lines of code carry out the exact same function:

```
gpio.SetPort(p)
gpio.ifMessagePort.SetPort(p)
```

BrightScript Objects consist *only* of interfaces, and interfaces define *only* methods. There is no concept of a "property" or variable at the object or interface level. These must be implemented as "set" or "get" methods in an interface.

Classes

A *class name* is used to create a BrightScript object. For example, the class name for a video playback instance is *roVideoPlayer*, so, to initialize a video playback instance, you would use code similar to the following:

```
Example  
  
video = CreateObject("roVideoPlayer")
```

Note that "video" can be any name that follows the syntax outlined in the next section.

Object and Class Name Syntax

Class names have the following characteristics:

- Must start with an alphabetic character (a – z).
- May consist of alphabetic characters, numbers, or the "_" (i.e. underscore) symbol.
- Are not case sensitive.
- May be of any reasonable length.

Zones

With the BrightSign Zones feature, you can divide the screen into rectangles and play different content in each rectangle.

Depending on the BrightSign model, zones can contain video, images, HTML content, audio, a clock, or text. 4Kx42, XDx32, and XDx30 models can display two video zones on screen, while the HDx22, HDx20, and LSx22 models can only display one. There can be multiple zones of other types on the screen. A text zone can contain simple text strings or can be configured to display an RSS feed in a ticker-type display.

As of firmware 6.0.x, zone support is enabled by default. When zones are enabled, the image layer is on top of the video layer by default. The default behavior can be modified using the *roVideoMode.SetGraphicsZOrder()* method.

Zone support can be disabled by calling `EnableZoneSupport(false)`. When zones are not enabled, the image layer is hidden whenever video is played, and the video layer is hidden whenever images are played.

Event Loops

When writing anything more than a very simple script, an "event loop" will need to be created. Event loops typically have the following structure:

1. Wait for an event.
2. Process the event.
3. Return to step 1.

An event can be any number occurrences: a button has been pressed; a timer has been triggered; a UDP message has been received; a video has finished playing back; etc. By convention, event scripting for BrightScript objects follows this work flow:

1. An object of the type *roMessagePort* is created by the script.
2. Objects that can send events (i.e. those that support the *ifMessagePort/ifSetMessagePort* interface) are instructed to send their events to this message port using the `SetPort()` method. You can set up multiple message ports and have each event go to its own message port, but it is usually simpler to create one message port and have all the events sent to this one port.
3. The script waits for an event using the built-in `wait()` statement (or the *ifMessagePort.WaitMessage()* method).
4. If multiple object instances are assigned to the same message port, the script determines from which instance the event originated, then processes it. The script then jumps back to the `wait()` statement.

```
Example  
  
'Listens for UDP messages on two different ports and displays the image file requested in the  
UDP message body.  
mp = CreateObject("roMessagePort")
```

```

udp1 = CreateObject("roDatagramReceiver", 3000)
udp1.SetPort(mp)
udp1.SetUserData("port 3000")

udp2 = CreateObject("roDatagramReceiver", 5000)
udp2.SetPort(mp)
udp2.SetUserData("port 5000")

r = CreateObject("roRectangle",0, 0, 1920, 1080)
img = CreateObject("roImagePlayer")
img.SetRectangle(r)

loop:
event = Wait(0, mp)
if type(event) = "roDatagramEvent"
    print "Image play command received on " + event.GetUserData() + "."
    img = DisplayFile("SD:" + event.GetString())
endif
endif
goto loop

```

Global Functions

ON THIS PAGE

- [ifGlobal](#)
 - [CreateObject\(name As String\) As Object](#)
 - [RestartScript\(\) As Void](#)
 - [RestartApplication\(\) As Void](#)
 - [Sleep\(milliseconds As Integer\)](#)
 - [asc\(letter As String\) As Integer](#)
 - [chr\(character As Integer\) As String](#)
 - [len\(target_string As String\) As Integer](#)
 - [str\(value As Double\) As String](#)
 - [strl\(value As Integer\) As String](#)
 - [val\(target_string As String\) As Double](#)
 - [abs\(x As Double\) As Double](#)
 - [atn\(x As Double\) As Double](#)
 - [csng\(x As Integer\) As Float](#)
 - [cdbl\(x As Integer\) As Double](#)
 - [cint\(x As Double\) As Integer](#)
 - [cos\(x As Double\) As Double](#)
 - [exp\(x As Double\) As Double](#)
 - [fix\(x As Double\) As Integer](#)
 - [int\(x As Double\) As Integer](#)
 - [log\(x As Double\) As Double](#)
 - [sgn\(x As Double\) As Integer](#)
 - [sgnl\(x As Integer\) As Integer](#)
 - [sin\(x As Double\) As Double](#)
 - [tan\(x As Double\) As Double](#)
 - [sqr\(x As Double\) As Double](#)
 - [Left\(target_string As String, n As Integer\) As String](#)
 - [Right\(target_string As String, n As Integer\) As String](#)
 - [Stringl\(n As Integer, character As Integer\) As String](#)
 - [String\(n As Integer, character As String\) As String](#)
 - [Mid\(target_string As String, start_position As Integer, length As Integer\) As String](#)
 - [Instr\(start_position As Integer, search_text As String, substring As String\) As Integer](#)
 - [GetInterface\(object As Object, ifname As String\) As Interface](#)
 - [Wait\(timeout As Integer, port As Object\) As Object](#)
 - [ReadAsciiFile\(file_path As String\) As String](#)
 - [WriteAsciiFile\(file_path As String, buffer As String\) As Boolean](#)

- `ListDir(path As String) As Object`
- `MatchFiles(path As String, pattern_in As String) As Object`
- `LCase(target_string As String) As String`
- `UCase(target_string As String) As String`
- `DeleteFile(file_path As String) As Boolean`
- `DeleteDirectory(directory As String) As Boolean`
- `CreateDirectory(directory As String) As Boolean`
- `RebootSystem() As Void`
- `ShutdownSystem() As Void`
- `UpTime(dummy As Integer) As Double`
- `FormatDrive(drive As String, fs_type As String) As Boolean`
- `EjectDrive(drive As String) As Boolean`
- `CopyFile(source As String, destination As String) As Boolean`
- `MoveFile(source As String, destination As String) As Boolean`
- `MapFilenameToNative(path As String) As String`
- `strtoi(target_string As String) As Integer`
- `rnd(a As Dynamic) As Dynamic`
- `RunGarbageCollector() As roAssociativeArray`
- `GetDefaultDrive() As String`
- `SetDefaultDrive(drive As String)`
- `EnableZoneSupport(enable As Boolean)`
- `EnableAudioMixer(enable As Boolean)`
- `Pi() As Double`
- `ParseJson(json_string As String) As Object`
- `FormatJson(json As roAssociativeArray, flags As Integer) As String`

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

BrightScript provides a set of standard, module-scope functions that are stored in the global object. If a global function is referenced, the compiler directs the runtime to call the appropriate global object member. When calling a global function, you do not need to use the **dot operator** to reference the *roGlobal* object.

Note

Global trigonometric functions accept and return values in radians, not degrees.

ifGlobal

CreateObject(name As String) As Object

Creates a BrightScript object corresponding to the specified class name. This method returns invalid if object creation fails. Some objects have parameters in their constructor, which must be passed after the class `name` in a comma-separated list.

```
sw = CreateObject("roGpioControlPort")
serial = CreateObject("roSerialPort", 0, 9600)
```

RestartScript() As Void

Exits the current script. The system then scans for a valid *autorun.brs* file to run.

RestartApplication() As Void

Restarts the BrightSign application.

Sleep(milliseconds As Integer)

Instructs the script to pause for a specified amount of time without wasting CPU cycles. The sleep interval is specified in milliseconds.

asc(letter As String) As Integer

Returns the ASCII code for the first character of the specified string. A null-string argument will cause an error.

chr(character As Integer) As String

Returns a one-character string containing a character reflected by the specified ASCII or control. For example, because quotation marks are normally used as string delimiters, you can pass ASCII code 34 to this function to add quotes to a string.

len(target_string As String) As Integer

Returns the number of characters in a string.

str(value As Double) As String

Converts a specified float value to a string. This method also returns a string equal to the character representation of a value. For example, if A is assigned a value of 58.5, then calling `str(A)` will return "58.5" as a string.

strl(value As Integer) As String

Converts a specified integer value to a string. This method also returns a string equal to the character representation of a value. For example, if A is assigned a value of 58.5, then calling `strl(A)` will return "58" as a string.

val(target_string As String) As Double

Returns a number represented by the characters in the string argument. This is the opposite of the `str()` function. For example, if A is assigned the string "58", and B is assigned the string "5", then calling `val(A+"."+B)` will return the float value 58.5.

abs(x As Double) As Double

Returns the absolute value of the argument `x`.

atn(x As Double) As Double

Returns the arctangent (in radians) of the argument `x` (i.e. `Atn(x)` returns "the angle whose tangent is `x`"). To get the arctangent in degrees, multiply `Atn(x)` by 57.29578.

csng(x As Integer) As Float

Returns a single-precision float representation of the argument `x`.

cdbl(x As Integer) As Double

Returns a double-precision float representation of the argument `x`.

cint(x As Double) As Integer

Returns an integer representation of the argument `x` by rounding to the nearest whole number.

cos(x As Double) As Double

Returns the cosine of the argument `x`. The argument must be in radians. To obtain the cosine of `x` when `x` is in degrees, use `Cos(x*.01745329)`.

exp(x As Double) As Double

Returns the natural exponential of `x`. This is the inverse of the `log()` function.

fix(x As Double) As Integer

Returns a truncated representation of the argument `x`. All digits to the right of the decimal point are removed so that the resultant value is an integer. For non-negative values of `x`, `fix(x)` is equal to `int(x)`. For negative values of `x`, `fix(x)` is equal to `int(x)+1`.

int(x As Double) As Integer

Returns an integer representation of the argument `x` using the largest whole number that is not greater than the argument. For example, `int(2.2)` returns 2, while `fix(-2.5)` returns -3.

log(x As Double) As Double

Returns the natural logarithm of the argument `i` (i.e. $\log_e(x)$). This is the inverse of the `exp()` function. To find the logarithm of a number to a base `b`, use the following formula: $\log_b(x) = \log_e(x) / \log_e(b)$.

sgn(x As Double) As Integer

Returns an integer representing how the float argument `x` is signed: -1 for negative, 0 for zero, and 1 for positive.

sgnl(x As Integer) As Integer

Returns an integer representing how the integer argument `x` is signed: -1 for negative, 0 for zero, and 1 for positive.

sin(x As Double) As Double

Returns the sine of the argument `x`. The argument must be in radians. To obtain the sine of `x` when `x` is in degrees, use `sin(x * 0.01745329)`.

tan(x As Double) As Double

Returns the tangent of the argument `x`. The argument must be in radians. To obtain the tangent of `x` when `x` is in degrees, use `tan(x * 0.01745329)`.

sqr(x As Double) As Double

Returns the square root of the argument `x`. This function is the same as $x^{(1/2)}$, but calculates the result faster.

Left(target_string As String, n As Integer) As String

Returns the first `n` characters of the specified string.

Right(target_string As String, n As Integer) As String

Returns the last `n` characters of the specified string.

Stringl(n As Integer, character As Integer) As String

Returns a string composed of a character symbol repeated `n` times. The character symbol is passed to the method as an ASCII code integer.

String(n As Integer, character As String) As String

Returns a string composed of a character symbol repeated `n` times. The character symbol is passed to the method as a string.

Mid(target_string As String, start_position As Integer, length As Integer) As String

Returns a substring of the target string. The first integer passed to the method specifies the starting position of the substring, and the second integer specifies the length of the substring. The start position of a string begins with 1.

Instr(start_position As Integer, search_text As String, substring As String) As Integer

Returns the position of a substring within a string. This function is case sensitive and returns 0 if the specified substring is not found. The start position of a string begins with 1.

Tip

The string object also offers an `Instr()` method (though it uses a zero-based index). See the [roString](#) documentation for more details.

GetInterface(object As Object, ifname As String) As Interface

Returns a value of the type `Interface`. All objects have one or more interfaces. In most cases, you can skip interface specification when calling an object component. This will not cause problems as long as the method names within a function are unique.

Wait(timeout As Integer, port As Object) As Object

Instructs the script to wait on an object that has an *ifMessagePort* interface. This method will return the event object that was posted to the message port. If the timeout is specified as zero, `wait()` will wait indefinitely; otherwise, `wait()` will return `Invalid` after the specified number of milliseconds if no messages have been received.

```
p = CreateObject("roMessagePort")
sw = CreateObject("roGpioControlPort")
sw.SetPort(p)
msg=wait(0, p)
print type(msg)      ' should be roGpioButton
print msg.GetInt()  ' button number
```

ReadAsciiFile(file_path As String) As String

Reads the specified text file and returns it as a string.

WriteAsciiFile(file_path As String, buffer As String) As Boolean

Creates a text file at the specified file path. The text of the file is passed as the second parameter. This method cannot be used to edit files: A preexisting text file will be overwritten if it has the same name and directory path as the one being created.

Note

The *roCreateFile* object provides more flexibility if you need to create or edit files.

ListDir(path As String) As Object

Returns an *roList* object containing the contents of the specified directory path. File names are converted to all lowercase.

MatchFiles(path As String, pattern_in As String) As Object

Takes a directory to look in (it can be as simple as "." or "/") and a pattern to be matched and then returns an *roList* containing the results. Each listed result contains only the part of the filename that is matched against the pattern, not the full path. The match is only applied in the specified directory; you will get no results if the pattern contains a directory separator. The pattern is a case insensitive wildcard expression. It may contain the following special characters:

- ? – Matches any single character.
- * – Matches zero or more arbitrary characters.
- [...] – Matches any single character specified within the brackets. The closing bracket is treated as a member of the character class if it immediately follows the opening bracket (i.e. "[]]" matches a single closed bracket). Within this class, "-" can be used to specify a range unless it is the first or last character (e.g. "[A-Cf-h]" is equivalent to "[ABCfgh]"). A character class may be negated by specifying "^" as the first character. To match a literal of this character, place it elsewhere in the class.

Note

The special characters "?", "*", and "[" lose their function if preceded by a single "\", and a single "\" can be matched using "\\".

LCase(target_string As String) As String

Converts the specified string to all lowercase.

UCase(target_string As String) As String

Converts the specified string to all uppercase.

DeleteFile(file_path As String) As Boolean

Deletes the file at the specified file path. This method returns `False` if the delete operation fails or if the file does not exist.

DeleteDirectory(directory As String) As Boolean

Deletes the specified directory. This method will recursively delete any files and directories that are necessary for removing the specified directory. This method returns `False` if it fails to delete the directory, but it may still delete some of the nested files or directories.

CreateDirectory(directory As String) As Boolean

Creates the specified directory. Only one directory can be created at a time. This method returns True upon success and False upon failure.

RebootSystem() As Void

Instructs the player to perform a soft reboot.

ShutdownSystem() As Void

UpTime(dummy As Integer) As Double

Returns the uptime of the system (in seconds) since the last reboot.

FormatDrive(drive As String, fs_type As String) As Boolean

Formats the specified drive using one of the file systems listed below. This function returns True upon success and False upon failure:

- `vfat` (DOS/Windows file system): Readable and writable by Windows, Linux, and MacOS.
- `ext2` (Linux file system): Writable by Linux and readable by Windows and MacOS with additional software.
- `ext3` (Linux file system): Writable by Linux and readable by Windows and MacOS with additional software.
- `ext4` (Linux file system): Writable by Linux and readable by Windows and MacOS with additional software. This is the recommended file system for SSD devices and USB hard drives.

EjectDrive(drive As String) As Boolean

Ejects the specified drive (e.g. "SD:") and returns True if successful. If the script is currently accessing files from the specified drive, the ejection process will fail.

CopyFile(source As String, destination As String) As Boolean

Copies the file at the specified source file-path to the specified destination directory. The function returns True if successful and False in the event of failure.

MoveFile(source As String, destination As String) As Boolean

Moves the specified source file to the specified destination directory. The function returns True if successful and False in the event of failure.

Note

Both path names must be on the same drive.

MapFilenameToNative(path As String) As String

Converts the specified BrightScript-style path to the corresponding native path and returns it as a string (e.g. the path "SD:/mydir" will be returned as "/storage/sd/mydir").

strtoi(target_string As String) As Integer

Converts the target string to an integer. Any non-integer characters (including decimal points and spaces), and any numbers to the right of a non-integer character, will not be part of the integer output.

rnd(a As Dynamic) As Dynamic

RunGarbageCollector() As roAssociativeArray

Destroys objects that are currently in a state of circular reference counting. BrightScript normally removes any objects that become unreferenced as part of its automated garbage collection algorithm. However, objects that reference each other will never reach a reference count of zero, and will need to be destroyed manually using this method.

This method is useful when destroying old presentation data structures and generating a new presentation. This method returns an associative array outlining the results of the garbage-collection process.

GetDefaultDrive() As String

Returns the current default drive complete with a trailing slash. When running as *autorun.brs*, the drive containing the autorun is designated as the current default.

SetDefaultDrive(drive As String)

Sets the current default drive, which does not need to include a trailing slash. This method does not fail; however, if the specified default drive does not exist, it will not be possible to retrieve anything.

This method accepts the following values:

- "USB1:" – The drive for USB storage devices connected to the player
- "SD:" – The primary SD or microSD drive on the player.
- "SD2:" – The internal microSD drive on the player (4Kx42, XDx32 models only)
- "SSD:" – The internal SSD on the player (XTx44, XTx34, XDx34, XDx33 models only)

EnableZoneSupport(enable As Boolean)

Allows for display of multiple video, HTML, image, and text zones. As of firmware 6.0.x, zone support is enabled by default.

EnableAudioMixer(enable As Boolean)

Pi() As Double

Returns the value of pi as a double-precision floating-point number.

ParseJson(json_string As String) As Object

Parses a string formatted according to the RFC4627 standard and returns an equivalent BrightScript object, which can consist of the following: Booleans, integers, floating point numbers, strings, *roArray* objects, and *roAssociativeArray* objects. The `ParseJson()` method has the following properties:

- Invalid will be returned if the string is not syntactically correct.
- Any *roAssociativeArray* objects that are returned will be case sensitive.
- An error will be returned if an *roArray* or *roAssociativeArray* is nested more than 256 levels deep.

The following script demonstrates how to use `ParseJson()` to process a JSON object containing the titles and URLs of a set of images.

JSON Script

```
{
  "photos" : [
    {
      "title" : "View from the hotel",
      "url" : "http://example.com/images/00012.jpg"
    },
    {
      "title" : "Relaxing at the beach",
      "url" : "http://example.com/images/00222.jpg"
    },
    {
      "title" : "Flat tire",
      "url" : "http://example.com/images/00314.jpg"
    }
  ]
}
```

BrightScript

```
searchRequest = CreateObject("roUrlTransfer")
searchRequest.SetURL("http://api.example.com/services/rest/getPhotos")
response = ParseJson(searchRequest.GetToString())
For Each photo In response.photos
  GetImage(photo.title, photo.url)
End For
```

FormatJson(json As roAssociativeArray, flags As Integer) As String

Converts an associative array to a JSON string (i.e. formatted according to the RFC4627 standard). The following are supported data types: Boolean, Integer, Float, String, *roArray*, and *roAssociativeArray*. If the `flags` parameter is set to 0 or not specified, non-ASCII characters are escaped in the output string as `"\uXXXX"`, where `"XXXX"` is the hexadecimal representation of the Unicode character value. If the `flags` parameter is set to 1, non-ASCII characters are not escaped.

If arrays or associative arrays are nested more than 256 levels deep, an error will occur. If an error occurs, an empty string will be returned.

Important

By default, using object-literal syntax (e.g. `aa={relativePath:"Foo"}`) to generate an associative array will convert keys to all lower case. To preserve camel case for JSON, use the *roAssociativeArray.AddReplace()* method instead of object literals or call *roAssociativeArray.SetModeCaseSensitive()* before adding entries.

BrightScript Core Objects

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This section describes objects that provide core BrightScript functionality.

- [roArray](#)
- [roAssociativeArray](#)
- [roBoolean](#)
- [roByteArray](#)
- [roDouble](#), [roIntrinsicDouble](#)
- [roFunction](#)
- [roInt](#), [roFloat](#), [roString](#)
- [roJRE](#)
- [roList](#)
- [roMessagePort](#)
- [roRegex](#)
- [roXMLElement](#)
- [roXMLList](#)

roArray

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

ON THIS PAGE

- [ifArray](#)
 - [Peek\(\) As Dynamic](#)

- `Pop()` As Dynamic
- `Push(entry As Dynamic)`
- `Shift()` As Dynamic
- `Unshift(entry As Dynamic)`
- `Delete(index As Integer) As Boolean`
- `Count()` As Integer
- `Clear()`
- `Append(array As roArray)`
- `ifEnum`
 - `Reset()`
 - `Next()` As Dynamic
 - `IsNext()` As Boolean
 - `IsEmpty()` As Boolean
- `ifArrayGet`
 - `GetEntry(index As Integer) As Dynamic`
- `ifArraySet`
 - `SetEntry(a As Integer, b As Dynamic)`

This object stores objects in a continuous array of memory locations. Since an *roArray* contains BrightScript components, and there are object wrappers for most intrinsic data types, entries can either be different types or all of the same type.

Object Creation: The *roArray* object is created with two parameters.

```
CreateObject("roArray", size As Integer, resize As Boolean)
```

- `size`: The initial number of entries allocated for the array.
- `resize`: If true, the array will be resized larger to accommodate more entries if needed. If the array is large, this process might take some time.

The `DIM` statement may be used instead of the `CreateObject()` function to create a new array. The `DIM` statement can be advantageous because it automatically creates array-of-array structures for multi-dimensional arrays.

`ifArray`

***Peek()* As Dynamic**

Returns the last (highest index) array entry without removing it.

***Pop()* As Dynamic**

Returns the last (highest index) entry and removes it from the array.

Push(entry As Dynamic)

Adds a new highest-index entry to the end of the array.

***Shift()* As Dynamic**

Removes index zero from the array and shifts all other entries down by one unit.

Unshift(entry As Dynamic)

Adds a new index zero to the array and shifts all other entries up by one unit.

Delete(index As Integer) As Boolean

Deletes the indicated array entry and shifts all above entries down by one unit.

***Count()* As Integer**

Returns the length of the array (i.e. the index of the highest entry in the array plus one).

Clear()

Deletes every entry in the array.

Append(array As roArray)

Appends one *roArray* to another. If the passed *roArray* contains entries that were never set to a value, they are not appended.

Note

The two appended objects must be of the same type.

ifEnum

Reset()

Resets the position to the first element of enumeration.

Next() As Dynamic

Returns a typed value at the current position and increments the position.

IsNext() As Boolean

Returns True if there is a next element.

IsEmpty() As Boolean

Returns True if there is not an exact statement.

ifArrayGet

GetEntry(index As Integer) As Dynamic

Returns an array entry of a given index. Entries start at zero. If the entry at the specified index has not been set, this method will return Invalid.

ifArraySet

SetEntry(a As Integer, b As Dynamic)

Sets an entry of a given index to the passed type value.

roAssociativeArray

ON THIS PAGE

- **ifEnum**
 - **Reset() As Void**
 - **Next() As Dynamic**
 - **IsNext() As Boolean**
 - **IsEmpty() As Boolean**
- **ifAssociativeArray**
 - **AddReplace(key As String, value As Object) As Void**
 - **Lookup(key As String) As Dynamic**
 - **DoesExist(key As String) As Boolean**
 - **Delete(key As String) As Boolean**
 - **Clear() As Void**
 - **SetModeCaseSensitive() As Void**
 - **LookupCi(key As String) As Dynamic**
 - **Append(aa As roAssociativeArray) As Void**

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object allows you to store objects in an associative array (also known as a map, dictionary, or hash table), a data structure that associates objects with string keys.

The `roAssociativeArray` object is created with no parameters:

```
CreateObject("roAssociativeArray")
```

Alternatively, an associative array can be created using brackets:

Example

```
aa1 = {}  
aa2 = {key1:"value", key2: 55, key3: 5+3 }
```

`ifEnum`

Reset() As Void

Resets the position to the first element of enumeration.

Next() As Dynamic

Returns a typed value at the current position and increments the position.

IsNext() As Boolean

Returns `true` if there is a next element.

IsEmpty() As Boolean

Returns `true` if the associative array contains no elements.

`ifAssociativeArray`

AddReplace(key As String, value As Object) As Void

Adds a new entry to the associative array, associating the supplied object with the supplied key string. Only one object may be associated with a key, so any existing object linked to that key is discarded. This method is always case-sensitive when creating keys, whereas object-literal syntax (e.g. `aa={bright:"Sign"}`) is case-insensitive when creating keys unless `SetModeCaseSensitive()` is called.

Lookup(key As String) As Dynamic

Looks up the specified key and returns the associated object. If there is no object associated with the key string, then this method will return `Invalid`.

Tip

In many cases, the [Dot Operator](#) can be used as shorthand for the `Lookup()` and `AddReplace()` methods when working with associative arrays.

DoesExist(key As String) As Boolean

Looks up the specified key in the associative array. If the key exists, `true` is returned; otherwise, `false` is returned.

Delete(key As String) As Boolean

Looks for an object in the associative array linked to the specified key. If there is such an object, it is deleted and `true` is returned; otherwise, `false` is returned.

Clear() As Void

Removes all objects from the associative array.

SetModeCaseSensitive() As Void

Makes all subsequent actions case sensitive. All lookups and created keys (with the exception of the `AddReplace()` method) are case insensitive by default.

LookupCi(key As String) As Dynamic

Looks for an object in the array associated with the specified key. This method functions similarly to `Lookup()`, with the exception that key comparisons are always case insensitive, regardless of case mode.

Append(aa As roAssociativeArray) As Void

Appends a second associative array to the first.

Example

```
aa = CreateObject("roAssociativeArray")
aa.AddReplace("Bright", "Sign")
aa.AddReplace("TMOL", 42)
print aa.Lookup("tmol")
print aa.Lookup("bright")
```

The above script returns the following:

```
42
Sign
```

Alternatively, you can use the **Dot Operator** in place of the `AddReplace()` and `Lookup()` methods:

Example

```
aa = {}
aa.bright = "Sign"
aa.tmol = 42
print aa.tmol
print aa.bright
```

You can also specify an associative array as a multiline object literal:

Example

```
aa = {
  bright : "Sign",
  tmol : 42,
```

```
pie : 3.14
}
```

roBoolean

ON THIS PAGE

- [ifBoolean](#)
 - [GetBoolean\(\) As Boolean](#)
 - [SetBoolean\(a As Boolean\)](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This is the object equivalent of the Boolean intrinsic type. It is useful in the following situations:

- **When an object is needed instead of an intrinsic value:** For example, if a Boolean is added to *roList*, it will be automatically wrapped in an *roBoolean* object by the language interpreter. When a function that expects a BrightScript component as a parameter is passed a Boolean, BrightScript automatically creates the equivalent BrightScript component.
- **When an object exposes the *ifBoolean* interface:** That object can then be used in any expression that expects an intrinsic value.

ifBoolean

GetBoolean() As Boolean

SetBoolean(a As Boolean)

roByteArray

ON THIS PAGE

- [ifByteArray](#)
 - [WriteFile\(file_path As String\) As Boolean](#)
 - [WriteFile\(file_path As String, start_index As Integer, length As Integer\) As Boolean](#)
 - [ReadFile\(file_path As String\) As Boolean](#)
 - [ReadFile\(file_path As String, start_index As Integer, length As Integer\) As Boolean](#)
 - [AppendFile\(file_path As String\) As Boolean](#)
 - [SetResize\(minimum_allocation_size As Integer, autoresize As Boolean\)](#)
 - [ToHexString\(\) As String](#)
 - [FromHexString\(hex_string As String\)](#)
 - [ToBase64String\(\) As String](#)
 - [FromBase64String\(base_64_string As String\)](#)
 - [ToAsciiString\(\) As String](#)
 - [FromAsciiString\(ascii_string As String\)](#)
 - [GetSignedByte\(index As Integer\) As Integer](#)
 - [GetSignedLong\(index As Integer\) As Integer](#)
 - [IsLittleEndianCPU\(\) As Boolean](#)
- [ifArray](#)
 - [Peek\(\) As Dynamic](#)
 - [Pop\(\) As Dynamic](#)
 - [Push\(entry As Dynamic\)](#)
 - [Shift\(\) As Dynamic](#)
 - [Unshift\(entry As Dynamic\)](#)
 - [Delete\(index As Integer\) As Boolean](#)

- `Count()` As Integer
- `Clear()`
- `Append(array As roArray)`
- `ifEnum`
 - `Reset()`
 - `Next()` As Dynamic
 - `IsNext()` As Boolean
 - `IsEmpty()` As Boolean
- `ifArrayGet`
 - `GetEntry(index As Integer)` As Dynamic
- `ifArraySet`
 - `SetEntry(index As Integer, entry As Dynamic)`

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object contains functions for converting strings to or from a byte array, as well as to or from ASCII hex or ASCII base64.

Note

If you are converting a byte array to a string, and the byte array contains a zero, the string conversion will end at that point.

The byte array will automatically resize to become larger as needed. If you wish to disable this behavior, use the `SetResize()` method. If an uninitialized index is read, `Invalid` is returned.

Since `roByteArray` supports the `ifArray` interface, it can be accessed with the `[]` array operator. The byte array is always accessed as unsigned bytes while this interface is being used. This object also supports the `ifEnum` interface, and so can be used with a `FOR EACH` statement.

`ifByteArray`

WriteFile(file_path As String) As Boolean

Writes the bytes contained in the byte array to the specified file. This method returns `True` if successful.

WriteFile(file_path As String, start_index As Integer, length As Integer) As Boolean

Writes a subset of the bytes contained in the byte array to the specified file. This method writes `length` bytes, beginning at `start_index` of the byte array.

ReadFile(file_path As String) As Boolean

Reads the specified file into the byte array. This operation will discard any data currently contained in the byte array.

ReadFile(file_path As String, start_index As Integer, length As Integer) As Boolean

Reads a section of the specified file into the byte array. This method reads `length` bytes, beginning at `start_index` of the file. This operation will discard any data currently contained in the byte array.

AppendFile(file_path As String) As Boolean

Appends the contents of the byte array to the specified file.

SetResize(minimum_allocation_size As Integer, autoresize As Boolean)

Expands the size of the byte array to the `minimum_allocation_size` if it is less than the `minimum_allocation_size`. This method also accepts a Boolean parameter that specifies whether the byte array should be resized automatically or not.

ToHexString() As String

Returns a hexadecimal string representation of the contents of the byte array. Each byte is represented as two hex digits.

FromHexString(hex_string As String)

Writes the contents of the passed hexadecimal string to the byte array. The passed string must contain an even number of hex digits. This operation will discard any data currently contained in the byte array.

ToBase64String() As String

Returns the contents of the byte array as a base64-formatted string.

FromBase64String(base_64_string As String)

Writes the contents of a valid base64-formatted string to the byte array. This operation will discard any data currently contained in the byte array.

ToAsciiString() As String

Returns the contents of the byte array as an ASCII-formatted string.

FromAsciiString(ascii_string As String)

Writes the contents of a valid ASCII-formatted string to the byte array. This operation will discard any data currently contained in the byte array.

GetSignedByte(index As Integer) As Integer

Returns the signed byte at the specified zero-based index in the byte array. To read an unsigned byte within a byte array, use the *ifArrayGet.GetEntry()* method or the [] array operator.

GetSignedLong(index As Integer) As Integer

Retrieves the integer located at the specified long-word index of the byte array. Note that this method cannot accept a byte index as its parameter.

IsLittleEndianCPU() As Boolean

Returns True if the CPU architecture is little-endian.

ifArray

Peek() As Dynamic

Returns the last (highest index) array entry without removing it.

Pop() As Dynamic

Returns the last (highest index) entry and removes it from the array.

Push(entry As Dynamic)

Adds a new highest index entry to the end of the array.

Shift() As Dynamic

Removes index zero from the array and shifts all other entries down by one unit.

Unshift(entry As Dynamic)

Adds a new index zero to the array and shifts all other entries up by one unit.

Delete(index As Integer) As Boolean

Deletes the indicated array entry and shifts all above entries down by one unit.

Count() As Integer

Returns the index of the highest entry in the array plus one (i.e. the length of the array).

Clear()

Deletes every entry in the array.

Append(array As roArray)

Appends one *roArray* to another. If the passed *roArray* contains entries that were never set to a value, they are not appended.

Note

The two appended objects must be of the same type.

ifEnum

Reset()

Resets the position to the first element of enumeration.

Next() As Dynamic

Returns a typed value at the current position and increment position.

IsNext() As Boolean

Returns True if there is a next element.

IsEmpty() As Boolean

Returns True if there is not an exact statement.

ifArrayGet

GetEntry(index As Integer) As Dynamic

Returns an array entry of a given index. Entries start at zero. If this method attempts to fetch an entry that has not been set, it will return Invalid.

ifArraySet

SetEntry(index As Integer, entry As Dynamic)

Sets an entry of a given index to the passed type value.

roDouble, roIntrinsicDouble

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

ON THIS PAGE

- [ifDouble](#)
 - [GetDouble\(\) As Double](#)
 - [SetDouble\(a As Double\)](#)

ifDouble

GetDouble() As Double

SetDouble(a As Double)

roFunction

ON THIS PAGE

- [ifFunction](#)
 - [GetSub\(\) As Function](#)
 - [SetSub\(value As Function\)](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

`ifFunction`

GetSub() As Function

SetSub(value As Function)

roInt, roFloat, roString

ON THIS PAGE

- [ifInt](#)
 - [GetInt\(\) As Integer](#)
 - [SetInt\(value As Integer\) As Void](#)
- [ifIntOps](#)
 - [ToStr\(\) As String](#)
- [ifFloat](#)
 - [GetFloat\(\) As Float](#)
 - [SetFloat\(value As Float\) As Void](#)
- [ifString](#)
 - [GetString\(\) As String](#)
 - [SetString\(value As String\) As Void](#)
- [ifStringOps](#)
 - [SetString\(str As String, str_len As Integer\)](#)
 - [AppendString\(str As String, str_len As Integer\)](#)
 - [Len\(\) As Integer](#)
 - [GetEntityEncode\(\) As String](#)
 - [Tokenize\(delim As String\) As roList](#)
 - [Trim\(\) As String](#)
 - [ToInt\(\) As Integer](#)
 - [ToFloat\(\) As Float](#)
 - [Left\(n As Integer\) As String](#)
 - [Right\(n As Integer\) As String](#)
 - [Mid\(start_index As Integer\) As String](#)
 - [Mid\(start_index As Integer, n As Integer\) As String](#)
 - [Instr\(substring As String\) As Integer](#)
 - [Instr\(start_index As Integer, substring As String\) As Integer](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)

- [Version 6.1](#)
- [Previous Versions](#)

The **intrinsic types** `Int32`, `Float`, and `String` have object and interface equivalents. These are useful in the following situations:

- An object is needed instead of a typed value (e.g. the `roList` object maintains a list of objects). When a function that expects a BrightScript object as a parameter is passed an integer, float, or string, BrightScript automatically creates the equivalent object.
- If any object exposes the `ifInt`, `ifFloat`, or `ifString` interfaces, that object can be used in any expression that expects a typed value. For example, an `roVideoEvent` can be used as an integer with a value representing the event ID.

Integer Operations

If "o" is of type `roInt`, then these statements will have the following effects:

- `print o`: Prints the value of `o.GetInt()`
- `i%=o`: Assigns the integer `i%` the value of `o.GetInt()`.
- `k=o`: Presumably `k` is automatically typed, so it becomes another reference to the `roInt` `o`.
- `o=5`: This is NOT the same as `o.SetInt(5)`. Instead it releases `o`, changes the type of `o` to `roInt32` (`o` is automatically typed), and assigns it to 5.

`ifInt`

`roInt` contains the `ifInt` interface, which provides the following:

GetInt() As Integer

Returns the integer value of the object.

SetInt(value As Integer) As Void

Sets the integer value of the object.

`ifIntOps`

`roInt` also contains the `ifIntOps` interface, which provides the following:

ToStr() As String

Returns the integer value as a string. A space is not appended to the front for positive numbers.

`ifFloat`

`roFloat` contains the `ifFloat` interface, which provides the following:

GetFloat() As Float

Returns the float value of the object.

SetFloat(value As Float) As Void

Sets the float value of the object.

`ifString`

`roString` contains the `ifString` interface, which provides the following:

GetString() As String

Returns the string value of the object.

SetString(value As String) As Void

Sets the string value of the object.

`ifStringOps`

`roString` also contains the `ifStringOps` interface, which provides the following:

Note

Some **global functions** offer the same functionality as *ifStringOps* methods. The function indexes of *ifStringOps* methods start at zero, while those of global functions start at one.

SetString(str As String, str_len As Integer)

Sets the string value of the object using the specified string and string-length values. This is similar to the `SetSetting()` method, which does not accept a parameter for string length.

AppendString(str As String, str_len As Integer)

Appends to the string value of the object using the specified string and string-length values. This method modifies itself—this can cause unexpected results when you pass an intrinsic string type, rather than a string object.

Example

```
x="string"
x.ifstringops.appendstring("ddd",3)
print x 'will print 'string'
y=box("string")
y.ifstringops.appendstring("ddd",3)
print y 'will print 'stringddd'
```

Len() As Integer

Returns the number of characters in a string.

GetEntityEncode() As String

Returns the string with certain characters replaced with HTML entity encoding sequences:

Character	Replaced with
" (double quote)	"
' (single quote)	'
<	<
>	>
&	&

Tokenize(delim As String) As roList

Splits a string into substrings using the specified delimiter character(s). The `delim` parameter can contain one or more characters to treat as delimiters. If the string object contains multiple contiguous delimiters, they will be treated as a single delimiter. This method returns the substrings as an *roList* object; the delimiters are not returned with the substrings.

Example

```
BrightScript> s = "one&&two"
BrightScript> print s.Tokenize("&")
one
two
```

Trim() As String

Returns the string with any leading and trailing whitespace characters (e.g. TAB, LF, CR, VT, FF, NO-BREAK SPACE) removed.

ToInt() As Integer

Returns the value of the string as an integer number.

ToFloat() As Float

Returns the value of the string as a floating point number.

Left(n As Integer) As String

Returns the first *n* characters of the string.

Right(n As Integer) As String

Returns the last *n* characters of the string.

Mid(start_index As Integer) As String

Returns a subset of the string that begins at the zero-based *start_index* and terminates at the end of the string.

Mid(start_index As Integer, n As Integer) As String

Returns a subset of the string, beginning at the zero-based *start_index* and consisting of *n* characters. If the string contains fewer than *n* characters after the specified *start_index*, this method will return all characters after the *start_index*.

Instr(substring As String) As Integer

Returns the zero-based index of the first occurrence of the substring in the string. If the substring does not occur in the string, this method returns -1.

Instr(start_index As Integer, substring As String) As Integer

Returns the zero-based index of the first occurrence of the substring after the specified *start_index* in the string. If the substring does not occur after the specified *start_index*, this method returns -1.

Tip

`Instr()` is also offered as a **global function** (note that the string index of the global function starts at 1).

Example

```
BrightScript> o=CreateObject("roInt")
BrightScript> o.SetInt(555)
BrightScript> print o
555
BrightScript> print o.GetInt()
555
BrightScript> print o-55
500
```

An integer value of 5 is converted to type *roInt* automatically because the `AddTail()` method expects a BrightScript object as its parameter:

Example

```
BrightScript> list=CreateObject("roList")
BrightScript> list.AddTail(5)
BrightScript> print type(list.GetTail())
```

Here the `ListDir()` method returns an *roList* object containing *roString* objects:

Example

```
BrightScript> l=ListDir("/")
BrightScript> for i=1 to l.Count():print l.RemoveHead():next
test_movie_3.vob
test_movie_4.vob
test_movie_1.vob
test_movie_2.vob
```

roJRE

This object allows you to load Java applications using the Java Runtime Environment (JRE) on the player. The JRE utilizes OpenJDK 7 and is available on XTx44, XTx43, XTx34, XDx33, HDx23, LS423, and 4Kx42 models.

OBJECT CREATION

The *roJRE* object is instantiated with the Java package filename and an optional associative array that specifies JVM system properties and program arguments. You can use the `Type()` global function to determine if the package was successfully loaded.

```
CreateObject("roJRE", filename As String, options As roAssociativeArray)
```

The associative array can contain two entries:

- `defines`: An associative array specifying system properties.
- `arguments`: An array specifying command-line arguments.

All property/argument values must be passed as strings. Note that associative-array keys are case insensitive (i.e. converted to all lowercase) by default; use the *roAssociativeArray.SetModeCaseSensitive()* method to enable case-sensitive keys.

Example

```
props = {}
props.SetModeCaseSensitive()
props["SYS_PROP_1"] = "system prop 1"
props["SYS_PROP_2"] = "system prop 2"
props["java.io.tmpdir"] = "/var/tmp"

jre = CreateObject("roJRE", "app.jar", { defines: props, arguments: [ "arg 1", "arg 2" ] })
if type(jre)="roJRE" then
    print "Successfully started Java runtime"
else
    print "Unable to start Java runtime"
end if
```

roList

ON THIS PAGE

- [ifList](#)
 - [Count\(\) As Integer](#)

- `ResetIndex()` As Boolean
- `AddTail(obj As Object)` As Void
- `AddHead(obj As Object)` As Void
- `RemoveIndex()` As Object
- `GetIndex()` As Object
- `RemoveTail()` As Object
- `RemoveHead()` As Object
- `GetTail()` As Object
- `GetHead()` As Object
- `Clear()`
- `ifEnum`
 - `Reset()`
 - `Next()` As Dynamic
 - `IsNext()` As Boolean
 - `IsEmpty()` As Boolean
- `ifArray`
 - `Peek()` As Dynamic
 - `Pop()` As Dynamic
 - `Push(entry As Dynamic)`
 - `Shift()` As Dynamic
 - `Unshift(entry As Dynamic)`
 - `Delete(index As Integer)` As Boolean
 - `Count()` As Integer
 - `Clear()`
 - `Append(list As roList)`
- `ifArrayGet`
 - `GetEntry(a As Integer)` As Dynamic
- `ifArraySet`
 - `SetEntry(a As Integer, b As Dynamic)`

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object functions as a general-purpose, doubly linked list. It can be used as a container for arbitrary-length lists of BrightSign objects. The array operator [] can be used to access any element in an ordered list.

`ifList`

Count() As Integer

Returns the number of elements in the list.

ResetIndex() As Boolean

Resets the current index or position in the list to the head element.

AddTail(obj As Object) As Void

Adds a typed value to the tail of the list.

AddHead(obj As Object) As Void

Adds a typed value to the head of the list.

RemoveIndex() As Object

Removes an entry from the list at the current index or position and increments the index or position in the list. It returns Invalid when the end of the list is reached.

GetIndex() As Object

Retrieves an entry from the list at the current index or position and increments the index or position in the list. It returns Invalid when the end of the list is reached.

RemoveTail() As Object

Removes the entry at the tail of the list.

RemoveHead() As Object

Removes the entry at the head of the list.

GetTail() As Object

Retrieves the entry at the tail of the list and keeps the entry in the list.

GetHead() As Object

Retrieves the entry at the head of the list and keeps the entry in the list.

Clear()

Removes all elements from the list.

ifEnum

Reset()

Resets the position to the first element of enumeration.

Next() As Dynamic

Returns a typed value at the current position and increment position.

IsNext() As Boolean

Returns True if there is a next element.

IsEmpty() As Boolean

Returns True if there is not an exact statement.

ifArray

Peek() As Dynamic

Returns the last (highest index) list entry without removing it.

Pop() As Dynamic

Returns the last (highest index) entry and removes it from the list.

Push(entry As Dynamic)

Adds a new highest-index entry to the end of the list.

Shift() As Dynamic

Removes index zero from the list and shifts all other entries down by one unit.

Unshift(entry As Dynamic)

Adds a new index zero to the list and shifts all other entries up by one unit.

Delete(index As Integer) As Boolean

Deletes the indicated list entry and shifts all above entries down by one unit.

Count() As Integer

Returns the length of the list (i.e. the index of the highest entry in the list plus one).

Clear()

Deletes every entry in the list.

Append(list As roList)

Appends one *roList* to another. If the passed *roList* contains entries that were never set to a value, they are not appended.

Note

The two appended objects must be of the same type.

`ifArrayGet`

GetEntry(a As Integer) As Dynamic

Returns a list entry of a given index. Entries start at zero. If an entry that has not been set is fetched, Invalid is returned.

`ifArraySet`

SetEntry(a As Integer, b As Dynamic)

Sets an entry of a given index to the passed type value.

Example

```
list = CreateObject("roList")
list.AddTail("a")
list.AddTail("b")
list.AddTail("c")
list.AddTail("d")
list.ResetIndex()
x= list.GetIndex()
while x <> invalid
print x
x = list.GetIndex()
end while
print list[2]
```

roMessagePort

ON THIS PAGE

- [ifMessagePort](#)
 - [GetMessage\(\) As Object](#)
 - [WaitMessage\(timeout As Integer\) As Object](#)
 - [PostMessage\(msg As Object\) As Void](#)
 - [PeekMessage\(\) As Object](#)
 - [SetWatchdogTimeout\(seconds As Integer\) As Integer](#)
 - [DeferWatchdog\(seconds As Integer\)](#)
 - [DeferWatchdog\(\)](#)

- `ifEnum`
 - `Reset()`
 - `Next() As Dynamic`
 - `IsNext() As Boolean`
 - `IsEmpty() As Boolean`

▼ Firmware Version 7.0

- `Version 7.0`
- `Version 6.2`
- `Version 6.1`
- `Previous Versions`

A message port is the destination where messages (events) are sent. See the explanation of [Event Loops](#) for more details. You do not call these functions directly when using BrightScript. Instead, use the `wait()` [global function](#).

`ifMessagePort`

GetMessage() As Object

Returns the event object if available. Otherwise, `Invalid` is returned. In either case, this method returns immediately without waiting.

WaitMessage(timeout As Integer) As Object

Waits until an event object is available or the specified amount of milliseconds have passed. When an event object becomes available, it will be returned. If the timeout is reached, `Invalid` will be returned. Passing a zero timeout value instructs this method to wait indefinitely for a message.

You can also use the `wait()` global function to retrieve event objects over a specified interval. The following two statements have the same effect:

```
msg = port.WaitMessage(timeout)
msg = wait(timeout, port)
```

PostMessage(msg As Object) As Void

PeekMessage() As Object

Returns the event object if available (or `Invalid` if otherwise), but does not remove the event object from the message queue; a later call to `GetMessage()`, `WaitMessage()`, or `PeekMessage()` will return the same event object.

SetWatchdogTimeout(seconds As Integer) As Integer

Enables a watchdog timeout on the `roMessagePort` instance if passed a positive integer. The watchdog will crash and reboot the player if the script does not call `GetMessage()` or `WaitMessage()` after the specified number of seconds (the timeout is blocked while the script waits on the message port). Passing zero to this method disables the watchdog.

Some BrightScript operations (e.g. `roAssetRealizer.Realize()`) can take a long time to return. In these cases, it may be better to use a short watchdog timeout in general but switch to a longer timeout before calling such operations.

The watchdog timeout only applies to its associated `roMessagePort` instance, so enabling the watchdog on one `roMessagePort` instance, then calling `WaitMessage()` on another, may cause the watchdog to trigger unexpectedly. The watchdog will not trigger while waiting on the BrightScript debugger prompt.

DeferWatchdog(seconds As Integer)

Defers the watchdog timeout set by the `SetWatchdogTimeout()` method. Passing an integer to this method defers the timeout for the specified number of seconds.

DeferWatchdog()

Defers the watchdog timeout by the amount of seconds set in the `SetWatchdogTimeout()` method.

Note

Calls to either `DeferWatchdog()` method cannot cause the watchdog to trigger earlier than it already will. For example, calling `DeferWatchdog(100)` followed by `DeferWatchdog(10)` will still cause the watchdog to trigger after 100 seconds.

`ifEnum`

Reset()

Resets the position to the first element of enumeration.

Next() As Dynamic

Returns the typed value at the current position and increment position.

IsNext() As Boolean

Returns True if there is a next element.

IsEmpty() As Boolean

Returns True if there is not a next element.

roRegex

ON THIS PAGE

- [ifRegex](#)
 - [IsMatch\(string As String\) As Boolean](#)
 - [Match\(string As String\) As roArray](#)
 - [Replace\(string As String, subset As String\) As String](#)
 - [ReplaceAll\(string As String, subset As String\) As String](#)
 - [Split\(string As String\) As roList](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object allows the implementation of the regular-expression processing provided by the PCRE library. This object is created with a string to represent the matching-pattern and a string to indicate flags that modify the behavior of one or more matching operations:

```
CreateObject("roRegex", "[a-z]+", "i")
```

The match string (in the example above, `"[a-z]+"`, which matches all lowercase letters) can include most Perl compatible regular expressions found in the [PCRE documentation](#).

This object supports any combination of the following behavior flags (in the example above, `"i"`, which can be modified to match both uppercase and lowercase letters):

- `"i"`: Case-insensitive match mode.
- `"m"`: Multiline mode. The start-line ("`^`") and end-line ("`$`") constructs match immediately before or after any newline in the subject string. They also match at the absolute beginning or end of a string.
- `"s"`: Dot-all mode, which includes a newline in the `"."` regular expression. This modifier is equivalent to `"/s"` in Perl.
- `"x"`: Extended mode, which ignores whitespace characters except when escaped or inside a character class. This modifier is equivalent to `"/x"` in Perl.

ifRegex

IsMatch(string As String) As Boolean

Returns True if the string is consistent with the matching pattern.

Match(string As String) As roArray

Returns an *roArray* of matched substrings from the string. The entire match is returned in the form `array[0]`. This will be the only entry in the array if there are no parenthetical substrings. If the matching pattern contains parenthetical substrings, the relevant substrings will be returned as an array of length `n+1`, where `array[0]` is the entire match and each additional entry in the array is the match for the corresponding parenthetical expression.

Replace(string As String, subset As String) As String

Replaces the first occurrence of a matching pattern in the string with the subset and returns the result.

The subset may contain numbered back-references to parenthetical substrings.

Example

```
> r = CreateObject("roRegex", "(\\d+)\\s+(\\w+)", "")
> ? r.Replace("123 abc", "word:\\2 number:\\1")
> word:abc number:123
```

ReplaceAll(string As String, subset As String) As String

Replaces all occurrences of a matching pattern in the string with the subset and returns the result.

Example

```
> r = CreateObject("roRegex", "a", "i")
> ? r.ReplaceAll("Abracadabra", "x")
xbrxcxdxbrx
```

Split(string As String) As roList

Uses the matching pattern as a delimiter and splits the string on the delimiter boundaries. The function returns an *roList* of strings that were separated by the matching pattern in the original string.

roXMLElement

ON THIS PAGE

- [ifXMLElement](#)
 - [GetBody\(\) As Object](#)
 - [GetAttributes\(\) As Object](#)
 - [GetName\(\) As String](#)
 - [GetText\(\) As String](#)
 - [GetChildElements\(\) As Object](#)
 - [GetNamedElements\(a As String\) As Object](#)
 - [GetNamedElementsCi\(a As String\) As Object](#)
 - [SetBody\(a As Object\)](#)
 - [AddBodyElement\(\) As Object](#)
 - [AddElement\(a As String\) As Object](#)
 - [AddElementWithBody\(a As String, b As Object\) As Object](#)
 - [AddAttribute\(a As String, b As String\)](#)
 - [SetName\(a As String\)](#)
 - [Parse\(a As String\) As Boolean](#)
 - [GenXML\(a As Object\) As String](#)

- [Clear\(\) As Void](#)
- [GenXMLHdr\(a As String\) As String](#)
- [IsName\(a As String\) As Boolean](#)
- [HasAttribute\(a As String\) As Boolean](#)
- [ParseFile\(a As String\) As Boolean](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is used to contain an XML tree.

The *roXMLElement* object is created with no parameters:

```
CreateObject("roXMLElement")
```

The following examples illustrate how XML elements are parsed in BrightScript:

```
<tag1>This is example text</tag1>
```

- Name = tag1
- Attributes = Invalid
- Body = *roString* containing "This is example text"

```
<tag2 caveman="barney"/>
```

- Name = tag2
- Attributes = *roAssociativeArray* with one entry, {caveman, barney}
- Body = Invalid

If the tag contains other tags, the body will be of type *roXMLList*.

To generate XML content, create an *roXMLElement* and call the `SetBody()` and `SetName()` methods to build it—then call the `GenXML()` method to generate it.

Example

```
root.SetName("myroot")
root.AddAttribute("key1", "value1")
root.AddAttribute("key2", "value2")
ne=root.AddBodyElement()
ne.SetName("sub")
ne.SetBody("this is the sub1 text")
ne=root.AddBodyElement()
ne.SetName("subelement2")
ne.SetBody("more sub text")
ne.AddAttribute("k", "v")
ne=root.AddElement("subelement3")
```

```
ne.SetBody("more sub text 3")
root.AddElementWithBody("sub", "another sub (#4)")
PrintXML(root, 0)
print root.GenXML(false)
```

ifXMLElement

GetBody() As Object

GetAttributes() As Object

GetName() As String

GetText() As String

GetChildElements() As Object

GetNamedElements(a As String) As Object

GetNamedElementsCi(a As String) As Object

SetBody(a As Object)

Generates an *roXMLList* for the body if needed. The method then adds the passed item (which should be an *roXMLElementtag*).

AddBodyElement() As Object

AddElement(a As String) As Object

AddElementWithBody(a As String, b As Object) As Object

AddAttribute(a As String, b As String)

SetName(a As String)

Parse(a As String) As Boolean

Parses the XML content passed to it. In the event of failure, this method returns False. However, it also populates *roXMLElement* with whatever text could be successfully parsed. To avoid passing along erroneous strings, it is always best to have the script check the return value of `Parse()` before using them.

GenXML(a As Object) As String

Generates XML content. This method takes a single Boolean parameter, indicating whether or not the XML should have an `<?xml ...>` tag at the top.

Clear() As Void

GenXMLHdr(a As String) As String

IsName(a As String) As Boolean

HasAttribute(a As String) As Boolean

ParseFile(a As String) As Boolean

The following is an example subroutine to print out the contents of an *roXMLElement* tree:

```
PrintXML(root, 0)

Sub PrintXML(element As Object, depth As Integer)
  print tab(depth*3);"Name: ";element.GetName()
  if not element.GetAttributes().IsEmpty() then
    print tab(depth*3);"Attributes: ";
    for each a in element.GetAttributes()
      print a;"=";left(element.GetAttributes()[a], 20);
      if element.GetAttributes().IsNext() then print ", ";
    end for
    print
  end if
  if element.GetText()<>invalid then
    print tab(depth*3);"Contains Text: ";left(element.GetText(), 40)
  end if
  if element.GetChildElements()<>invalid
    print tab(depth*3);"Contains roXMLList:"
    for each e in element.GetChildElements()
      PrintXML(e, depth+1)
    end for
  end if
  print
end sub
```

roXMLList

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

ON THIS PAGE

- [ifXMLList](#)
 - [Simplify\(\) As Object](#)
 - [GetAttributes\(\) As Object](#)
 - [GetText\(\) As String](#)
 - [GetChildElements\(\) As Object](#)
 - [GetNamedElements\(a As String\) As Object](#)
 - [GetNamedElementsCi\(a As String\) As Object](#)
- [ifList](#)
 - [Count\(\) As Integer](#)

- `ResetIndex()` As Boolean
- `AddTail(obj As Object)` As Void
- `AddHead(obj As Object)` As Void
- `RemoveIndex()` As Object
- `GetIndex()` As Object
- `RemoveTail()` As Object
- `RemoveHead()` As Object
- `GetTail()` As Object
- `GetHead()` As Object
- `Clear()` As Void
- `ifEnum`
 - `Reset()`
 - `Next()` As Dynamic
 - `IsNext()` As Boolean
 - `IsEmpty()` As Boolean
- `ifArray`
 - `Peek()` As Dynamic
 - `Pop()` As Dynamic
 - `Push(a As Dynamic)`
 - `Shift()` As Dynamic
 - `Unshift(a As Dynamic)`
 - `Delete(a As Integer)` As Boolean
 - `Count()` As Integer
 - `Clear()` As Void
 - `Append(a As Object)` As Void
- `ifArrayGet`
 - `GetEntry(a As Integer)` As Dynamic
- `ifArraySet`
 - `SetEntry(a As Integer, b As Dynamic)` As Void

`ifXMLList`

***Simplify()* As Object**

Returns an *roXmlElement* if the list contains exactly one element. Otherwise, it will return itself.

***GetAttributes()* As Object**

***GetText()* As String**

***GetChildElements()* As Object**

***GetNamedElements(a As String)* As Object**

Returns a new XMLList that contains all *roXmlElements* that match the name of the passed element. This action is the same as using the dot operator on an *roXMLList*.

***GetNamedElementsCi(a As String)* As Object**

`ifList`

***Count()* As Integer**

Returns the number of elements in the list.

***ResetIndex()* As Boolean**

Resets the current index or position in the list to the head element.

***AddTail(obj As Object)* As Void**

Adds a typed value to the tail of the list.

AddHead(obj As Object) As Void

Adds a typed value to the head of the list.

RemoveIndex() As Object

Removes an entry from the list at the current index or position and increments the index or position in the list. It returns Invalid when the end of the list is reached.

GetIndex() As Object

Retrieves an entry from the list at the current index or position and increments the index or position in the list. It returns Invalid when the end of the list is reached.

RemoveTail() As Object

Removes the entry at the tail of the list.

RemoveHead() As Object

Removes the entry at the head of the list.

GetTail() As Object

Retrieves the entry at the tail of the list and keeps the entry in the list.

GetHead() As Object

Retrieves the entry at the head of the list and keeps the entry in the list.

Clear() As Void

Removes all elements from the list.

ifEnum

Reset()

Resets the position to the first element of enumeration.

Next() As Dynamic

Returns a typed value at the current position and increment position.

IsNext() As Boolean

Returns True if there is a next element.

IsEmpty() As Boolean

Returns True if there is not an exact statement.

ifArray

Peek() As Dynamic

Returns the last (highest index) array entry without removing it.

Pop() As Dynamic

Returns the last (highest index) entry and removes it from the array.

Push(a As Dynamic)

Adds a new highest index entry to the end of the array.

Shift() As Dynamic

Removes index zero from the array and shifts all other entries down by one unit.

Unshift(a As Dynamic)

Adds a new index zero to the array and shifts all other entries up by one unit.

Delete(a As Integer) As Boolean

Deletes the indicated array entry and shifts all above entries down by one unit.

Count() As Integer

Returns the index of the highest entry in the array plus one (i.e. the length of the array).

Clear() As Void

Deletes every entry in the array.

Append(a As Object) As Void

Appends one *roArray* to another. If the passed *roArray* contains entries that were never set to a value, they are not appended.

Note

The two appended objects must be of the same type.

`ifArrayGet`

GetEntry(a As Integer) As Dynamic

Returns an array entry of a given index. Entries start at zero. If an entry that has not been set is fetched, Invalid is returned.

`ifArraySet`

SetEntry(a As Integer, b As Dynamic) As Void

Sets an entry of a given index to the passed type value.

[Presentation and Widget Objects](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This section describes objects that relate directly to audio/video playback on BrightSign players.

- [roAudioConfiguration](#)
- [roAudioOutput](#)
- [roAudioPlayer](#)
- [roAudioPlayerMx](#)
- [roAudioEventMx](#)
- [roCanvasWidget](#)
- [roClockWidget](#)
- [roHdmiInputChanged, roHdmiOutputChanged](#)
- [roHtmlWidget](#)
- [roHtmlWidgetEvent](#)
- [roImageBuffer](#)
- [roImagePlayer](#)
- [roImageWidget](#)

- [roRectangle](#)
- [roStreamQueue](#)
- [roTextField](#)
- [roTextWidget](#)
- [roTextWidgetEvent](#)
- [roTouchScreen](#)
- [roTouchEvent](#), [roTouchCalibrationEvent](#)
- [roVideoEvent](#), [roAudioEvent](#)
- [roVideoInput](#)
- [roVideoMode](#)
- [roVideoPlayer](#)

roAudioConfiguration

ON THIS PAGE

- [ifAudioConfiguration](#)
 - [ConfigureAudio\(audio_routing As roAssociativeArray\) As Boolean](#)
 - [GetConfiguration\(\) As roAssociativeArray](#)
 - [GetClockStatus\(\) As roAssociativeArray](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object allows for mixing and leveling of audio streams before they are passed to audio outputs.

Object Creation: The *roAudioConfiguration* object is created with no parameters.

```
CreateObject("roAudioConfiguration")
```

ifAudioConfiguration

ConfigureAudio(audio_routing As roAssociativeArray) As Boolean

Configures the audio routing. This method will fail if called when audio resources are in use (i.e. there are active *roVideoPlayer* or *roAudioPlayer* instances). It returns `true` on success and `false` on failure. The passed associative array can have the following parameters:

- [string] `mode`: Sets the audio routing mode:
 - "dynamic": The default mode. Mixing audio streams with differing sampling rates will cause playback to fail; differing volume levels will not be normalized; and audio streams cannot be added to an output that currently has audio playing on it. Other parameters in the associative array are ignored.
 - "prerouted": This setting was implemented in firmware 7.0. You can add and remove audio streams on an output that currently has audio playing on it; additional audio routing behavior is determined with the `autolevel`, `pcmonly`, and `srcrate` parameters.
- [string] `autolevel`: Enables ("on") or disables ("off") volume leveling for audio outputs in the "prerouted" audio mode. When this setting is enabled, all PCM audio streams on a particular output will play at a similar volume.
- [string] `pcmonly`: Enables ("true") or disables ("false") compressed audio support in the "prerouted" audio mode.
- [int] `srcrate`: Sets the sample rate to which all PCM audio streams are converted in the the "prerouted" audio mode. This value can be either 44100 or 48000.

GetConfiguration() As roAssociativeArray

Returns audio configuration information as an associative array:

- [string] `mode`: The audio routing mode ("dynamic" or "prerouted")

- [int] `decoder_count`: The number of available decoders
- [int] `compressed_capable_count`: The number of decoders that can route compressed audio
- [Boolean] `autolevel`: A flag indicating whether the volume leveling setting is enabled
- [Boolean] `pcmonly`: A flag indicating whether the compressed audio support setting is enabled
- [int] `srate`: The sample rate to which all PCM audio streams are converted

Note

If the mode is "dynamic", the associative array will contain the `mode` and `decoder_count` parameter only.

GetClockStatus() As *roAssociativeArray*

Returns audio clock information about the I2C, Spdif, and Hdmi outputs.

roAudioOutput

ON THIS PAGE

- [ifAudioOutput](#)
 - [SetVolume\(a As Integer\) As Boolean](#)
 - [SetTone\(treble As Integer, bass As Integer\) As Boolean](#)
 - [SetMute\(a As Boolean\) As Boolean](#)
 - [GetOutput\(\) As String](#)
 - [SetAudioDelay\(delay_in_milliseconds As Integer\) As Boolean](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object allows individual control of audio outputs on the player.

Object Creation: The *roAudioOutput* object requires a single output parameter upon creation.

```
CreateObject("roAudioOutput", output As String)
```

The `output` parameter can take the following strings:

- "none"
- "hdmi"
- "usb"
- "spdif"
- "analog"
- "analog:N" (N specifies the port enumeration, which is useful for models with multiple analog-audio ports; you can also use "analog:1" to specify the analog output on a model with a single analog-audio port)

You can create any number of *roAudioOutput* objects. There can be multiple instances of this object that represent the same audio output, but in these cases one object will override another.

ifAudioOutput

SetVolume(a As Integer) As Boolean

Sets the volume of the specified output as a percentage represented by an integer between 0 and 100.

SetTone(treble As Integer, bass As Integer) As Boolean

Sets the treble and bass of the specified output. The treble and bass integers can range from -1000 to 1000, with 0 indicating no modification to the audio signal. Each increment represents a change of 0.01db.

SetMute(a As Boolean) As Boolean

Mutes the specified output if True. This method is set to False by default.

GetOutput() As String

Returns the string with which the *roAudioOutput* object was created.

SetAudioDelay(delay_in_milliseconds As Integer) As Boolean

Delays the audio for a specific audio output by lagging decoded samples before they reach that output. Delays are limited to 150ms or less. Currently, the system software only supports positive delays; therefore, if you need to set the audio ahead of the video, you will need to use *roVideoPlayer.SetVideoDelay()* instead.

The *SetVolume()* and *SetMute()* methods work in conjunction with the volume and mute functionality offered by *roAudioPlayer*. The *roAudioPlayer* volume settings affect the audio decoder volume. The audio stream is then sent to the assigned outputs, which have an additional level of volume control enabled by *roAudioOutput*.

The *roAudioOutput* object affects the absolute volume (as well as mute settings) for an audio output. If two players are streaming to the same output, both will be affected by any settings implemented through *roAudioOutput*.

Note

To control which audio outputs connect to audio player outputs generated by *roAudioOutput*, use the *SetPcmAudioOutputs()* and *SetCompressedAudioOutputs()* methods, which can be used for *roVideoPlayer* and *roAudioPlayer*. See the *roAudioPlayer* entry for further explanation of these methods.

roAudioPlayer

ON THIS PAGE

- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\) As Void](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [IfIdentity](#)
 - [GetIdentity\(\) As Integer](#)
- [ifMediaTransport](#)
- [ifAudioControl](#)
 - [SetPcmAudioOutputs\(outputs As roArray\) As Boolean](#)
 - [SetCompressedAudioOutputs\(outputs As roArray\) As Boolean](#)
 - [SetMultichannelAudioOutputs\(array As Object\) As Boolean](#)
 - [SetAudioOutput\(audio_output As Integer\) As Boolean](#)
 - [SetAudioMode\(audio_mode As Integer\) As Boolean](#)
 - [MapStereoOutput\(mapping As Integer\) As Boolean](#)
 - [MapDigitalOutput\(mapping As Integer\) As Boolean](#)
 - [SetVolume\(volume As Dynamic\) As Boolean](#)
 - [SetChannelVolumes\(channel_mask As Integer, volume As Integer\) As Boolean](#)
 - [SetPreferredAudio\(description As String\) As Boolean](#)
 - [SetUsbAudioPort\(a As Integer\) As Boolean](#)
 - [SetSpdifMute\(a As Boolean\) As Boolean](#)
 - [SetStereoMappingSpan\(a As Integer\) As Boolean](#)
 - [ConfigureAudioResources\(\) As Boolean](#)
 - [SetAudioStream\(stream_index As Integer\) As Boolean](#)
 - [SetAudioDelay\(delay_in_milliseconds As Integer\) As Boolean](#)

- [SetVideoDelay\(delay_in_milliseconds As Integer\) As Boolean](#)
- [StoreEncryptionKey\(a As String, b As String\) As Boolean](#)
- [StoreObfuscatedEncryptionKey\(a As String, b As String\) As Boolean](#)
- [SetAudioOutputAux\(audio_output As Integer\) As Boolean](#)
- [SetAudioModeAux\(audio_mode As Integer\) As Boolean](#)
- [MapStereoOutputAux\(mapping As Integer\) As Boolean](#)
- [SetVolumeAux\(volume As Integer\) As Boolean](#)
- [SetChannelVolumesAux\(channel_mask As Integer, volume As Integer\) As Boolean](#)
- [SetAudioStreamAux\(stream_index As Integer\) As Boolean](#)
- [Configuring Audio Outputs](#)
- [Playing Multiple Audio Files Simultaneously](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

An audio player is used to play back audio files using the generic *ifMediaTransport* interface. If the message port is set, the object will send events of the type *roAudioEvent*. All object calls are asynchronous. In other words, audio playback is handled in a different thread from the script. The script may continue to run while audio is playing.

`ifMessagePort`

SetPort(port As roMessagePort) As Void

Posts messages of type *roAudioEvent* to the attached message port.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via *SetUserData()*. It will return *Invalid* if no data has been set.

`IfIdentity`

GetIdentity() As Integer

Note

The *ifIdentity* interface has been deprecated. We recommend using the *ifUserData* interface instead.

`ifMediaTransport`

See *roVideoPlayer* for a description of *ifMediaTransport* methods.

`ifAudioControl`

SetPcmAudioOutputs(outputs As roArray) As Boolean

Specifies which audio connectors should output PCM audio. This method accepts one or more outputs in the form of an *roArray* of *roAudioOutput* instances.

SetCompressedAudioOutputs(outputs As roArray) As Boolean

Specifies which audio connectors should output compressed audio (e.g. Dolby AC3 encoded audio). This method accepts one or more outputs in the form of an *roArray* of *roAudioOutput* instances. When one or both of the above output methods are called, they will override the settings of the following *ifAudioControl* methods: *SetAudioOutput()*, *MapStereoOutput()*, *SetUsbAudioPort()*, *MapDigitalOutput()*.

SetMultichannelAudioOutputs(array As Object) As Boolean

SetAudioOutput(audio_output As Integer) As Boolean

Configures the audio output of the *roAudioPlayer* object. This method accepts the following values:

- 0: Analog audio
- 1: USB audio
- 2: Digital audio, stereo PCM
- 3: Digital audio, raw AC3
- 4: Onboard analog audio with HDMI mirroring raw AC3

SetAudioMode(audio_mode As Integer) As Boolean

Sets the audio mode of the *roAudioPlayer* object. This method accepts the following values:

- 0: AC3 Surround
- 1: AC3 mixed down to stereo
- 2: No audio
- 3: Left
- 4: Right

Note

Options 0 and 1 only apply to video files, while options 2, 3, and 4 apply to all audio sources.

MapStereoOutput(mapping As Integer) As Boolean

Determines which output to use when the output is set to the analog audio (i.e. `SetAudioOutput(0)`).

- 0: Stereo audio is mapped to onboard analog output.
- 1: Stereo audio is mapped to left output of the expansion module.
- 2: Stereo audio is mapped to middle output of the expansion module.
- 3: Stereo audio is mapped to right output of the expansion module.

MapDigitalOutput(mapping As Integer) As Boolean

Maps the digital audio output from the *roAudioPlayer* object. This method accepts the following values:

- 0: Onboard HDMI
- 1: SPDIF from expansion module

SetVolume(volume As Dynamic) As Boolean

Specifies the volume of the audio output as either a percentage or decibel amount. To use a percentage measurement, pass an integer value between 0 and 100. To use a decibel measurement, pass an *roAssociativeArray* containing a single `{db:<value As Float>}` parameter. The decibel measurement is an absolute value: passing 0 specifies no change to the audio output, and the effective range of measurements is from approximately -80 to 20 decibels.

The volume value is clipped prior to use (i.e. `SetVolume(101)` will set the volume to 100 and return `True`). The volume is the same for all mapped outputs and USB/SPDIF/analog.

Separate volume levels are stored for *roAudioPlayer* and *roVideoPlayer*.

SetChannelVolumes(channel_mask As Integer, volume As Integer) As Boolean

You can control the volume of individual audio channels. This volume command takes a hex channel mask, which determines the channels to apply the volume to, and a level, which is a percentage of the full scale. The volume control works according to audio channel rather than the output. The channel mask is a bit mask with the following bits for MP3 output:

- `&H01`: Left
- `&H02`: Right
- `&H03`: Both left and right

SetPreferredAudio(description As String) As Boolean

Selects which audio track to use if there are multiple audio tracks in the stream. See the **Preferred Streams** section on the *roVideoPlayer* page for more details.

SetUsbAudioPort(a As Integer) As Boolean

SetSpdifMute(a As Boolean) As Boolean

SetStereoMappingSpan(a As Integer) As Boolean

ConfigureAudioResources() As Boolean

SetAudioStream(stream_index As Integer) As Boolean

SetAudioDelay(delay_in_milliseconds As Integer) As Boolean

Adds a presentation time stamp (PTS) offset to the audio. This makes it possible to adjust for file multiplexing differences. Delays are limited to 150ms or less. Currently, the system software only supports positive delays; therefore, if you need to set the audio ahead of the video, you will need to use `SetVideoDelay()` instead.

SetVideoDelay(delay_in_milliseconds As Integer) As Boolean

Adds a presentation time stamp (PTS) offset to the video. This makes it possible to adjust for file multiplexing differences. Delays are limited to 150ms or less.

StoreEncryptionKey(a As String, b As String) As Boolean

StoreObfuscatedEncryptionKey(a As String, b As String) As Boolean

HD2000 Only:

SetAudioOutputAux(audio_output As Integer) As Boolean

SetAudioModeAux(audio_mode As Integer) As Boolean

MapStereoOutputAux(mapping As Integer) As Boolean***SetVolumeAux(volume As Integer) As Boolean***

SetChannelVolumesAux(channel_mask As Integer, volume As Integer) As Boolean

SetAudioStreamAux(stream_index As Integer) As Boolean

Configuring Audio Outputs

If a audio/video file is playing or has played, you need to call `Stop()` before changing the audio output.

The following script shows how to use the `SetPcmAudioOutputs()` and `SetCompressedAudioOutputs()` methods in conjunction with *roAudioOutput*. The audio/video player is configured to output decoded audio to the analog output or compressed audio to the HDMI and SPDIF outputs.

```
ao1=CreateObject("roAudioOutput", "Analog")
ao2=CreateObject("roAudioOutput", "HDMI")
ao3=CreateObject("roAudioOutput", "SPDIF")
```

```

a1=CreateObject("roAudioPlayer")
a1.SetPcmAudioOutputs(ao1)

ar = CreateObject("roArray", 2, true)
ar[0] = ao2
ar[1] = ao3
a1.SetCompressedAudioOutputs(ar)

```

Important

In most cases, rerouting audio outputs during audio/video playback will cause playback to stop. The system software will still be responsive, so you can use commands to exit playback during or after an audio output modification.

This code sets audio output to the rightmost expansion module audio port:

```

video = CreateObject("roVideoPlayer")
video.SetAudioOutput(0)
video.MapStereoOutput(3)

```

This code sets the volume level for individual channels:

```

audio = CreateObject("roAudioPlayer")
audio.SetChannelVolumes(&H01, 60)      'left channel to 60%
audio.SetChannelVolumes(&H02, 75)      'right channel to 75%
audio.SetChannelVolumes(&H03, 65)      'all channels to 65%

```

Playing Multiple Audio Files Simultaneously

Multiple MP3 files, as well as the audio track of a video file, can be played to any combination of the following:

- Analog outputs
- SPDIF / HDMI
- USB

Only a single file can be sent to an output at any given time. For example, two *roAudioPlayers* cannot simultaneously play to the SPDIF output. The second one to attempt a *PlayFile* will get an error. To free an output, the audio or video stream must be stopped (using the *ifMediaTransport* *Stop* or *StopClear* calls).

Note the following about multiple audio-file functionality:

- The onboard analog audio output and HDMI output are clocked by the same sample-rate clock. Therefore, if different content is being played out of each, the content must have the same sample rate.
- Currently, only a single set of USB speakers is supported.
- Each audio and video stream played consumes some of the finite CPU resources. The amount consumed depends on the bitrates of the streams. Testing is the only way to determine whether a given set of audio files can be played at the same time as a video. The maximum recommended usage is a 16Mbps video file with three simultaneous MP3 160kbps streams.

This code plays a video with audio over HDMI and an MP3 file to the onboard analog port.

```

video=CreateObject("roVideoPlayer")

video.SetAudioOutput(3)
video.PlayFile("video.mpg")

audio=CreateObject("roAudioPlayer")

```

```
audio.MapStereoOutput(0)
audio.PlayFile("audio.mp3")
```

roAudioPlayerMx

ON THIS PAGE

- [ifMediaTransport](#)
 - [PlayFile\(a As Object\) As Boolean](#)
 - [Stop\(\) As Boolean](#)
 - [Play\(\) As Boolean](#)
 - [Pause\(\) As Boolean](#)
 - [Resume\(\) As Boolean](#)
 - [SetLoopMode\(a As Boolean\) As Boolean](#)
 - [GetPlaybackStatus\(\) As Object](#)
- [ifAudioControl](#)
 - [MapStereoOutput\(a As Integer\) As Boolean](#)
 - [SetVolume\(a As Integer\) As Boolean](#)
 - [SetChannelVolumes\(a As Integer, b As Integer\) As Boolean](#)
 - [SetAudioOutput\(a As Integer\) As Boolean](#)
 - [SetAudioMode\(a As Integer\) As Boolean](#)
 - [SetAudioStream\(a As Integer\) As Boolean](#)
 - [SetUsbAudioPort\(a As Integer\) As Boolean](#)
 - [SetSpdifMute\(a As Boolean\) As Boolean](#)
 - [MapDigitalOutput\(a As Integer\) As Boolean](#)
 - [StoreEncryptionKey\(a As String, b As String\) As Boolean](#)
 - [StoreObfuscatedEncryptionKey\(a As String, b As String\) As Boolean](#)
 - [SetStereoMappingSpan\(a As Integer\) As Boolean](#)
 - [ConfigureAudioResources\(\) As Boolean](#)
 - [SetPcmAudioOutputs\(a As Object\) As Boolean](#)
 - [SetCompressedAudioOutputs\(a As Object\) As Boolean](#)
- [ifSetMessagePort](#)
 - [SetPort\(a As Object\)](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifIdentity](#)
 - [GetIdentity\(\) As Integer](#)
- [ifAudioControlMx](#)
 - [SetDecoderCount\(a As Integer\) As Boolean](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

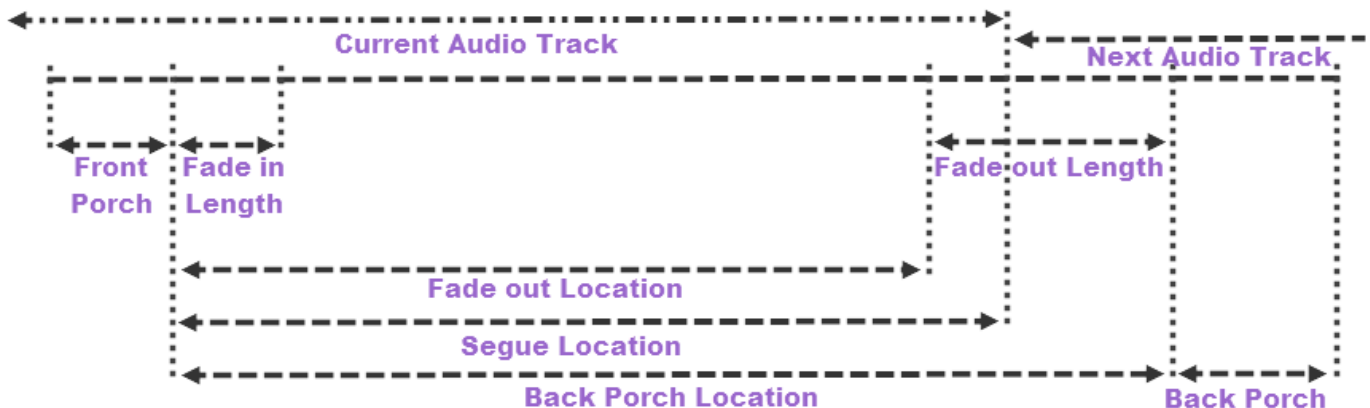
This object allows you to mix audio files, as well as HLS audio streams. Each `roAudioPlayerMx` object contains two internal audio players: The main audio playlist consists of queued audio tracks that play sequentially, while the audio overlay plays files on top of the main playlist. A fade will not occur if it is called while an overlay is playing, but the next audio track will start playing as expected.

Tracks are queued to `PlayFile` with their fade parameters specified in an [associative array](#). These are the parameters you can pass to `PlayFile`:

- `Filename`: The filename of the track
- `FrontPorch`: The length (in milliseconds) to skip from the start of the track. This value is 0 by default.

- **FadeOutLocation**: The location (in milliseconds) of the fade out relative to the value of the FrontPorch. If the FrontPorch value is 0 (which is the default setting), and the FadeOutLength value is non-zero, then the fade out is calculated back from the end of the file.
- **FadeOutLength**: The length of the fade out (in milliseconds). This value is 0 by default.
- **SegueLocation**: The location (in milliseconds) of the event that triggers the next audio file to play. This location is relative to the first audio file that is played. If the SegueLocation parameter is not included, the value defaults to the FadeOutLocation.
- **BackPorchLocation**: The location (in milliseconds) of the termination point for the audio track. This location is relative to the first audio file that is played. If the BackPorchLocation parameter is not included, the audio file plays to the end. The value is 0 by default, which disables the back porch.
- **TrackVolume**: The relative volume of the audio track, measured as a percentage. Specify the percentage using values between 0 and 100.
- **EventID**: The ID for an audio event.
- **EventTimeStamp**: The timestamp for the audio event. There can only be one event per audio file.
- **QueueNext**: The queuing of an audio track. Set the parameter value to 1 to queue an audio file to play after the current track.
- **Overlay**: The overlay specification of an audio track. Set the parameter value to 1 to fade down the main audio playlist while playing the audio track as an overlay. Overlays have additional parameters:
 - **AudioBedLevel**: The volume-level percentage of the main audio playlist while the overlay is playing. Specify the percentage using values between 0 and 100.
 - **AudioBedFadeOutLength**: The fade-out length of the main audio playlist.
 - **AudioBedFadeInLength**: The fade-in length for the length of the underlying audio track once the segue is triggered.
- **FadeCurrentPlayNext**: A fade command. Set the parameter value to 1 to fade out the current main audio playlist track and fade in the designated audio file.
- **CrossfadeCurrentPlayNext**: A crossfade command. Set the parameter value to 1 to force an immediate crossfade between the current main audio playlist track and the designated audio file.
- **UserString**: A string that can be set to a unique value for each *roAudioPlayerMx* instance. This string is returned with every event generated by the instance. Since all current platforms can support multiple *roAudioPlayerMx* instances running at the same time, the UserString allows the script to distinguish between event returns.

The following diagram illustrates how some of these timing parameters work together:



The following script illustrates a simple crossfade between audio tracks:

```

a = CreateObject("roAudioPlayerMx")

track1 = CreateObject("roAssociativeArray")
track1["Filename"] = "file1.mp3"
track1["FadeInLength"] = 4000
track1["FadeOutLength"] = 4000
track1["QueueNext"] = 1

track2 = CreateObject("roAssociativeArray")
track2["Filename"] = "file2.mp3"
track2["FadeInLength"] = 4000
track2["FadeOutLength"] = 4000
track2["QueueNext"] = 1

```

```
a.PlayFile(track1)
a.PlayFile(track2)
```

ifMediaTransport

PlayFile(a As Object) As Boolean

Stop() As Boolean

Play() As Boolean

Pause() As Boolean

Resume() As Boolean

SetLoopMode(a As Boolean) As Boolean

GetPlaybackStatus() As Object

ifAudioControl

MapStereoOutput(a As Integer) As Boolean

SetVolume(a As Integer) As Boolean

SetChannelVolumes(a As Integer, b As Integer) As Boolean

SetAudioOutput(a As Integer) As Boolean

SetAudioMode(a As Integer) As Boolean

SetAudioStream(a As Integer) As Boolean

SetUsbAudioPort(a As Integer) As Boolean

SetSpdifMute(a As Boolean) As Boolean

MapDigitalOutput(a As Integer) As Boolean

StoreEncryptionKey(a As String, b As String) As Boolean

StoreObfuscatedEncryptionKey(a As String, b As String) As Boolean

SetStereoMappingSpan(a As Integer) As Boolean

ConfigureAudioResources() As Boolean

SetPcmAudioOutputs(a As Object) As Boolean

SetCompressedAudioOutputs(a As Object) As Boolean

ifSetMessagePort

SetPort(a As Object)

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifIdentity`

GetIdentity() As Integer

Note

The `ifIdentity` interface has been deprecated. We recommend using the `ifUserData` interface instead.

`ifAudioControlMx`

SetDecoderCount(a As Integer) As Boolean

roAudioEventMx

ON THIS PAGE

- [ifInt](#)
 - [GetInt\(\) As Integer](#)
 - [SetInt\(value As Integer\) As Void](#)
- [ifAudioUserData](#)
 - [SetUserData\(user_data As String\)](#)
 - [GetUserData\(\) As String](#)
- [ifSourceIdentity](#)
 - [GetSourceIdentity\(\) As Integer](#)
 - [SetSourceIdentity\(\) As Integer](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

The `roAudioPlayerMx` object can generate `roAudioEventMx` messages with the following values:

- 8 `EVENT_MEDIAENDED`
- 14 `EVENT_OVERLAY_MEDIAENDED`
- 16 `EVENT_MEDIAERROR`
- 17 `EVENT_OVERLAY_MEDIAERROR`

"Media ended" events are sent when a track finishes and there are no more queued tracks for the player, while "Media error" events are sent when a queued file is not found (e.g. when it does not exist).

`ifInt`

GetInt() As Integer

Returns the event ID.

SetInt(value As Integer) As Void

Sets the value of the event.

ifAudioUserData

SetUserData(user_data As String)

Sets the user data.

GetUserData() As String

Returns the user data that has previously been set via `SetUserData()` (either on the source or event object). It will return `Invalid` if no data has been set.

ifSourceIdentity

GetSourceIdentity() As Integer

SetSourceIdentity() As Integer

Note

The `ifSourceIdentity` interface has been deprecated. We recommend using the `ifAudioUserData` interface instead.

roCanvasWidget

ON THIS PAGE

- [ifCanvasWidget](#)
 - [Hide\(\) As Boolean](#)
 - [Show\(\) As Boolean](#)
 - [SetRectangle\(r As roRectangle\) As Boolean](#)
 - [SetLayer\(content As Object, z-level As Integer\) As Boolean](#)
 - [ClearLayer\(z-level As Integer\) As Boolean](#)
 - [Clear\(\) As Boolean](#)
 - [EnableAutoRedraw\(enable As Boolean\) As Boolean](#)
- [Object Content](#)
 - [Background color](#)
 - [Text](#)
 - [Image](#)
 - [QR Codes](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object composites background color, text, and images into a single rectangle, allowing you to layer images on a z-axis.

Object Creation: Like other widgets, `roCanvasWidget` is created with an `roRectangle` to set its size and position on the screen.

```
CreateObject ("roCanvasWidget", r As roRectangle) As Object
```

ifCanvasWidget

Hide() As Boolean

Hides the widget.

Show() As Boolean

Shows the widget.

SetRectangle(r As roRectangle) As Boolean

Changes the size and positioning of the widget rectangle using the passed *roRectangle* object.

SetLayer(content As Object, z-level As Integer) As Boolean

Sets the contents of a layer within the widget. The lowest z-level is drawn first, and the highest z-level is drawn last. The object content is described below.

ClearLayer(z-level As Integer) As Boolean

Clears the specified layer.

Clear() As Boolean

Clears all of the layers.

EnableAutoRedraw(enable As Boolean) As Boolean

Enables or disables the automatic redrawing of the widget.

- When this function is enabled, each call to *SetLayer*, *ClearLayer*, or *Clear* results in a redraw. If you need to change multiple layers, then you should disable auto redraw while calling the *SetLayer* function.
- *SetLayer* enables or disables redrawing of the widget when layer content is changed. When auto-redraw is enabled, each call to *SetLayer*, *ClearLayer*, or *Clear* results in a redraw. To batch multiple updates together, you should first suspend drawing using *EnableAutoRedraw(false)*, then make the changes to the content, and finally re-enable drawing using *EnableAutoRedraw(true)*. The redraw happens in a separate thread, so *EnableAutoRedraw* returns almost immediately.

Object Content

The content specified in each layer can consist of one or more objects. Each object is defined by an *roAssociativeArray*. If there is more than one object, then each is placed into an *roArray* prior to passing to the *SetLayer()* method. Currently, there are four object types.

Background color

- *color*: The *#[aa]rrggbb* hex value of the background color
- *targetRect*: A target rectangle, which is another *roAssociativeArray* consisting of *x*, *y*, *w*, and *h* values. These values are relative to the top left corner of the widget.

Text

- *text*: A string of text to display
- *targetRect*: The rectangle in which the text is displayed
- *textAttrs*: An *roAssociativeArray* containing attributes to be applied to the text. The attributes can be any of the following:
 - *font*: A string indicating whether the text should be displayed as "small"/"medium"/"large"/"huge"
 - *fontSize*: A point size that is used directly when creating the font. If the value is set to 0, then the font automatically resizes to fit the *targetRect*.
 - *fontfile*: The filename for a non-system font to use
 - *hAlign*: A string indicating the "left"/"center"/"right" alignment of the text on a line
 - *vAlign*: A string indicating the "top"/"center"/"bottom" alignment of the text perpendicular to the line
 - *rotation*: A string indicating the "0"/"90"/"180"/"270" degree rotation of the text
 - *color*: The *#[aa]rrggbb* hex value of the text

Image

- *filename*: The filename of the image
- *encryptionAlgorithm*: The file-encryption algorithm. Currently the options are "AesCtr" and "AesCtrHmac".
- *encryptionKey*: The key to decrypt the image file. This is a byte array consisting of 128 bits of key, followed by 128 bits of IV.

Note

See the [Image Decryption](#) section in the *roImagePlayer* entry for details on displaying encrypted images.

- `targetRect`: The rectangle in which the image is displayed. The image will be automatically resized to fit into the target area.
- `sourceRect`: The source rectangle to clip from a source image
- `compositionMode`: Enter either source or source_over. The latter alpha blends with underlying objects. The former replaces the underlying values completely.
- `imgAttrs`: An *roAssociativeArray* containing attributes to be applied to the image:
 - `rotation`: A string indicating the "0"/"90"/"180"/"270" degree rotation of the image

QR Codes

QR (quick response) codes appear as squares of black dots on a white background. They are used to encode URLs, email addresses, etc, and they can be scanned using readily available software for smart phones. Although the codes usually appear as black on white, you can, in theory, use any two contrasting colors.

- `targetRect`: The rectangle in which the QR code is displayed. Regardless of the aspect ratio of this rectangle, the QR code itself will always be squared with the background color that fills the gaps.
- `QrCode` (simple form): Contains the string to encode into the QR code.
- `QrCode` (complex form): Contains an array of parameters for the QR code. The parameters can be any of the following:
 - `color`: The foreground color in the QR code (the default is black)
 - `backgroundColor`: The background color in the QR code (the default is white)
 - `rotation`: A string indicating the "0"/"90"/"180"/"270" degree rotation of the code. The code will scan regardless of rotation.
 - `qrText`: Contains the text to encode into the QR code.

This code contains most of the *roCanvasWidget* features outlined above:

```
rect=CreateObject("roRectangle", 0, 0, 1920, 1080)
cw=CreateObject("roCanvasWidget", rect)

aa=CreateObject("roAssociativeArray")
aa["text"] = "Primal Scream"
aa["targetRect"] = { x: 280, y: 180, w: 500, h: 30 }
aa["textAttrs"] = { Color:"#AAAAAA", font:"Medium", HAlign:"Left", VAlign:"Top"}

aal=CreateObject("roAssociativeArray")
aal["text"] = "Movin' on up, followed by something else, followed by something else, followed
by something else, followed by something else"
aal["targetRect"] = { x: 282, y: 215, w: 80, h: 500 }
aal["textAttrs"] = { Color:"#ffffff", font:"Large", fontfile:"usb1:/GiddyupStd.otf", HAlign:"
Left", VAlign:"Top", rotation:"90"}

array=CreateObject("roArray", 10, false)
array.Push({ color: "5c5d5f" })
array.Push({ filename: "transparent-balls.png" })
array.Push(aa)

aa2=CreateObject("roAssociativeArray")
aa2["filename"] = "transparent-balls.png"
aa2["CompositionMode"] = "source_over"
aa2["targetRect"] = { x: 400, y: 200, w: 200, h: 200 }

aa3=CreateObject("roAssociativeArray")
aa3["QrCode"] = "www.brightsign.biz"
aa3["targetRect"] = { x: 100, y: 100, w: 400, h: 400 }

aa4=CreateObject("roAssociativeArray")
aa4["QrCode"] = { qrText:"www.brightsign.biz", rotation:"90" }
aa4["targetRect"] = { x: 1200, y: 100, w: 400, h: 600 }
```

```

aa5=CreateObject("roAssociativeArray")
aa5["QRcode"] = { color:"#964969", backgroundColor:"#FFFF77", qrText:"www.brightsign.biz",
rotation:"180" }
aa5["targetRect"] = { x: 100, y: 600, w: 400, h: 400 }

cw.Show()
cw.EnableAutoRedraw(0)
cw.SetLayer(array, 0)
cw.SetLayer(aa1, 1)
cw.SetLayer(aa1, 2)
cw.SetLayer(aa3, 3)
cw.SetLayer(aa4, 4)
cw.SetLayer(aa5, 5)
cw.EnableAutoRedraw(1)

cw.ClearLayer(0)

```

roClockWidget

ON THIS PAGE

- [Control Characters](#)
- [ifWidget](#)
 - [SetForegroundColor\(color As Integer\) As Boolean](#)
 - [SetBackgroundColor\(color As Integer\) As Boolean](#)
 - [SetFont\(font_filename As String\) As Boolean](#)
 - [SetBackgroundBitmap\(bitmap_filename As String, stretch As Boolean\) As Boolean](#)
 - [SetBackgroundBitmap\(parameters As roAssociativeArray, stretch As Boolean\) As Boolean](#)
 - [SetSafeTextRegion\(region As roRectangle\) As Boolean](#)
 - [Show\(\) As Boolean](#)
 - [Hide\(\) As Boolean](#)
 - [GetFailureReason\(\) As String](#)
 - [SetRectangle\(r As roRectangle\) As Boolean](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object places a clock on the screen. It has construction arguments only.

Object Creation: The *roClockWidget* object is created with several parameters.

```

CreateObject("roClockWidget", rect As roRectangle, res As roResourceManager, display_type As
Integer)

```

- **rect**: The rectangle in which the clock is displayed. The widget picks a font based on the size of the rectangle.
- **res**: A *resources.txt* file that allows localization via the *roResourceManager* object (see below for further details).
- **display_type**: Use 0 for date only, and 1 for clock only. To show both on the screen, you need to create two widgets.

Example

```

rect=CreateObject("roRectangle", 0, 0, 300, 60)
res=CreateObject("roResourceManager", "resources.txt")

```

```
c=CreateObject("roClockWidget", rect, res, 1)
c.Show()
```

The resource manager is passed into the widget, which uses the following resources within the *resources.txt* file to display the time and date correctly. Here are the "eng" entries:

```
[CLOCK_DATE_FORMAT]
eng "%A, %B %e, %Y"
[CLOCK_TIME_FORMAT]
eng "%l:%M"
[CLOCK_TIME_AM]
eng "AM"
[CLOCK_TIME_PM]
eng "PM"
[CLOCK_DATE_SHORT_MONTH]
eng "Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec"
[CLOCK_DATE_LONG_MONTH]
eng
"January|February|March|April|May|June|July|August|September|October|November|December"
[CLOCK_DATE_SHORT_DAY]
eng "Sun|Mon|Tue|Wed|Thu|Fri|Sat"
[CLOCK_DATE_LONG_DAY]
eng "Sunday|Monday|Tuesday|Wednesday|Thursday|Friday|Saturday"
```

Control Characters

The following are the control characters for the date/time format strings:

```
// Date format
//
// %a Abbreviated weekday name
// %A Long weekday name
// %b Abbreviated month name
// %B Full month name
// %d Day of the month as decimal 01 to 31
// %e Like %d, the day of the month as a decimal number, but without leading zero
// %m Month name as a decimal 01 to 12
// %n Like %m, the month as a decimal number, but without leading zero
// %y Two digit year
// %Y Four digit year

// Time format
//
// %H The hour using 24-hour clock (00 to 23)
// %I The hour using 12-hour clock (01 to 12)
// %k The hour using 24-hour clock (0 to 23); single digits are preceded by a blank.
// %l The hour using 12-hour clock (1 to 12); single digits are preceded by a blank.
// %M Minutes (00 to 59)
// %S Seconds (00 to 59)
```

ifWidget

SetForegroundColor(color As Integer) As Boolean

Sets the foreground color in ARGB format. Hex color values should be converted to integers before being passed to this method (e.g. the value &hFFFFFF should be passed as 4294967295). You can use the `HexToInteger()` method (available in the [core library extension](#)) to convert a hex string to an integer.

SetBackgroundColor(color As Integer) As Boolean

Sets the background color in ARGB format. Hex color values should be converted to integers before being passed to this method (e.g. the value `&hFFFFFF` should be passed as `4294967295`). You can use the `HexToInteger()` method (available in the [core library extension](#)) to convert a hex string to an integer.

Note

The top 8 bits of the `SetForegroundColor()` and `SetBGackgroundColor()` values are "alpha" parameters. Zero is equivalent to fully transparent and 255 to fully non-transparent. This feature allows for effects similar to subtitles. For example, you can create a semi-transparent black box containing text over video.

SetFont(font_filename As String) As Boolean

Sets the `font_filename` using a TrueType font (for example, `SD:/Arial.ttf`).

SetBackgroundBitmap(bitmap_filename As String, stretch As Boolean) As Boolean

Sets the background bitmap image. If `stretch` is `True`, then the image is stretched to the size of the window.

SetBackgroundBitmap(parameters As roAssociativeArray, stretch As Boolean) As Boolean

Sets the background bitmap image. If `stretch` is `True`, then the image is stretched to the size of the window. The associative array can contain the following parameters:

- `Filename`: The name of the image file
- `EncryptionAlgorithm`: The file-encryption algorithm. Currently the options are "AesCtr" and "AesCtrHmac".
- `EncryptionKey`: The key to decrypt the image file. This is a byte array consisting of 128 bits of key, followed by 128 bits of IV.

Note

See the [Image Decryption](#) section in the `roImagePlayer` entry for details on displaying encrypted images.

SetSafeTextRegion(region As roRectangle) As Boolean

Specifies the rectangle within the widget where the text can be drawn safely.

Show() As Boolean

Displays the widget. After creation, the widget is hidden until `Show()` is called.

Hide() As Boolean

Hides the widget.

GetFailureReason() As String

Yields additional useful information if a function return indicates an error.

SetRectangle(r As roRectangle) As Boolean

Changes the size and positioning of the widget rectangle using the passed `roRectangle` object.

roHdmiInputChanged, roHdmiOutputChanged

ON THIS PAGE

- [ifInt](#)
 - [GetInt\(\) As Integer](#)
 - [SetInt\(value As Integer\) As Void](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

The *roVideoMode* object generates an *roHdmiInputChanged* or *roHdmiOutputChanged* event object whenever the hotplug status of the HDMI input or output changes.

At least one *roHdmiOutputChanged* event object will always be generated for a hotplug event, even for very quick disconnect/connect hotplug events. In most cases, a disconnect/connect hotplug event will generate two event objects.

ifInt

***GetInt()* As Integer**

Returns the event ID.

SetInt(value As Integer) As Void

Sets the value of the event.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

***GetUserData()* As Object**

Returns the user data that has previously been set via `SetUserData()`. It will return Invalid if no data has been set.

roHtmlWidget

ON THIS PAGE

- [Object Creation](#)
- [Initialization Parameters](#)
 - [\[boolean\] nodejs_enabled](#)
 - [\[boolean\] focus_enabled](#)
 - [\[boolean\] mouse_enabled](#)
 - [\[boolean\] scrollbar_enabled](#)
 - [\[boolean\] force_gpu_rasterization_enabled](#)
 - [\[boolean\] canvas_2d_acceleration_enabled](#)
 - [\[boolean\] javascript_enabled](#)
 - [\[boolean\] brightsign_js_objects_enabled](#)
 - [\[string\] transform](#)
 - [\[string\] user_agent](#)
 - [\[string\] url](#)
 - [\[string\] user_stylesheet](#)
 - [\[string\] hwz_default](#)
 - [\[string\] storage_path](#)
 - [\[string or double\] storage_quota](#)
 - [\[roMessagePort\] port](#)
 - [\[roArray\] fonts](#)
 - [\[roArray\] pcm_audio_outputs](#)
 - [\[roArray\] compressed_audio_outputs](#)
 - [\[roArray\] multi_channel_audio_outputs](#)
 - [\[roAssociativeArray\] inspector_server](#)
 - [\[roAssociativeArray\] security_params](#)
 - [\[roAssociativeArray\] javascript_injection](#)
 - [\[roArray\] assets](#)

- `ifHtmlWidget`
 - `GetFailureReason()` As String
 - `Hide()` As Boolean
 - `Show()` As Boolean
 - `SetRectangle(r As roRectangle)` As Boolean
 - `SetURL(URL As String)` As Boolean
 - `MapFilesFromAssetPool(asset_pool As roAssetPool, asset_collection As roAssetCollection, pool_prefix As String, uri_prefix As String)` As Boolean
 - `SetZoomLevel(scale_factor As Float)` As Boolean
 - `EnableSecurity(enable As Dynamic)` As Boolean
 - `EnableMouseEvents(enable As Boolean)` As Boolean
 - `SetPortrait(portrait_mode As Boolean)` As Boolean
 - `SetTransform(transform As String)` As Boolean
 - `SetAlpha(alpha As Integer)` As Boolean
 - `EnableScrollbars(scrollbars As Boolean)` As Boolean
 - `AddFont(filename As String)` As Boolean
 - `SetPcmAudioOutputs(outputs As roArray)` As Boolean
 - `SetCompressedAudioOutputs(outputs As roArray)` As Boolean
 - `SetMultichannelAudioOutputs(outputs As roArray)` As Boolean
 - `SetHWZDefault(default As String)` As Void
 - `SetVideoPlayerDefaults(defaults As roAssociativeArray)` As Boolean
 - `ForceGpuRasterization(enable As Boolean)` As Boolean
 - `EnableCanvas2dAcceleration(enable As Boolean)` As Boolean
 - `SetUserStylesheet(URI As String)` As Boolean
 - `SetAppCacheDir(file_path As String)` As Boolean
 - `SetAppCacheSize(maximum As Integer)` As Boolean
 - `FlushCachedResources()` As Boolean
 - `SetLocalStorageDir(file_path As String)` As Boolean
 - `SetLocalStorageQuota(maximum As Dynamic)` As Boolean
 - `SetWebDatabaseDir(file_path As String)` As Boolean
 - `SetWebDatabaseQuota(maximum As Dynamic)` As Boolean
 - `EnableJavaScript(enable As Boolean)` As Boolean
 - `AllowJavaScriptURLs(url_collection As roAssociativeArray)`
 - `PostJSMMessage(data As roAssociativeArray)` As Boolean
 - `InjectJavaScript(code As String)` As Boolean
 - `StartInspectorServer(port As Integer)` As Boolean
 - `SetUserAgent(user_agent As String)` As Boolean
 - `GetUserAgent()` As String
 - `SetUserAgentSuffix(suffix As String)` As Boolean
 - `SetProxy(proxy As String)` As Boolean
 - `SetProxyBypass(hostnames As String)` As Boolean
- `ifMessagePort`
 - `SetPort(port As roMessagePort)`
- `ifUserData`
 - `SetUserData(user_data As Object)`
 - `GetUserData()` As Object

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object embeds the Chromium HTML rendering engine, which can be rendered at full screen or as a widget. You can display multiple *roHtmlWidget* instances at the same time.

Tip

Use the *roKeyStore* object to provide client certificates for websites. Use the *roVirtualMemory* object to set up a virtual memory repository for Chromium.

Object Creation

The *roHtmlWidget* object is initialized with an *roRectangle* object, which specifies the size and positioning of the widget on the screen, and an optional associative array, which defines properties for the widget.

```
CreateObject("roHtmlWidget", rect As roRectangle, properties As roAssociativeArray)
```

The properties of an *roHtmlWidget* instance can be set with an associative array at initialization or with equivalent methods after initialization. Because many *roHtmlWidget* properties cannot be changed without reloading the page (and can produce unpredictable results while the page is running), we recommend setting properties at initialization when possible, rather than using the equivalent methods.

Important

Defining initialization properties for an *roHtmlWidget* instance disables all property methods for that instance (e.g. `SetTransform()`, `AddFont()`, `SetUserStylesheet()`). Other methods that do not affect properties (e.g. `Show()`, `FlushCachedResources()`) can still be used.

Initialization Parameters

The associative array passed during initialization can have the following parameters:

[boolean] nodejs_enabled

Enables **Node.js** on the widget. This value is `false` by default.

[boolean] focus_enabled

Enables focus for mouse/touchscreen events. This value is `true` by default.

[boolean] mouse_enabled

Enables mouse/touchscreen events. This value is `false` by default.

[boolean] scrollbar_enabled

Enables automatic scrollbars for content that does not fit into the viewport. This value is `false` by default.

[boolean] force_gpu_rasterization_enabled

Enables GPU rasterization for HTML graphics. By default, the decision to use GPU rasterization is based on **internal Chromium logic**. Setting this value to `true/false` will enable/disable it for all content.

[boolean] canvas_2d_acceleration_enabled

Enables 2D canvas acceleration. This will improve the framerate of most HTML pages that use 2D animations, but can cause out-of-memory issues with pages that use a large number of off-screen canvas surfaces. This value is `true` by default in firmware versions 7.0.x and later and `false` by default in firmware versions 6.2.x and earlier.

[boolean] javascript_enabled

Enables JavaScript on the widget. This value is `true` by default.

[boolean] brightsign_js_objects_enabled

Enables **BrightScript-JavaScript objects**. This value is `false` by default.

[string] transform

Sets the screen orientation of content in the widget (note that the coordinates and dimensions of the *roRectangle* containing the widget are not affected by rotation). The following values are accepted:

- "identity": There is no transform (i.e. the widget content is oriented as landscape). This is the default setting.
- "rot90": The widget content is rotated to portrait at 90 degrees (clockwise).
- "rot270": The widget content is rotated to portrait at 270 degrees (counter-clockwise).

[string] user_agent

Modifies the default user-agent string for the *roHtmlWidget* instance.

[string] url

The URL to use for display. See the `SetUrl()` entry below for more information on using URIs to access files from local storage.

[string] user_stylesheet

Applies the specified user stylesheet to pages in the widget. The parameter is a URI specifying any `file:` resource in the storage. The stylesheet can also be specified as inline data.

[string] hwz_default

Specifies the default HWZ behavior. See the `SetHWZDefault()` entry below for more information.

[string] storage_path

Creates a "Local Storage" subfolder in the specified directory. This folder is used by local storage applications such as the JavaScript *storage* class.

[string or double] storage_quota

Sets the total size (in bytes) allotted to all local storage applications (including IndexedDB). A number literal can only represent byte numbers up to 2GB; use a string to specify larger amounts. The default total size is 10GB (or 1GB less than the total size of the storage device, whichever is less).

[roMessagePort] port

Configures the message port to which the *roHtmlWidget* instance will send events. When using initialization parameters, the `port` parameter should be used instead of the `SetPort()` method to ensure the script can catch `load-started`, `load-finished`, and `load-error` events.

[roArray] fonts

Specifies a list of font files that can be accessed by the webpage. Font files are specified as an array of string filenames. Supported font types include TrueType Font files (*.ttf*) and Web Open Font files (*.woff*, *.woff2*).

[roArray] pcm_audio_outputs

Configures the PCM audio output for the HTML widget. Outputs are specified as an array of *roAudioOutput* instances.

[roArray] compressed_audio_outputs

Configures compressed audio output (e.g. Dolby AC3 encoded audio) for the HTML widget. Outputs are specified as an array of *roAudioOutput* instances.

[roArray] multi_channel_audio_outputs

Configures multi-channel audio output for the HTML widget. Outputs are specified as an array of *roAudioOutput* instances.

[roAssociativeArray] inspector_server

Configures the **Chromium Inspector** for the widget.

- `[string] ip_addr`: The Inspector IP address. This value is useful if the player is assigned more than one IP address (i.e. there are multiple network interfaces) and you wish to limit the Inspector server to one. The default value is "0.0.0.0", which allows the Inspector to accept connections using either IP address.
- `[int] port`: The port for the Inspector server.

[roAssociativeArray] security_params

Enables or disables Chromium security checks for cross-origin requests, local video playback from HTTP, etc.

- `[Boolean] websecurity`: Enables Chromium security checks.

- [Boolean] `camera_enabled`: Enables webpage access to USB cameras connected to the player (access is disabled by default). This allows support for WebRTC applications.
- [Boolean] `insecure_https_enabled`: Instructs the widget to ignore security errors when connecting to insecure HTTPS hosts (insecure HTTPS is disabled by default). Enabling this feature makes the player insecure; it is not suitable for production environments and should only be used for testing.

Note

The `camera_enabled` parameter is currently supported on the XTx44, XTx43, 4Kx42, XDx34, XDx33, HDx23, and LS423 models.

[roAssociativeArray] javascript_injection

Specifies JavaScript code to inject at different initialization points (JavaScript code can also be injected during runtime using the `InjectJavaScript()` method). The associative array can contain three parameters (described below). Each parameter value is an array of associative arrays, each containing a single key/value pair. The array must contain a `source` key. The `source` value is a string that can contain any of the following: pure JavaScript code, a path to a JavaScript file, or a base64-encoded string (i.e. beginning with `data:text/javascript; charset=utf-8;base64,`). Multiple `source` keys can be included, but the load order will be unpredictable. The array can also contain the optional `world` key, which can be assigned one of the following values: "application", "user", or "main" (see [this page](#) for more details); if the `world` parameter is not included in the array, "application" is selected by default.

- `document_creation`: The script will run as soon as the document is created. This behavior is not suitable for any DOM operation.
- `document_ready`: The script will run as soon as the DOM is ready. This behavior is equivalent to the `DOMContentLoaded` event firing in JavaScript.
- `deferred`: The script will run when the page load finishes. The DOM cannot be changed at this point.

```
config = {
  javascript_injection: {
    document_creation: [{source: "0.js" }],
    document_ready: [{source: "1a.js" }, {source: "1b.js" }],
    deferred: [{source: "2.js" }]
  },
  url: "..."}
}
```

[roArray] assets

Allows the `roHtmlWidget` instance to [access one or more asset pools](#). If a file exists in multiple specified asset pools, the asset pool with the lowest index in the array has precedence. Each array entry is an associative array containing information about an asset pool:

- [roAssetPool] `pool`: An asset pool containing files
- [roAssetCollection] `collection`: A manifest identifying the files in the pool
- [string] `uri_prefix`: The URI prefix of the webpage resources to retrieve from the pool
- [string] `pool_prefix`: The pool prefix that will replace the URI prefix when looking up the resource in the pool

`ifHtmlWidget`

GetFailureReason() As String

Gives more information when a member function returns `false`.

Hide() As Boolean

Hides the widget.

Show() As Boolean

Shows the widget.

SetRectangle(r As roRectangle) As Boolean

Changes the size and positioning of the widget rectangle using the passed `roRectangle` object.

SetURL(URL As String) As Boolean

Displays content from the specified URL. When using this method to retrieve content from local storage, specify the file location as follows: "file : /<drive> : /<directory> /<filename> ". For example, an *index.html* file in the "Content" folder on the SD card can be selected with the string "file : /SD : /Content /index.html ". You can also omit the **drive specification** to select the currently active drive (i.e. whichever drive the current autorun as loaded from).

MapFilesFromAssetPool(asset_pool As roAssetPool, asset_collection As roAssetCollection, pool_prefix As String, uri_prefix As String) As Boolean

Sets the mapping between the URL space and the pool files. HTML content that has been deployed via BrightAuthor will typically reside in the pool and have encrypted SHA1-based filenames. A mapping mechanism is required to allow any relative URIs contained in the HTML content to continue working and to locate the appropriate resources in their respective pool locations.

You can use this method to bind part of the resource URI space onto pool locations. This method accepts the following arguments: an *roAssetPool* object containing assets, an *roAssetCollection* object identifying the assets, and two semi-arbitrary strings (URI_PREFIX and POOL_PREFIX).

Any URI in the form "file : / [URI_PREFIX] [RESOURCE_ID] " will be rewritten into the form " [POOL_PREFIX] [RESOURCE_ID] ". It will then be located in the pool as if that name had been passed to the *roAssetPoolFiles.GetPoolFilePath()* method. This binding occurs for every instance of *roHtmlWidget*, so different mappings can be used for different bundles of content.

SetZoomLevel(scale_factor as Float) As Boolean

Adjusts the scale factor for the displayed page (the default equals 1.0).

EnableSecurity(enable As Dynamic) As Boolean

Enables or disables Chromium security checks for cross-origin requests, local video playback from HTTP, etc. (if the argument is Boolean). This method can also accept an associative array, which can contain the following parameters:

- [Boolean] *websecurity*: Enables Chromium security checks. This parameter is identical to passing a Boolean argument to the method itself.
- [Boolean] *camera_enabled*: Enables webpage access to USB cameras connected to the player (access is disabled by default). This allows support for WebRTC applications.
- [Boolean] *insecure_https_enabled*: Instructs the widget to ignore security errors when connecting to insecure HTTPS hosts (insecure HTTPS is disabled by default). Enabling this feature makes the player insecure; it is not suitable for production environments and should only be used for testing.

Note

The *camera_enabled* parameter is currently supported on the XTx44, XTx43, 4Kx42, XDx34, XDx33, HDx23, and LS423 models.

EnableMouseEvents(enable As Boolean) As Boolean

Enables response to mouse/touchscreen presses if *true*. Setting this method to *false* (the default) disables this feature.

SetPortrait(portrait_mode As Boolean) As Boolean

Sets the widget orientation to portrait if *true*. If this method is *false* (the default), the widget is oriented as a landscape.

Important

The *SetPortrait()* method has been deprecated in firmware 6.1. We recommend using the *SetTransform()* method instead.

SetTransform(transform As String) As Boolean

Sets the screen orientation of content in the widget (note that the coordinates and dimensions of the *roRectangle* containing the widget are not affected by rotation). This method accepts the following strings:

- "identity": There is no transform (i.e. the widget content is oriented as landscape). This is the default setting.
- "rot90": The widget content is rotated to portrait at 90 degrees (clockwise).
- "rot270": The widget content is rotated to portrait at 270 degrees (counter-clockwise).

SetAlpha(alpha As Integer) As Boolean

Sets the overall alpha level for the widget (the default equals 255).

EnableScrollbars(scrollbars As Boolean) As Boolean

Enables automatic scrollbars for content that does not fit into the viewport if `true`. Setting this method to `false` (the default) disables this feature.

AddFont(filename As String) As Boolean

Supplies additional or custom typefaces for the HTML rendering engine. Supported font types include TrueType Font files (`.ttf`) and Web Open Font files (`.woff`, `.woff2`).

SetPcmAudioOutputs(outputs As roArray) As Boolean

Configures the PCM audio output for the HTML widget. This method accepts one or more outputs in the form of an *roArray* of *roAudioOutput* instances.

SetCompressedAudioOutputs(outputs As roArray) As Boolean

Configures compressed audio output (e.g. Dolby AC3 encoded audio) for the HTML widget. This method accepts one or more outputs in the form of an *roArray* of *roAudioOutput* instances.

Note

When one or more audio-output methods are called, they will override the settings of the following *ifAudioControl* methods: `SetAudioOutput()`, `MapStereoOutput()`, `SetUsbAudioPort()`, `MapDigitalOutput()`. Calls to HTML audio-output methods will only take effect with subsequent calls to `SetUrl()`. Audio-output settings on an *roHtmlWidget* instance will be overwritten by **equivalent settings in the HTML**.

SetMultichannelAudioOutputs(outputs As roArray) As Boolean

SetHWZDefault(default As String) As Void

Sets the default **HWZ mode** for HTML video. Normally, HWZ must be enabled in each `<video>` tag, but passing "on" to this string enables HWZ for all `<video>` elements. This method can also accept a semicolon-separated list of HWZ parameters:

- `on/off`: Enables or disables HWZ mode.
- `z-index`: Sets the default z-ordering for `<video>` elements. A positive integer places the video in front of all graphics; a negative integer places the video behind all graphics; and a zero value disables HWZ mode completely. You can customize the z-ordering of individual video elements with respect to each other by **inserting the "z-index" parameter into the `<video>` tag**.
- `transform`: Sets the default rotation for `<video>` elements. HWZ mode must be enabled for transforms to work.
 - `identity`: No transformation (default behavior)
 - `rot90`: 90 degree clockwise rotation
 - `rot180`: 180 degree rotation
 - `rot270`: 270 degree clockwise rotation
 - `mirror`: Horizontal mirror transformation
 - `mirror_rot90`: Mirrored 90 degree clockwise rotation
 - `mirror_rot180`: Mirrored 180 degree clockwise rotation
 - `mirror_rot270`: Mirrored 270 degree clockwise rotation
- `fade`: Sets the fading behavior for `<video>` elements.
 - `auto`: Videos transition without fade effects. This is the default behavior.
 - `always`: When a video ends, the video window will go black. The new video will then fade in.
- `luma-key/cr-key/cb-key`: Enables luma and/or chroma keying for `<video>` elements. HWZ mode must be enabled for luma/chroma keying to work.

Example

```
html.SetHWZDefault("on; transform:rot90; luma-key:#ff0020;")
```

SetVideoPlayerDefaults(defaults As roAssociativeArray) As Boolean

Sets default playback settings for `<video>` elements in the widget. Available parameters are identical to the *roVideoPlayer.PlayFile()* method. Default settings will be overwritten by parameters specified in individual `<video>` elements.

ForceGpuRasterization(enable As Boolean) As Boolean

Enables GPU rasterization for HTML graphics. This method will take effect for subsequent page loads only. If Chromium determines that a page is not compatible, it will refuse to enable GPU rasterization for that page.

Note

GPU rasterization is enabled by default in firmware versions 6.2.x and later.

EnableCanvas2dAcceleration(enable As Boolean) As Boolean

Enables 2D canvas acceleration. This will improve the framerate of most HTML pages that use 2D animations, but can cause out-of-memory issues with pages that use a large number of off-screen canvas surfaces. We recommend thoroughly testing this method with your HTML content before deploying it.

SetUserStylesheet(uri As String) As Boolean

Applies the specified user stylesheet to the page(s) loaded in the widget. The parameter is a **URI** specifying any `file:` resource in the storage. The stylesheet can also be specified as inline data in the following form:

```
"data:text/css;charset=utf-8;base64,<base64 encoded data>"
```

This method will fail if you specify the inline data in any other order or if you use any data format other than base64.

SetAppCacheDir(file_path As String) As Boolean

Sets the directory to use for storing the application cache (which services `<html manifest="example.appcache">` tags). The file path is passed to the method as a string (e.g. "SD:/appcache").

SetAppCacheSize(maximum As Integer) As Boolean

Sets the maximum size (in bytes) for the application cache. Changing the storage size of the application cache will clear the cache and rebuild the cache storage. Depending on database-specific attributes, you will only be able to set the size in units that are equal to the page size of the database, which is established at creation. These storage units will occur only in the following increments: 512, 1024, 2048, 4096, 8192, 16384, 32768.

FlushCachedResources() As Boolean

Discards any resources that Chromium has cached in memory.

SetLocalStorageDir(file_path As String) As Boolean

Creates a "Local Storage" subfolder in the specified directory. This folder is used by local storage applications such as the JavaScript `storage` class.

SetLocalStorageQuota(maximum As Dynamic) As Boolean

Sets the total size (in bytes) allotted to all local storage applications. This method can accept a string, double, or integer. A number literal can only represent byte numbers up to 2GB; use a string to specify larger amounts. The default total size is 10GB (or 1GB less than the total size of the storage device, whichever is less).

SetWebDatabaseDir(file_path As String) As Boolean

Specifies the directory that should be used for web database applications (e.g. "SD:/webdb"). This method must be called before using web database applications such as Web SQL or IndexedDB.

SetWebDatabaseQuota(maximum As Dynamic) As Boolean

Sets the total size (in bytes) allotted to all web database applications. This method can accept a string, double, or integer. A number literal can only represent byte numbers up to 2GB; use a string to specify larger amounts. The default total size is 5MB.

EnableJavaScript(enable As Boolean) As Boolean

Enables/disables JavaScript on the widget. JavaScript is enabled by default.

AllowJavaScriptURLs(url_collection As roAssociativeArray)

Allows the specified JavaScript BrightScript classes to be used by the specified URLs (all BrightScript classes in JavaScript are disabled by default). This method accepts an associative array that maps JavaScript BrightScript classes to the URL(s) that are allowed to use them.

- An `all` key indicates that all classes are authorized for the associated URL(s).
- An asterisk "*" value indicates that all URLs are authorized for the associated BrightScript class.
- A "local" value indicates that all local pages are authorized for the associated BrightScript class.

The following will enable all BrightScript classes for all URLs:

```
html.AllowJavaScriptUrls({ all: "*" })
```

The following will enable all BrightScript classes for local pages and the BrightSign homepage:

```
html.AllowJavaScriptUrls({ all: ["local", "http://www.brightsign.biz"]})
```

PostJSMessages(data As roAssociativeArray) As Boolean

Posts a collection of key:value pairs to the `BSMessagePort` JavaScript class (see the JavaScript Objects for BrightScript tech note for more details). This method does not support passing nested associative arrays.

InjectJavaScript(code As String) As Boolean

Immediately injects a user script into the JavaScript engine. The passed string can contain any of the following: pure JavaScript code, a path to a JavaScript file, or a base64-encoded string (i.e. beginning with `data:text/javascript;charset=utf-8;base64,`).

This method can be used to simulate the `bind_ready` option:

```
if type(event) = "roHtmlWidgetEvent" then
    if event.GetData().reason = "load-finished" then
        h.InsertJavascript("fillPasswordFields()")
    end if
end if
```

Note

Changing the DOM is only possible at `bind_dom_loaded` and `bind_ready`. Also, since JavaScript is only guaranteed to be ready at `bind_ready`, your callbacks must use this event if they invoke any global functions.

StartInspectorServer(port As Integer) As Boolean

Enables the [Chromium Inspector](#), which allows you to debug JavaScript applications while a webpage is running. To access the console, navigate to the player IP address at the specified port number. See [this page](#) for documentation relating to the JavaScript console.

SetUserAgent(user_agent As String) As Boolean

Changes the default user-agent string for the `roHtmlWidget` instance.

GetUserAgent() As String

Returns the currently active user-agent string for the `roHtmlWidget` instance.

SetUserAgentSuffix(suffix As String) As Boolean

Appends text to the current user-agent string. Subsequent calls to this method will replace the suffix that was previously specified.

SetProxy(proxy as String) As Boolean

Sets the name or address of the proxy server that the *roHtmlWidget* instance will use to make HTTP requests. This method takes effect immediately. It does not affect network operations performed by other components in the firmware. The string can be used to specify either the proxy URL or the location of a *.pac* proxy file:

- **Proxy URL:** The proxy address should be formatted as "http://user:password@hostname:port". The hostname can contain up to four "*" characters; each "*" character can be used to replace one octet from the current IP address. For example, if the IP address is currently 192.168.1.2, and the proxy is set to "proxy-*-*", then the player will attempt to use a proxy named "proxy-192.168".
- **Proxy File:** The *.pac* proxy file can be located on either the local file system or the network. If the file is local, there are no file-name restrictions; if the file is located on the network, the file name should have a *.pac* extension. If the URL is a hostname only, it will be considered a proxy-server address rather than a file URL.

Example

The following are examples of valid location formats for *.pac* files:

- file:///storage/sd/asset_pool/a/b/ababababababa
- http://example:8080/setup/proxy.pac
- https://example/encrypted/proxy.pac
- https://example:433/encrypted/proxy.pac
- ftp://example/files/proxy.pac
- file:///storage/sd/myproxypac.txt

SetProxyBypass(hostnames As String) As Boolean

Exempts the specified hosts from the proxy configuration on the *roHtmlWidget* instance. The passed array should consist of one or more hostnames. The player will attempt to reach the specified hosts directly rather than using the proxy that has been specified with the *SetProxy()* method. For example, the hostname "example.com" would exempt "example.com", "example.com:80", and "www.example.com" from the proxy setting.

ifMessagePort

SetPort(port As roMessagePort)

Posts messages of type *roHtmlWidgetEvent* to the attached message port.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via *SetUserData()*. It will return *Invalid* if no data has been set.

The following examples show how to configure an *roHtmlWidget* instance using an associative array or methods. Note that these techniques are mutually exclusive.

Example (with initialization properties)

```
x = 0
y = 0
width = 1920
height = 1080
url = "http://www.brightsign.biz"

rect = CreateObject("roRectangle", x, y, width, height)

config = {
url: url,
```

```

mouse_enabled: true,
storage_path: "/local/",
}

html = CreateObject("roHtmlWidget", rect, config)

html.Show()

```

Example (with methods)

```

x = 0
y = 0
width = 1920
height = 1080
url = "http://www.brightsign.biz"

rect = CreateObject("roRectangle", x, y, width, height)
html = CreateObject("roHtmlWidget", rect)

html.SetUrl(url)
html.EnableMouseEvents(true)
html.SetLocalStorageDir("/local/")
html.Show()

```

roHtmlWidgetEvent

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifHTMLWidgetEvent](#)
 - [GetData\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

If an *roMessagePort* is attached to an *roHtmlWidget*, it will receive *roHtmlWidgetEvent* objects when something happens to the parent *roHtmlWidget* instance.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

ifHTMLWidgetEvent

GetData() As Object

Returns an associative array of key/value pairs. The `reason` key identifies the cause of the event:

- `load-started`: The HTML widget has started loading a page.
- `load-finished`: The HTML widget completed loading a page.
- `load-error`: The HTML widget has failed to load a page. Use the `uri` key to identify the failing resource and the `message` key to retrieve some explanatory text.

roImageBuffer

ON THIS PAGE

- [ifImageBufferControl](#)
 - [DisplayBuffer\(x As Integer, y As Integer\) As Boolean](#)
 - [GetBufferByteArray\(\) As roByteArray](#)
 - [GetBufferMetadata\(\) As roAssociativeArray](#)
 - [ConvertFormat\(a As String\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object allows you to access decoded image-file data. You can then copy or manipulate that data.

Object Creation: An `roImageBuffer` object is instantiated with an `roImagePlayer` object and a string specifying the file path of an image file.

```
CreateObject("roImageBuffer", image_player As Object, file_path As String)
```

Example

```
imgPlayer = CreateObject("roImagePlayer")  
imgBuffer = CreateObject("roImageBuffer", imgPlayer, "SD:/content/image.png")
```

`ifImageBufferControl`

DisplayBuffer(x As Integer, y As Integer) As Boolean

Displays the image on screen. The `x` and `y` integers specify the coordinates of the top-left corner of the image.

GetBufferByteArray() As roByteArray

Returns the decoded image-file data as an `roByteArray`.

GetBufferMetadata() As roAssociativeArray

Returns an associative array containing information about the image file. The associative array contains the following keys:

- `width`: The width of the image file
- `height`: The height of the image file
- `acceptable`: A Boolean integer value indicating whether the image can be displayed by the `roImagePlayer` instance
- `format`: The color space (ARGB/CMYK) of the image file

ConvertFormat(a As String) As Object

rolmagePlayer

ON THIS PAGE

- [ifImageControl](#)
 - [DisplayFile\(image_filename As String\) As Boolean](#)
 - [DisplayFile\(parameters As roAssociativeArray\) As Boolean](#)
 - [PreloadFile\(image_filename As String\) As Boolean](#)
 - [PreloadFile\(parameters As roAssociativeArray\) As Boolean](#)
 - [DisplayPreload\(\) As Boolean](#)
 - [StopDisplay\(\) As Boolean](#)
 - [DisplayFileEx\(filename As String, mode As Integer, x As Integer, y As Integer\) As Boolean](#)
 - [PreloadFileEx\(filename As String, mode As Integer, x As Integer, y As Integer\) As Boolean](#)
 - [SetDefaultMode\(mode As Integer\) As Boolean](#)
 - [SetDefaultTransition\(transition As Integer\) As Boolean](#)
 - [OverlayImage\(image_filename As String, x As Integer, y As Integer\) As Boolean](#)
 - [SetRectangle\(r As roRectangle\) As Boolean](#)
 - [GetRectangle\(\) As roRectangle](#)
 - [CreateTestHole\(hole As roRectangle\) As Boolean](#)
 - [SetTransitionDuration\(duration As Integer\) As Boolean](#)
 - [DisplayBuffer\(a As Object, b As Integer, c As Integer\) As Boolean](#)
 - [Hide\(\) As Boolean](#)
 - [Show\(\) As Boolean](#)
- [X, Y](#)
- [Testing Display Modes](#)
- [Preloading Images](#)
- [Image Decryption](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object displays static bitmap images on the video display. The simplest way to use *rolmagePlayer* is to make calls to `DisplayFile()` with the filename as a String. Alternatively, you can use `PreloadFile()` in conjunction with `DisplayPreload()` to have more control. For more pleasing aesthetics when generating an image player, use the *rolmageWidget* object.

Object Creation: The image player is displayed by first creating *roRectangle* and *rolmagePlayer* instances, then calling `SetRectangle()` using the *roRectangle* instance as the argument.

```
rectangle = CreateObject("roRectangle", 0, 0, 1024, 768)
i = CreateObject("rolmagePlayer")
i.SetRectangle(rectangle)
```

ifImageControl

DisplayFile(image_filename As String) As Boolean

Displays the image with the specified filename. The `image_filename` string must point to a `.png`, `.jpeg`, or 8-bit, 24-bit, or 32-bit `.bmp` file. Note that `.jpeg` image files with CMYK color profiles are not supported.

DisplayFile(parameters As roAssociativeArray) As Boolean

Displays an image using an associative array of display parameters:

- `Filename`: The name of the image file
- `Mode`: The image mode. See the entry for `SetDefaultMode()` below for more details.

- **Transition:** The image transition setting. See the entry for `SetDefaultTransition()` below for more details.
- **EncryptionAlgorithm:** The file-encryption algorithm. Currently the options are "AesCtr" and "AesCtrHmac".
- **EncryptionKey:** The key to decrypt the image file. This is a byte array consisting of 128 bits of key, followed by 128 bits of IV.

See the [Image Decryption](#) section below for details on displaying encrypted images.

PreloadFile(image_filename As String) As Boolean

Loads the specified image file into an offscreen memory buffer.

PreloadFile(parameters As roAssociativeArray) As Boolean

Loads an image file into an offscreen memory buffer. Image display properties are determined by an associative array of parameters:

- **Filename:** The name of the image file
- **Mode:** See the entry for `SetDefaultMode()` below for more details.
- **Transition:** See the entry for `SetDefaultTransition()` below for more details.

DisplayPreload() As Boolean

Uses the on-screen memory buffer to display the image stored in the offscreen memory buffer using `PreloadFile()`. There are only two memory buffers: one is displayed on screen; and the other is used for preloading images. `PreloadFile()` can be called multiple times before `DisplayPreload()` is called, and will keep loading into the same off-screen buffer. The `DisplayFile()` method calls `PreloadFile()` followed immediately by `DisplayPreload()`, so any previously preloaded image will be lost. If no image is preloaded, `DisplayPreload()` will have no effect.

StopDisplay() As Boolean

Removes an image from the display.

DisplayFileEx(filename As String, mode As Integer, x As Integer, y As Integer) As Boolean

PreloadFileEx(filename As String, mode As Integer, x As Integer, y As Integer) As Boolean

SetDefaultMode(mode As Integer) As Boolean

Sets the default image display mode for `DisplayFile()` and `PreloadFile()`. If `SetDefaultMode()` is not called, then the default mode is set to 0 (equivalent to the image being centered without scaling). The supported display mode are listed below:

- **0 – Center image:** No scaling takes place. Cropping only occurs if the image is bigger than the window.
- **1 – Scale to fit:** The image is scaled so that it is fully viewable, with its aspect ratio maintained.
- **2 – Scale to fill and crop:** The image is scaled so that it completely fills the window, with its aspect ratio maintained.
- **3 – Scale to fill:** The image is stretched so that it fills the window and the whole image is viewable. The aspect ratio will not be maintained if it is different from the window.

SetDefaultTransition(transition As Integer) As Boolean

Sets the transition to be used when the next image is displayed. The following are available transitions:

- 0: No transition: immediate blit
- 1-4: Wipes from top, bottom, left, or right.
- 5-8: Explodes from centre, top left, top right, bottom left, or bottom right.
- 10-11: Uses vertical or horizontal venetian-blind effect.
- 12-13: Combs vertical or horizontal.
- 14: Fades out to background color, then back in.
- 15: Fades between current image and new image.
- 16-19: Slides from top, bottom, left or right.
- 20-23: Slides entire screen from top, bottom, left, or right.
- 24-25: Scales old image in, then the new one out again (this works as a pseudo rotation around a vertical or horizontal axis).
- 26-29: Expands a new image onto the screen from right, left, bottom, or top.

`SetTransform(transform As String) As Boolean`

Applies one of eight transforms to the image. Calls to this method only take effect when the next file is displayed. Note that the image rectangle itself does not change to accommodate the new height and width ratio of a transformed image. This method can be called separately on multiple *roImagePlayer* or *roImageWidget* instances.

- *identity*: No transformation (default behavior)
- *rot90*: 90 degree clockwise rotation
- *rot180*: 180 degree rotation
- *rot270*: 270 degree clockwise rotation
- *mirror*: Horizontal mirror transformation
- *mirror_rot90*: Mirrored 90 degree clockwise rotation
- *mirror_rot180*: Mirrored 180 degree clockwise rotation
- *mirror_rot270*: Mirrored 270 degree clockwise rotation

OverlayImage(image_filename As String, x As Integer, y As Integer) As Boolean

Composites the image with the specified filename on top of the primary *DisplayFile()* image. Use the *x* and *y* integers to specify its location within the image widget.

SetRectangle(r As roRectangle) As Boolean

Changes the size and positioning of the image rectangle using the passed *roRectangle* object.

GetRectangle() As roRectangle

Returns an *roRectangle* object that has the same location and dimensions as the *roRectangle* object used to define the image window.

CreateTestHole(hole As roRectangle) As Boolean

Creates a hole in the image with the location and dimensions specified in the passed *roRectangle* instance. Any video windows located directly beneath the image will show through. This method will disrupt image playback and should be used for test purposes only.

SetTransitionDuration(duration As Integer) As Boolean

Sets the amount of time it takes (in milliseconds) for a specified transition effect to take place. The default transition duration is 1000 milliseconds.

DisplayBuffer(a As Object, b As Integer, c As Integer) As Boolean

Hide() As Boolean

Hides the image currently being displayed by the *roImagePlayer* widget.

Show() As Boolean

Shows the image currently being displayed by the *roImagePlayer* widget.

X, *Y*

The *x* and *y* values indicate which position of the image to center as near as possible, or both *x* and *y* can be set to -1, which uses the center of the image as the point to position nearest to the center.

To display images in a zone, *SetRectangle()* must be called, and *EnableZoneSupport()* must be included in a script to use the zones functionality.

Testing Display Modes

Here are some example shell commands you can use to test the different display modes:

```
BrightSign> image filename.bmp 0
BrightSign> image filename.bmp 1
BrightSign> image filename.bmp 2
BrightSign> image filename.bmp 3

BrightSign> image filename.bmp 0 0 0
BrightSign> image filename.bmp 2 0 0
```

Preloading Images

The following example script uses preloaded images to improve the UI speed when the user hits a key on the keyboard. As soon as a key is struck, the display switches to the new image, which has already been preloaded. The only possible delay occurs if the key is hit while the image is preloading. In this case, the image will display as soon as it is loaded.

```
i = CreateObject("roImagePlayer")
p = CreateObject("roMessagePort")
k = CreateObject("roKeyboard")
k.SetPort(p)

i.PreloadFile("one.bmp")

loop:
i.DisplayPreload
i.PreloadFile("two.bmp")
Wait(0,p)
i.DisplayPreload
i.PreloadFile("one.bmp")
Wait(0,p)
goto loop
```

Image Decryption

The *roImagePlayer*, *roImageWidget*, *roClockWidget*, *roTextWidget*, and *roCanvasWidget* objects can be used to display encrypted images. Each object has an image playback method that accepts an associative array, which can include the *EncryptionAlgorithm* and *EncryptionKey* decryption parameters.

Note

Contact support@brightsign.biz to learn more about generating a key for obfuscation and storing it on the player.

You can call `roDeviceInfo.HasFeature("media_decryption")` to determine if a player model and firmware version supports image decryption.

Example

```
print "Play ENCRYPTED image in an image widget"

imagePlayer = CreateObject("roImageWidget", r1)

aa=CreateObject("roAssociativeArray")
aa.filename = "sd:/images_enc.jpg"
aa.encryptionalgorithm = "AesCtr"
aa.encryptionkey = CreateObject("roByteArray")
aa.encryptionkey.fromhexstring
("01030507090b0d0f00020406080a0c0e000000000000000000000000000000")

imagePlayer.DisplayFile(aa)
sleep(10000)
imagePlayer.Hide()

print "Play ENCRYPTED image with PlayStaticImage"

videoPlayer = CreateObject("roVideoPlayer")

aa=CreateObject("roAssociativeArray")
aa.filename = "sd:/images_enc.jpg"
aa.encryptionalgorithm = "AesCtr"
aa.encryptionkey = CreateObject("roByteArray")
aa.encryptionkey.fromhexstring
("01030507090b0d0f00020406080a0c0e000000000000000000000000000000")
```

```

videoPlayer.PlayStaticImage(aa)
sleep(10000)
videoPlayer = invalid

print "Show CLOCK image"

resourceManager = CreateObject("roResourceManager", "sd:/resources.txt")

clockWidget = CreateObject("roClockWidget", r1, resourceManager, {})

aa=CreateObject("roAssociativeArray")
aa.filename = "sd:/images_enc.jpg"
aa.encryptionalgorithm = "AesCtr"
aa.encryptionkey = CreateObject("roByteArray")
aa.encryptionkey.fromhexstring
("01030507090b0d0f00020406080a0c0e000000000000000000000000000000")

clockWidget.SetBackgroundBitmap(aa, True)
clockWidget.Show()
sleep(10000)
clockWidget.Hide()

print "Text widget with encrypted background image"

twParams = CreateObject("roAssociativeArray")
twParams.LineCount = 1
twParams.TextMode = 1
twParams.Rotation = 0
twParams.Alignment = 1

tw=CreateObject("roTextWidget", r1, 1, 2, twParams)
tw.SetBackgroundColor(&h00ff0000)
tw.SetForegroundColor(&hff00ff00)
tw.PushString("Encrypted Background")
'tw.SetRectangle(r)

aa=CreateObject("roAssociativeArray")
aa.filename = "sd:/images_enc.jpg"
aa.encryptionalgorithm = "AesCtr"
aa.encryptionkey = CreateObject("roByteArray")
aa.encryptionkey.fromhexstring
("01030507090b0d0f00020406080a0c0e000000000000000000000000000000")

tw.SetBackgroundBitmap(aa, True)
tw.Show()
sleep(10000)
tw.Hide()

cw=CreateObject("roCanvasWidget", rect)
canvas_aa=CreateObject("roAssociativeArray")
canvas_aa.FileName = "sd:/images_enc.jpg"
canvas_aa.Encryptionalgorithm = "AesCtr"
canvas_aa.EncryptionKey = CreateObject("roByteArray")
canvas_aa.EncryptionKey.FromHexString
("01030507090b0d0f00020406080a0c0e000000000000000000000000000000")
cw.SetLayer(canvas_aa, 1)
cw.Show()

```

rolmageWidget

ON THIS PAGE

- [ifImageControl](#)

- `DisplayFile(image_filename As String) As Boolean`
- `DisplayFile(parameters As roAssociativeArray) As Boolean`
- `PreloadFile(image_filename As String) As Boolean`
- `PreloadFile(parameters As roAssociativeArray) As Boolean`
- `DisplayPreload() As Boolean`
- `StopDisplay() As Boolean`
- `DisplayFileEx(filename As String, mode As Integer, x As Integer, y As Integer) As Boolean`
- `PreloadFileEx(filename As String, mode As Integer, x As Integer, y As Integer) As Boolean`
- `SetDefaultMode(mode As Integer) As Boolean`
- `SetDefaultTransition(transition As Integer) As Boolean`
- `SetTransform(transform As String) As Boolean`
- `OverlayImage(image_filename As String, x As Integer, y As Integer) As Boolean`
- `SetRectangle(r As roRectangle) As Boolean`
- `GetRectangle() As roRectangle`
- `CreateTestHole(hole As roRectangle) As Boolean`
- `SetTransitionDuration(duration As Integer) As Boolean`
- `DisplayBuffer(a As Object, b As Integer, c As Integer) As Boolean`
- `Hide() As Boolean`
- `Show() As Boolean`
- **Multiscreen Images**

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object can be used in place of `rolmagePlayer` in cases where the image is displayed within a rectangle. Using an `rolmageWidget` can result in more pleasing aesthetics for image player creation; it can also be used to display images in a multi-screen array. Beyond this, `rolmageWidget` behaves identically to `rolmagePlayer`.

Object Creation: The image widget area is generated using an `roRectangle` object.

```
rectangle = CreateObject("roRectangle", 0, 0, 1024, 768)
i = CreateObject("rolmageWidget", rectangle)
```

`ifImageControl`

`DisplayFile(image_filename As String) As Boolean`

Displays the image with the specified filename. The `image_filename` string must point to a `.png`, `.jpeg`, or 8-bit, 24-bit, or 32-bit `.bmp` file. Note that `.jpeg` image files with CMYK color profiles are not supported.

`DisplayFile(parameters As roAssociativeArray) As Boolean`

Displays an image using an associative array of display parameters:

- **Filename:** The name of the image file
- **Mode:** The image mode. See the entry for `SetDefaultMode()` below for more details.
- **Transition:** The image transition setting. See the entry for `SetDefaultTransition()` below for more details.
- **EncryptionAlgorithm:** The file-encryption algorithm. Currently the options are "AesCtr" and "AesCtrHmac".
- **EncryptionKey:** The key to decrypt the image file. This is a byte array consisting of 128 bits of key, followed by 128 bits of IV.

See the [Image Decryption](#) description in the `rolmagePlayer` entry for details on displaying encrypted images.

`PreloadFile(image_filename As String) As Boolean`

Loads the specified image file into an offscreen memory buffer.

PreloadFile(parameters As roAssociativeArray) As Boolean

Loads an image file into an offscreen memory buffer. Image display properties are determined by an associative array of parameters:

- **Filename:** The name of the image file
- **Mode:** See the entry for SetDefaultMode() below for more details.
- **Transition:** See the entry for SetDefaultTransition() below for more details.

DisplayPreload() As Boolean

Uses the on-screen memory buffer to display the image stored in the offscreen memory buffer using PreloadFile(). There are only two memory buffers: one is displayed on screen; and the other is used for preloading images. PreloadFile() can be called multiple times before DisplayPreload() is called, and will keep loading into the same off-screen buffer. The DisplayFile() method calls PreloadFile() followed immediately by DisplayPreload(), so any previously preloaded image will be lost. If no image is preloaded, DisplayPreload() will have no effect.

StopDisplay() As Boolean

Removes an image from the display.

DisplayFileEx(filename As String, mode As Integer, x As Integer, y As Integer) As Boolean

PreloadFileEx(filename As String, mode As Integer, x As Integer, y As Integer) As Boolean

SetDefaultMode(mode As Integer) As Boolean

Sets the default image display mode for DisplayFile() and PreloadFile(). If SetDefaultMode() is not called, then the default mode is set to 0 (equivalent to the image being centered without scaling). The supported display mode are listed below:

- **0 – Center image:** No scaling takes place. Cropping only occurs if the image is bigger than the window.
- **1 – Scale to fit:** The image is scaled so that it is fully viewable, with its aspect ratio maintained.
- **2 – Scale to fill and crop:** The image is scaled so that it completely fills the window, with its aspect ratio maintained.
- **3 – Scale to fill:** The image is stretched so that it fills the window and the whole image is viewable. The aspect ratio will not be maintained if it is different from the window.

SetDefaultTransition(transition As Integer) As Boolean

Sets the transition to be used when the next image is displayed. The following are available transitions:

- 0: No transition: immediate blit
- 1-4: Wipes from top, bottom, left, or right.
- 5-8: Explodes from centre, top left, top right, bottom left, or bottom right.
- 10-11: Uses vertical or horizontal venetian-blind effect.
- 12-13: Combs vertical or horizontal.
- 14: Fades out to background color, then back in.
- 15: Fades between current image and new image.
- 16-19: Slides from top, bottom, left or right.
- 20-23: Slides entire screen from top, bottom, left, or right.
- 24-25: Scales old image in, then the new one out again (this works as a pseudo rotation around a vertical or horizontal axis).
- 26-29: Expands a new image onto the screen from right, left, bottom, or top.

SetTransform(transform As String) As Boolean

Applies one of eight transforms to the image. Calls to this method only take effect when the next file is displayed. Note that the image rectangle itself does not change to accommodate the new height and width ratio of a transformed image. This method can be called separately on multiple *roImagePlayer* or *roImageWidget* instances.

- **identity:** No transformation (default behavior)
- **rot90:** 90 degree clockwise rotation
- **rot180:** 180 degree rotation
- **rot270:** 270 degree clockwise rotation
- **mirror:** Horizontal mirror transformation
- **mirror_rot90:** Mirrored 90 degree clockwise rotation
- **mirror_rot180:** Mirrored 180 degree clockwise rotation

- `mirror_rot270`: Mirrored 270 degree clockwise rotation

OverlayImage(image_filename As String, x As Integer, y As Integer) As Boolean

Composites the image with the specified filename on top of the primary `DisplayFile()` image. Use the `x` and `y` integers to specify its location within the image widget.

SetRectangle(r As roRectangle) As Boolean

Changes the size and positioning of the image rectangle using the passed `roRectangle` object.

GetRectangle() As roRectangle

Returns an `roRectangle` object that has the same location and dimensions as the `roRectangle` object used to define the image window.

CreateTestHole(hole As roRectangle) As Boolean

Creates a hole in the image with the location and dimensions specified in the passed `roRectangle` instance. Any video windows located directly beneath the image will show through. This method will disrupt image playback and should be used for test purposes only.

SetTransitionDuration(duration As Integer) As Boolean

Sets the amount of time it takes (in milliseconds) for a specified transition effect to take place. The default transition duration is 1000 milliseconds.

DisplayBuffer(a As Object, b As Integer, c As Integer) As Boolean

Hide() As Boolean

Hides the image currently being displayed by the `roImageWidget` instance.

Show() As Boolean

Shows the image currently being displayed by the `roImageWidget` instance.

Multiscreen Images

This object includes overloaded `PreloadFile()` and `DisplayFile()` methods. These methods receive an `roAssociativeArray` object that stores various options to be passed. They must be used when displaying images across multiple screens in an array, or displaying a portion of an image—though they can also be used in place of the original method calls in all cases.

The following code uses the `PreloadFile()` method for a multiscreen display:

```
i=CreateObject("roImageWidget")
a=CreateObject("roAssociativeArray")
a["Filename"] = "test.jpg"
a["Mode"] = 1
a["Transition"] = 14
a["MultiscreenWidth"] = 3
a["MultiscreenHeight"] = 2
a["MultiscreenX"] = 0
a["MultiscreenY"] = 0
i.PreloadFile(a)
i.DisplayPreload
```

The filename, mode, and transition values are the same as those documented for the `DisplayFile()` and `PreloadFile()` methods above. The `MultiscreenWidth` and `MultiscreenHeight` parameters specify the width and height of the multi-screen matrix. For example, 3x2 would be three screens wide and two screens high. The `MultiscreenX` and `MultiscreenY` specify the position of the current screen within that matrix.

In the above case, on average only 1/6 of the image is drawn on each screen, though the image mode still applies so that, depending on the shape of the image, it may have black bars on the side screens. It is relatively simple, therefore, for an image widget to display part of an image based on its position in the multiscreen array. The following are default values for the parameters:

- Mode = 0

- Transition = 0
- MultiscreenWidth = 1
- MultiscreenHeight = 1
- MultiscreenX = 0
- MultiscreenY = 0

This code uses `DisplayFile()` to display a portion of an image:

```
i=CreateObject("roImageWidget")
a=CreateObject("roAssociativeArray")
a["Filename"] = "test.JPG"
a["Mode"] = 0
a["SourceX"] = 600
a["SourceY"] = 600
a["SourceWidth"] = 400
a["SourceHeight"] = 400
i.DisplayFile(a)
```

This displays just a portion of the image test JPG starting at coordinates `SourceX`, `SourceY`, and `SourceWidth` by `SourceHeight` in size. The viewmode is still honored as if it were displaying the whole file.

roRectangle

ON THIS PAGE

- [Rectangles with 4K Video Modes](#)
- [ifRectangle](#)
 - [SetX\(x As Integer\) As Void](#)
 - [SetY\(y As Integer\) As Void](#)
 - [SetWidth\(width As Integer\) As Void](#)
 - [SetHeight\(height As Integer\) As Void](#)
 - [GetX\(\) As Integer](#)
 - [GetY\(\) As Integer](#)
 - [GetWidth\(\) As Integer](#)
 - [GetHeight\(\) As Integer](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is passed to various video and graphics widgets (*roVideoPlayer*, *roImageWidget*, *roHtmlWidget*, etc.) to specify their size and positioning.

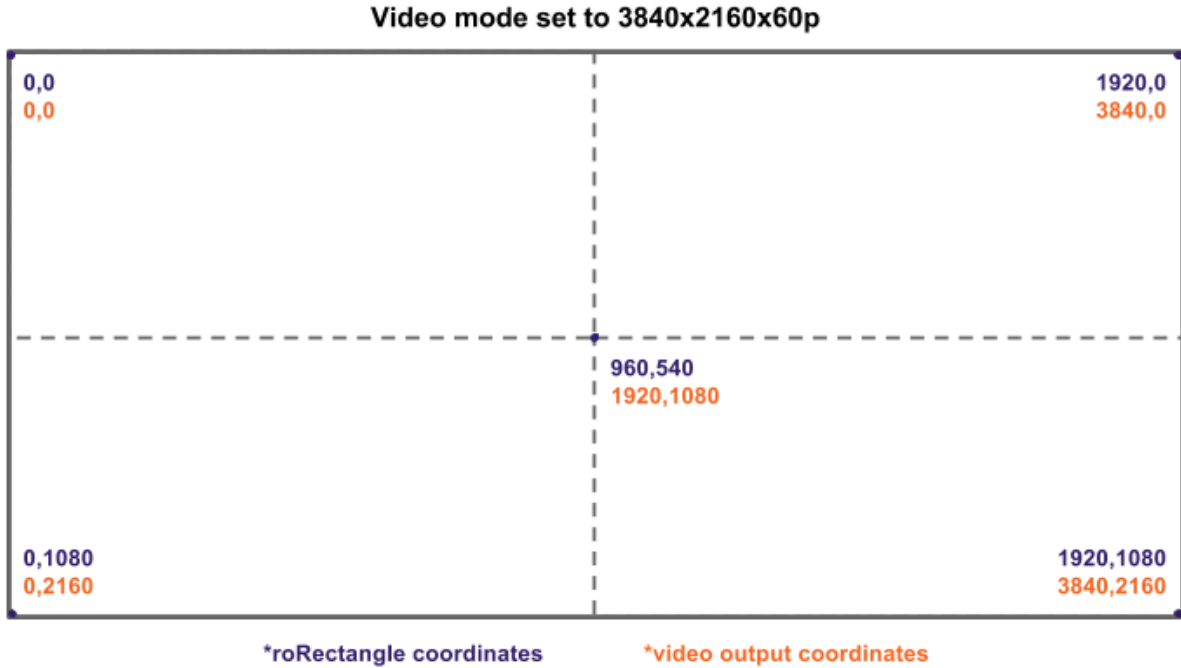
Object Creation: This object is created with coordinate and dimension parameters.

```
CreateObject("roRectangle", x As Integer, y As Integer, width As Integer, height As Integer)
```

`SetRectangle()` calls made by widget objects (e.g. *roImageWidget.SetRectangle()*) honor the view-mode or aspect-ratio conversion mode set by the user. If the user has set the video player for letterboxing, it will occur if the video does not fit in the new rectangle exactly.

Rectangles with 4K Video Modes

When the player is using a 4K video mode (e.g. 3840x2160x60p), rectangles still operate as if the total screen area is 1920x1080 (unless **full-resolution graphics** is enabled). They are then scaled by a factor of 2 when the video is output. For example, to display a full-screen 4K video, set the *roRectangle* instance to 0, 0, 1920, 1080; to display an additional HD video in the top-right quadrant of the screen, set another *roRectangle* instance to 960, 0, 960, 540.



Note that, while 4K video is output at a 1:1 pixel ratio, graphics elements (*roImageWidget*, *roCanvasWidget*, etc.) are upscaled to match the size of the 4K video output. This occurs irrespective of the size of the original image and widget rectangle: For example, an image in a 960x540 rectangle will first be downsampled to 960x540, then upscaled to 1920x1080. If you want to display images without upscaling on 4K video modes, display them in a video window using the *roVideoPlayer.PlayStaticImage()* method (or enable **full-resolution graphics** on the XTx43).

ifRectangle

SetX(x As Integer) As Void

Specifies a new x coordinate for the rectangle.

SetY(y As Integer) As Void

Specifies a new y coordinate for the rectangle.

SetWidth(width As Integer) As Void

Specifies a new width value for the rectangle.

SetHeight(height As Integer) As Void

Specifies a new height value for the rectangle.

GetX() As Integer

Returns the x coordinate of the rectangle.

GetY() As Integer

Returns the y coordinate of the rectangle.

GetWidth() As Integer

Returns the width of the rectangle.

GetHeight() As Integer

Returns the height of the rectangle.

roStreamQueue

ON THIS PAGE

- [File Requirements](#)
- [ifStreamQueue](#)
 - [QueueFile\(filename As String\) As Boolean](#)
 - [Loop\(loop As Boolean\) As Boolean](#)
 - [LoopLast\(loop_last As Boolean\) As Boolean](#)
 - [SetPreferredAudio\(description As String\) As Boolean](#)
 - [SetPreferredVideo\(description As String\) As Boolean](#)
 - [NextFile\(a As Boolean\) As Boolean](#)
 - [Start\(\) As Boolean](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifIdentity](#)
 - [GetIdentity\(\) As Integer](#)
- [Playing and Streaming Queues](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object allows you to play a list of video files seamlessly (i.e. without any blank frames or interrupts between one video and the next). You can link *roStreamQueue* to an *roVideoPlayer* instance for seamless video playback or to an *roMediaStreamer* instance for seamless video streaming.

Object Creation: This object is created with no parameters.

```
CreateObject("roStreamQueue")
```

File Requirements

Video files must meet the following requirements to work with the *roStreamQueue* object:

- Files must be Transport Stream (.ts) formatted.
- All files in a queue should have identical PIDs.
- Files must begin and end with a closed GOP.
- Files must have regular PCRs at intervals of 100ms or shorter. Note that the default interval for FFmpeg is 1000ms unless it is configured for CBR muxing (in which case the interval is set to 20ms). The following is an example of an appropriate command-line to remux:

```
avconv -i clip_short.ts -vcodec copy -acodec copy -muxrate 15M clip_short2.ts
```

ifStreamQueue

QueueFile(filename As String) As Boolean

Adds the specified video file to the queue.

Loop(loop As Boolean) As Boolean

Specifies that playback/streaming should return to the beginning of the queue once it reaches the end. If `Loop(false)` and `LoopLast(false)` are both called, playback/streaming will stop once the end of the queue is reached. This is also the default behavior.

LoopLast(loop_last As Boolean) As Boolean

Specifies that playback/streaming should loop the last file in the queue once it reaches the end. This method has no effect if `Loop(true)` is called as well.

SetPreferredAudio(description As String) As Boolean

Chooses a video stream from the video input based on the `parameters` in the passed string.

SetPreferredVideo(description As String) As Boolean

Chooses an audio stream from the video input based on the `parameters` in the passed string.

NextFile(a As Boolean) As Boolean

Start() As Boolean

`ifMessagePort`

SetPort(port As roMessagePort)

Posts messages of type `roStreamQueueEvent` to the attached message port. An event is raised whenever the end of the queue is reached.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifIdentity`

GetIdentity() As Integer

Returns a unique number that can be used to identify when events originate from this object.

Note

The `ifIdentity` interface has been deprecated. We recommend using the `ifUserData` interface instead.

Playing and Streaming Queues

To use `roStreamQueue` as a streaming playlist, include it as the source component in an `roMediaStreamer.SetPipeline()` call. For video playback, use an `roVideoPlayer` instance as the destination component in the `SetPipeline()` call.

Example (roVideoPlayer)

```
q=createobject("rostreamqueue")
q.queuefile("sd:/Test_Count_Up_Blue_Frames.ts")
q.queuefile("sd:/Test_Count_Up_Green_Frames.ts")
q.loop(true)
c=createobject("romediastreamer")
r=createobject("rorectangle",0,0,1920,1080)
v=createobject("rovideoplayer")
```

```
v.setrectangle(r)
c.setpipeline([q, v])
c.start()
```

Example (roMediaStreamer)

```
queue = CreateObject("roStreamQueue")
queue.queueFile("file1.ts")
queue.queueFile("file2.ts")
queue.loop(true)
m = CreateObject("roMediaStreamer")
m.setpipeline([queue, "udp://239.0.156.101:5000/?maxbitrate=0"])
m.Start()
```

Important

We strongly suggest appending the "maxbitrate=0" parameter to UDP streams when using *roStreamQueue*. This prevents the *roMediaStreamer* instance from rate-limiting an *roStreamQueue* stream that is already running at a precise, correct rate.

roTextField

ON THIS PAGE

- [ifTextField](#)
 - [Cls\(\) As Void](#)
 - [GetWidth\(\) As Integer](#)
 - [GetHeight\(\) As Integer](#)
 - [SetCursorPos\(x As Integer, y As Integer\) As Void](#)
 - [GetValue\(\) As Integer](#)
- [ifStreamSend](#)
 - [SendByte\(byte As Integer\) As Void](#)
 - [SendLine\(string As String\) As Void](#)
 - [SendBlock\(string As Dynamic\) As Void](#)
 - [SetSendEol\(string As String\) As Void](#)
- [Printing a Text Field](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

A text field represents an area of the screen that can contain arbitrary text. This feature is intended for presenting diagnostic and usage information. Use the *roTextWidget* object to generate text for user interfaces and signage.

Object Creation: The object is created with several parameters:

```
CreateObject("roTextField", xpos As Integer, ypos As Integer, width_in_chars As Integer,
height_in_chars As Integer, metadata As Object)
```

- *xpos*: The horizontal coordinate for the top left of the text field.
- *ypos*: The vertical coordinate for the top left of the text field. The top of the screen is equivalent to zero.

- `width_in_chars`: The width of the text field in character cells.
- `height_in_chars`: The height of the text field in character cells.
- `metadata`: An optional *roAssociativeArray* containing extra parameters for the text field. You can pass zero if you do not require this.

The metadata associative array supports the following extra parameters:

- "CharWidth": The width of each character cell in pixels.
- "CharLength": The height of each character cell in pixels.
- "BackgroundColor": The background color of the text field as an integer specifying eight bits (for each) for red, green and blue in the form &Hrrggbb.
- "TextColor": The color of the text as an integer specifying eight bits (for each) for red, green and blue in the form &Hrrggbb.
- "Size": An alternative to "CharWidth" and "CharLength" for specifying either normal size text (0) or double-sized text (1).

Note

In TV modes, a border around the screen may not be displayed due to overscanning. You may want to use the *roVideoMode* object functions `GetSafeX()` and `GetSafeY()` to ensure that the coordinates you use will be visible.

`ifTextField`

Cls() As Void

Clears the text field.

GetWidth() As Integer

Returns the width of the text field

GetHeight() As Integer

Returns the height of the text field.

SetCursorPos(x As Integer, y As Integer) As Void

Moves the cursor to the specified position. Subsequent output will appear at this position.

GetValue() As Integer

Returns the value of the character currently under the cursor.

`ifStreamSend`

SendByte(byte As Integer) As Void

Writes the character indicated by the specified number at the current cursor position within the text field. It then advances the cursor.

SendLine(string As String) As Void

Writes the characters specified at the current cursor position followed by the end-of-line sequence.

SendBlock(string As Dynamic) As Void

Writes the characters specified at the current cursor position and advances the cursor to one position beyond the last character. This method can support either a string or an *roByteArray*. If the block is a string, any null bytes will terminate the block.

SetSendEol(string As String) As Void

Sets the sequence sent at the end of a `SendLine()` value. You should leave this at the default ASCII value of 13 (Carriage Return) for normal use. If you need to change this value to another non-printing character, use the `chr` [global function](#).

Note

The `ifStreamSend` interface is also described in the section documenting the various file objects. The interface is described again here in a manner more specific to the `roTextField` object.

Printing a Text Field

As with any object that implements the *ifStreamSend* interface, a text field can be written to using the PRINT #textfield syntax. See the example below for more details.

It is also possible to write to a text field using the syntax PRINT #textfield, @pos, where *pos* is the character position in the *textfield*. For example, if your *textfield* object has 8 columns and 3 rows, writing to position 17 writes to row 3, column 2 (positions 0-7 are in row 1; positions 8-15 are in row 2; and positions 16-23 are in the last row).

When output reaches the bottom of the text field, it will automatically scroll.

Example

```
meta = CreateObject("roAssociativeArray")
meta.AddReplace("CharWidth", 20)
meta.AddReplace("CharLength", 32)
meta.AddReplace("BackgroundColor", &H101010) ' Dark grey
meta.AddReplace("TextColor", &Hffff00) ' Yellow
vm = CreateObject("roVideoMode")
tf = CreateObject("roTextField", vm.GetSafeX(), vm.GetSafeY(), 20, 20, meta)
print #tf, "Hello World"
tf.SetCursorPos(4, 10)
print #tf, "World Hello"
```

roTextWidget

ON THIS PAGE

- [ifTextWidget](#)
 - [PushString\(str As String\) As Boolean](#)
 - [PopStrings\(number_of_string_to_pop As Integer\) As Boolean](#)
 - [GetStringCount\(\) As Integer](#)
 - [Clear\(\) As Boolean](#)
 - [SetStringSource\(file_path As String\) As Boolean](#)
 - [SetAnimationSpeed\(speed As Integer\) As Boolean](#)
 - [SetSeparator\(separator As String\) As Boolean](#)
 - [SetSyncManager\(domain As String\) As Boolean](#)
 - [SetMultiscreen\(offset As Integer, size As Integer, ip_address As String, port As Integer\) As Boolean](#)
- [ifWidget](#)
 - [Show\(\) As Boolean](#)
 - [Hide\(\) As Boolean](#)
 - [SetForegroundColor\(color As Integer\) As Boolean](#)
 - [SetBackgroundColor\(color As Integer\) As Boolean](#)
 - [SetFont\(font_filename As String\) As Boolean](#)
 - [SetBackgroundBitmap\(bitmap_filename As String, stretch As Boolean\) As Boolean](#)
 - [SetBackgroundBitmap\(parameters As roAssociativeArray, stretch As Boolean\) As Boolean](#)
 - [SetSafeTextRegion\(region As roRectangle\) As Boolean](#)
 - [SetRectangle\(r As roRectangle\) As Boolean](#)
 - [GetFailureReason\(\) As String](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is used to display text on the screen.

Object Creation: This object is created using one of two variants.

Option 1

```
CreateObject("roTextWidget", r As roRectangle, line_count As Integer, text_mode As Integer, pause_time As Integer)
```

- *r*: An *roRectangle* instance that contains the text
- *line_count*: The number of lines of text to show within the rectangle
- *text_mode*: The animation characteristics of the text:
 - 0: An animated view similar to teletype
 - 1: Static text
 - 2: Simple text with no queue of strings
 - 3: Scrolling ticker (strings are separated with a diamond by default; the separator can be modified using the `SetSeparator()` method). Note that this option is not available for LSx22, HDx20, or HDx10 models.
- *pause_time*: The length of time each string is displayed before displaying the next string. This does not apply to text mode 2 or 3 because the strings on screen are updated immediately.

Option 2

```
CreateObject("roTextWidget", r As roRectangle, line_count As Integer, text_mode As Integer, parameters As roAssociativeArray)
```

- *r*: An *roRectangle* instance that contains the text
- *line_count*: The number of lines of text to show within the rectangle
- *text_mode*: The animation characteristics of the text:
 - 0: An animated view similar to teletype
 - 1: Static text
 - 2: Simple text with no queue of strings
 - 3: Scrolling ticker (strings are separated with a diamond by default; the separator can be modified using the `SetSeparator()` method). Note that this option is not available for LSx22, HDx20, or HDx10 models.
- *parameters*: An associative array that can include the following values:
 - *LineCount*: The number of lines of text to show within the rectangle.
 - *TextMode*: The animation characteristics of the text:
 - 0: An animated view similar to teletype
 - 1: Static text
 - 2: Simple text with no queue of strings
 - 3: Scrolling ticker (strings are separated with a diamond by default; the separator can be modified using the `SetSeparator()` method). Note that this option is not available for LSx22, HDx20, or HDx10 models.
 - *PauseTime*: The length of time each string is displayed before displaying the next string. This does not apply to text mode 2 or 3 because the strings on screen are updated immediately.
 - *Rotation*: The rotation of the text within the widget (note that the coordinates and dimensions of the rectangle itself are not rotated):
 - 0: 0 degrees
 - 1: 90 degrees. This value can also be represented in degrees (90) or radians (.5).
 - 2: 180 degrees. This value can also be represented in degrees (180) or radians ().
 - 3: 270 degrees. This value can also be represented in degrees (270) or radians (1.5).
 - *Alignment*: The alignment of the text:
 - 0: Left
 - 1: Center
 - 2: Right

Tip

Modes 0, 1, and 3 are useful for displaying RSS feeds and ticker-type text. However, for dynamic data where immediate screen updates are required, mode 2 may be more appropriate. Mode 2 allows text to be drawn immediately to the screen.

Note

Text-mode 3 (scrolling ticker) supports both right-to-left and left-to-right (e.g. Arabic, Hebrew) scrolling modes, depending on the language of the first string or file added to the widget. To change the scrolling direction, you will first need to remove all strings from the ticker.

`ifTextWidget`

PushString(str As String) As Boolean

Adds the string to the list of strings to display in modes 0, 1, and 3. Strings are displayed in order, and when the end is reached, the object loops, returning to the beginning of the list. In mode 2, the string is displayed immediately.

Note

Adding too many strings without popping them from the stack will cause `PushString()` to return `False` (the exact number depends on the font size for each string). The maximum recommended number of stored strings is 64.

PopStrings(number_of_string_to_pop As Integer) As Boolean

Pops strings off the front of the list (using "last in, first out" ordering) in modes 0, 1, and 3. This occurs the next time the widget wraps so that strings can be added to and removed from the widget seamlessly. In mode 2, the string is cleared from the widget immediately.

GetStringCount() As Integer

Returns the number of strings that will exist once any pending pops have taken place.

Clear() As Boolean

Clears the list of strings, leaving the widget blank and able to accept more `PushString()` calls.

SetStringSource(file_path As String) As Boolean

Displays the text file at the specified path as a single, continuous string. This method is only applicable to text mode 3 (scrolling ticker). When the end of the file is reached, the text widget loops to the beginning, using a diamond symbol as the separator.

SetAnimationSpeed(speed As Integer) As Boolean

Sets the speed at which animated text displays. This method is applicable to text modes 0 and 3 only:

- Mode 0: The default speed value is 10000. Setting an integer above this value decreases the speed of the teletype-style ticker: For example, specifying a value of 20000 will decrease the default speed at which text displays by half, while a value of 5000 will double the default speed.
- Mode 3: The default speed value is 10000. Because the speed of a scrolling ticker is measured in pixels per second (PPS), the speed must be a multiple of the current framerate, or else it will be rounded down to the nearest multiple (for example, a framerate of 60p will honor PPS values of 60, 120, 180, etc.). The software determines the speed of the scrolling ticker by performing the following calculation on the passed `speed` parameter:
$$\text{PPS} = (\text{speed} * 60) / 10000$$

SetSeparator(separator As String) As Boolean

Changes the separator between strings. The default diamond separator will be replaced by the contents of the passed string. This method applies to Text Mode 3 (smooth scrolling ticker) only. The following strings indicate special symbols: `":diamond:"`, `":circle:"`, `":square:"`.

SetSyncManager(domain As String) As Boolean

Specifies an *roSyncManager* domain to use when executing the `SetMultiscreen()` call. When this method is called, it creates an internal domain that is based on the existing *roSyncManager* domain and uses the same network parameters. We recommend using this method instead of passing a multicast IP address and port to the `SetMultiscreen()` method itself.

SetMultiscreen(offset As Integer, size As Integer, ip_address As String, port As Integer) As Boolean

Displays a smooth scrolling ticker across multiple screens. To specify network synchronization parameters, you can either call the `SetSyncManager()` method first or pass the optional `ip_address` and `port` values to the `SetMultiscreen()` method (calling `SetSyncManager()` is preferred). If you use `SetSyncManager()`, do not specify the `ip_address/port`; doing so may lead to unexpected behavior.

The master screen is designated as the instance with the rightmost offset of all the players in the multiscreen array; all `PushString()` and `Show()` calls (as well as any other changes) must be made on the master instance. Slave instances of the text widget will remain blank until the master starts. This method requires the following parameters:

- `offset`: The offset (in pixels) of the display in the multiscreen array. For example, using an offset of 1920 in a two-screen array of 1920x1080 screens would specify this player as the right-hand (master) display.
- `size`: The total width (in pixels) of the multiscreen array. For example, defining a size of 3840 would specify a two-screen array of 1920x1080 screens.
- `ip_address`: A string specifying the multicast IP address for the PTP synchronization process (e.g. "239.192.0.0")
- `port`: A string specifying the multicast port for the PTP synchronization process (e.g. "1234").

Note

Players can support more than one multiscreen ticker at a time.

`ifWidget`

Show() As Boolean

Displays the widget. After creation, the widget is hidden until `Show()` is called.

Hide() As Boolean

Hides the widget.

SetForegroundColor(color As Integer) As Boolean

Sets the foreground color in ARGB format. The top 8 bits are "alpha" parameters. Zero is equivalent to fully transparent and 255 to fully non-transparent.

Note

Foreground alpha values are currently unsupported with ticker mode 3 (scrolling ticker) and will be ignored by this method. This behavior may change in future firmware versions.

Hex color values should be converted to integers before being passed to this method (e.g. the value `&hFFFFFF` should be passed as 4294967295). You can use the `HexToInteger()` method (available in the [core library extension](#)) to convert a hex string to an integer.

SetBackgroundColor(color As Integer) As Boolean

Sets the background color in ARGB format. The top 8 bits are "alpha" parameters. Zero is equivalent to fully transparent and 255 to fully non-transparent. This feature allows for effects similar to subtitles. For example, you can create a semi-transparent black box containing text over video.

Hex color values should be converted to integers before being passed to this method (e.g. the value `&hFFFFFF` should be passed as 4294967295). You can use the `HexToInteger()` method (available in the [core library extension](#)) to convert a hex string to an integer.

SetFont(font_filename As String) As Boolean

Specifies a custom font for the widget using a TrueType font file from local storage (e.g. "SD:/ComicSans.ttf").

SetBackgroundBitmap(bitmap_filename As String, stretch As Boolean) As Boolean

Sets the background bitmap image. If `stretch` is `True`, then the image is stretched to the size of the widget.

SetBackgroundBitmap(parameters As roAssociativeArray, stretch As Boolean) As Boolean

Sets the background bitmap image. If `stretch` is `True`, then the image is stretched to the size of the widget. The associative array can contain the following parameters:

- `Filename`: The name of the image file
- `EncryptionAlgorithm`: The file-encryption algorithm. Currently the options are "AesCtr" and "AesCtrHmac".
- `EncryptionKey`: The key to decrypt the image file. This is a byte array consisting of 128 bits of key, followed by 128 bits of IV.

Note

See the [Image Decryption](#) section in the *roImagePlayer* entry for details on displaying encrypted images.

SetSafeTextRegion(region As roRectangle) As Boolean

Specifies the rectangle within the widget where the text can be drawn safely.

SetRectangle(r As roRectangle) As Boolean

Changes the size and positioning of the widget rectangle using the passed *roRectangle* object.

GetFailureReason() As String

Yields additional useful information if a function return indicates an error.

roTextWidgetEvent

ON THIS PAGE

- [ifWidgetEvent](#)
 - [GetStringId\(\) As Integer](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\) As Void](#)
 - [GetUserData\(\) As Objecta](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This event object is generated when a string or surface leaves the screen area of an *roTextWidget* instance.

`ifWidgetEvent`

GetStringId() As Integer

Returns the ID of the string or surface that has disappeared from the screen.

`ifUserData`

SetUserData(user_data As Object) As Void

Sets the user data.

GetUserData() As Objecta

Returns the user data that has previously been set via `SetUserData()` (on either the event or source object). It will return `Invalid` if no data has been set.

roTouchScreen

ON THIS PAGE

- [ifTouchScreen](#)
 - [SetResolution\(x As Integer, y As Integer\) As Void](#)
 - [AddRectangleRegion\(x As Integer, y As Integer, w As Integer, h As Integer, region_id As Integer\) As Void](#)
 - [AddCircleRegion\(x As Integer, y As Integer, radius As Integer, region_id As Integer\) As Void](#)
 - [ClearRegions\(\) As Void](#)
 - [GetDeviceName\(\) As String](#)

- `SetCursorPosition(x As Integer, y As Integer) As Void`
- `SetCursorBitmap(filename As String, x As Integer, y As Integer) As Void`
- `EnableCursor(enable As Boolean) As Void`
- `EnableRollover(region_id As Integer, on_image As String, off_image As String, cache_image As Boolean, image_player As Object) As Void`
- `EnableRegion(region_id As Integer, enabled As Boolean) As Void`
- `SetRollOverOrigin(region_id As Integer, x As Integer, y As Integer) As Void`
- `IsMousePresent() As Boolean`
- `SetMouseRotation(rotation As Integer) As Boolean`
- `EnableSerialTouchscreen(a As Integer) As Boolean`
- `SetSerialTouchscreenConfiguration(a As String) As Boolean`
- `GetDiagnosticInfo() As String`
- `ifSetMessagePort`
 - `SetPort(port As roMessagePort)`
- `ifTouchScreenCalibration`
 - `StartCalibration() As Boolean`
 - `GetCalibrationStatus() As Integer`
 - `GetDiagnosticInfo() As String`
 - `ClearStoredCalibration() As Boolean`
 - `StartEventLogging() As Boolean`
 - `StopEventLogging() As Boolean`
 - `ClearEventLogs() As Boolean`
 - `SetCalibrationRanges(x-min As Integer, x-max As Integer, y-min As Integer, y-max As Integer) As Boolean`
- `ifSerialControl`
 - `SetBaudRate(baud_rate As Integer) As Boolean`
 - `NotUsed1(a As String)`
 - `SetMode(a As String) As Boolean`
 - `NotUsed2(a As Boolean) As Boolean`
- `Examples`

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object accepts inputs from touchscreen panels or mice. For each recognized input, the object will generate an *roTouchEvent* object.

Not all touchscreens are supported, but we are always working to extend driver support. Please see this [FAQ](#) for a full list of supported touchscreens, or contact sales@brightsign.biz if you want to know whether a specific touch-screen model is supported. The *roTouchScreen* object responds to the clicks of a USB mouse in the same way it responds to touch events on a touchscreen.

To set up touchscreen/mouse interactivity, follow this outline:

1. Create an *roTouchScreen* instance.
2. Use `SetPort()` to specify an *roMessagePort* instance to receive the *roTouchScreen* events.
3. Define one or more touch regions.
 1. A touch region may be rectangular or circular.
 2. When a touch or click occurs anywhere inside the area of a touch region, an event will be sent to the message port.
 3. If touch regions overlap such that a click or touch hits multiple regions, an event for each affected region will be sent.
4. Process the events.

The *roTouchScreen* object supports rollover regions. Rollovers are based around touch regions. When a rectangular or circular region is added, it defaults to having no rollover. You can use the `EnableRollover()` method to add an *on* and *off* image for a region. Whenever the mouse cursor is within that region, the *on* image is displayed. In all other cases, the *off* image is displayed. This allows buttons to be highlighted as the mouse cursor moves over them.

`ifTouchScreen`

SetResolution(x As Integer, y As Integer) As Void

AddRectangleRegion(x As Integer, y As Integer, w As Integer, h As Integer, region_id As Integer) As Void

Adds a rectangular touch region to the screen. The `region_id` is used to associate the touch region with `roTouchEvent` events and to link the region with rollover images.

AddCircleRegion(x As Integer, y As Integer, radius As Integer, region_id As Integer) As Void

Adds a circular touch region to the screen. The `region_id` is used to associate the touch region with `roTouchEvent` events and to link the region with rollover images.

ClearRegions() As Void

Clears the list of regions added using `AddRectangleRegion()` or `AddCircleRegion()` so that any contacts in those regions no longer generate events. This call has no effect on the rollover graphics.

GetDeviceName() As String

SetCursorPosition(x As Integer, y As Integer) As Void

SetCursorBitmap(filename As String, x As Integer, y As Integer) As Void

Specifies a BMP or PNG file as the mouse cursor icon. This method also accepts a "hot spot" (i.e. the point within the icon rectangle that will trigger events when the mouse is clicked) as a set of `x,y` coordinates. The icon can be a rectangle of any width or height. The colors are specified internally in YUV (6-4-4 bits respectively), but pixels in the passed image file can be one of 16 different colors. These colors are 16 bits, with 14 bits of color and 2 bits of alpha. If you use all of the alpha levels on all shades, then you limit the number of available shades to five (five shades at three alpha levels plus one fully transparent color gives 16).

EnableCursor(enable As Boolean) As Void

Displays a cursor on screen if passed True.

EnableRollover(region_id As Integer, on_image As String, off_image As String, cache_image As Boolean, image_player As Object) As Void

Enables a rollover for a touch region. This method accepts the ID of the touch region, as well as two strings specifying the names of the *on* and *off* bitmap images, a cache setting, and the image player that draws the rollover. The `cache_image` parameter simply tells the script whether to keep the bitmaps loaded in memory or not. This setting uses up memory very quickly, so we recommend that `cache_image` normally be set to 0.

EnableRegion(region_id As Integer, enabled As Boolean) As Void

Enables or disables a rollover region. This method accepts the ID of the touch region, as well as a Boolean value (True or False). The rollover regions default to "enabled" when created, but you can set up all of the regions at the start of your script and then enable regions as required.

SetRollOverOrigin(region_id As Integer, x As Integer, y As Integer) As Void

Changes the origin so that more (or less) of the screen changes when the mouse rolls in and out of the region. This means that bitmaps that are larger than the region can be drawn. The default requirement is that rollover bitmaps be the same size and position as the touch region. Note that the bitmap is square for circular regions. The default origin for circular regions is $[x - r]$, $[y - r]$, where `x, y` is the center of the circle, and `r` is the radius.

IsMousePresent() As Boolean

Returns True if a relative pointing device is attached to the player. This does not work for absolute devices like touchscreens.

SetMouseRotation(rotation As Integer) As Boolean

Transforms mouse-movement inputs to account for screen rotation. This method can accept the following integers:

- 0: Inputs are unchanged (i.e. landscape orientation).
- 1, 90: Rotated 90 degrees (i.e. clockwise portrait orientation).
- 2, 180: Rotated 180 degrees.
- 3, 270: Rotated 270 degrees (i.e. counter-clockwise portrait orientation).

EnableSerialTouchscreen(a As Integer) As Boolean

SetSerialTouchscreenConfiguration(a As String) As Boolean

GetDiagnosticInfo() As String

Returns an HTML string with captured information describing hardware that was connected and events that occurred during the calibration process. This method is used by the calibration script to diagnose touchscreen issues.

ifSetMessagePort

SetPort(port As roMessagePort)

Posts messages of type *roTouchEvent* and *roTouchCalibrationEvent* to the attached message port.

ifTouchScreenCalibration

StartCalibration() As Boolean

GetCalibrationStatus() As Integer

GetDiagnosticInfo() As String

ClearStoredCalibration() As Boolean

StartEventLogging() As Boolean

StopEventLogging() As Boolean

ClearEventLogs() As Boolean

SetCalibrationRanges(x-min As Integer, x-max As Integer, y-min As Integer, y-max As Integer) As Boolean

Overrides the screen range values provided by the touchscreen. This method is useful when the entirety of the video output is not being displayed on the touch surface. Practical use of this method usually requires a custom calibration script, appropriate images, and a calibration setting matched to a particular setup.

ifSerialControl

SetBaudRate(baud_rate As Integer) As Boolean

Sets the baud rate of the device. The supported baud rates are as follows: 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400.

NotUsed1(a As String)

SetMode(a As String) As Boolean

NotUsed2(a As Boolean) As Boolean

Examples

This script loops a video and waits for a mouse click or touchscreen input. It outputs the coordinates of the click or touch to the shell if it is located within the defined region.

```
v=CreateObject("roVideoPlayer")
t=CreateObject("roTouchScreen")
p=CreateObject("roMessagePort")

v.SetPort(p)
```

```

t.SetPort(p)
v.SetLoopMode(True)
v.PlayFile("testclip.mp2v")

t.AddRectangleRegion(0,0,100,100,2)

loop:
  msg=Wait(0, p)
  print "type: ";type(msg)
  print "msg=";msg
  if type(msg)="roTouchEvent" then
    print "x,y=";msg.GetX();msg.GetY()
  endif
  goto loop:

```

This script includes mouse support.

```

t=CreateObject("roTouchScreen")
t.SetPort(p)
REM Puts up a cursor if a mouse is attached
REM The cursor must be a 16 x 16 BMP
REM The x,y position is the "hot spot" point
t.SetCursorBitmap("cursor.bmp", 16, 16)
t.SetResolution(1024, 768)
t.SetCursorPosition(512, 389)
REM
REM Pass enable cursor display: TRUE for on, and FALSE for off
REM The cursor will only enable if there is a mouse attached
REM
t.EnableCursor(TRUE)

```

This script includes a rollover region and mouse support.

```

img=CreateObject("roImagePlayer")
t=CreateObject("roTouchScreen")
p=CreateObject("roMessagePort")
t.SetPort(p)

t.SetCursorBitmap("cursor.bmp", 16, 16)
t.SetResolution(1024, 768)
t.SetCursorPosition(512, 389)
t.EnableCursor(1)

img.DisplayFile("\menu.bmp")

REM Adds a rectangular touch region
REM Enables rollover support for that region
REM Sets the rollover origin to the same position as the touch region REM
t.AddRectangleRegion(0, 0, 100, 100, 1)
t.EnableRollOver(1, "on.bmp", "off.bmp", true, img)
t.SetRollOverOrigin(1, 0, 0)

```

roTouchEvent, roTouchCalibrationEvent

ON THIS PAGE

- [ifInt](#)
 - [GetInt\(\) As Integer](#)
 - [SetInt\(a As Integer\)](#)
- [ifPoint](#)
 - [GetX\(\) As Integer](#)
 - [GetY\(\) As Integer](#)
 - [SetX\(a As Integer\)](#)
 - [SetY\(a As Integer\)](#)
- [ifEvent](#)
 - [GetEvent\(\) As Integer](#)
 - [SetEvent\(a As Integer\)](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

The *roTouchEvent* object is generated by the *roTouchScreen* object whenever a touch or mouse event is detected within a defined region.

`ifInt`

GetInt() As Integer

Retrieves the region ID of the event.

SetInt(a As Integer)

Sets the region ID of the event.

`ifPoint`

The *ifPoint* interface is not available on the *roTouchCalibrationEvent* object.

GetX() As Integer

Retrieves the x coordinate of the mouse/touch event.

GetY() As Integer

Retrieves the y coordinate of the mouse/touch event.

SetX(a As Integer)

Sets the x coordinate of the event.

SetY(a As Integer)

Sets the y coordinate of the event.

`ifEvent`

The *ifEvent* interface is not available on the *roTouchCalibrationEvent* object.

GetEvent() As Integer

SetEvent(a As Integer)

roVideoEvent, roAudioEvent

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

Video and audio events are declared as separate classes. Events can have one of the following values, which are retrieved using the `GetInt()` method:

3	Playing	The current media item has started playing.
8	MediaEnded	The media item has completed playback.
12	TimeHit	A particular timecode has been reached. See the entry on Timecode Events for more details.
13	Overlay_Playing	An <i>roAudioPlayerMx</i> instance has begun playback of an audio file.
14	Overlay_MediaEnded	An <i>roAudioPlayerMx</i> instance has completed playback of an audio file.
15	Overlay_TimeHit	The EventTimeStamp of an <i>roAudioPlayerMx</i> instance has been reached.
16	MediaError	A media error has been detected. As opposed to code 30, this event usually indicates transient errors (e.g. low buffer levels).
17	Overlay_MediaError	A media error has been detected during <i>roAudioPlayerMx</i> playback.
18	FadingOut	The current media item has completed fading out. See the <i>roVideoPlayer.SetFade()</i> entry for more details.
19	DecoderEOS	
20	Overlay_FadingOut	The FadeOutLocation of an <i>roAudioPlayerMx</i> instance has been reached.
21	Overlay_DecoderEOS	
26	Underflow	The stream seems to be underflowing. This event usually indicates that the streaming latency is set too low. It will be generated every few seconds as long as underflow is detected.
30	MediaError_NotStarted	A fatal error has been encountered while attempting to start playback (e.g. the video format is not supported).

Note

The [Diagnostic Web Server log](#) may return more specific information about media error states.

`ifInt`

GetInt() As Integer

Returns the event ID as an integer value.

SetInt(value As Integer) As Void

Sets the integer value.

ifUserData

SetUserData(user_data As Object) As Void

Sets the user data.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()` (on either the event or source object). It will return `Invalid` if no data has been set.

ifSourceIdentity

GetSourceIdentity() As Integer***SetSourceIdentity() As Integer*****Note**

The *ifSourceIdentity* interface has been deprecated. We recommend using the *ifUserData* interface instead.

ifData

GetData() As Integer***SetData(a As Integer)***

Example

```
vp_msg_loop:
  msg=Wait(tiut, p)
  if type(msg)="roVideoEvent" then
    if debug then print "Video Event";msg.GetInt()
    if msg.GetInt() = 8 then
      if debug then print "VideoFinished"
      retcode=5
      return
    endif
  else if type(msg)="roGpioButton" then
    if debug then print "Button Press";msg
    if escm and msg=BM then retcode=1:return
    if esc1 and msg=B1 then retcode=2:return
    if esc2 and msg=B2 then retcode=3:return
    if esc3 and msg=B3 then retcode=4:return
  else if type(msg)="rotINT32" then
    if debug then print "TimeOut"
    retcode=6
    return
  endif
  goto vp_msg_loop
```

roVideoInput

ON THIS PAGE

- [ifVideoInput](#)
 - [GetStandards\(\) As roArray](#)
 - [GetInputs\(\) As roArray](#)
 - [SetStandard\(standard As String\) As Boolean](#)
 - [GetCurrentStandard\(\) As String](#)
 - [SetInput\(input As String\) As Boolean](#)
 - [GetCurrentInput\(\) As String](#)
 - [GetControls\(\) As roArray](#)
 - [SetControlValue\(control_name As String, value As Integer\) As Boolean](#)
 - [GetCurrentControlValue\(control_name As String\) As roAssociativeArray](#)
 - [GetFormats\(\) As Object](#)
 - [SetFormat\(a As String, b As Integer, c As Integer\) As Boolean](#)
 - [GetCurrentFormat\(\) As String](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object allows playback of HDMI input or video provided by a video capture dongle. Note that the *ifVideoInput* methods do not apply to HDMI input, which can be achieved by passing an unmodified *roVideoInput* instance to the *roVideoPlayer.PlayFile()* method (see below for examples).

Object Creation: *roVideoInput* is created with no parameters:

```
CreateObject("roVideoInput")
```

ifVideoInput

GetStandards() As roArray

GetInputs() As roArray

These return an array of strings describing the various inputs and video standards that the video capture device supports. The following are the possible standards that can be returned: PAL-D/K, PAL-G, PAL-H, PAL-I, PAL-D, PAL-D1, PAL-K, PAL-M, PAL-N, PAL-Nc, PAL-60, SECAM-B/G, ECAM-B, SECAM-D, SECAM-G, SECAM-H, SECAM-K, SECAM-K1, SECAM-L, SECAM-LC, SECAM-D/K, NTSC-M, NTSC-Mj, NTSC-443, NTSC-Mk, PAL-B and PAL-B1. Inputs returned are s-video and composite.

SetStandard(standard As String) As Boolean

GetCurrentStandard() As String

SetInput(input As String) As Boolean

GetCurrentInput() As String

Use the above to get and set the input and video standard.

GetControls() As roArray

Returns the possible controls on the input. These include "Brightness," "Contrast," "Saturation," "Hue," and others.

SetControlValue(control_name As String, value As Integer) As Boolean

Sets the value of the specified control.

GetCurrentControlValue(control_name As String) As roAssociativeArray

Returns an associative array with 3 members: "Value," "Minimum," and "Maximum." "Value" is the current value, and the possible range is specified by "Minimum" and "Maximum."

GetFormats() As Object

SetFormat(a As String, b As Integer, c As Integer) As Boolean

GetCurrentFormat() As String

This script uses the HDMI Input as the video source to create a full-screen display.

```
v = CreateObject("roVideoPlayer")
i = CreateObject("roVideoInput")
p = CreateObject("roMessagePort")

vm = CreateObject("roVideoMode")
vm.SetMode("1920x1080x60p")

r = CreateObject("roRectangle", 0, 0, 1920, 1080)
v.SetRectangle(r)

v.PlayFile(i)
```

This script uses the video capture dongle as the video source to create a full-screen display.

```
v=CreateObject("roVideoPlayer")
i=CreateObject("roVideoInput")
p=CreateObject("roMessagePort")

vm=CreateObject("roVideoMode")
vm.SetMode("1280x720x60p")

r = CreateObject("roRectangle", 0, 0, 1280, 720)
v.SetRectangle(r)

i.SetInput("s-video")
i.SetStandard("ntsc-m")

v.PlayFile(i)
```

roVideoMode

ON THIS PAGE

- **ifVideoMode**
 - **SetMode(mode As String) As Boolean**
 - **SetModeForNextBoot(video_mode As String) As Boolean**
 - **GetModeForNextBoot() As String**
 - **GetBestMode(connector As String) As String**
 - **GetMode() As String**
 - **GetAvailableModes() As Array**
 - **GetActiveMode() As AssociativeArray**
 - **GetConfiguredMode() As AssociativeArray**
 - **GetFPS() As Integer**
 - **SetDecoderMode(decoder As String, timeslice_mode As String, z_order As Integer, friendly_name As String, enable_mosaic_deinterlacer As Boolean) As Boolean**
 - **GetDecoderModes() As roArray**
 - **Set3dMode(mode As Integer) As Boolean**
 - **Screenshot(parameters As roAssociativeArray) As Boolean**
 - **GetResX() As Integer**
 - **GetResY() As Integer**
 - **GetVideoResX() As Integer**
 - **GetVideoResY() As Integer**
 - **GetOutputResX() As Integer**
 - **GetOutputResY() As Integer**
 - **GetSafeY() As Integer**
 - **GetSafeWidth() As Integer**
 - **GetSafeHeight() As Integer**
 - **SetGraphicsZOrder(order As String) As Boolean**
 - **PauseGraphics(timeout_in_ms As Integer) As Boolean**
 - **ResumeGraphics() As Boolean**
 - **SetImageSizeThreshold(parameters As roAssociativeArray) As Boolean**
 - **AdjustGraphicsColor(parameters As roAssociativeArray) As Boolean**
 - **ConfigureHdmiInput(parameters As roAssociativeArray) As Boolean**
 - **GetHdmiOutputStatus() As roAssociativeArray**
 - **GetHdmiInputStatus() As roAssociativeArray**
 - **GetCompositorCRC() As Integer**
 - **GetTxHDCPStatus() As roAssociativeArray**
 - **ForceHDCPOn(force As Boolean) As Boolean**
 - **DisableHDCPRepeater(disable As Boolean) As Boolean**
 - **SetBackgroundColor(color As Integer) As Boolean**
 - **SetPowerSaveMode(power_save_enable As Boolean) As Boolean**
 - **IsAttached(connector As String) As Boolean**
 - **HdmiAudioDisable(disable As Boolean) As Boolean**
 - **SetMultiscreenBezel(x_pct As Integer, y_pct As Integer) As Boolean**
 - **SaveEdids(filename As String) As Boolean**
 - **GetEdidIdentity(video_connector As Boolean) As roAssociativeArray**
 - **SetMpcdi(parameters As roAssociativeArray) As Boolean**
 - **SetSyncDomain(domain As String) As Boolean**
- **ifMessagePort**
 - **SetPort(port As Object) As Void**
- **ifUserData**
 - **SetUserData(user_data As Object)**
 - **GetUserData() As Object**
- "Auto" Video Mode
- **Selecting Decoders for Playback**

▼ Firmware Version 7.0

- **Version 7.0**
- **Version 6.2**

- [Version 6.1](#)
- [Previous Versions](#)

This object allows you to configure resolution and other video output settings. The same video resolution is applied to all video outputs on a BrightSign player. Video or images that are subsequently decoded and displayed will be scaled (using the hardware scalar) to this output resolution if necessary.

Object Creation: The *roVideoMode* object is created with no parameters.

```
CreateObject("roVideoMode")
```

The *roVideoMode* object generates *roHdmiInputChanged* and *roHdmiOutputChanged* event objects whenever the hotplug status of the HDMI input or output changes.

ifVideoMode

SetMode(mode As String) As Boolean

Sets the video output mode. See [here](#) for a list of supported video modes. If the specified video mode is different from the current video mode of the object, the unit will reboot and change the video mode to the new setting during system initialization. This method also accepts "auto" as a mode parameter. The following optional parameters can be appended to the string

- `<resolution>:<color_space>:<depth>bit`: Sets the [video profile](#) for HDMI output. For example, to output 4Kp60 at the 4:2:0 color space with 10 bits of depth, you would pass the following string: "3840x2160x60p:420:10bit".
- `<resolution>:preferred`: Enables the preferred video mode setting. This instructs the player to only use the video mode if the display EDID indicates that it is supported. Otherwise, the output will default to "auto" mode. If no EDID is detected at bootup, the player will output the preferred video mode. If an HDMI hotplug event occurs afterward, then the player will perform the preferred video-mode check again. The `:preferred` flag currently ignores video profile settings (i.e. color space and bit depth).
- `<resolution>:fullres`: Enables full-resolution graphics. This instructs the player to match the graphics plane to the video mode; otherwise, video modes larger than 1920x1200 upscale the graphics plane to match the video output. See [here](#) for more details and a list of supported modes/models.

Note

If the `fullres` mode is used with graphics-intensive HTML applications, we recommend enabling the `gfxmemlarge` setting as well.

- `<resolution>:gfxmemlarge`: Enables the large graphics memory configuration for XTx44 and XTx43 models (at the expense of general-purpose memory). This setting requires a reboot to take effect.
- `<resolution>:gfxmemdefault`: Resets the graphics memory configuration to default. This setting requires a reboot to take effect.
- `<resolution>:dbv`: Enables Dolby Vision on video output (XTx44 only). Only one video decoder is available when Dolby Vision is enabled.
- `<resolution>:rgb:<range>`: Sets the color space to RGB. The optional `<range>` parameter can be one of the following values:
 - `auto`: The default setting. Over HDMI, the player will output RGB Full for VESA modes and RGB Limited for TV modes. Over DVI, the player will output RGB Full for all modes.
 - `fullrange`: The RGB Full (0-255) setting
 - `limitedrange`: The RGB Limited (16-235) setting

BrightSign hardware has a video anti-aliasing low-pass filter that is set automatically.

SetModeForNextBoot(video_mode As String) As Boolean

Specifies the target video mode of the device the next time it reboots. Once a video mode is specified using `SetMode()`, it can only be changed by a device reboot.

GetModeForNextBoot() As String

Returns the target video mode of the device the next time it reboots. The return value is specified with the `SetModeForNextBoot()` method.

GetBestMode(connector As String) As String

Returns the highest supported [video mode](#), as reported by the display via EDID. The video connector can be specified as "hdmi" or "vga" (these values are case sensitive). If the display does not return a resolution value over EDID (or if no display is connected), this method returns a blank string.

GetMode() As String

Returns the current video mode of the device, which is specified using the `SetMode()` method.

GetAvailableModes() As Array

Returns all video modes supported by the player as an array of entries. Each entry is an associative array with the following values:

- `videomode:(string)` A description of the video mode (a full list of modes can be found [here](#))
- `width:(integer)` The width of the video output
- `height:(integer)` The height of the video output
- `graphicsPlaneWidth:(integer)` The width of the graphics plane
- `graphicsPlaneHeight:(integer)` The height of the graphics plane

Note

In 4K modes, the graphics plane may be smaller than the video plane and output. In these cases, the graphics plane is upscaled to match the video plane/output.

- `framerate:(integer)` The frame rate of the video output
- `interlaced:(Boolean)` A flag indicating whether the video output is interlaced (`true`) or progressive (`false`)
- `overscan:(Boolean)` A flag indicating whether the video output is using an overscan setting or not
- `colorspace:(string)` The color space of the video signal ("8bit", "10bit", or "12bit")
- `colordepth:(string)` The color depth of the video signal ("rgb", "yuv420", or "yuv422")
- `dropframe:(Boolean)` A flag indicating whether the video timecode utilizes drop frames
- `preferred:(Boolean)` A flag indicating whether the video mode is the preferred mode, which is configured using the `SetMode()` method

GetActiveMode() As AssociativeArray

Returns information about the current video mode as an associative array. All values in the `roAssociativeArray` are strings:

- `videomode`: The current video mode (e.g. "3840x2160x60p")
- `colordepth`: The current color depth ("8bit", "10bit", or "12bit")
- `colorspace`: The current color space ("rgb", "yuv420", or "yuv422")
- `preferred`: A "true" or "false" string indicating whether the current video mode is the preferred mode, which is set using the `SetMode()` method.

GetConfiguredMode() As AssociativeArray

Returns information about the video mode configured using the `SetMode()` method. This method returns an `roAssociativeArray`; see the `GetAvailableModes()` entry for more details about returned values. If the video mode is set to "auto", this method will return `Invalid`.

GetFPS() As Integer

Returns the current framerate of the video output.

SetDecoderMode(decoder As String, timeslice_mode As String, z_order As Integer, friendly_name As String, enable_mosaic_deinterlacer As Boolean) As Boolean

Configures a video decoder for either standard mode or Mosaic mode. In standard mode, a single decoder is used to play a single video; in Mosaic mode, the decoder can be used to decode multiple videos from different local or remote sources. See the [Selecting Decoders](#) section below for more information on assigning video players to decoders in HTML or BrightScript.

- `decoder`: The video decoder to be used (decoder availability differs by model).
 - "4K1": The primary 4K decoder (XTx44 models only)
 - "4K2": The secondary 4K decoder (XTx44 models only)
 - "4K": The sole 4K decoder (XTx43, 4Kx42, XDx34, and XDx33 models only)
 - "HD1": The primary HD decoder
 - "HD2": The secondary HD decoder
- `timeslice_mode`: The maximum resolution that the decoder will accept (at framerates up to 60p). If this resolution is the same as the decoder's maximum resolution limit, the decoder will use standard mode, not Mosaic mode.
 - "4K": 4096x2160
 - "HD": 1920x1080
 - "SD": 720x576
 - "CIF": 352x288

- "QCIF": 176x144

Important

Upscaling videos in Mosaic mode currently causes severe performance degradation.

- `z_order`: The z-order of the video window (in standard mode) or group of video windows (in Mosaic mode) relative to the graphics plane:
 - 1: The video window (or group of windows) is positioned in front of the graphics plane.
 - -1: The video window (or group of windows) is positioned behind the graphics plane.

Tip

This configuration can be modified using the `SetGraphicsZOrder()` method.

- `friendly_name`: A human-readable name for referencing the decoder in HTML and scripts
- `enable_mosaic_deinterlacer`: A Boolean value indicating whether Mosaic-mode videos can be interlaced or not. Enabling the deinterlacer will allow playback of interlaced videos in Mosaic mode, but will reduce the number of Mosaic-mode videos that can be decoded simultaneously as well.

GetDecoderModes() As roArray

Returns an array of associative arrays, each one corresponding to a single decoder. Each associative array contains the following entries:

- `decoder_name`: The system name of the decoder
- `friendly_name`: The name of the decoder as specified when calling `SetDecoderMode()`
- `max_decode_size`: The maximum resolution of the decoder, as set by system software. This value can be either "4K" or "HD"
- `configured_decode_size`: The maximum resolution of the decoder that is specified when calling `SetDecoderMode()`
- `mode`: The current mode of the decoder, which can be either "Regular" or "Mosaic"
- `usage_count`: The number of videos currently being decoded by the decoder
- `max_usage`: The maximum number of videos that can be decoded simultaneously by the decoder. The optimum `max_usage` limits are described below; the limit may be lower depending on a number of factors, including interlacing and frame rate.
 - 4K decoder:
 - 1 4K video
 - 2 HD videos
 - 4 SD videos
 - 8 CIF videos
 - 10 QCIF videos
 - HD decoder:
 - 0 4K videos
 - 1 HD video
 - 3 SD videos
 - 4 CIF videos
 - 5 QCIF videos
- `mosaic_mode_interlace`: The current deinterlacing mode of the decoder, which can be either "Enabled" or "Disabled". This value is specified when calling `SetDecoderMode()`.

Set3dMode(mode As Integer) As Boolean

Sets the 3D video output mode, which is specified by passing one the following parameters:

- 0: Standard mono video (default)
- 1: Side-by-side stereo video
- 2: Top-and-bottom stereo video

Screenshot(parameters As roAssociativeArray) As Boolean

Captures a screenshot of the video and graphics layer as a `.jpeg` or `.bmp` file. The screenshot is configured by passing an associative array of parameters to the method:

- `filename`: A string specifying the name and path of the image file that will be saved (e.g. "SD:/myscreenshots/screen.jpg"). If the specified directory does not exist, the screenshot will fail.
- `width`: An integer specifying the width of the image file.
- `height`: An integer specifying the height of the image file.

Note

The default dimensions of the image file is 640x480.

- `filetype`: A string determining whether the image is a "JPEG" or "BMP" file type. Note that the file extension (".jpg" or ".bmp") is not appended to the filename by default and, if needed, should be included in the `filename` string.
- `quality`: An integer value (between 0 and 100) that determines the image quality of the screenshot. This parameter is set to 50 by default.
- `async`: An integer value that determines whether the screenshot should be taken synchronously or asynchronously. If set to 0, the function returns True after the image file has successfully finished writing. If set to 1, the function will return True prior to saving the file, then return an `roScreenShotComplete` event once the file has finished writing.
- `rotation`: The rotation of the screenshot image (in degrees). The default value is 0. Accepted values are 0, 90, 180, and 270.

GetResX() As Integer

Returns the current width of the graphics plane.

GetResY() As Integer

Returns the current height of the graphics plane.

GetVideoResX() As Integer

Returns the current width of the video plane.

GetVideoResY() As Integer

Returns the current height of the video plane.

GetOutputResX() As Integer

Returns the width of the display for the current video mode.

GetOutputResY() As Integer

Returns the height of the display for the current video mode.

Note

In most cases, the values returned by `GetResX()/GetResY()`, `GetVideoResX()/GetVideoResY()`, and `GetOutputResX()/GetOutputResY()` will be identical. The "GetRes" and "GetVideoRes" values will sometimes differ when using 4K modes if the graphics plane (i.e. "GetRes") remains at 1920x1080, while the video plane (i.e. "VideoRes") expands (e.g. to 3840x2160). The "OutputRes" value will differ from the other two values when the video output is upscaled: For example, when upscaling from HD to 4K, the "GetRes" and "GetVideoRes" values will remain at 1920x1080, while the "OutputRes" values will indicate 3840x2160.

GetSafeX() As Integer

Returns the horizontal coordinate for the upper-left corner of the "safe area". For modes that are generally displayed with no overscan, this will be zero.

GetSafeY() As Integer

Returns the vertical coordinate for the upper-left corner of the "safe area". For modes that are generally displayed with no overscan, this will be zero.

GetSafeWidth() As Integer

Returns the width of the "safe area." For modes that are generally displayed with no overscan, this will return the same as `GetResX`.

GetSafeHeight() As Integer

Returns the height of the "safe area." For modes that are generally displayed with no overscan, this will return the same as `GetResY`.

More information about safe areas can be found here:

- http://en.wikipedia.org/wiki/Safe_area

- http://en.wikipedia.org/wiki/Overscan_amounts

SetGraphicsZOrder(order As String) As Boolean

Specifies the order of the graphics plane (which includes all graphical elements) in relation to the video plane(s). This method accepts three parameters:

- "front": Places the graphics plane in front of the video plane(s).
- "middle": Places the graphics plane between two video planes. This option is only applicable for models that have two video decoders (e.g. XTx44, XDx34).
- "back": Places the graphics plane behind the video plane(s).

If the player is rendering two videos, the `front` and `back` options will always place the graphics plane in front of or behind both video planes. To determine the z-order of video planes in relation to one another, use the `ToFront()` and `ToBack()` methods provided by the `roVideoPlayer` object. The following table shows all possible video and graphics z-order arrangements that can be specified using the `SetGraphicsZOrder()` method and calling the `ToFront()` and `ToBack()` methods on a "Video1" `roVideoPlayer` instance.

SetGraphicsZOrder()	front		middle		back	
ToFront()/ToBack()	ToFront()	ToBack()	ToFront()	ToBack()	ToFront()	ToBack()
Z-Order	Graphics	Graphics	Video1	Video2	Video1	Video2
	Video1	Video2	Graphics	Graphics	Video2	Video1
	Video2	Video1	Video2	Video1	Graphics	Graphics

PauseGraphics(timeout_in_ms As Integer) As Boolean

Suspends graphics compositor updates for the specified number of milliseconds (or until the `ResumeGraphics()` method is called), up to a maximum interval of 10 seconds. While the graphics compositor is paused, no visual elements will be updated (except for **HWZ video**, **scrolling tickers**, and off-screen Chromium textures). Use this method to combine `Show()`, `Hide()`, and `SetRectangle()` operations into a single v-sync update.

ResumeGraphics() As Boolean

Resumes the graphics compositor if it has been paused with the `PauseGraphics()` method.

SetImageSizeThreshold(parameters As roAssociativeArray) As Boolean

Changes the maximum allowed size for images. Images sizes are measured in bytes using the following formula: $\text{image_width} * \text{image_height} * 4$.

The default image size limit is 35389440 bytes (equivalent to 4096x2160x32bpp) for XTx44, XTx43, XDx34, XDx33, and 4Kx42 models and 2621440 bytes (equivalent to 2048x1280x32bpp) for other models. Displaying images larger than the default value may deplete the graphics memory and cause a crash, so we recommend testing a script that uses this method thoroughly before deploying it in a production environment. This method accepts an associative array with the following parameters:

- [int] `width`: The image width value
- [int] `height`: The image height value
- [int] `ignore`: A flag specifying whether the image size limit is enabled (0) or disabled (1)

Tip

You can also increase the default maximum width/height by reducing the bpp value (e.g. using a 4096x1280x16bpp on non-4K-capable players is allowed); however, this method is not applicable to HTML pages because Chromium always decodes images to RGBA 8888.

AdjustGraphicsColor(parameters As roAssociativeArray) As Boolean

Adjusts the video and graphics output of the player using the following parameters, which can be passed to the method as an associative array: "brightness", "hue", "contrast", "saturation". Each parameter has a default value of 0 and can accept a range of values between -1000 and 1000.

ConfigureHdmiInput(parameters As roAssociativeArray) As Boolean

Configures EDID reporting for the HDMI input. By default, the input EDID includes video modes from the display attached to the HDMI output, some video modes supported by the player, and support for PCM audio up to 48kHz; it does not report proprietary media codecs that can be

decoded by the device connected to the HDMI output, so you can use this method to announce such support if available at the endpoint. Using this method to change the default configuration will trigger a reboot on the player. The passed associative array can contain the following parameters:

- [int] MaxSampleRate: The maximum supported PCM audio sampling rate in Hz (e.g. the default sampling rate is 48000)
- [int] MaxChannelCount: The number of PCM channels that are advertised over EDID. The default value is 2, which allows for stereo mixdown. Increasing this value to 6 allows the source to send multichannel PCM.

Note

If AC-3 or E-AC-3 is enabled on the player, multichannel audio is supported regardless of the MaxChannelCount setting.

- [int] EnableAC3: A flag specifying whether AC-3 is not supported (0) or supported (1)
- [int] EnableEAC3: A flag specifying whether E-AC-3 is not supported (0) or supported (1)
- [int] EnableTrueHDMlp: A flag specifying whether TrueHD MLP is not supported (0) or supported (1)
- [int] EnableDTS: A flag specifying whether DTS is not supported (0) or supported (1)
- [int] EnableDTSHD: A flag specifying whether DTS-HD is not supported (0) or supported (1)
- [int] LockAudioToVideoClock: A flag specifying whether the audio sample rate clock is locked to the audio clock (0) or video clock (1) of the incoming HDMI signal.

GetHdmiOutputStatus() As roAssociativeArray

Returns an associative array of Boolean and integer values if an HDMI output is currently connected to a display device. This method will return Invalid if the HDMI output is currently not connected to a display device. The associative array contains the following parameters:

- [bool] output_present: A flag indicating whether the HDMI output is connected to a display device
- [bool] output_powered: A flag indicating whether the display device is on (i.e. RX powered)
- [string] EOTF: A string indicating the current electro-optical transfer function (EOTF) used by the display. The following are possible values:
 - "HDR (GAMMA)"
 - "SDR (GAMMA)"
 - "SMPTE 2084 (PQ)"
 - "Future (BBC/NHK)"
 - "unspecified"
- [int] audio_bits_per_sample: The number of bits per audio sample
- [string] audio_format: The format of the audio output. A "PCM" value indicates that the player is sending decoded output.
- [int] audio_channel_count: The number of audio channels in the output
- [int] audio_sample_rate: The audio sample rate (in hertz)

Note

If an HDR video is playing, the following values will be retrieved from the video file: "max_cli", "max_fall", "red_primary_x", "red_primary_y", "green_primary_x", "green_primary_y", "blue_primary_x", "blue_primary_y", "white_point_x", "white_point_y", "min_mastering_luminance", "max_mastering_luminance".

GetHdmiInputStatus() As roAssociativeArray

Returns an associative array of Boolean and integer values if an HDMI input is currently connected to the device. This method will return Invalid if there is currently no HDMI input source. The associative array contains the following parameters:

- [Boolean] device_present: A flag indicating whether an HDMI input source is present
- [int] width: The width of the source video
- [int] height: The height of the source video
- [Boolean] interlaced: A flag indicating whether the video source is interlaced
- [float] frame_rate: The framerate of the source video
- [float] pixelClock: The pixel-clock rate of the source video (in MHz)
- [string] colorspace: The color space of the source video
- [string] audio_type: The audio encoding of the source video
- [int] audio_sampling_rate: The audio sampling rate of the source video (in Hz)

GetCompositorCRC() As Integer

Returns the CRC of the Y and Cb signals as a single integer.

***GetTxHDCPStatus()* As roAssociativeArray**

Returns an associative array indicating the current HDCP status of the HDMI output. The associative array currently contains a single key labeled `state`, which can have the following values:

- "not-required": HDCP is not required by the player. HDCP is required by the player if the video has been decoded locally and needs protection or if the script has called the `forceHdcpOn()` method. Note that, even if the "not-required" value is returned, HDCP might still be active in passthrough mode if an upstream HDMI source (i.e. a device connected to the HDMI input port on the player) has requested it.
- "authenticated": HDCP has been enabled and successfully negotiated.
- "authentication-in-progress": HDCP has been enabled, but authentication has not been completed.
- "authentication-failed": HDCP has been requested but could not be negotiated.

ForceHDCPOn(force As Boolean) As Boolean

Forces HDCP authentication on the HDMI output if passed True. Passing False to this method will prevent forced authentication attempts with subsequent hotplug events. This method will return False if the player does not support HDCP or if `ForceHDCPOn()` has already been called with the same value.

DisableHDCPRepeater(disable As Boolean) As Boolean

Prevents HDCP authentication from taking place on the HDMI input if passed True. The HDMI source will treat the player like any other non-HDCP authenticated HDMI sink. This method returns False if the HDCP state could not be changed, indicating that there's no HDMI input on the player or that HDCP has already been disabled.

SetBackgroundColor(color As Integer) As Boolean

Specifies the background color using an `#rrggbb` hex value (8 bits for each color).

SetPowerSaveMode(power_save_enable As Boolean) As Boolean

Disables HDMI output, the syncs for VGA output, and the DAC output for component video. The absence of a signal will cause some monitors to go into standby mode.

IsAttached(connector As String) As Boolean

Returns True if the specified video connector is attached to an output device (i.e. the display EDID can be read successfully). This method can be passed the following parameters (note that they are case sensitive):

- "hdmi"
- "vga"

HdmiAudioDisable(disable As Boolean) As Boolean

Disables audio output if True. This method is set to False by default.

SetMultiscreenBezel(x_pct As Integer, y_pct As Integer) As Boolean

Adjusts the size of the bezel used in calculations for multiscreen displays, allowing for users to compensate for the width of their screen bezels. The calculations for the percentages are as follows:

```
x_percentage = (width_of_bezel_between_active_screens / width_of_active_screen) * 100
```

```
y_percentage = (height_of_bezel_between_active_screens / height_of_active_screen) * 100
```

The bezel measurement is therefore the total of the top and bottom bezels in the y case, or the left and right bezels in the x case. When this value is set correctly, images spread across multiple screens take account of the bezel widths, leading to better alignment of images.

SaveEdids(filename As String) As Boolean

Saves the EDID information of the display(s) connected via HDMI and/or VGA. The EDID fields are saved sequentially as raw binaries into the specified file. The EDID sets are two 2kb each, resulting in a maximum file size of 4kb. This method returns True upon success and False upon failure.

GetEdidIdentity(video_connector As Boolean) As roAssociativeArray

Returns an associative array with EDID information from a compatible monitor/television. Passing True with this method specifies EDID over HDMI, while passing False specifies EDID over VGA. These are the possible parameters returned in the associative array:

- `serial_number_string`

- year_of_manufacture
- monitor_name
- manufacturer
- text_string
- serial_number
- product
- week_of_manufacture

The system will generate an *roHdmiEdidChanged* event when an HDMI cable is hotplugged and the EDID information changes. Calling `GetEdidIdentity(true)` at this point retrieves the new EDID information.

SetMpcdi(parameters As roAssociativeArray) As Boolean

Enables MPCDI using the passed parameters. See the [MPCDI tech note](#) for more information about configuring MPCDI on BrightSign players.

SetSyncDomain(domain As String) As Boolean

Enables Genlock synchronization on the specified *roSyncManager* domain. To disable Genlock on the domain, pass an empty string to this method. To reconfigure an active Genlock synchronization, call `SetSyncDomain()` again using the domain name of the new or edited *roSyncManager* instance.

`ifMessagePort`

SetPort(port As Object) As Void

Posts events to the attached message port

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

"Auto" Video Mode

If the mode is set to "auto," the BrightSign player will use the following algorithm to determine the best video mode to use based on connected hardware:

1. Try VGA: If VGA is attached, use the highest-resolution mode (as reported by the monitor) that the player supports.
2. Try HDMI: If HDMI is attached, use the highest-resolution mode (as reported by the monitor) that the player supports.
3. Default to 640x480x60p.
4. If an HDMI hotplug event occurs at any point, recheck the monitor EDID to determine if the highest-resolution mode has changed. If it has changed, reboot the player and use the new video mode.

Selecting Decoders for Playback

The system software selects which video decoder to use based on the resolution probed from the video file. In standard mode, it will attempt to select the decoder that has the closest maximum supported resolution (i.e. 1920x1080 for the HD decoder and 3840x2160 for the 4K decoder), without exceeding that maximum resolution. If a decoder has been configured for Mosaic mode, it will match the video resolution against the specified `timeslice_mode` instead. If both decoders support the same maximum resolution, you can select a decoder by matching the z-order of the *roVideoPlayer* instance (set using the `ToFront()` and `ToBack()` methods) with the z-order of the decoder (set using the *roVideoMode.SetDecoderMode()* method).

You can also select the decoder manually. First, configure the decoder(s) using the *roVideoMode.SetDecoderMode()* method. Then, use the `friendly_name` specified when calling the method to designate a decoder to use for video playback.

To select a decoder in BrightScript, pass an associative array to the *roVideoPlayer.PlayFile()* method containing the `decoder`: `[friendly_name]` parameter:

```
PlayFile({filename:"text_1.mov", decoder:"main-video"})
```

To select a decoder for HTML video, include the `decoder:[friendly_name]` property with the `hwz` attribute:

```
<video hwz="decoder:main-video;" > </video>
<video hwz="decoder:sd-video;" > </video>
```

The `max_usage` of a decoder determines how many video players can be assigned to the decoder using the system software algorithm described above—video players beyond the `max_usage` limit may be assigned to another decoder or not displayed at all. On the other hand, if you manually assign video players using the `friendly_name` of the decoder, you can assign more video players to the decoder than the `max_usage` limit, but this may cause unpredictable video-display behavior.

roVideoPlayer

ON THIS PAGE

- **ifVideoControl**
 - `PlayStaticImage(filename As String) As Boolean`
 - `PlayStaticImage(parameters As roAssociativeArray) As Boolean`
 - `SetViewMode(mode As Integer) As Boolean`
 - `SetRectangle(r As roRectangle) As Void`
 - `Hide() As Boolean`
 - `Show() As Boolean`
 - `EnableSafeRegionTrimming(enable As Boolean) As Boolean`
 - `AdjustVideoColor(parameters As roAssociativeArray) As Boolean`
 - `SetKeyingValue(keying_settings As roAssociativeArray) As Boolean`
 - `SetTransform(transform As String) As Boolean`
 - `GetFilePlayability(filename As String) As roAssociativeArray`
 - `GetProbePlayability(probe_string As String) As roAssociativeArray`
 - `GetStreamInfo() As roAssociativeArray`
 - `GetStreamStatistics() As roAssociativeArray`
 - `SetPreferredVideo(description As String) As Boolean`
 - `SetPreferredAudio(description As String) As Boolean`
 - `SetPreferredCaptions(description As String) As Boolean`
- **ifMediaTransport**
 - `PlayFile(source As Object) As Boolean`
 - `PlayFile(parameters As roAssociativeArray) As Boolean`
 - `SetProperties(parameters As roAssociativeArray) As Boolean`
 - `GetProperties() As roAssociativeArray`
 - `SetPropertiesString(parameters As String) As Boolean`
 - `GetPropertiesString() As String`
 - `SetPlaybackSpeed(speed As Float) As Boolean`
 - `PreloadFile(parameters As roAssociativeArray) As Boolean`
 - `Play() As Boolean`
 - `Stop() As Boolean`
 - `StopClear() As Boolean`
 - `Pause(parameters As roAssociativeArray) As Boolean`
 - `Resume(parameters As roAssociativeArray) As Boolean`
 - `SetLoopMode(mode As Dynamic) As Boolean`
 - `AddEvent(user_data As Integer, time_in_ms As Integer) As Boolean`
 - `ClearEvents() As Boolean`
 - `GetEvents() As roArray`
 - `PlayEx(a As Object) As Boolean`
 - `GetPlaybackPosition() As Integer`
 - `GetDuration() As Integer`
 - `Seek(position As Integer) As Boolean`
 - `SetFade(parameters As roAssociativeArray) As Boolean`
 - `ProbeFile(filename As String) As roAssociativeArray`
- **ifZorderControl**
 - `ToFront() As Boolean`
 - `ToBack() As Boolean`
- **ifAudioControl**

- **ifAudioAuxControl**
 - **MapStereoOutputAux(mapping As Integer) As Boolean**
 - **SetVolumeAux(a As Integer) As Boolean**
 - **SetChannelVolumesAux(channel_mask As Integer, b As Integer) As Boolean**
 - **SetAudioOutputAux(audio_output As Integer) As Boolean**
 - **SetAudioModeAux(audio_mode As Integer) As Boolean**
 - **SetAudioStreamAux(stream_index As Integer) As Boolean**
 - **SetUsbAudioPortAux(a As Integer) As Boolean**
- **ifUserData**
 - **SetUserData(user_data As Object)**
 - **GetUserData() As Object**
- **ifIdentity**
 - **GetIdentity() As Integer**
- **ifMessagePort**
 - **SetPort(port As roMessagePort)**
- **Timecode Events**
- **Multiscreen Video Playback**
 - **Multiscreen with Portrait Mode**
- **RF Channel Scanning**
- **Playing Encrypted Files**
- **Preferred Streams**
 - **SetPreferredVideo(description As String) As Boolean**
 - **SetPreferredAudio(description As String) As Boolean**
 - **SetPreferredCaptions(description As String) As Boolean**
- **Pattern Matching Rules**
- **Pattern Matching Examples**
- **Examples**

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is used to play back video files (using the generic *ifMediaTransport* interface). If the message port is set, the object will send events of the type *roVideoEvent*. All object calls are asynchronous. That is, video playback is handled in a different thread from the script, and the script will continue to run while video is playing. Decoded video will be scaled to the output resolution specified by *roVideoMode*.

To display video in a zone/window, you must call `SetRectangle()`. In firmware versions 6.0.x and later, zone support is enabled by default

`ifVideoControl`

PlayStaticImage(filename As String) As Boolean

Uses the video decoder to display an image. On 4Kx42 models, you can use the video decoder to display 4K images.

PlayStaticImage(parameters As roAssociativeArray) As Boolean

Uses the video decoder to display an image. The passed associative array can contain the following parameters:

- **Filename:** The name of the image file
- **EncryptionAlgorithm:** The file-encryption algorithm. Currently the options are "AesCtr" and "AesCtrHmac".
- **EncryptionKey:** The key to decrypt the image file. This is a byte array consisting of 128 bits of key, followed by 128 bits of IV.

See the [Image Decryption](#) section in the `roImagePlayer` entry for details on displaying encrypted images.

SetViewMode(mode As Integer) As Boolean

Sets the scaling of the video in relation to the video window. The passed integer can be one of the following values:

- 0: Scales the video to fill the window. The aspect ratio of the source video is ignored, so the video may appear stretched/squashed.
- 1: Letterboxes and centers the window. The aspect ratio of the source window is maintained.

- 2:(Default) Scales the video to fill the window. The aspect ratio is maintained, so the video may appear cropped.

Note that view modes rely on correct aspect-ratio marking from video files, and not all files may be marked correctly.

SetRectangle(r As roRectangle) As Void

Specifies the placement and dimensions of the video window using a passed *roRectangle* instance.

Hide() As Boolean

Hides the video window.

Show() As Boolean

Shows the video window.

EnableSafeRegionTrimming(enable As Boolean) As Boolean

AdjustVideoColor(parameters As roAssociativeArray) As Boolean

Adjusts the video and graphics output of the player using the following parameters, which can be passed to the method as an associative array: "brightness", "hue", "contrast", "saturation". Each parameter has a default value of 0 and can accept a range of values between -1000 and 1000.

SetKeyingValue(keying_settings As roAssociativeArray) As Boolean

Applies a mask to each pixel in the video window. If the pixel value falls within the specified range of chroma and luma key values, the pixel will appear transparent, allowing video and graphics behind it to show through. If the pixel value does not fall within the specified range, the pixel is unaltered. The chroma and luma key values are set using integers contained in the passed associative array:

- luma
- cr
- cb

Each integer value is arranged as follows: [8 bits of mask][8 bits of high-end range][8 bits of low-end range]. For example, an 0xff8040 value for luma would mask luma at 0xff (no change) and then apply a range from 0x40 to 0x80 for changing to transparent alpha. Note that chroma and luma keying work well with simple shapes and patterns, while complex patterns like hair or grass will not be masked effectively.

SetTransform(transform As String) As Boolean

Applies one of eight transforms to the video plane. This method works equally well with all video sources (files, streams, HDMI input) and can be called separately on multiple *roVideoPlayer* instances. Calls to this method only take effect when the next file/source is played, and transitions to a transformed video do not take place seamlessly.

- `identity`: No transformation (default behavior)
- `rot90`: 90 degree clockwise rotation
- `rot180`: 180 degree rotation
- `rot270`: 270 degree clockwise rotation
- `mirror`: Horizontal mirror transformation
- `mirror_rot90`: Mirrored 90 degree clockwise rotation
- `mirror_rot180`: Mirrored 180 degree clockwise rotation
- `mirror_rot270`: Mirrored 270 degree clockwise rotation

Note

The coordinates and dimensions of the *roRectangle* instance containing the video are not affected by rotation.

GetFilePlayability(filename As String) As roAssociativeArray

Returns an associative array indicating the playability of the video file. For the following keys, a "playable" value indicates that the component is playable, while a "no media" value indicates that there is no media—any other value indicates that the media is unplayable.

- `audio`: The audio file associated with the video
- `video`: The video file associated with the video
- `file`: The video container file

GetProbePlayability(probe_string As String) As roAssociativeArray

Returns an associative array indicating the playability of the probe string. For the following keys, a "playable" value indicates that the component is playable, while a "no media" value indicates that there is no media—any other value indicates that the media is unplayable.

- audio: The audio file associated with the video
- video: The video file associated with the video
- file: The video container file

GetStreamInfo() As roAssociativeArray

Returns an associative array containing information about the current video. To retrieve metadata about a video file that is not currently playing, use the `ProbeFile()` method instead. The associative array can contain the following parameters:

- [string] Source: The URI of the video file
- [string] SrcAddress: The source IP address of the video stream
- [string] DstAddress: The multicast address on which the IP stream is being transmitted. This value may be absent if the RTSP service has not redirected the stream (in this case, the IP address of the player may be displayed instead).
- [string] Encapsulation: The encapsulation of the video. This value can be "ES" (elementary stream), "TS" (transport stream), or "UNKNOWN" for streaming video.
- [string] AudioFormat: The format of the audio file
- [int] AudioSampleRate: The audio sample rate (in hertz)
- [int] AudioChannelCount: The number of audio channels
- [int] AudioDuration: The duration of the audio track (in milliseconds)
- [string] VideoFormat: The format of the video file
- [int] VideoFramerate: The video frame rate (in frames per second)
- [int] VideoColorDepth: The color depth of the video (in bits)
- [int] VideoWidth: The width of the video (in pixels)
- [int] VideoHeight: The height of the video (in pixels)
- [float] VideoAspectRatio: The aspect ratio of the video
- [int] VideoDuration: The duration of the video (in milliseconds)
- [int] PreferredVideo: The current preferred video track, as determined by the `SetPreferredVideo()` method
- [int] PreferredAudio: The current preferred audio track, as determined by the `SetPreferredAudio()` method
- [int] PreferredSubtitle: The current preferred subtitle track, as determined by the `SetPreferredCaptions()` method
- [roArray] Programs: A list of programs that are part of the video. Each entry contains the program ID, along with indices of video, audio, and subtitle tracks that are part of the program. These track lists can be used—along with the `Video`, `Audio`, and `Subtitle` parameters and the `SetPreferred<>()` methods—to scan and select tracks (see the [example at the bottom of this page](#) for more details). Each entry can contain the following parameters:
 - [int] ProgramId: The program ID
 - [roArray] Video: An index of integers corresponding to video tracks that are part of the program
 - [roArray] Audio: An index of integers corresponding to audio tracks that are part of the program
 - [roArray] Subtitle: An index of integers corresponding to subtitle tracks that are part of the program

Tip

To retrieve more information about an individual video/audio/subtitle track, use the integer value to look up the associated `Video/Audio/Subtitle` index (e.g. `print streaminfo.subtitle[streaminfo.programs[0].subtitle[1]]`).

- [roArray] Video: A list of video tracks that are part of the video. Each entry in the list is an associative array with the following parameters:
 - [float] AspectRatio: The aspect ratio of the video track
 - [string] Name: The name of the video track
 - [int] Width: The video width (in pixels)
 - [int] Height: The video height (in pixels)
 - [int] ColorDepth: The aspect ratio of the video track
 - [int] Duration: The duration of the video track (in milliseconds)
 - [int] Program: The ID of the program to which the video track belongs
 - [float] FrameRate: The framerate of the video track
 - [int] Pid: The packet identifier (PID) of the video track
 - [string] Format: The format of the video track
- [roArray] Audio: A list of audio tracks that are part of the video. Each entry in the list is an associative array with the following parameters:

- [string] Name: The name of the audio track
- [int] ChannelCount: The number of audio channels
- [string] Format: The format of the audio track
- [int] Pid: The packet identifier (PID) of the audio track
- [int] Program: The ID of the program to which the audio track belongs
- [int] Duration: The duration of the audio track (in milliseconds)
- [int] SampleRate: The audio sample rate (in hertz)
- [string] Language: A code specifying the language of the audio track (e.g. "eng", "spa"). The language codes are specified in the ISO 639-2 standard.
- [roArray] Subtitle: A list of subtitle tracks that are part of the video. Each entry in the list is an associative array with the following parameters:
 - [string] Language: A code specifying the language of the subtitle track (e.g. "eng", "spa"). The language codes are specified in the ISO 639-2 standard.
 - [int] Program: The ID of the program to which the subtitle track belongs
 - [int] Pid: The packet identifier (PID) of the subtitle track
 - [string] Type: The encoding standard of the subtitles (e.g. "CEA708", "DVB")

GetStreamStatistics() As roAssociativeArray

Returns an associative array containing statistics associated with the IP stream. The associative array contains the following parameters:

Note

All counters are reset every time PlayFile() is called. The audio keys will not be included in the associative array if there is no audio in the stream.

- [int] Bitrate: The video bitrate
- [int] NumDisplayed: The number of video frames displayed. This is based on the refresh rate of the monitor.
- [int] NumUnderflowed: The number of times the video FIFO has under-flowed. This usually indicates that the **buffer size** needs to be increased.
- [int] NumDecodeErrors: The number of video frames with decode errors
- [int] NumDecoded: The total number of video frames decoded
- [int] NumAudioDecoded: The total number of audio frames decoded
- [int] NumAudioDecodeErrors: The number of audio frames with decode errors
- [int] NumAudioDummy: The total number of missing audio frames. This value will increment when an audio frame goes missing or a timestamp is incorrect. A couple of frames will often be registered when streaming begins.
- [int] NumAudioUnderflows: The number of times the audio FIFO has under-flowed. This usually indicates that the **buffer size** needs to be increased.
- [int] VideoFramesPerSecond: The video frame rate (in frames per second)
- [int] VideoInterlaced: A flag indicating whether the video frames are interlaced or progressive

SetPreferredVideo(description As String) As Boolean

Chooses a video stream from the video input based on the **parameters** in the passed string.

SetPreferredAudio(description As String) As Boolean

Chooses an audio stream from the video input based on the **parameters** in the passed string.

SetPreferredCaptions(description As String) As Boolean

Chooses a data stream from the video input based on the **parameters** in the passed string.

ifMediaTransport

PlayFile(source As Object) As Boolean

Plays a video file or HDMI Input. To play a file, pass a string specifying the file name and path. To play HDMI Input, pass an *roVideoInput* instance

PlayFile(parameters As roAssociativeArray) As Boolean

Plays video using the parameters passed as an associative array. All settings specified with this method are transient: They will only last for the duration of the file/stream playback. To specify persistent settings, use the **SetProperties()** method or the equivalent "Set" methods (SetTransform(), SetViewMode(), etc.).

- **Filename:** The name/path of a file to be used for playback
- **Url:** The URL of a **video stream** to be used for playback
- **FadeInLength:** The length (in milliseconds) of fade-in at the beginning of the media
- **FadOutLength:** The length (in milliseconds) of fade-out at the end of the media
- **Transform:** The rotation of the video. See the `SetTransform()` entry for a list of applicable values.
- **Decoder:** The `friendly_name` of the **decoder** that you wish to use to play the video.
- **LoopMode:** The looping mode for media playback. See the `SetLoopMode()` entry for a list of applicable values.
- **ViewMode:** The view mode of the video window. See the `SetViewMode()` entry for a list of applicable values.
- **StreamLatency:** The amount of deviation (in milliseconds) from the default latency value: For example, a value of -500 will reduce the latency by half a second; a 500 value will increase the latency by half a second; and a 0 value will specify the default latency. Specifying a negative value will not change the buffer size; instead, it will give the buffer less time to fill up before playback begins. Usable values extend to approximately -750, though this value may differ depending on the network environment. Reducing the latency too much will result in obvious playback stutter.
- **StreamFadeIn:** The length (in milliseconds) of audio/video fade-in for streams.
- **StreamLowLatency:** Low-latency mode for RTSP streams. Setting this parameter to True will achieve the lowest possible latency for a stream, but at a reduced maximum bitrate.
- **StreamProbe:** The stream probe type. This parameter can be set to "deep" (to include video dimensions, audio sample rate, etc.) or "shallow".
- **StreamMaxBitrate:** The maximum initial bitrate (in bytes) for adaptive streaming.

These parameters are used to set they keying value of the video (see the `SetKeyingValue()` entry for more details)

- LumaKey
- CrKey
- CbKey

These parameters are used to **play encrypted video**:

- EncryptionAlgorithm
- EncryptionKey

These parameters are used to parse **preferred streams**:

- PreferredVideo
- PreferredAudio
- PreferredCaptions

These parameters are used in conjunction with the `roSyncManager` object to synchronize playback:

- SyncDomain
- SyncId
- SyncIsoTimestamp

These parameters are used for **multiscreen** playback:

- MultiscreenWidth
- MultiscreenHeight
- MultiscreenX
- MultiscreenY
- SourceX
- SourceY
- SourceWidth
- SourceHeight

SetProperties(parameters As roAssociativeArray) As Boolean

Sets persistent properties for video playback. These properties can be temporarily overridden by the parameters in a `PlayFile()` call. See the `PlayFile()` entry for a list of available parameters.

GetProperties() As roAssociativeArray

Returns the current video-playback properties as an associative array. See the `PlayFile()` entry for a description of parameters.

SetPropertiesString(parameters As String) As Boolean

Sets persistent properties for video playback using a comma-separated list. These properties can be temporarily overridden by the parameters in a `PlayFile()` call. See the `PlayFile()` entry for a list of available parameters.

Example

```
vp = CreateObject("roVideoPlayer")
vp.SetPropertiesString("Transform=rot90,StreamLowLatency=true")
```

GetPropertiesString() As String

Returns the current video-playback properties as a string (e.g. "`<key>=<value>`, `<key>=<value>`"). See the `PlayFile()` entry for a description of parameters.

SetPlaybackSpeed(speed as Float) As Boolean

Modulates the playback speed of the video, using the float 1.0 as the value for standard playback speed. To fast-forward the video, pass a value greater than 1.0; to rewind the video, pass a negative value. A value between 0 and 1.0 will play the video in slow motion.

PreloadFile(parameters As roAssociativeArray) As Boolean

Play() As Boolean

Plays the currently loaded file or stream.

Stop() As Boolean

Stops playback of the currently loaded file or stream.

StopClear() As Boolean

Stops video playback and clears the currently loaded file or stream.

Pause(parameters As roAssociativeArray) As Boolean

Pauses the video file or stream. This method accepts an optional associative array containing the following parameter:

- `SyncIsoTimeStamp`: The time stamp for pausing synchronized video. This value is provided by the `roSyncManager.Synchronize()` method on the master unit and the `roSyncManagerEvent.GetIsoTimeStamp()` method on slave unit(s).

Resume(parameters As roAssociativeArray) As Boolean

Resumes a paused video file or stream. This method accepts an optional associative array containing the following parameter:

- `SyncIsoTimeStamp`: The time stamp for resuming synchronized video. This value is provided by the `roSyncManager.Synchronize()` method on the master unit and the `roSyncManagerEvent.GetIsoTimeStamp()` method on slave unit(s).

SetLoopMode(mode As Dynamic) As Boolean

Specifies the looping mode for media playback. If this method is passed `True`, a single media file will loop seamlessly if possible. If the video file **cannot be looped seamlessly**, then the video will loop with seams. Setting this method to `False`, which is the default behavior, allows for playback of multiple files in a playlist—with noticeable gaps between the end and beginning of the file. Alternatively, this method can accept an associative array with three Boolean parameters: `enable`, `enable_if_seamless`, `allow_seamless`. The following table describes how these parameters interact:

<code>enable</code>	<code>enable_if_seamless</code>	<code>allow_seamless</code>	Behavior
False	X	X	Looping is disabled in all cases (the <code>enable_if_seamless</code> and <code>allow_seamless</code> parameters are ignored).
True	False*	True*	The video is looped seamlessly if possible; otherwise, it is looped with seams.
True*	True	True*	The video is looped seamlessly if

			possible; otherwise, it is not looped at all.
True*	True	False	Looping is disabled in all cases.
True	False*	False	The video is looped with seams.

*or not specified.

Note

Media End events are only sent if seamless looping is disabled, or if `enable_if_seamless` is enabled and the file cannot be looped seamlessly.

AddEvent(user_data As Integer, time_in_ms As Integer) As Boolean

Adds a trigger that will generate an *roVideoEvent* when it reaches the specified time. The user data will be passed with the event and can be retrieved using the *roVideoEvent.GetData()* method. See the [Video Timecode Events](#) section below for more details.

ClearEvents() As Boolean

Removes all timecode events that have been added using the `AddEvent()` method.

GetEvents() As roArray

Returns an array of timecode events added to the *roVideoPlayer* instance using the `AddEvent()` method. Each entry in the array consists of an associative array with the following values:

- `id`: The `user_data` of the event (as an Integer)
- `timestamp`: The timestamp (in milliseconds)

PlayEx(a As Object) As Boolean

This object has been deprecated. We suggest using the `PlayFile()` method for video playback instead.

GetPlaybackPosition() As Integer

Returns the amount of time the current file or IP stream has been playing (in milliseconds). If `SetLoopMode()` is set to `True`, the value will not reset when playback loops. If looping playback or IP streaming continues uninterrupted for approximately 25 days, the value will wrap around and become negative.

GetDuration() As Integer

Returns the total playback duration (in milliseconds) of the current file.

Seek(position As Integer) As Boolean

Seeks to the specified position in the audio/video file (measured in milliseconds). If the file is currently playing, then it will continue to play; otherwise, it will remain paused after seeking. This method only supports the MP4/MOV video container; all standard audio formats are supported.

SetFade(parameters As roAssociativeArray) As Boolean

Fades out both the video and audio when the method is called. When the fade completes, an *roVideoEvent* object with the `18 - FadeOut` value will be posted to the message port. This method accepts an associative array, which can currently contain only one parameter:

- `FadeOutLength`: The length of time (in milliseconds) over which the audio/video fades out.

ProbeFile(filename As String) As roAssociativeArray

Returns an associative array containing metadata about the specified video file. To retrieve metadata about a file that is currently playing, use the `GetStreamInfo()` method instead. The returned associative array can contain the following parameters:

- `Source`: The URI of the file
- `Encapsulation`: The encapsulation of the video
- `AudioFormat`: The format of the audio file
- `AudioSampleRate`: The audio sample rate (in hertz)
- `AudioChannelCount`: The number of audio channels
- `AudioDuration`: The duration of the audio track (in milliseconds)

- `VideoFormat`: The format of the video file
- `VideoColorDepth`: The color depth of the video (in bits)
- `VideoWidth`: The width of the video (in pixels)
- `VideoHeight`: The height of the video (in pixels)
- `VideoAspectRatio`: The aspect ratio of the video
- `VideoDuration`: The duration of the video (in milliseconds)

`ifZorderControl`

ToFront() As Boolean

Places the video layer of the `roVideoPlayer` instance in front of the other video player.

ToBack() As Boolean

Places the video layer of the `roVideoPlayer` instance behind the other video player.

Note

This feature is not available on HD/LS players, which only support a single video player. For more information on ordering video layers relative to the graphics layer, refer to the `roVideoMode.SetGraphicsZOrder()` entry.

`ifAudioControl`

See the `roAudioPlayer` entry for documentation of `ifAudioControl`.

`ifAudioAuxControl`

MapStereoOutputAux(mapping As Integer) As Boolean

SetVolumeAux(a As Integer) As Boolean

SetChannelVolumesAux(channel_mask As Integer, b As Integer) As Boolean

SetAudioOutputAux(audio_output As Integer) As Boolean

SetAudioModeAux(audio_mode As Integer) As Boolean

SetAudioStreamAux(stream_index As Integer) As Boolean

SetUsbAudioPortAux(a As Integer) As Boolean

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifIdentity`

GetIdentity() As Integer

Note

The *ifIdentity* interface has been deprecated. We recommend using the *ifUserData* interface instead.

`ifMessagePort`

SetPort(port As roMessagePort)

Posts messages of type *roVideoEvent* to the attached message port.

Timecode Events

You can use the `AddEvent()` method to add triggers for *roVideoEvent* events, which will generate the `12 - Timecode Hit` value at the specified millisecond times in a video file. Use the *roVideoEvent.GetData()* method to retrieve the user data passed with `AddEvent()`.

The following example script uses timecode events. The script prints 2, 5, and 10 at 2 seconds, 5 seconds, and 10 seconds into the video, respectively. The "msg" is approaching frame accurate.

```
v = CreateObject("roVideoPlayer")
p = CreateObject("roMessagePort")
v.SetPort(p)

ok = v.AddEvent(2, 2000) ' Add timed events to video
ok = v.AddEvent(5, 5000)
ok = v.AddEvent(10, 10000)
ok = v.AddEvent(100, 100000)
ok = v.PlayFile("SD:/C5_d5_phil.vob")

waitloop:
msg = Wait(0,p) ' Wait for all events
if msg.GetInt() = 8 then stop ' End of file
if msg.GetInt() <> 12 goto waitloop      ' I only care about time events
print msg.GetData() ' Print out index when the time event happens
goto waitloop
```

Multiscreen Video Playback

The `PreloadFile()` and `PlayFile()` methods can be used in conjunction with *roSyncManager* to stretch an image across multiple screens in an array or display windowed portions of a video.

The following example script uses the `PreloadFile()` method for multiscreen display:

```
v=CreateObject("roVideoPlayer")
a=CreateObject("roAssociativeArray")
a["Filename"] = "test.ts"
a["MultiscreenWidth"] = 3
a["MultiscreenHeight"] = 2
a["MultiscreenX"] = 0
a["MultiscreenY"] = 0
v.PreloadFile(a)
...
...
v.Play()
```

The `MultiscreenWidth` and `MultiscreenHeight` values specify the width and height of the multiple-screen matrix. For example, 3x2 would be 3 screens wide and 2 high. `MultiscreenX` and `MultiscreenY` specify the position of the current screen within that matrix. In the case above, on average only 1/6th of the video is drawn on each screen (though the view mode still applies), so depending on the shape of the video, it may have black bars on the side screens. In this way, it is relatively simple for a video player to display part of an image based on its position in the multiscreen array.

`PreloadFile()` does all of the preliminary work to get ready to play the specified video clip, including stopping the playback of the previous video file. The call to "Play" starts the playback. This is good for synchronizing video across multiple players as they can all be prepared ready to play and then will immediately start playing when the "Play" command is issued. This reduces synchronization latencies.

The following are the default values for the parameters:

- MultiscreenWidth= 1
- MultiscreenHeight= 1
- MultiscreenX= 0
- MultiscreenY= 0

This script uses `PlayFile()` to display a portion of a video. This displays a windowed portion of the `test.ts` video file starting at coordinates `SourceX`, `SourceY`, and `SourceWidth` by `SourceHeight` in size. The `SetViewMode()` setting is still honored as if displaying the whole file.

```
v=CreateObject("roVideoPlayer")
a=CreateObject("roAssociativeArray")
a["Filename"] = "test.ts"
a["SourceX"] = 100
a["SourceY"] = 100
a["SourceWidth"] = 1000
a["SourceHeight"] = 500
v.PlayFile(a)
```

Multiscreen with Portrait Mode

To create a multiple-screen matrix in portrait mode, call `SetTransform("rot90")` or `SetTransform("rot270")` before calling `PlayFile()`.

This script creates a 2x1 portrait-mode multiscreen display:

```
v1=CreateObject("roVideoPlayer")
v1.SetViewMode(1)
r=CreateObject("roRectangle", 0, 0, 1920, 1080)
v1.SetRectangle(r)
v1.SetTransform("rot90")

a1=CreateObject("roAssociativeArray")
a1.MultiscreenWidth = 2
a1.MultiscreenHeight = 1
a1.MultiscreenX = 1
a1.MultiscreenY = 0
a1.Filename = "example.mp4"
v1.PlayFile(a1)
```

RF Channel Scanning

The `PlayFile()` method can be used for channel scanning and handling functionality similar to `roChannelManager`. To use `PlayFile()` for channel scanning, pass an `roAssociativeArray` with the following possible parameters:

- VirtualChannel
- RfChannel
- SpectralInversion
 - INVERSION_ON
 - INVERSION_OFF
 - INVERSION_AUTO
- ModulationType
 - QAM_64
 - QAM_256
 - QAM_AUTO
 - 8VSB
- VideoCodec
 - MPEG1-Video
 - MPEG2-Video
 - MPEG4Part2-Video
 - H264
 - H264-SVC
 - H264-MVCAVSC

- AudioCodec
 - MPEG-Audio
 - AAC
 - AAC+
 - AC3AC3+DTS
- VideoPid
- AudioPid
- PcrPid

The `VirtualChannel` and `RfChannel` parameters must be present for `PlayFile()` to scan correctly. If you specify only these parameters, the player will scan the RF channel for a QAM/ATSC signal and attempt to retrieve the specified virtual channel from the results. The results from this action are cached so that subsequent calls to `PlayFile()` will take much less time. Providing the `SpectralInversion` and/or `ModulationType` parameters will further speed up the scanning process.

If all parameters are supplied, then no scanning is required and the player can tune to the channel immediately. If one or more of the optional parameters is missing, then the player must parse the transport stream metadata to find the appropriate values for the supplied `VirtualChannel` and `RfChannel`.

Playing Encrypted Files

The `roVideoPlayer` object can be used to play audio/video files or streams that have been encrypted using AES.

Note

Streaming decryption requires firmware version 6.2.x and is currently only supported with the UDP protocol and the HTTP protocol (when HTTP is paired with an MPEG2 transport stream). If using a UDP multicast MPEG2 transport stream, one of the elemental streams should provide the PCR to the player.

The associative array passed to the `PlayFile()` method can accept two parameters for file decryption:

- `EncryptionAlgorithm`: The file-encryption algorithm. The following are the current options:
 - "AesCtr": The AES algorithm in CTR mode.
 - "AesCtrHmac": The AES algorithm in CTR mode with HMAC.
 - "TsAesEcb": The AES algorithm in ECB mode (e.g. with a Harmonic Prostream). This algorithm is currently used for streaming encryption/decryption.
- `EncryptionKey`: A byte array consisting of 128 bits of key. If the encryption algorithm is AES-CTR or AES-CTR-HMAC, this is followed by 128 bits of IV.

Note

File decryption is supported on the XTx44, XTx43, 4Kx42, XDx34, XDx33, XDx32, XDx30, HDx23, HDx22, and LS423 platforms. Contact support@brightsign.biz to learn more about generating a key for obfuscation and storing it on the player.

Example

```
v = CreateObject("roVideoPlayer")
aa=CreateObject("roAssociativeArray")
aa.filename = "wall-sync2.mp4"
aa.encryptionalgorithm = "AesCtr"
aa.encryptionkey = CreateObject("roByteArray")
aa.encryptionkey.fromhexstring
("01030507090b0d0f00020406080a0c0e000000000000000000000000000000")

v.PlayFile(aa)
```

Preferred Streams

If multiple video, audio, or data streams are encapsulated in the video input, you can use the `SetPreferredVideo()`, `SetPreferredAudio()`, and `SetPreferredCaptions()` methods to determine which stream to use. For example, if a video may contain English and Spanish audio tracks, you can call `SetPreferredAudio()` to specify that the Spanish track should be played if it exists, with the video defaulting to English otherwise.

Preferred streams are chosen by matching the patterns in the passed string(s) against the textual description of the stream:

1. The passed string is a semicolon-separated list of templates.
2. Each template is a comma-separated list of patterns.
3. Each pattern is a [field_name]=[field_value] pair that is matched directly against the stream description.

SetPreferredVideo(description As String) As Boolean

Each template in the passed video description string can contain the following patterns:

- pid=[integer]: The packet identifier (PID) of the video stream you wish to display
- program=[integer]: The program number of the video stream
- codec=[video_codec]: The preferred video codec, which can be any of the following:
 - MPEG1
 - MPEG2
 - MPEG4Part2
 - H263
 - H264
 - VC1
 - H265
- width=[integer]: The preferred video width
- height=[integer]: The preferred video height
- aspect=[float(x.yy)]: The preferred aspect ratio of the video stream as a floating-point number with two fractional digits.
- colordepth=[integer]: The preferred color depth of the video.

Example

```
"pid=7680, codec=H264, width=1280, height=720, aspect=1.78, colordepth=8;"
```

SetPreferredAudio(description As String) As Boolean

Each template in the passed description string can contain the following patterns:

- pid=[integer]: The packet identifier (PID) of the audio stream you wish to play
- program=[integer]: The program number of the audio stream
- codec=[audio_codec]: The preferred audio codec, which can be any of the following:
 - MPEG
 - MP3
 - AAC
 - AAC-PLUS
 - AC3
 - AC3-PLUS
 - DTS
 - PCM
 - FLAC
 - Vorbis
- channels=[integer]: The preferred number of audio channels (from 1 to 8)
- freq=[frequency]: The preferred sample frequency of the audio track, which can be any of the following:
 - 32000
 - 44100
 - 48000
- lang=[language]: A code that determines the preferred language of the audio track (e.g. eng, spa). The language codes are specified in the ISO 639-2 standard.
- type=[audio_type]: The preferred audio type, which can be one of the following:
 - Main audio
 - Clean effects
 - Hearing impaired
 - Visual impaired commentary

Example

```
"pid=4192, codec=AC3, channels=5, freq=48000, lang=eng, type=Main audio;;"
```

SetPreferredCaptions(description As String) As Boolean

Each template in the passed description string can contain the following patterns:

- `pid=[integer]`: The packet identifier (PID) of the caption stream you wish to play
- `type=[subtitle_type]`: The encoding standard of the subtitles. This value can be one of the following:
 - CEA708: If the CEA-708 standard is not present, the `subtitle_type` will default to CEA-608 (if it is present).
 - CEA608
 - DVB
- `lang=[language]`: A code that determines the preferred language of the subtitles (e.g. eng, spa). The language codes are specified in the ISO 639-2 standard.
- `service=[integer]`: The preferred service number of the caption stream

Example

```
"pid=0, type=Cea708, lang=eng service=1;;"
```

Pattern Matching Rules

Note the following rules when matching templates to video, audio, or caption stream descriptions:

- For a template to match a stream description, every pattern within the template must match.
- The first listed template to match the stream description (if any) will be used.
- An empty template string will match any stream description.
- All value comparisons are case-insensitive.
- Numerical values must match the stream description exactly (without leading zeroes). For example, the pattern `pid=016` will never match the stream PID value of 16.
- To indicate logical negation, apply the "!" exclamation mark to the beginning of a pattern. For example, specifying `SetPreferredVideo(" !codec=H265")` will match only streams that are not encoded using H.265.
- Apply the ">" greater-than symbol before an integer to indicate that, for a successful match, the value in the stream description must be *greater than* the value following the symbol. For example, specifying `SetPreferredVideo("width=<1921,height=<1081")` will match only videos that are no larger than full-HD.
- Apply the "<" less-than symbol before an integer to indicate that, for a successful match, the value in the stream description must be *less than* the value following the symbol.

Pattern Matching Examples

The following examples illustrate some of the pattern matching behavior described above:

- The following template list contains three patterns: `lang=eng`, `lang=spa`, and an empty template. The first pattern specifies an English language channel; if the English channel does not exist, the second pattern specifies a Spanish language channel. The third pattern specifies any other channel if the first two don't exist (the empty template matches anything).

```
SetPreferredAudio("lang=eng;lang=spa;;")
```

- Since the following template list is empty, no captions are specified. This can be used to disable captions altogether.

```
SetPreferredCaptions("")
```

- The following template list contains an empty template. Since an empty template matches anything, the first video stream encountered will be played. This is the default behavior of all attributes.

```
SetPreferredVideo(";")
```

- The following template list specifies a 48KHz audio stream if there is one; otherwise, no audio stream will be played. Observe that the list is not correctly terminated with a semicolon; in this case, the semi-colon is implicitly supplied.

```
SetPreferredAudio("freq=48000")
```

- The following template list contains two templates. Note that all patterns within a template must match the stream description for the entire template to match. In this example, an AAC-encoded English track is preferred; an MP3-encoded English track is designated as the second option; and any track will be chosen if neither template is matched.

```
SetPreferredAudio("codec=aac, lang=eng; codec=mp3, lang=eng; ;")
```

Examples

The following script selects a program from a video and sets preferred video and audio tracks for playback:

```
v1 = CreateObject("roVideoPlayer")
v1.PlayFile("example.ts")
si = v1.GetStreamInfo()

' Pick the program
prog = si.Programs[2].ProgramId
v1 = CreateObject("roVideoPlayer")

' Select the program
v1.SetPreferredVideo("prog=" + prog.ToStr() + ";")
v1.SetPreferredAudio("prog=" + prog.ToStr() + ";")

' Play the stream/file
v1.PlayFile("example.ts")
```

File Objects

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This section describes objects for creating, deleting, and manipulating files.

- [roAppendFile](#)
- [roCreateFile](#)
- [roReadFile](#)
- [roReadWriteFile](#)

roAppendFile

ON THIS PAGE

- [ifStreamSend](#)
 - [SetSendEol\(eol_sequence As String\) As Void](#)
 - [SendByte\(byte As Integer\) As Void](#)

- [SendLine\(string As String\) As Void](#)
- [SendBlock\(a As Dynamic\) As Void](#)
- [Flush\(\)](#)
- [AsyncFlush\(\)](#)
- [ifStreamPosition](#)
 - [CurrentPosition\(\) As Integer](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object can be used to create a new file or append information to the end of an existing file.

Object Creation: Creating an *roAppendFile* object opens an existing file or creates a new file. The current position is set to the end of the file, and all writes are made to the end of the file.

```
CreateObject("roAppendFile", filename As String)
```

ifStreamSend

SetSendEol(eol_sequence As String) As Void

Sets the EOL sequence when writing to the stream. The default value is CR+LF. If you need to set this value to a non-printing character, use the [chr \(\) global function](#).

SendByte(byte As Integer) As Void

Writes the specified byte to the stream.

SendLine(string As String) As Void

Writes the specified characters to the stream followed by the current EOL sequence.

SendBlock(a As Dynamic) As Void

Writes the specified characters to the stream. This method can support either a string or an *roByteArray*. If the block is a string, any null bytes will terminate the block.

Flush()

AsyncFlush()

ifStreamPosition

CurrentPosition() As Integer

Returns the current position within the file.

roCreateFile

ON THIS PAGE

- [ifReadStream](#)
 - [SetReceiveEol\(eol_sequence As String\) As Void](#)
 - [ReadByte\(\) As Integer](#)
 - [ReadByteIfAvailable\(\) As Integer](#)

- [ReadLine\(\)](#) As String
- [ReadBlock\(size As Integer\)](#) As String
- [AtEof\(\)](#) As Boolean
- [ifStreamSend](#)
 - [SetSendEol\(eol_sequence As String\)](#) As Void
 - [SendByte\(byte As Integer\)](#) As Void
 - [SendLine\(string As String\)](#) As Void
 - [SendBlock\(a As Dynamic\)](#) As Void
 - [Flush\(\)](#)
 - [AsyncFlush\(\)](#)
- [ifStreamSeek](#)
 - [SeekAbsolute\(offset As Integer\)](#) As Void
 - [SeekRelative\(offset As Integer\)](#) As Void
 - [SeekToEnd\(\)](#) As Void
 - [CurrentPosition\(\)](#) As Integer

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object can be used to write a new file or overwrite an existing file.

Object Creation: Creating an *roCreateFile* object opens an existing file or creates a new file. If the file exists, it is truncated to a size of zero.

```
CreateObject("roCreateFile", filename As String)
```

[ifReadStream](#)

SetReceiveEol(eol_sequence As String) As Void

Sets the EOL sequence when reading from the stream.

ReadByte() As Integer

Reads a single byte from the stream, blocking if necessary. If the EOF is reached or there is an error condition, then a value less than 0 is returned.

ReadByteIfAvailable() As Integer

Reads a single byte from the stream if one is available. If no bytes are available, it returns immediately. A return value less than 0 indicates either that the EOF has been reached or no byte is available.

ReadLine() As String

Reads until it finds a complete end of the line sequence. If it fails to find the sequence within 4096 bytes, then it returns the 4096 bytes that are found. No data is discarded in this case.

ReadBlock(size As Integer) As String

Reads the specified number of bytes. The number is limited to 65536 bytes. In the event of an EOF or an error, fewer bytes than requested will be returned. Any null bytes in the file will mask any further bytes.

AtEof() As Boolean

Returns True if an attempt has been made to read beyond the end of the file. If the current position is at the end of the file, but no attempt has been made to read beyond it, this method will return False.

[ifStreamSend](#)

SetSendEol(eol_sequence As String) As Void

Sets the EOL sequence when writing to the stream. The default value is CR+LF. If you need to set this value to a non-printing character, use the `chr()` [global function](#).

SendByte(byte As Integer) As Void

Writes the specified byte to the stream.

SendLine(string As String) As Void

Writes the specified characters to the stream followed by the current EOL sequence.

SendBlock(a As Dynamic) As Void

Writes the specified characters to the stream. This method can support either a string or an *roByteArray*. If the block is a string, any null bytes will terminate the block.

Flush()

AsyncFlush()

ifStreamSeek

SeekAbsolute(offset As Integer) As Void

Seeks the specified offset. If the offset is beyond the end of the file, then the file will be extended upon the next write and any previously unoccupied space will be filled with null bytes.

SeekRelative(offset As Integer) As Void

Seeks to the specified offset relative to the current position. If the ultimate offset is beyond the end of the file, then the file will be extended as described in `SeekAbsolute()`.

SeekToEnd() As Void

Seeks to the end of the file.

CurrentPosition() As Integer

Retrieves the current position within the file.

roReadFile

ON THIS PAGE

- [ifStreamRead](#)
 - [SetReceiveEol\(eol_sequence As String\) As Void](#)
 - [ReadByte\(\) As Integer](#)
 - [ReadByteIfAvailable\(\) As Integer](#)
 - [ReadLine\(\) As String](#)
 - [ReadBlock\(size As Integer\) As String](#)
 - [AtEof\(\) As Boolean](#)
- [ifStreamSeek](#)
 - [SeekAbsolute\(offset As Integer\) As Void](#)
 - [SeekRelative\(offset As Integer\) As Void](#)
 - [SeekToEnd\(\) As Void](#)
 - [CurrentPosition\(\) As Integer](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

Object Creation: Creating an *roReadFile* object opens the specified file for reading only. Object creation fails if the file does not exist.

```
CreateObject("roReadFile", filename As String)
```

ifStreamRead

SetReceiveEol(eol_sequence As String) As Void

Sets the EOL sequence when reading from the stream. The default EOL character is LF (ASCII value 10). If you need to set this value to a non-printing character, use the `chr()` [global function](#).

ReadByte() As Integer

Reads a single byte from the stream, blocking if necessary. If the EOF is reached or there is an error condition, then a value less than 0 is returned.

ReadByteIfAvailable() As Integer

Reads a single byte from the stream if one is available. If no bytes are available, it returns immediately. A return value less than 0 indicates either that the EOF has been reached or no byte is available.

ReadLine() As String

Reads until it finds a complete end of the line sequence. If it fails to find the sequence within 4096 bytes, then it returns the 4096 bytes that are found. No data is discarded in this case.

ReadBlock(size As Integer) As String

Reads the specified number of bytes. The number is limited to 65536 bytes. In the event of an EOF or an error, fewer bytes than requested will be returned. Any null bytes in the file will mask any further bytes.

AtEof() As Boolean

Returns True if an attempt has been made to read beyond the end of the file. If the current position is at the end of the file, but no attempt has been made to read beyond it, this method will return False.

ifStreamSeek

SeekAbsolute(offset As Integer) As Void

Seeks the specified offset. If the offset is beyond the end of the file, then the file will be extended upon the next write and any previously unoccupied space will be filled with null bytes.

SeekRelative(offset As Integer) As Void

Seeks to the specified offset relative to the current position. If the ultimate offset is beyond the end of the file, then the file will be extended as described in `SeekAbsolute()`.

SeekToEnd() As Void

Seeks to the end of the file.

CurrentPosition() As Integer

Retrieves the current position within the file.

roReadWriteFile

ON THIS PAGE

- **ifReadStream**
 - **SetReceiveEol(eol_sequence As String) As Void**
 - **ReadByte() As Integer**
 - **ReadByteIfAvailable() As Integer**
 - **ReadLine() As String**
 - **ReadBlock(size As Integer) As String**
 - **AtEof() As Boolean**
- **ifStreamSend**
 - **SetSendEol(eol_sequence As String) As Void**
 - **SendByte(byte As Integer) As Void**
 - **SendLine(string As String) As Void**
 - **SendBlock(a As Dynamic) As Void**
 - **Flush()**
 - **AsyncFlush()**
- **ifStreamSeek**
 - **SeekAbsolute(offset As Integer) As Void**
 - **SeekRelative(offset As Integer) As Void**
 - **SeekToEnd() As Void**
 - **CurrentPosition() As Integer**

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

The object opens a file and allows both reading and writing operations on that file.

Object Creation: Creating an *roReadWriteFile* object opens an existing file for both reading and writing. Object creation fails if the file does not exist. The current position is set to the beginning of the file.

```
CreateObject("roReadWriteFile", filename As String)
```

ifReadStream

SetReceiveEol(eol_sequence As String) As Void

Sets the EOL sequence when reading from the stream.

ReadByte() As Integer

Reads a single byte from the stream, blocking if necessary. If the EOF is reached or there is an error condition, then a value less than 0 is returned.

ReadByteIfAvailable() As Integer

Reads a single byte from the stream if one is available. If no bytes are available, it returns immediately. A return value less than 0 indicates either that the EOF has been reached or no byte is available.

ReadLine() As String

Reads until it finds a complete end of the line sequence. If it fails to find the sequence within 4096 bytes, then it returns the 4096 bytes that are found. No data is discarded in this case.

ReadBlock(size As Integer) As String

Reads the specified number of bytes. The number is limited to 65536 bytes. In the event of an EOF or an error, fewer bytes than requested will be returned. Any null bytes in the file will mask any further bytes.

AtEof() As Boolean

Returns True if an attempt has been made to read beyond the end of the file. If the current position is at the end of the file, but no attempt has been made to read beyond it, this method will return False.

ifStreamSend

SetSendEol(eol_sequence As String) As Void

Sets the EOL sequence when writing to the stream. The default value is CR+LF. If you need to set this value to a non-printing character, use the `chr()` [global function](#).

SendByte(byte As Integer) As Void

Writes the specified byte to the stream.

SendLine(string As String) As Void

Writes the specified characters to the stream followed by the current EOL sequence.

SendBlock(a As Dynamic) As Void

Writes the specified characters to the stream. This method can support either a string or an *roByteArray*. If the block is a string, any null bytes will terminate the block.

Flush()

AsyncFlush()

ifStreamSeek

SeekAbsolute(offset As Integer) As Void

Seeks the specified offset. If the offset is beyond the end of the file, then the file will be extended upon the next write and any previously unoccupied space will be filled with null bytes.

SeekRelative(offset As Integer) As Void

Seeks to the specified offset relative to the current position. If the ultimate offset is beyond the end of the file, then the file will be extended as described in `SeekAbsolute()`.

SeekToEnd() As Void

Seeks to the end of the file.

CurrentPosition() As Integer

Retrieves the current position within the file.

Hashing and Storage Objects

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This section describes objects related to local storage and the player registry, as well as generating data structures and hashing files.

- [roBlockCipher](#)
- [roBrightPackage](#)
- [roDiskErrorEvent](#)
- [roDiskMonitor](#)
- [roHashGenerator](#)

- [roPassKey](#)
- [roRegistry](#)
- [roRegistrySection](#)
- [roSqliteDatabase](#)
- [roSqliteEvent](#)
- [roSqliteStatement](#)
- [roStorageAttached, roStorageDetached](#)
- [roStorageHotplug](#)
- [roStorageInfo](#)
- [roVirtualMemory](#)

roBlockCipher

ON THIS PAGE

- [ifBlockCipher](#)
 - [SetIV\(iv As Object\) As Void](#)
 - [Encrypt\(key As Object, plaintext As Object\) As roByteArray](#)
 - [Decrypt\(key As Object, cipher_text As Object\) As roByteArray](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object provides a means for symmetric block encryption. It currently supports AES and CBC ciphers, at block sizes of 128, 192, or 256 bits.

Object Creation: The *roBlockCipher* object is created with an associative array representing a set of parameters.

```
CreateObject("roBlockCipher", parameters As roAssociativeArray)
```

The associative array should contain the following parameters:

- `mode`: "aes-128-cbc", "aes-192-cbc", or "aes-256-cbc"
- `padding`: "zero" or "pkcs7". The object defaults to zero padding if this parameter is omitted.

Padding is required for inputs that are not an exact multiple of the cipher block size. Specifying "zero" will add padding only when needed, while specifying "pkcs7" always adds padding, even if the data is already a multiple of the block size (in this case, an entire block will be added). PKCS#7 padding is automatically removed upon decryption, and zero padding will be retained since there are no means to unambiguously distinguish pad values from data.

ifBlockCipher

SetIV(iv As Object) As Void

Sets the Initialization Vector (IV) for CBC (Cipher-Block-Chaining) modes. If the supplied IV is shorter than required, then it will be zero padded (passing an empty string will set the vector to all zeroes). The IV will typically contain arbitrary characters and be in the form of an *roByteArray*, though it can also be a string.

Encrypt(key As Object, plaintext As Object) As roByteArray

Uses the specified key to encrypt the plaintext parameter, which can be passed as either a string or an *roByteArray*.

Decrypt(key As Object, cipher_text As Object) As roByteArray

Uses the specified key to decrypt cipher text, which should be passed as an *roByteArray*. Because the cipher text is encrypted, it can contain any character.

Example

```
' This is Case#4 from RFC3602
key = CreateObject("roByteArray")
iv = CreateObject("roByteArray")
plain = CreateObject("roByteArray")
key.FromHexString("56e47a38c5598974bc46903dba290349")
iv.FromHexString("8ce82eefbea0da3c44699ed7db51b7d9") plain.FromHexString
("a0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbefbc0c1c2c3c4c5c6c7c8c9cacbcccd
cecf0d1d2d3d4d5d6d7d8d9dadbdcdededf")
c = CreateObject("roBlockCipher", { mode: "aes-128-cbc" })
c.SetIV(iv)
crypt = c.Encrypt(key, plain)
result = crypt.ToHexString()
expected = UCase
("c30e32ffedc0774e6aff6af0869f71aa0f3af07a9a31a9c684db207eb0ef8e4e35907aa632c3ffdf868bb7b29d3d
46ad83ce9f9a102ee99d49a53e87f4c3da55")

' Decrypt example to recover the encrypted data
c.SetIV(iv)
roundtrip = c.Decrypt(key, crypt)

' Second example selecting PKCS#7 padding
c = CreateObject("roBlockCipher", { mode: "aes-128-cbc", padding: "pkcs7" })
```

roBrightPackage

ON THIS PAGE

- [Limitations](#)
- [ifBrightPackage](#)
 - [Unpack\(path As String\) As Void](#)
 - [SetPassword\(password As String\) As Void](#)
 - [GetFailureReason\(\) As String](#)
 - [UnpackFile\(a As String, b As String\) As Boolean](#)
- [Using roBrightPackage for Content Updates](#)
 - [Unpacking Encrypted Archives](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

An *roBrightPackage* object represents a *.zip* file, which can include arbitrary content or be installed on a storage device to provide content and script updates (for example, to distribute updates via USB thumb drives).

Object Creation: The *roBrightPackage* object is created with a filename parameter that specifies the name of the *.zip* file.

```
CreateObject("roBrightPackage", filename As String)
```

Limitations

The *roBrightPackage* object supports *.zip* files that are smaller than 4GB only.

Supported Technologies

- deflate32 (with default options)
- PPMd (with default options)
- WinZip (with "No compression", "Maximum", and "SuperFast" options; other options, including "Enhanced Deflate", are not supported)
- AES encryption

Unsupported Technologies

- bzip2
- LZMA
- Deflate64
- zip64 (i.e. the compression software built in to Windows Explorer)

`ifBrightPackage`

ifBrightPackage is a legacy interface. We recommend you use *roAssetPool* instead to achieve better functionality.

Unpack(path As String) As Void

Extracts the *.zip* file to the specified destination path. Any preexisting files in the target directory will be deleted as part of this operation. Providing a destination path of "SD:" will wipe all preexisting files from the card and extract the *.zip* contents to the root folder.

SetPassword(password As String) As Void

Provides the password specified when the *.zip* file was created. This method supports AES 128 and 256 bit encryption, as generated by WinZip.

GetFailureReason() As String

UnpackFile(a As String, b As String) As Boolean

Example

```
package = CreateObject("roBrightPackage", "newfiles.zip")
package.SetPassword("test")
package.Unpack("SD:/")
```

Using `roBrightPackage` for Content Updates

BrightSign players check storage devices for autorun scripts in the following order:

1. External USB devices 1 through 9
2. SD
3. μ SD

In addition to looking for *autorun.brs* scripts, BrightSign players look for *autorun.zip* files that contain the script name *autozip.brs*. If an *autorun.zip* file with an *autozip.brs* file is found, and *autozip.brs* can be decrypted, then the player will execute the *autozip.brs* file.

Important

The *autozip.brs* file cannot reference any external files, as it is the only file to be automatically uncompressed by a BrightSign player prior to execution.

The *autozip.brs* script should unpack the contents of the *autorun.zip* file to an installed storage device and reboot to complete the update:

Example

```
package = CreateObject("roBrightPackage", "SD:/autorun.zip")
package.Unpack("SD:/")
MoveFile("SD:/autorun.zip", "SD:/autorun.zip_invalid")
RebootSystem()
```

Unpacking Encrypted Archives

If the *autorun.zip* file is encrypted, then the player uses the password stored in the **registry**, in the section "security" under the name "autozipkey," to decrypt the file.

Extended Example

```
' Content update application

r=CreateObject("roRectangle", 20, 668, 1240, 80)
t=CreateObject("roTextWidget",r,1,2,1)
r=CreateObject("roRectangle", 20, 20, 1200, 40)
t.SetSafeTextRegion(r)
t.SetForegroundColor(&hff303030)
t.SetBackgroundColor(&hfffffff)
t.PushString("Updating content from USB drive, please wait...")

package = CreateObject("roBrightPackage", "autorun.zip")
package.SetPassword("test")
package.Unpack("SD:/")
package = 0

t.Clear()
t.PushString("Update complete - remove USB drive to restart.")

while true
    sleep(1000)

    usb_key = CreateObject("roReadFile", "USB1:/autorun.zip")
    if type(usb_key) <> "roReadFile" then
        a=RebootSystem()
    endif
    usb_key = 0
end while
```

roDiskErrorEvent

ON THIS PAGE

- **ifUserData**
 - **SetUserData(user_data As Object)**
 - **GetUserData() As Object**
- **ifDiskErrorEvent**
 - **GetDiskError() As Object**

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is returned while waiting on a message port that is connected to an *roDiskMonitor* object.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via *SetUserData()*. It will return *Invalid* if no data has been set.

ifDiskErrorEvent

GetDiskError() As Object

Returns an *roAssociativeArray* containing the following parameters:

Key	Type	Description
source	<i>roString</i>	The error type
time	<i>roDateTime</i>	The time at which the error occurred (with millisecond accuracy)
device	<i>roString</i>	The internal name for the device generating the error
error	<i>roString</i>	A description of the error (e.g. "Timeout")
param	<i>roString</i>	The error parameter (use depends on type of error (e.g. the sector number))

Example

```
aa = msgp.GetDiskError()  
report = "Time: " + aa["Time"] + "Error: " + aa["source"] + " " + aa["error"] + " " + aa  
["device"] + " " + aa["param"]
```

This example uses an implicit conversion of *roDateTime*. You could also use *roDateTime.GetString()*.

roDiskMonitor

ON THIS PAGE

- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object provides access to low-level information about disk errors. It provides an event-based interface that delivers *roDiskErrorEvent* objects via *roMessagePort*. Error messages are held for five seconds before delivery to minimize the chance of spurious error reports. Errors are not reported if the disk is removed during this five second interval because disk-removal detection takes several seconds. This allows for long-term monitoring of occasional media errors.

Object Creation: The *roDiskMonitor* object is created with no parameters.

```
CreateObject("roDiskMonitor")
```

ifMessagePort

SetPort(port As roMessagePort)

Posts messages of type *roDiskErrorEvent* to the attached message port.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via *SetUserData()*. It will return Invalid if no data has been set.

Example

```
diskmon=CreateObject("roDiskMonitor")  
  
msgp=CreateObject("roMessagePort")  
diskmon.Setport(msgp)
```

roHashGenerator

ON THIS PAGE

- [ifHashGenerator](#)
 - [Hash\(obj As Object\) As Object](#)
 - [SetHmacKey\(key As Dynamic\) As Boolean](#)
 - [SetObfuscatedHmacKey\(key As String\) As Boolean](#)
 - [GetFailureReason\(\) As String](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)

- [Version 6.1](#)
- [Previous Versions](#)

This object allows you to generate a variety of message digests.

Object Creation: The hashing algorithm is specified when creating the *roHashGenerator* object.

```
CreateObject("roHashGenerator", algorithm As String)
```

The algorithm parameter accepts the following strings:

- "SHA256"
- "SHA384"
- "SHA512"
- "SHA1"
- "MD5"
- "CRC32"

Note

CRC32 is only available on firmware versions 4.4.x or later.

ifHashGenerator

Hash(obj As Object) As Object

Hashes the payload, which can be supplied in the form of a string (or any object implementing *ifString*) or an *roByteArray*. The hash is returned as an *roByteArray*.

SetHmacKey(key As Dynamic) As Boolean

Supplies a cryptographic key for the hashing function. This method accepts a plain-text key.

SetObfuscatedHmacKey(key As String) As Boolean

Supplies a cryptographic key for the hashing function. This method accepts a key that is obfuscated using a shared secret.

GetFailureReason() As String

roPassKey

ON THIS PAGE

- [ifPassKey](#)
 - [GenerateKey\(password As Object, salt As Object\) As roByteArray](#)
 - [GenerateSalt\(length As Integer\) As roByteArray](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object provides a means for generating keys (hashes) from a password and salt.

Object Creation: The object is passed an associative array that specifies the generation methods and cipher.

```
CreateObject("roPassKey", parameters As roAssociativeArray)
```

The associative array should contain the following parameters:

- **method**: The key derivation method. Currently, only "pbkdf2" can be specified.
- **keyfn**: The pseudorandom function (PRF). Currently, only "hmac-sha256" can be specified.
- **keylen**: The key length
- **iterations**: The number of iterations

ifPassKey

GenerateKey(password As Object, salt As Object) As roByteArray

Generates a key using the supplied password and salt. The parameters may be passed as either strings or *roByteArray* instances. The generated *roByteArray* instance may contain all possible byte values, including NUL.

GenerateSalt(length As Integer) As roByteArray

Generates a salt of the specified length. This salt can be used when calling the `GenerateKey()` method. The generated *roByteArray* instance may contain all possible byte values, including NUL.

Example

```
' Create input test data
salt = CreateObject("roByteArray")
pass = CreateObject("roByteArray")
pass.FromAsciiString("password")
salt.FromAsciiString("salt")
' Create the key generator
pk = CreateObject("roPassKey", { method: "pbkdf2", keyfn: "hmac-sha256", keylen: 32,
iterations: 4096 } )
' key will be a roByteArray
key = pk.GenerateKey(pass, salt)
```

roRegistry

ON THIS PAGE

- [ifRegistry](#)
 - [GetSectionList\(\) As roList](#)
 - [Delete\(section As String\) As Boolean](#)
 - [Flush\(\) As Boolean](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

The registry is an area of memory where a small number of persistent settings can be stored. Access to the registry is available through the *roRegistry* object.

This object is created with no parameters:

```
CreateObject("roRegistry")
```

ifRegistry

GetSectionList() As roList

Returns a list with one entry for each registry section.

Delete(section As String) As Boolean

Deletes the specified section and returns an indication of success.

Flush() As Boolean

Flushes the registry out to persistent storage.

roRegistrySection

ON THIS PAGE

- [ifRegistrySection](#)
 - [Read\(key As String\) As String](#)
 - [Write\(key As String, value As String\) As Boolean](#)
 - [Delete\(key As String\) As Boolean](#)
 - [Exists\(key As String\) As Boolean](#)
 - [Flush\(\) As Boolean](#)
 - [GetKeyList\(\) As roList](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object represents a section of the registry, enabling the organization of settings within the registry. It allows the section to be read or written.

Object Creation: This object must be supplied with a registry-section name upon creation.

```
CreateObject("roRegistrySection", section As String)
```

Writes do not always take effect immediately to prevent the system from exceeding the maximum number of writes on the onboard persistent storage. At most, 60 seconds after a write to the registry, the newly written data will be automatically written out to persistent storage. If, for some reason, the change must be written immediately, then the `Flush()` method should be called. All changes will be written automatically prior to exiting the application.

ifRegistrySection

Read(key As String) As String

Reads and returns the value of the specified key. Performing `Read()` on an entry that does not exist, or on a key within a section that does not exist, will return an empty string (`""`).

Write(key As String, value As String) As Boolean

Replaces the value of the specified key.

Delete(key As String) As Boolean

Deletes the specified key.

Exists(key As String) As Boolean

Returns True if the specified key exists.

Flush() As Boolean

Flushes the contents of the registry out to persistent storage.

GetKeyList() As roList

Returns a list containing one entry per registry key in this section.

Example

```
registrySection = CreateObject("roRegistrySection", "widget-usage")
' An empty entry will read as an empty string and therefore be converted to zero.
hits = val(registrySection.Read("big-red-button-hits"))
hits = hits + 1
registrySection.Write("big-red-button-hits", strI(hits))
```

roSqliteDatabase

ON THIS PAGE

- [ifSqliteDatabase](#)
 - [Open\(path As String\) As Boolean](#)
 - [Create\(path As String\) As Boolean](#)
 - [Close\(\)](#)
 - [CreateStatement\(sql_text As String\) As Object](#)
 - [RunBackground\(sql_text As String, associative_array As Object\) As Integer](#)
 - [SetMemoryLimit\(limit As Integer\)](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This is the main SQLite object that "owns" the database. You can create as many of these objects as you need.

`ifSqliteDatabase`

Open(path As String) As Boolean

Opens an existing database file. This method returns True upon success.

Create(path As String) As Boolean

Creates a new, empty database file. This method returns True upon success.

Passing " :memory: " to this method creates a database in non-persistent storage. This is useful for high-volume database applications that might cause wear issues on an SD card.

Close()

Closes an open database.

CreateStatement(sql_text As String) As Object

Creates a new *roSQLiteStatement* object using the specified SQL string.

RunBackground(sql_text As String, associative_array As Object) As Integer

Runs the specified SQL statement in the background and binds variables using the passed *roAssociativeArray*.

SetMemoryLimit(limit As Integer)

Sets the "soft" memory limit under which SQLite will attempt to remain (see the SQLite documentation for details).

The `SetMemoryLimit()` method sets global parameters. It must, therefore, be called before any other calls are made on the database object.

`ifMessagePort`

SetPort(port As roMessagePort)

Posts messages of type *roSQLiteEvent* to the attached message port.

Example: Creating a Database

```
db = CreateObject("roSQLiteDatabase")

print db

openResult = db.Create("SD:/test.db")

if openResult
    print "Created OK"
else
    print "Creation FAILED"
end
endif
```

Example: Creating a Table in a Database

```
createStmt = db.CreateStatement("CREATE TABLE playback (md5 text PRIMARY KEY, path PATH,
playback_count INT);")

print createStmt

if type(createStmt) <> "roSQLiteStatement" then
    print "We didn't get a statement returned!!"
end
endif

sqlResult = createStmt.Run()

print sqlResult

if sqlResult = SQLITE_COMPLETE
    print "Table Created OK"
else
    print "Table Creation FAILED"
```

```
endif  
  
createStmt.Finalise()
```

roSQLiteEvent

ON THIS PAGE

- [ifSQLiteEvent](#)
 - [GetTransactionId\(\) As Integer](#)
 - [GetSqlResult\(\) As Integer](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This event object is returned when a `RunBackground()` operation is called by the associated `roSQLiteDatabase` object.

`ifSQLiteEvent`

GetTransactionId() As Integer

Returns an integer that matches the result of the originating `RunBackground()` operation.

GetSqlResult() As Integer

Returns the result code returned by the `roSQLiteStatement.Run()` method. The possible return values are identical to the `Run()` method:

- 100: Statement complete
- 101: Busy
- 102: Rows available

Note

This method can be used as the asynchronous alternative to the `Run()` method.

roSQLiteStatement

ON THIS PAGE

- [ifSQLiteStatement](#)
 - [BindByName\(associative_array As Object\) As Boolean](#)
 - [BindByOffset\(associative_array/enumerable As Object\) As Boolean](#)
 - [BindText\(variable/index As Object, value As String\) As Boolean](#)
 - [BindInteger\(variable/index As Object, value As Integer\) As Boolean](#)
 - [Run\(\) As Integer](#)
 - [RunBackground\(\) As Integer](#)
 - [GetData\(\) As Object](#)
 - [Finalise\(\)](#)

▼ Firmware Version 7.0

- [Version 7.0](#)

- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is created by calling the `CreateStatement()` method on an `roSQLiteDatabase` object.

`ifSQLiteStatement`

All *bind* methods return True upon success.

BindByName(associative_array As Object) As Boolean

Binds the SQL variable(s) using the names contained in the SQL statement.

BindByOffset(associative_array/enumerable As Object) As Boolean

Binds the SQL variable(s) using the index contained in the SQL statement. If passed an associative array, this method will convert the keys of the associative array into numeric offsets when binding. If passed an enumerable object (e.g. `roArray`), it will bind the values of the enumerable in the order that they are stored.

BindText(variable/index As Object, value As String) As Boolean

Binds the SQL variable indicated by the name or index parameter to the passed string value.

BindInteger(variable/index As Object, value As Integer) As Boolean

Binds the SQL variable indicated by the name or index parameter to the passed integer value.

Run() As Integer

Runs the SQL statement immediately and waits for the integer result. The following are possible integer result codes:

- 100: Statement complete
- 101: Busy
- 102: Rows available

RunBackground() As Integer

Runs the SQL statement in the background. You can use `roSQLiteDatabase.SetPort()` to set a message port that will receive an `roSQLiteEvent` message at a later point. The `RunBackground()` call will result in an integer transaction ID, which will appear in the `roSQLiteEvent` message that matches the transaction.

GetData() As Object

Returns an associative array of name/value pairs that are available after a SELECT (or similar) operation.

Finalise()

Finalizes the statement. This method should be applied to statements before the parent database is closed. The object should not be used after this method is called. Also note that objects are automatically finalized when they are deleted.

The following script inserts into a table using the `BindByName()` method:

```
insertStmt = db.CreateStatement("INSERT INTO playback (md5,path,playback_count) VALUES(:
md5_param,:path_param,:pc_param);")

print insertStmt

if type(insertStmt) <> "roSQLiteStatement" then
    print "We didn't get a statement returned!!"
end
endif
```

```

params = { md5_param: "ABDEF12346", path_param: "/foo/bar/bing/bong", pc_param: 11 }

bindResult = insertStmt.BindByName(params)

if bindResult
    print "BindByName OK"
else
    print "BindByName FAILED"
end
endif

sqlResult = insertStmt.Run()

print sqlResult

if sqlResult = SQLITE_COMPLETE
    print "Table Insertion OK"
else
    print "Table Insertion FAILED"
endif

insertStmt.Finalise()

```

The following script inserts into a table in the background:

```

' This examples assume you have set a message port on your roSqliteDatabase instance
'

insertStmt = db.CreateStatement("INSERT INTO playback (md5,path,playback_count) VALUES(:
md5_param,:path_param,:pc_param);")

print insertStmt

if type(insertStmt) <> "roSqliteDatabase" then
    print "We didn't get a statement returned!!"
end
endif

params = { md5_param: "ABDEF12348", path_param: "/foo/bar/bing/bong", pc_param: 13 }

bindResult = insertStmt.BindByName(params)

if bindResult
    print "BindByName OK"
else
    print "BindByName FAILED"
end
endif

expectedId = insertStmt.RunBackground()

e = mp.WaitMessage(10000)
if e <> invalid then
    if type(e) = "roSqliteDatabase" then
        transId = e.GetTransactionId()
        sqlResult = e.GetSqlResult()
        print transId
        print sqlResult
        if transId <> expectedId then
            print "Incorrect transaction Id"
        end
    endif
    if sqlResult <> SQLITE_COMPLETE then

```



```

        print "SQL Insertion Failed"
    end
endif
else
    print "RunBackground() - Wrong event - FAILED"
end
endif
else
    print "RunBackground() - No Response - FAILED"
end
endif
' You don't need to call Finalise() since that'll be done by the background processor.

```

The following script queries from a table:

```

selectStmt = db.CreateStatement("SELECT * FROM playback;")

if type(selectStmt) <> "roSqliteStatement" then
    print "We didn't get a statement returned!!"
end
endif

sqlResult = selectStmt.Run()

print sqlResult

while sqlResult = SQLITE_ROWS
    resultsData = selectStmt.GetData()
    print resultsData;
    sqlResult = selectStmt.Run()
end while

selectStmt.Finalise()

```

roStorageAttached, roStorageDetached

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifString](#)
 - [GetString\(\) As String](#)
 - [SetString\(a As String\)](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

These event objects are generated by the *roStorageHotplug* object whenever a storage device becomes attached or detached from the player.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via SetUserData(). It will return Invalid if no data has been set.

ifString

GetString() As String

SetString(a As String)

roStorageHotplug

ON THIS PAGE

- [ifStorageHotplug](#)
 - [GetStorageStatus\(drive As String\) As roAssociativeArray](#)
- [ifFailureReason](#)
 - [GetFailureReason\(\) As String](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object provides *roStorageAttached* messages when storage devices appear and *roStorageDetached* messages when storage devices disappear. An *roStorageAttached* event is not delivered until the filesystem is mounted.

There is currently no way to poll for media.

Object Creation: The *roStorageHotplug* object is created with no parameters.

```
CreateObject("roStorageHotplug")
```

ifStorageHotplug

GetStorageStatus(drive As String) As roAssociativeArray

Returns the current status of a storage device ("SD:", "SSD:", "USB:"), even if it is not mounted. This method returns an associative array containing the following:

Key	Value Type	Description
present	Boolean	Indicates whether the specified device is present.
mounted	Boolean	Indicates whether the specified device is mounted.
	Boolean	Indicates whether the specified device is

corrupt		believed to be corrupt.
checking	Boolean	Indicates whether the specified device is currently being checked.

Note

The results of the `GetStorageStatus()` method are unreliable when called with a "USBn:" parameter, where "n" is a positive integer indicating a USB storage device when multiple USB devices are connected to the player.

Example

```
status = CreateObject("roStorageHotplug").GetStorageStatus("SSD:")
if not status.mounted and not status.checking and status.present then
    ' Should ask for confirmation here
    FormatDrive("SSD:")
end if
```

`ifFailureReason`

GetFailureReason() As String

Returns additional diagnostic information if a method returns False.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return Invalid if no data has been set.

`ifMessagePort`

SetPort(port As roMessagePort)

Posts messages of type `roStorageAttached` and `roStorageDetached` to the attached message port.

In order to avoid race conditions at startup, you should check for any storage devices that might have existed prior to the message port being set. We recommend doing this after the object is created and the message port is set, but before instructing the script to wait for any events.

Example

```
Sub Main()
    mp = CreateObject("roMessagePort")
    sh = CreateObject("roStorageHotplug")
    gpio = CreateObject("roControlPort", "brightsign")

    sh.SetPort(mp)
    gpio.SetPort(mp)

    finished = false
    while not finished
        ev = mp.WaitMessage(0)
        if type(ev) = "roControlDown"
            finished = true
        end if
    end while
End Sub
```

```

else if type(ev) = "roStorageAttached"
    print "ATTACHED "; ev.GetString()
else if type(ev) = "roStorageDetached"
    print "DETACHED "; ev.GetString()
else
    print type(ev)
    stop
end if
end while
End Sub

```

roStorageInfo

ON THIS PAGE

- [Drive Specifications](#)
- [ifStorageInfo](#)
 - [GetFailureReason\(\) As String](#)
 - [GetBytesPerBlock\(\) As Integer](#)
 - [GetSizeInMegabytes\(\) As Integer](#)
 - [GetUsedInMegabytes\(\) As Integer](#)
 - [GetFreeInMegabytes\(\) As Integer](#)
 - [GetFileSystemType\(\) As String](#)
 - [GetStorageCardInfo\(\) As roAssociativeArray](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is used to report usage information for storage devices.

Object Creation: The *roStorageInfo* object is created with a parameter that specifies the path of the storage device. The path does not need to extend to the root of the storage device.

```
CreateObject("roStorageInfo", path As String)
```

Drive Specifications

Use the following string values to specify different storage drives:

- "USB1:" – The drive for USB storage devices connected to the player
- "SD:" – The primary SD or microSD drive on the player.
- "SD2:" – The internal microSD drive on the player (4Kx42, XDx32 models only)
- "SSD:" – The internal SSD on the player (XTx44, XTx43, XDx34, and XDx33 models only)

ifStorageInfo

GetFailureReason() As String

Yields additional useful information if a function return indicates an error.

GetBytesPerBlock() As Integer

Returns the size of a native block on the filesystem used by the specified storage device.

GetSizeInMegabytes() As Integer

Returns the total size (in mebibytes) of the storage device.

Important

On some filesystems that have a portion of space reserved for the super user, the following expression may not be true: `GetUsedInMegabytes() + GetFreeInMegabytes() == GetSizeInMegabytes()`

GetUsedInMegabytes() As Integer

Returns the amount (in mebibytes) of space currently used on the storage device. This amount includes the size of the pool because this class does not integrate pools into its calculations.

GetFreeInMegabytes() As Integer

Returns the available space (in mebibytes) on the storage device.

GetFileSystemType() As String

Returns a string describing the type of filesystem used on the specified storage. The following are potential values:

- "exfat"
- "ext3"
- "ext4"
- "fat12"
- "fat16"
- "fat32"
- "hfs"
- "hfsplus"
- "ntfs"

GetStorageCardInfo() As roAssociativeArray

Returns an associative array containing details of the storage device hardware. For SD cards, the returned data may include the following:

<code>sd_mfr_id</code>	Integer	Card manufacturer ID as assigned by the SD Card Association
<code>sd_oem_id</code>	String	Two-character card OEM identifier as assigned by the SD Card Association
<code>sd_product_name</code>	String	Product name, assigned by the card manufacturer (5 bytes for SD, 6 bytes for MMC)
<code>sd_spec_vers</code>	Integer	Version of SD spec to which the card conforms
<code>sd_product_rev</code>	String	Product revision assigned by the card manufacturer
<code>sd_speed_class</code>	String	Speed class (if any) declared by the card
<code>sd_au_size</code>	Integer	Size of the SD AU in bytes.

Example

```
si=CreateObject("roStorageInfo", "SD:/")
Print si.GetFreeInMegabytes(); "MiB free"
```

roVirtualMemory

ON THIS PAGE

- [ifVirtualMemory](#)
 - [AddSwapFile\(parameters As roAssociativeArray\) As Boolean](#)
 - [RemoveSwapFile\(filename As String\) As Boolean](#)
- [ifFailureReason](#)
 - [GetFailureReason\(\) As String](#)

The *roVirtualMemory* object allows you to create a virtual memory repository for the Chromium HTML engine. This allows Chromium to automatically swap unused elements out of physical memory.

This object is only available on platforms that can support SSD/mSATA drives: XTx44, XTx43, XDx34, XDx33, and 4Kx42.

Object Creation: This object is instantiated without parameters.

```
CreateObject("roVirtualMemory")
```

ifVirtualMemory

AddSwapFile(parameters As roAssociativeArray) As Boolean

Adds a virtual memory repository to local storage. This method accepts an associative array with the following parameters:

- `filename:(string)` The URI of the swap file to be created. It is possible to specify a file location other than the SSD/mSATA drive, but it is not recommended.
- `megabytes:(Integer)` The size of the swap file to be created.

If the specified file does not already exist, it will be created. If it already exists, the size will be modified. Either way, this operation can take some time, but it should only need to be called once.

RemoveSwapFile(filename As String) As Boolean

Removes a swap file with the specified filename.

ifFailureReason

GetFailureReason() As String

Returns additional useful information when a method on the *ifVirtualMemory* interface returns False.

Content Management Objects

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This section describes objects that enable downloading, storage, and retrieval of content from a remote CMS.

- [roAssetCollection](#)
- [roAssetFetcher](#)
- [roAssetFetcherEvent](#)
- [roAssetFetcherProgressEvent](#)

- [roAssetPool](#)
- [roAssetPoolFiles](#)
- [roAssetRealizer](#)
- [roAssetRealizerEvent](#)
- [roSyncSpec](#)

roAssetCollection

ON THIS PAGE

- [ifAssetCollection](#)
 - [GetFailureReason\(\) As String](#)
 - [AddAsset\(asset_info As Dynamic\) As Boolean](#)
 - [AddAssets\(asset_info_array As Dynamic\) As Boolean](#)
 - [GetAssetList\(\) As roList](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is used to represent a collection of assets.

Object Creation: The *roAssetCollection* object is created with no parameters.

```
CreateObject("roAssetCollection")
```

You can populate an asset collection with individual calls to `AddAsset()` or `AddAssets()`. You can also populate an asset collection using the *roSyncSpec*.`GetAssets()` method, as shown below:

```
assetCollection = CreateObject("roAssetCollection")

localCurrentSync = CreateObject("roSyncSpec")
if localCurrentSync.ReadFromFile("local-sync.xml") then
    assetCollection = localCurrentSync.GetAssets("download")
endif
```

ifAssetCollection

GetFailureReason() As String

AddAsset(asset_info As Dynamic) As Boolean

Adds a single asset from an associative array or JSON-formatted string. If the argument is a string, it should match the format used for a single asset in a JSON sync spec.

AddAssets(asset_info_array As Dynamic) As Boolean

Adds multiple assets from an enumerable object (an *roList*, *roArray*, or string containing a JSON-formatted array) that contains compatible associative arrays.

GetAssetList() As roList

Returns an *roList* instance containing associative arrays of asset metadata. This method is not efficient and is, therefore, recommended for debugging and diagnostic purposes only.

Each associative array contains the following:

name	String	Required	The name of the asset. For a file to be realized, it must have a valid filename (i.e. no slashes).
link	String	Required	The download location of the asset
size	Integer/String	Optional	The size of the asset. Use a string if you want to specify a number that is too large to fit into an integer (this allows file sizes larger than 2 GB).
hash	String	Optional	A string in the form of "hash_algorithm:hash". See the next table for available hash algorithms.
change_hint	String	Optional	Any string that will change in conjunction with the file contents. This is not necessary if the link or hash is supplied and always changes.
auth_inherit	Boolean	Optional	Indication of whether or not this asset uses <i>roAssetFetcher</i> authentication information. The default is set to True.
auth_user	Boolean	Optional	User to utilize for authentication when downloading only this asset. This automatically disables "auth_inherit".
auth_password	Boolean	Optional	Password to use when downloading only this asset. This automatically disables "auth_inherit".
headers_inherit	Boolean	Optional	The command to pass any header supplier to <i>roAssetFetcher</i> when fetching this asset. The default is true.

Important

Any "optional" fields that are specified when populating the pool must also be specified when retrieving assets from the pool (i.e. they become "mandatory" once they are used for an asset). For example, if the hash value is specified when fetching into the pool, then it must also be specified when attempting to refer to files in the pool.

Hash algorithms:

sha1	If a sha1 is available, you can validate the hash as the file is downloaded. If such a hash is available, it should be used. The link and change_hint properties have no effect on the pool file name, so the file is shared even if it is downloaded from different locations.
besha1	This algorithm hashes some of the file along with the file size in order to verify the contents. It also moves the link and change_hint properties into the pool filename.
MD5	Uses the MD5 hash algorithm to validate files.
(none)	Without any hash, the file cannot be verified as it is downloaded, and the system will rely on the link and change_hint properties to give the pool a unique filename.

roAssetFetcher

ON THIS PAGE

- [ifAssetFetcher](#)

- `GetFailureReason()` As String
- `SetUserAndPassword(user As String, password As String)` As Boolean
- `EnableUnsafeAuthentication(enable As Boolean)` As Boolean
- `EnableUnsafeProxyAuthentication(enable As Boolean)` As Boolean
- `EnableEncodings(enable As Boolean)` As Boolean
- `AsyncDownload(assets As roAssetCollection)` As Boolean
- `AsyncSuggestCache(a As Object)` As Boolean
- `AsyncCancel()` As Boolean
- `EnablePeerVerification(verification As Boolean)` As Boolean
- `EnableHostVerification(verification As Boolean)` As Boolean
- `SetCertificatesFile(filename As String)` As Boolean
- `AddHeader(name As String, value As String)` As Boolean
- `SetHeaders(headers As roAssociativeArray)` As Boolean
- `SetProxy(proxy As String)` As Boolean
- `SetProxyBypass(hostnames As Array)` As Boolean
- `SetFileProgressIntervalSeconds(interval As Integer)` As Boolean
- `SetFileRetryCount(count As Integer)` As Boolean
- `SetRelativeLinkPrefix(prefix As String)` As Boolean
- `BindToInterface(interface As Integer)` As Boolean
- `SetMaximumReceiveBytesPerSecond(bytes_per_second As Double)` As Boolean
- `SetMinimumTransferRate(bytes_per_second As Integer, period_in_seconds As Integer)` As Boolean
- `ifMessagePort`
 - `SetPort(port As roMessagePort)`
- `ifUserData`
 - `SetUserData(user_data As Object)`
 - `GetUserData()` As Object

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object contains functions for downloading files to the pool.

Object Creation: The `roAssetFetcher` object must be passed an `roAssetPool` instance upon creation.

```
CreateObject("roAssetFetcher", pool As roAssetPool)
```

Example

```
Pool = CreateObject("roAssetPool", "pool")
Fetcher = CreateObject("roAssetFetcher", Pool)
```

`ifAssetFetcher`

***GetFailureReason()* As String**

Returns an error string if an `roAssetFetcher` method has failed (this is usually indicated by returning `False`). The error string may help diagnose the failure.

***SetUserAndPassword(user As String, password As String)* As Boolean**

Sets the default user and password strings to be used for all download requests that are not otherwise marked using the following attributes: `<auth inherit="no">` or `<auth user="user" password="password">`.

***EnableUnsafeAuthentication(enable As Boolean)* As Boolean**

Supports basic HTTP authentication if True. HTTP authentication uses an insecure protocol, which might allow others to easily determine the password. The *roAssetFetcher* object will still prefer the stronger digest HTTP if it is supported by the server. If this method is False (which is the default setting), it will refuse to provide passwords via basic HTTP authentication, and any requests requiring this authentication will fail.

EnableUnsafeProxyAuthentication(enable As Boolean) As Boolean

Supports basic HTTP authentication against proxies if True (which, unlike *EnableUnsafeAuthentication()*, is the default setting). HTTP authentication uses an insecure protocol, which might allow others to easily determine the password. If this method is False, it will refuse to provide passwords via basic HTTP authentication, and any requests requiring this authentication type will fail.

EnableEncodings(enable As Boolean) As Boolean

Enables HTTP compression, which communicates to the server that the system can accept any encoding that the *roAssetFetcher* object is capable of decoding by itself (this behavior is enabled by default). Supported encodings currently include "deflate" and "gzip", which allow for transparent compression of responses. Clients of the *roAssetFetcher* instance see only the decoded data and are unaware of the encoding being used.

AsyncDownload(assets As roAssetCollection) As Boolean

Begins populating the asset pool using the files listed in the passed *roAssetCollection* instance. Files that are not already in the pool will be downloaded automatically. Events are raised during the download process to indicate whether individual file downloads have succeeded or failed; finally, a single event will be raised indicating whether the entire asset collection has been downloaded successfully or not. See the *roAssetFetcherEvent* and *roAssetFetcherProgressEvent* entries for more details.

AsyncSuggestCache(a As Object) As Boolean

AsyncCancel() As Boolean

Cancels any pending "Async" requests. Note that, prior to and during this method call, events associated with an asynchronous action may be queued. No more events will be queued once this call returns. We recommend collecting any pending events prior to calling any further "Async" methods on the same object to avoid confusion.

EnablePeerVerification(verification As Boolean) As Boolean

Enables checking of TLS/SSL certificates. This method is set to `true` by default. Disabling peer verification allows you to bypass an expired certificate check.

EnableHostVerification(verification As Boolean) As Boolean

Enables checking of the TLS/SSL certificate for the correct hostname. This method is set to `true` by default. Disabling host verification allows you to accept a certificate being sent for the wrong hostname.

Important

Peer verification and host verification are important security checks that prevent "man-in-the-middle" attacks. These features should only be disabled after careful consideration of the security implications.

SetCertificatesFile(filename As String) As Boolean

Configures an alternative set of CA certificates for the connection. This method is useful if the connection certificates are signed by a CA that is not on the the default trusted list (for example, if your organization uses a private CA hierarchy that is not signed by a well known root CA). This method replaces the default list, so the passed certificate file must contain all acceptable CA certificates required for the connection.

AddHeader(name As String, value As String) As Boolean

Specifies a header that will be passed to HTTP requests made by the *roAssetFetcher* instance. A particular download will not include the header if it has the `<headers inherit="no">` attribute in the sync spec.

SetHeaders(headers As roAssociativeArray) As Boolean

Specifies all headers that will be passed to HTTP requests made by the *roAssetFetcher* instance. This method removes any previously set headers. A particular download will not include the headers if it has the `<headers inherit="no">` attribute in the sync spec.

SetProxy(proxy As String) As Boolean

Sets the name or address of the proxy server that will be used by the *roAssetFetcher* instance. The proxy string should be formatted as "http://user:password@hostname:port". It can contain up to four "*" characters; each "*" character can be used to replace one octet from the current IP address. For example, if the IP address is currently 192.168.1.2, and the proxy is set to "proxy-*-**", then the player will attempt to use a proxy named "proxy-192.168".

SetProxyBypass(hostnames As Array) As Boolean

Exempts the specified hosts from the proxy setting. The passed array should consist of one or more hostnames. The *roAssetFetcher* instance will attempt to reach the specified hosts directly rather than using the proxy that has been specified with the `SetProxy()` method. For example, the hostname "example.com" would exempt "example.com", "example.com:80", and "www.example.com" from the proxy setting.

SetFileProgressIntervalSeconds(interval As Integer) As Boolean

Specifies the interval (in seconds) between progress events when an individual file is being downloaded. Setting the interval to -1 disables all progress events. Setting the interval to 0 specifies that events should be generated as often as possible, though this will slow down the transfer process. If the interval is set to 0 or any positive integer, events will always be generated at the start and end of the file download irrespective of elapsed time. The default interval is 300 seconds.

SetFileRetryCount(count As Integer) As Boolean

Specifies the maximum number of times each file download will be retried before moving on to the next file download. The default retry count is five.

SetRelativeLinkPrefix(prefix As String) As Boolean

Specifies a prefix that will be appended to the front of relative links in the sync spec. Normally, this method is used to make file:/// URIs drive agnostic, but it can also be used to reduce the size of the sync spec if all files are stored in the same place. Non-relative links are not affected by this method.

BindToInterface(interface As Integer) As Boolean

Ensures that the HTTP request goes out over the specified network interface (0 for Ethernet or 1 for WiFi). The default behavior (which can be specified by passing -1) is to send requests using the most appropriate network interface, which may depend on the routing metric configured via the *roNetworkConfiguration* object. If both interfaces are on the same layer 2 network, this method may not work as expected due to the Linux weak-host model.

SetMaximumReceiveBytesPerSecond(bytes_per_second As Double) As Boolean

Limits the rate at which downloads are performed by the *roAssetFetcher* instance. The source data rate isn't under the direct control of the BrightSign player, but download rates should average below the specified value over time.

This method returns `true` on success and `false` on failure. In event of failure, the `GetFailureReason()` method may provide more information.

SetMinimumTransferRate(bytes_per_second As Integer, period_in_seconds As Integer) As Boolean

Sets the minimum transfer rate for each file download. A transfer will be terminated if the rate drops below *bytes_per_second* when averaged over *period_in_seconds*. Note that if the transfer is over the Internet, you may not want to set *period_in_seconds* to a small number in case network problems cause temporary drops in performance. For large file transfers and a small *bytes_per_second* limit, averaging fifteen minutes or more may be appropriate.

`ifMessagePort`

SetPort(port As roMessagePort)

Posts messages of type *roAssetFetcherEvent* and *roAssetFetcherProgressEvent* to the attached message port.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

roAssetFetcherEvent

ON THIS PAGE

- [ifAssetFetcherEvent](#)
 - [GetEvent\(\) As Integer](#)
 - [GetName\(\) As String](#)
 - [GetFailureReason\(\) As String](#)
 - [GetFileIndex\(\) As Integer](#)
 - [GetResponseCode\(\) As Integer](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This event is generated by an *roAssetFetcher* instance when a file transfer succeeds or fails or when population of the asset pool as a whole succeeds or fails.

`ifAssetFetcherEvent`

GetEvent() As Integer

Returns an integer indicating the result of an *roAssetFetcher* download attempt:

- 1: POOL_EVENT_FILE_DOWNLOADED
- -1: POOL_EVENT_FILE_FAILED
- 2: POOL_EVENT_ALL_DOWNLOADED
- -2: POOL_EVENT_ALL_FAILED

GetName() As String

GetFailureReason() As String

Returns additional failure information associated with the event (if any).

GetFileIndex() As Integer

Retrieves the zero-based index from the sync spec of the file associated with the event.

GetResponseCode() As Integer

Returns the protocol response code associated with an event. The following codes indicate success:

- 200: Successful HTTP transfer
- 226: Successful FTP transfer
- 0: Successful local file transfer

For unexpected errors, the return value is negative. There are many possible negative errors from the CURL library, but it is often best to look at the text version by calling `GetFailureReason()`.

Here are some potential errors. Not all of them can be generated by a BrightSign player:

Status	Name	Description
-1	CURLE_UNSUPPORTED_PROTOCOL	
-2	CURLE_FAILED_INIT	

-3	CURLE_URL_MALFORMAT	
-5	CURLE_COULDNT_RESOLVE_PROXY	
-6	CURLE_COULDNT_RESOLVE_HOST	
-7	CURLE_COULDNT_CONNECT	
-8	CURLE_FTP_WEIRD_SERVER_REPLY	
-9	CURLE_REMOTE_ACCESS_DENIED	A service was denied by the server due to lack of access. When login fails, this is not returned.
-11	CURLE_FTP_WEIRD_PASS_REPLY	
-13	CURLE_FTP_WEIRD_PASV_REPLY	
-14	CURLE_FTP_WEIRD_227_FORMAT	
-15	CURLE_FTP_CANT_GET_HOST	
-17	CURLE_FTP_COULDNT_SET_TYPE	
-18	CURLE_PARTIAL_FILE	
-19	CURLE_FTP_COULDNT_RETR_FILE	
-21	CURLE_QUOTE_ERROR	Failed quote command
-22	CURLE_HTTP_RETURNED_ERROR	
-23	CURLE_WRITE_ERROR	
-25	CURLE_UPLOAD_FAILED	Failed upload command.
-26	CURLE_READ_ERROR	Could not open/read from file.
-27	CURLE_OUT_OF_MEMORY	
-28	CURLE_OPERATION_TIMEDOUT	The timeout time was reached.
-30	CURLE_FTP_PORT_FAILED	FTP PORT operation failed.
-31	CURLE_FTP_COULDNT_USE_REST	REST command failed.
-33	CURLE_RANGE_ERROR	RANGE command did not work.
-34	CURLE_HTTP_POST_ERROR	
-35	CURLE_SSL_CONNECT_ERROR	Wrong when connecting with SSL.
-36	CURLE_BAD_DOWNLOAD_RESUME	Could not resume download.
-37	CURLE_FILE_COULDNT_READ_FILE	
-38	CURLE_LDAP_CANNOT_BIND	
-39	CURLE_LDAP_SEARCH_FAILED	
-41	CURLE_FUNCTION_NOT_FOUND	
-42	CURLE_ABORTED_BY_CALLBACK	
-43	CURLE_BAD_FUNCTION_ARGUMENT	
-45	CURLE_INTERFACE_FAILED	CURLOPT_INTERFACE failed.
-47	CURLE_TOO_MANY_REDIRECTS	Catch endless re-direct loops.
-48	CURLE_UNKNOWN_TELNET_OPTION	User specified an unknown option.
-49	CURLE_TELNET_OPTION_SYNTAX	Malformed telnet option.
-51	CURLE_PEER_FAILED_VERIFICATION	Peer's certificate or fingerprint wasn't verified correctly.

-52	CURLE_GOT_NOTHING	When this is a specific error.
-53	CURLE_SSL_ENGINE_NOTFOUND	SSL crypto engine not found.
-54	CURLE_SSL_ENGINE_SETFAILED	Cannot set SSL crypto engine as default.
-55	CURLE_SEND_ERROR,	Failed sending network data.
-56	CURLE_RECV_ERROR	Failure in receiving network data.
-58	CURLE_SSL_CERTPROBLEM	Problem with the local certificate.
-59	CURLE_SSL_CIPHER	Could not use specified cipher.
-60	CURLE_SSL_CACERT	Problem with the CA cert (path?)
-61	CURLE_BAD_CONTENT_ENCODING	Unrecognized transfer encoding.
-62	CURLE_LDAP_INVALID_URL	Invalid LDAP URL.
-63	CURLE_FILESIZE_EXCEEDED,	Maximum file size exceeded.
-64	CURLE_USE_SSL_FAILED,	Requested FTP SSL level failed.
-65	CURLE_SEND_FAIL_REWIND,	Sending the data requires a rewind that failed.
-66	CURLE_SSL_ENGINE_INITFAILED	Failed to initialize ENGINE.
-67	CURLE_LOGIN_DENIED	User, password, or similar field was not accepted and login failed .
-68	CURLE_TFTP_NOTFOUND	File not found on server.
-69	CURLE_TFTP_PERM	Permission problem on server.
-70	CURLE_REMOTE_DISK_FULL	Out of disk space on server.
-71	CURLE_TFTP_ILLEGAL	Illegal TFTP operation.
-72	CURLE_TFTP_UNKNOWNID	Unknown transfer ID.
-73	CURLE_REMOTE_FILE_EXISTS	File already exists.
-74	CURLE_TFTP_NOSUCHUSER	No such user.
-75	CURLE_CONV_FAILED	Conversion failed.
-76	CURLE_CONV_REQD	Caller must register conversion callbacks using the following URL_easy_setopt options: CURLOPT_CONV_FROM_NETWORK_FUNCTION CURLOPT_CONV_TO_NETWORK_FUNCTION CURLOPT_CONV_FROM_UTF8_FUNCTION
-77	CURLE_SSL_CACERT_BADFILE	Could not load CACERT file, missing or wrong format.
-78	CURLE_REMOTE_FILE_NOT_FOUND	Remote file not found.
-79	CURLE_SSH	Error from the SSH layer (this is somewhat generic, so the error message will be important when this occurs).
-80	CURLE_SSL_SHUTDOWN_FAILED	Failed to shut down the SSL connection.

The following error codes are generated by the system software and are outside the range of CURL events:

Status	Name	Description

-1002	ENOENT	The specified file does not exist or cannot be created.
-10001	Cancelled	The operation has been cancelled.
-10002	Exception	The operation caused a local exception. Call <code>GetFailureReason()</code> for more details.
-10003	ERROR_EXCEPTION	An unexpected exception occurred.
-10004	ERROR_DISK_ERROR	A disk error occurred (usually as a result of the disk being full).
-10005	ERROR_POOL_UNSATISFIED	The expected files are not present in the pool.
-10006	ERROR_DOWNLOADING_ELSEWHERE	The file is being downloaded by another <i>roAssetFetcher</i> instance.
-10007	ERROR_HASH_MISMATCH	A downloaded file did not match its checksum or file size.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

roAssetFetcherProgressEvent

ON THIS PAGE

- [ifAssetFetcherProgressEvent](#)
 - [GetFileName\(\) As String](#)
 - [GetFileIndex\(\) As Integer](#)
 - [GetFileCount\(\) As Integer](#)
 - [GetCurrentFileTransferredMegabytes\(\) As Integer](#)
 - [GetCurrentFileSizeMegabytes\(\) As Integer](#)
 - [GetCurrentFilePercentage\(\) As Float](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\) As Void](#)
 - [GetUserData\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This event is generated by the *roAssetFetcher* object at regular intervals during file downloads. Use the *roAssetFetcher.SetFileProgressIntervalSeconds()* method to customize how often progress events are generated.

`ifAssetFetcherProgressEvent`

GetFileName() As String

Returns the name of the file associated with the event. The file name is retrieved from the sync spec associated with the *roAssetFetcher* that generated the event.

GetFileIndex() As Integer

Returns the zero-based index from the sync spec of the file associated with the event.

GetFileCount() As Integer

Returns the total number of files within the sync spec.

GetCurrentFileTransferredMegabytes() As Integer

Returns the number of transferred megabytes belonging to the file associated with the event.

GetCurrentFileSizeMegabytes() As Integer

Returns the size of the file associated with the event.

GetCurrentFilePercentage() As Float

Returns a floating-point number representing the download percentage of the file associated with the event.

Note

The file size/percentage returned by the methods above is derived from the `asset_size` in the associated `roAssetCollection` or sync spec—not from the HTTP "Content-Length" header. Specifying an `asset_size` of zero will cause these methods to return zero.

`ifUserData`

SetUserData(user_data As Object) As Void

Sets the user data.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()` (on either the event or source object). It will return `Invalid` if no data has been set.

roAssetPool

ON THIS PAGE

- [ifAssetPool](#)
 - [GetFailureReason\(\) As String](#)
 - [ProtectAssets\(name As String, collection As Object\) As Boolean](#)
 - [UnprotectAssets\(name As String\) As Boolean](#)
 - [UnprotectAllAssets\(\) As Boolean](#)
 - [ReserveMegabytes\(size As Integer\) As Boolean](#)
 - [SetMaximumPoolSizeMegabytes\(maximum_size As Integer\) As Boolean](#)
 - [GetPoolSizeInMegabytes\(\) As Integer](#)
 - [Validate\(sync_spec As Object, options As roAssociativeArray\) As Boolean](#)
 - [QueryFiles\(sync_spec As roAssociativeArray\) As roAssociativeArray](#)
 - [AssetsReady\(collection As Object\) As Boolean](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

An `roAssetPool` instance represents a pool of files for synchronization. You can instruct this object to populate the pool based on a sync spec and then realize it in a specified directory when required.

Object Creation: The `roAssetPool` object is created with a single parameter representing the rooted path of the pool.


```
CreateObject("roAssetPool", pool_path As String)
```

Example

```
pool = CreateObject ("roAssetPool", "SD:/pool")
```

ifAssetPool

GetFailureReason() As String

ProtectAssets(name As String, collection As Object) As Boolean

Requests that the files specified in the "download" section of a sync spec receive a certain amount of protection. Specified files will not be deleted when the system software needs to reduce the size of the pool to make space.

UnprotectAssets(name As String) As Boolean

Removes the protected status placed on the specified files by the `ProtectAssets()` method. **Asset collections** are reference counted at the system-software level. As a result, when calling `UnprotectAssets()`, you must pass the same object that you previously passed to `ProtectAssets()`.

UnprotectAllAssets() As Boolean

Removes the protected status placed on any files in the asset pool using the `ProtectAssets()` method.

ReserveMegabytes(size As Integer) As Boolean

Reserves the specified amount of storage space. This method is dynamic: The system software attempts to keep the space free even when parallel processes are filling up the storage.

SetMaximumPoolSizeMegabytes(maximum_size As Integer) As Boolean

Specifies the maximum size of an `roAssetPool` instance in megabytes. This method is more resource-intensive than `ReserveMegabytes()`, but it is useful when creating multiple pools on a storage device.

GetPoolSizeInMegabytes() As Integer

Returns the current size of the asset pool (in megabytes).

Validate(sync_spec As Object, options As roAssociativeArray) As Boolean

Checks the SHA1, BESHA1, or MD5 hash value of files that are in the sync spec and are currently present in the pool. This method returns `True` if all checks pass and `False` if one or more checks fail. Calling `GetFailureReason()` will return information about the corrupt file(s). Note that a `True` return may not mean that all files in the sync spec are currently present in the pool. The second parameter represents a table of validation options: The key specifies the option and the value specifies whether the option is enabled or not (as a Boolean value). Currently, the only option is "DeleteCorrupt", which determines whether the method should automatically delete corrupt files or not.

QueryFiles(sync_spec As roAssociativeArray) As roAssociativeArray

Determines whether the specified files are present in the asset pool. This method returns an associative array with filename keys and corresponding Boolean values.

AssetsReady(collection As Object) As Boolean

Returns `true` if the specified files are ready in the asset pool.

roAssetPoolFiles

ON THIS PAGE

- **ifAssetPoolFiles**
 - **GetFailureReason() As String**
 - **GetPoolFilePath(asset_name As String) As String**
 - **GetPoolFileInfo(asset_name As String) As Object**

▼ **Firmware Version 7.0**

- **Version 7.0**
- **Version 6.2**
- **Version 6.1**
- **Previous Versions**

Object Creation: The *roAssetPoolFiles* object is created with two parameters.

```
CreateObject("roAssetPoolFiles", pool As roAssetPool, assets As Dynamic)
```

The *assets* argument can be either an *roAssetCollection* or *roSyncSpec* object. If more than one object requires use of the *roAssetCollection* object, it will be more efficient to convert *roSyncSpec* to *roAssetCollection* by calling *GetAssets()* once and then passing that collection to all objects requiring it.

This object works similarly to the *roSyncPoolFiles* object.

ifAssetPoolFiles

GetFailureReason() As String

Returns explanatory text if *GetPoolFilePath()* returns an empty string or *GetPoolFileInfo()* returns Invalid.

GetPoolFilePath(asset_name As String) As String

Looks up the specified file name in the asset collection and uses the information to determine the actual name of the file in the pool. This method returns an empty string if the name is not found in the asset collection, or if the file is not found in the pool.

GetPoolFileInfo(asset_name As String) As Object

Looks up the specified file name in the asset collection and returns all available information, including the pool file path, as an associative array. This method returns Invalid if the asset name is not found in the asset collection. If the file is not found in the pool, information from the asset collection will be returned without the pool path. See the table below for a description of assets in the associative array.

Field	Value	Description
name	String	Asset name
link	String	Asset URL
size	String	
hash	String	Hash in algorithm ":" hash format
change_hint	String	Only present if set
auth_user	String	Only present if set
auth_password	String	Only present if set
auth_inherit	Boolean	
headers_inherit	Boolean	
probe	String	Probe data
path	String	Absolute path of the file in the pool (or "invalid" if the file is not in the pool)

roAssetRealizer

ON THIS PAGE

- [IfUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifAssetRealizer](#)
 - [GetFailureReason\(\) As String](#)
 - [EstimateRealizedSizeInMegabytes\(spec As Object\) As Integer](#)
 - [Realize\(spec As roSyncSpec/roAssetCollection\) As roAssetRealizerEvent](#)
 - [ValidateFiles\(spec As Object, options As Object\) As roAssociativeArray](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object contains functions for realizing files (i.e. extracting files from an asset pool and placing them into the standard file directory). Realizing a complete asset pool is not recommended because the process can be slow. Instead, the script should retrieve pool file locations using the *roAssetPoolFiles* object and use them to refer to files directly. The *roAssetRealizer* object is still useful for realizing files that must be in the root directory to work (e.g. autorun and firmware update files).

Object Creation: The *roAssetRealizer* object requires two parameters upon creation: an *roAssetPool* object and a destination directory.

```
CreateObject("roAssetRealizer", pool As roAssetPool, destination_directory As String)
```

Example

```
pool = CreateObject("roAssetPool", "pool")
realizer = CreateObject("roAssetRealizer", pool, "/")
```

IfUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return Invalid if no data has been set.

ifAssetRealizer

GetFailureReason() As String

Yields additional useful information if a function return indicates an error.

EstimateRealizedSizeInMegabytes(spec As Object) As Integer

Returns the estimated amount of space that would be taken up by the specified sync spec.

Realize(spec As roSyncSpec/roAssetCollection) As roAssetRealizerEvent

Places the files into the destination directory specified in the passed *roSyncSpec* or *roAssetCollection*. If the pool does not contain the full set of required files, then this method will immediately fail before any files are changed (this method attempts to do as much work as possible before destructively modifying the file system). This method automatically checks the length of the file and any hashes that match the specification. If

there is a mismatch, then the affected file is deleted and realization fails. This method indicates success or failure by returning an *roAssetRealizerEvent* object.

Note

The pool and the destination must be in the same file system.

ValidateFiles(spec As Object, options As Object) As roAssociativeArray

Checks the hash of every file in the spec against the corresponding file in the destination path and returns an associative array containing each mismatched file name mapped to the reason. The options parameter is an *roAssociativeArray*, which can currently support a single option:

- "DeleteCorrupt": Automatically deletes any files that do not match the expected hash. By default, these files are not deleted.

roAssetRealizerEvent

ON THIS PAGE

- [ifAssetRealizerEvent](#)
 - [GetEvent\(\) As Integer](#)
 - [GetName\(\) As String](#)
 - [GetResponseCode\(\) As Integer](#)
 - [GetFailureReason\(\) As String](#)
 - [GetFileIndex\(\) As Integer](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\) As Void](#)
 - [GetUserData\(\) As Object](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This event object is returned by the *roAssetRealizer.Realize()* method. It yields information about the success or failure of the realization process.

ifAssetRealizerEvent

GetEvent() As Integer

Returns an integer value indicating the type of the event:

101	EVENT_REALIZE_SUCCESS	The specified sync list was successfully realized.
-102	EVENT_REALIZE_INCOMPLETE	Realization could not begin because at least one of the required files is not available in the pool.
-103	EVENT_REALIZE_FAILED_SAFE	Realization has failed. Nothing has been written to the destination, so it is likely safe to continue the realization process. More information about the failure is available via the <i>GetFailureReason()</i> and <i>GetName()</i> methods.
-104	EVENT_REALIZE_FAILED_UNSAFE	Realization has failed while running, and changes have been made to destination files. It may not be safe to continue the realization process. More information about the failure is available via the <i>GetFailureReason()</i> and <i>GetName()</i> methods.

GetName() As String

Retrieves the name of the affected file if the realization process fails.

GetResponseCode() As Integer

Retrieves the *roUrlTransfer* response code associated with the event (if any).

GetFailureReason() As String

Returns additional information if the realization process fails.

GetFileIndex() As Integer

Retrieves the zero-based index number of the the file in the sync spec.

`ifUserData`

SetUserData(user_data As Object) As Void

Sets the user data.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()` (on either the event or source object). It will return `Invalid` if no data has been set.

roSyncSpec

ON THIS PAGE

- [ifSyncSpec](#)
 - [GetFailureReason\(\) As String](#)
 - [ReadFromFile\(filename As String\) As Boolean](#)
 - [ReadFromString\(spec As String\) As Boolean](#)
 - [WriteToFile\(filename As String\) As Boolean](#)
 - [WriteToFile\(filename As String, options As roAssociativeArray\) As Boolean](#)
 - [WriteToString\(\) As String](#)
 - [WriteToString\(options As roAssociativeArray\) As String](#)
 - [GetMetadata\(section As String\) As roAssociativeArray](#)
 - [LookupMetadata\(section As String, field As String\) As String](#)
 - [GetFileList\(section As String\) As roList](#)
 - [GetFile\(section As String, index As Integer\) As roAssociativeArray](#)
 - [GetName\(\) As String](#)
 - [EqualTo\(other As roSyncSpec\) As Boolean](#)
 - [VerifySignature\(signature as String, obfuscated_passphrase as String\) As Boolean](#)
 - [FilterFiles\(section As String, criteria As roAssociativeArray\) As roSyncSpec](#)
 - [FilesEqualTo\(spec As roSyncSpec\) As Boolean](#)
 - [GetAssets\(section As String\) As Object](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object represents a parsed sync spec. It allows you to retrieve various parts of the specification with methods.

ifSyncSpec

GetFailureReason() As String

Returns information if an *roSyncSpec* method indicates failure.

ReadFromFile(filename As String) As Boolean

Populates the sync spec by reading the specified file. This method returns True upon success and False upon failure.

ReadFromString(spec As String) As Boolean

Populates the sync spec by reading the passed string. This method returns True upon success and False upon failure.

WriteToFile(filename As String) As Boolean

Writes out the current sync spec to the specified file as XML. Because the XML is regenerated, it is possible this file may not be textually identical to the specification that was read. This method returns True upon success and False upon failure.

WriteToFile(filename As String, options As roAssociativeArray) As Boolean

Writes out the current sync spec to the specified file as XML or JSON. The *options* associative array can currently contain the *format* parameter, which can have a value of "xml" or "json". Because the XML/JSON is regenerated, it is possible this file may not be textually identical to the specification that was read. This method returns True upon success and False upon failure.

WriteToString() As String

Writes out the current sync spec to a string and returns it as XML. This method returns an empty string if the write operation fails.

WriteToString(options As roAssociativeArray) As String

Writes out the current sync spec to a string and returns it as XML or JSON. The *options* associative array can currently contain the *format* parameter, which can have a value of "xml" or "json". This method returns an empty string if the write operation fails.

GetMetadata(section As String) As roAssociativeArray

Returns an *roAssociativeArray* object containing the information stored in the specified metadata section of the sync spec (typically "client" or "server"). This method returns 0 if the read operation fails.

If the sync spec is formatted as XML, then all returned values will be strings. If the spec spec is formatted as JSON, the JavaScript types will be converted as follows:

JavaScript type	BrightScript type
Array	roArray
Object	roAssociativeArray
String	String
Boolean	Boolean
Number	Double
Null	Invalid

LookupMetadata(section As String, field As String) As String

Provides a shortcut for looking up specified metadata items in the specified section without needing to create a temporary *roAssociativeArray* object. This method returns an empty string if the read operation fails.

GetFileList(section As String) As roList

Returns an *roList* object containing *roAssociativeArray* objects for each file in the specified section of the sync spec. This method returns Invalid if the read operation fails.

GetFile(section As String, index As Integer) As roAssociativeArray

Returns an *roAssociativeArray* object for the file in the specified section and at the specified index. This method returns Invalid if the read operation fails.

***GetName()* As String**

Returns the name supplied for the sync spec in the <sync> XML element.

EqualTo(other As roSyncSpec) As Boolean

Compares the contents of the *roSyncSpec* object with another *roSyncSpec* object. This method compares the parsed contents of each sync spec rather than the XML files themselves.

VerifySignature(signature as String, obfuscated_passphrase as String) As Boolean

De-obfuscates the passphrase and uses it to verify the signature of the sync spec. This method returns True upon success and False upon failure.

FilterFiles(section As String, criteria As roAssociativeArray) As roSyncSpec

Returns a new *roSyncSpec* object that is a copy of the existing object, except that the specified section is filtered using the specified criteria. The criteria are matched against the file metadata. Multiple criteria can be specified in the passed associative array, and all criteria must be met for a file to be returned with the new *roSyncSpec*.

The following call will yield an *roSyncSpec* object with a "download" section that has been filtered so that only files of the group "scripts" will remain.

```
filtered_spec = spec.FilterFiles("download", { group: "scripts" })
```

FilesEqualTo(spec As roSyncSpec) As Boolean

Compares the contents of the sync spec to another sync spec and returns true if they're identical. This method compares the parsed contents of the sync specs, rather than the XML/JSON itself.

GetAssets(section As String) As Object

Retrieves a list of assets from the sync spec in a form that's compatible with *roAssetPool* and its associated objects.

Networking Objects

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This section describes objects related to networking, client/server applications, and feeds.

- [roDatagramReceiver](#)
- [roDatagramSender](#)
- [roDatagramSocket](#)
- [roDatagramEvent](#)
- [roHttpServer](#)
- [roHttpEvent](#)
- [roKeyStore](#)
- [roMediaServer](#)
- [roMediaStreamer](#)
- [roMediaStreamerEvent](#)
- [roMimeStream](#)
- [roMimeStreamEvent](#)
- [roNetworkAdvertisement](#)
- [roNetworkConfiguration](#)
- [roNetworkAttached](#)
- [roNetworkDetached](#)

- [roNetworkDiscovery](#)
- [roNetworkHotplug](#)
- [roNetworkStatistics](#)
- [roPtp](#)
- [roPtpEvent](#)
- [roRssArticle](#)
- [roRssParser](#)
- [roRtspStream](#)
- [roSnmAgent](#)
- [roSnmEvent](#)
- [roStreamByteEvent](#)
- [roStreamConnectResultEvent](#)
- [roStreamEndEvent](#)
- [roStreamLineEvent](#)
- [roSyncManager](#)
- [roSyncManagerEvent](#)
- [roTCPStream](#)
- [roTCPConnectEvent](#)
- [roUPnPActionResult](#)
- [roUPnPController](#)
- [roUPnPDevice](#)
- [roUPnPSearchEvent](#)
- [roUPnPService](#)
- [roUPnPServiceEvent](#)
- [roURLTransfer](#)
- [roURLEvent](#)

roDatagramReceiver

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifIdentity](#)
 - [GetIdentity\(\) As Integer](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object sends *roDatagramEvent* instances to a message port when UDP packets are received on a specified port.

Object Creation: The *roDatagramReceiver* object is created with a `port` parameter, which specifies the port on which to receive UDP packets.

```
CreateObject("roDatagramReceiver ", port As Integer)
```

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifIdentity`

GetIdentity() As Integer

Note

The `ifIdentity` interface has been deprecated. We recommend using the `ifUserData` interface instead.

`ifMessagePort`

SetPort(port As roMessagePort)

Posts messages of type `roDatagramEvent` to the attached message port.

This example script listens for UDP packets on port 21075:

```
receiver = CreateObject("roDatagramReceiver", 21075)
mp = CreateObject("roMessagePort")
receiver.SetPort(mp)
while true
    event = mp.WaitMessage(0)
    if type(event) = "roDatagramEvent" then
        print "Datagram: "; event
    endif
end while
```

roDatagramSender

ON THIS PAGE

- [ifDatagramSender](#)
 - [SetDestination\(destination_address As String, destination_port As Integer\) As Boolean](#)
 - [Send\(packet As Object\) As Integer](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object allows UDP packets to be sent to a specified network destination.

Object Creation: The `roDatagramSender` object is created with no parameters.

```
CreateObject("roDatagramSender")
```

ifDatagramSender

SetDestination(destination_address As String, destination_port As Integer) As Boolean

Specifies the destination IP address in dotted quad form along with the destination port. This function returns True if successful.

Send(packet As Object) As Integer

Sends the specified data packet as a datagram. The packet may be a string or an *roByteArray*. This method returns 0 upon success and a negative error code upon failure.

This example script broadcasts a single UDP packet containing "HELLO" to anyone on the network listening on port 21075:

```
sender = CreateObject("roDatagramSender")
sender.SetDestination("255.255.255.255", 21075)
sender.Send("Hello")
```

roDatagramSocket

ON THIS PAGE

- [ifDatagramSocket](#)
 - [GetFailureReason\(\) As String](#)
 - [BindToLocalPort\(port As Integer\) As Boolean](#)
 - [GetLocalPort\(\) As Integer](#)
 - [SendTo\(destination_address As String, destination_port As Integer, packet As Object\) As Integer](#)
 - [JoinMulticastGroup\(address as String\) as Boolean](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)
- [ifIdentity](#)
 - [GetIdentity\(\) As Integer](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object both sends and receives UDP packets. Use *roDatagramSocket* if you need the player to communicate using protocols such as SSDP, which only allow a server to respond to the source of a received request.

Received packets are delivered to the message port as *roDatagramEvent* objects.

ifDatagramSocket

GetFailureReason() As String

Returns additional information if the *BindToLocalPort* or *Sendto* methods fail.

BindToLocalPort(port As Integer) As Boolean

Binds the socket to the specified local port. Use this method to receive packets sent to a specific port. Alternatively, if you want to receive replies to sent packets (and it doesn't matter which local port is used), pass a port number of 0, and the player will select an unused port. This method returns True upon success and False upon failure

GetLocalPort() As Integer

Returns the local port to which the socket is bound. Use this method if you passed a port number of 0 to BindToLocalPort and need to determine which port the player has selected.

SendTo(destination_address As String, destination_port As Integer, packet As Object) As Integer

Sends a single UDP packet, which can be an *roString* or *roByteArray*, to the specified address and port. This method returns 0 upon success and a negative error code upon failure.

JoinMulticastGroup(address as String) as Boolean

Joins the multicast group for the specified address on all interfaces that are currently up. This method returns True upon success and False upon failure. In the event of failure, GetFailureReason() may provide additional information. To ensure that you are joined on all network interfaces, you should register for *roNetworkHotplug* events and call the JoinMulticastGroup() method in response to the arrival of new networks.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via SetUserData(). It will return Invalid if no data has been set.

ifMessagePort

SetPort(port As roMessagePort)

Posts messages of type *roDatagramEvent* to the attached message port.

ifIdentity

GetIdentity() As Integer

Note

The *ifIdentity* interface has been deprecated. We recommend using the *ifUserData* interface instead.

roDatagramEvent

ON THIS PAGE

- [ifDatagramEvent](#)
 - [GetByteArray\(\) as Object](#)
 - [GetSourceHost\(\) as String](#)
 - [GetSourcePort\(\) as Integer](#)
- [ifString](#)
 - [GetString\(\) As String](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifSourceIdentity](#)
 - [GetSourceIdentity\(\) As Integer](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This event object is generated when datagram packets are received by the *roDatagramReceiver* or *roDatagramSocket* objects.

ifDatagramEvent

GetByteArray() as Object

Returns the contents of the packet as an *roByteArray*.

GetSourceHost() as String

Returns the source IP address of the packet in dotted form.

GetSourcePort() as Integer

Returns the source port of the packet.

ifString

GetString() As String

ifUserData

SetUserData(user_data As Object)

Sets the user data.

GetUserData() As Object

Returns the user data that has previously been set via *SetUserData()* (either on the source or event object). It will return *Invalid* if no data has been set.

ifSourceIdentity

GetSourceIdentity() As Integer

Note

The *ifSourceIdentity* interface has been deprecated. We recommend using the *ifUserData* interface instead.

roHttpServer

ON THIS PAGE

- [ifHttpServer](#)
 - [GetFailureReason\(\) As String](#)
 - [AddGetFromString\(parameters As roAssociativeArray\) As Boolean](#)
 - [AddGetFromFile\(parameters As roAssociativeArray\) As Boolean](#)
 - [AddGetFromFolder\(parameters As roAssociativeArray\) As Boolean](#)
 - [AddGetFromEvent\(parameters As roAssociativeArray\) As Boolean](#)
 - [AddPostToString\(parameters As roAssociativeArray\) As Boolean](#)
 - [AddPostToFile\(parameters As roAssociativeArray\) As Boolean](#)
 - [AddPostToFormData\(parameters As roAssociativeArray\) As Boolean](#)
 - [AddMethodFromEvent\(parameters As roAssociativeArray\) As Boolean](#)
 - [AddMethodToFile\(parameters As roAssociativeArray\) As Boolean](#)
 - [AddMethodToString\(parameters As roAssociativeArray\) As Boolean](#)
 - [SetupDWSLink\(title As String\) As Boolean](#)
 - [Parameters for "Add" Handler Methods](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object allows for processing of RESTful HTTP requests from remote URLs to the embedded web server of the BrightSign player. Many of the requests are provided to the script as *roHttpRequest* objects for handling.

Object Creation: The *roHttpRequest* object is created with an *roAssociativeArray*.

```
CreateObject("roHttpRequest", parameters As roAssociativeArray)
```

Currently, the associative array can contain a single parameter:

- `port`: The port number of the embedded web server

roHttpRequest

Each "Add" handler method described below takes an associative array as its parameter. Values in the associative array specify how the handler behaves. See the [table](#) at the end of this section for common `key:value` pairs.

GetFailureReason() As String

Yields additional useful information if an *roHttpRequest* method fails.

AddGetFromString(parameters As roAssociativeArray) As Boolean

Causes any HTTP GET requests for the specified URL path to be met directly with the contents of the "body" member of the parameter associative array. The MIME type (and potentially the entire character set) should be specified if the request is expected to come from a web browser. The request is handled entirely within the *roHttpRequest* method; no events are sent to the message port.

AddGetFromFile(parameters As roAssociativeArray) As Boolean

Causes any HTTP GET requests for the specified URL path to be met directly from the specified file. You should always specify the MIME type (and possibly the character set) if you expect the request to come from a web browser. The request is handled entirely within the *roHttpRequest* method; no events are sent to the message port.

AddGetFromFolder(parameters As roAssociativeArray) As Boolean

Constructs a dynamic handler that serves up static files, which will appear as children of the defined storage folder. This method accepts an associative array with the following parameters:

- `folder`: The file path of the folder that will act as the root directory of the server. If this parameter is absent, everything on the storage device will be served.
- `url_prefix`: The URL prefix under which files will be served. If this parameter is absent, the URL will match from root.
- `filters`: An array of filters. Each filter is an associative array with the following parameters:
 - `re`: The regular expression to match against the request URL.
 - `ext`: The extension to match against the leaf file of the request URL.

If the "re" and "ext" parameters are absent, the filter will match everything.

- `headers`: An associative array containing arbitrary headers to be included with the automatic response.
- `content_type`: The contents of the "Content-Type" header included with the automatic response. This cannot be set in the same filter as the headers. The MIME type and character set can be specified together (e.g. "text/plain; charset=utf-8").

AddGetFromEvent(parameters As roAssociativeArray) As Boolean

Requests that an event of type *roHttpRequest* be sent to the configured message port. This occurs when an HTTP GET request is made for the specified URL path.

AddPostToString(parameters As roAssociativeArray) As Boolean

Requests that an event of type *roHttpRequest* be sent to the configured message port. This occurs when an HTTP POST request is made for the specified URL path. Use the *roHttpRequest.GetRequestBodyString()* method to retrieve the posted body.

AddPostToFile(parameters As roAssociativeArray) As Boolean

Requests that, when an HTTP POST request is made to the specified URL path, the request body be stored in a temporary file according to the parameters["destination_directory"] value in the associative array. When this request is complete, an *roHttpRequest* event is sent to the configured message port. Use the *roHttpRequest.GetRequestBodyFile()* method to retrieve the name of the temporary file. If the file still exists at the time the response is sent, it will be automatically deleted. However, if the player reboots or loses power during the POST process, the file will not be deleted. For this reason, we recommend using a dedicated subdirectory as the "destination_directory" and wiping this subdirectory during startup (using *DeleteDirectory()*) before adding handlers that refer to it.

AddPostToFormData(parameters As roAssociativeArray) As Boolean

Requests that, when an HTTP POST request is made to the specified URL path, an attempt be made to store form data (passed as application/x-www-form-urlencoded or multipart/form-data) in an associative array that can be retrieved by calling the *roHttpRequest.GetFormData()* method.

AddMethodFromEvent(parameters As roAssociativeArray) As Boolean

Requests that an event of type *roHttpRequest* be sent to the configured message port. Unlike *AddGetFromEvent()*, this method can support arbitrary HTTP methods. The HTTP method is specified using the method member in the associative array.

AddMethodToFile(parameters As roAssociativeArray) As Boolean

Requests that, when an arbitrary HTTP request is made to the specified URL path, the request body be stored in a temporary file according to the parameters["destination_directory"] value in the associative array. The HTTP method is specified using the method member in the associative array. When the request is complete, an *roHttpRequest* event is sent to the configured message port. Use the *roHttpRequest.GetRequestBodyFile()* method to retrieve the name of the temporary file. If the file still exists at the time the response is sent, it will be automatically deleted.

AddMethodToString(parameters As roAssociativeArray) As Boolean

Attempts to support an arbitrary HTTP method. The request body is placed in a string and an event is raised. This makes the request body available via the *roHttpRequest.GetRequestBodyString()* method. A response can be sent in the same manner as the *AddGetToEvent()* method.

SetupDWSLink(title As String) As Boolean

Generates a tab in the **Diagnostic Web Server** (DWS) that links directly to the base <ip_address:port> of the *roHttpRequest* instance. The passed string specifies the title of the tab.

Example

```
server1.SetupDWSLink("My AWS Link")
server2.SetupDWSLink("My Other AWS Link")
```

Parameters for "Add" Handler Methods

The following table describes common key:value pairs for "Add" handler methods:

Name	Applies to	Value
url_path	All	The path for which the handler method will be used
user_data	GetFromEvent() PostToString() PostToFile() MethodToString()	A user-defined value that can be retrieved by calling <i>roHttpRequest.GetUserData()</i>
method	AddMethodFromEvent() AddMethodToFile()	The HTTP method associated with the generated <i>roHttpRequest</i> . The method type can then be retrieved using <i>roHttpRequest.GetMethod()</i> .
passwords	All	

		An associative array that contains a mapping between usernames and passwords
auth	All	The authentication type to use when passwords are set. This value can be either "basic" or "digest". The value defaults to "digest" if not specified.
realm	All	The authentication realm, which will be displayed by web browsers when prompting for a username and password
headers	GetFromFile()	An associative array that contains arbitrary headers to be included with the automated response
content_type	GetFromFile()	The contents of the "Content-Type" header that is included with the automated response. This may not be set at the same time as the headers member. You can set both the MIME type and character set together (e.g. "text/plain; charset=utf-8")
body	GetFromString()	The response body
filename	GetFromFile()	The path to the file used for the response body.
destination_directory	PostToFile()	The path to the directory used for the temporary file containing the request body. A random filename will be generated automatically.

ifMessagePort

SetPort(port As roMessagePort)

Posts messages of type *roHttpEvent* to the attached message port.

roHttpEvent

ON THIS PAGE

- **ifHttpEvent**
 - **GetFailureReason() As String**
 - **GetMethod() As String**
 - **SetResponseBodyString(body As String)**
 - **SetResponseBodyFile(filename As String) As Boolean**
 - **GetRequestBodyString() As String**
 - **GetRequestBodyFile() As String**
 - **GetRequestHeader(header_name As String) As String**
 - **GetRequestHeaders() As Object**
 - **GetRequestParam(URI_parameter As String) As String**
 - **GetRequestParams() As Object**
 - **AddResponseHeader(header As String, value As String) As Boolean**
 - **AddResponseHeaders(a As Object) As Boolean**
 - **SendResponse(http_status_code As Integer) As Boolean**
 - **GetFormData() As Object**
 - **GetUrl() As String**
- **ifUserData**
 - **SetUserData(user_data As Object)**
 - **GetUserData() As Object**

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This event object is used to handle requests generated by the *roHttpRequest* object.

roHttpRequest

GetFailureReason() As String

Yields additional useful information if a function return indicates an error.

GetMethod() As String

Returns the type of HTTP method that triggered the event on the *roHttpRequest* instance.

SetResponseBodyString(body As String)

Sets the response body for an event generated via the *AddGetFromEvent()* or *AddMethodToString()* method on the *roHttpRequest* object. This call is ignored with any other event.

SetResponseBodyFile(filename As String) As Boolean

Specifies the name of a file to use as the source response body for an event generated via the *AddGetFromEvent()* or *AddMethodToString()* method on the *roHttpRequest* object. This call is ignored with any other event. This function will return *False* if the file cannot be opened or another failure occurs.

The specified file is read gradually as it is sent to the client.

GetRequestBodyString() As String

Returns the string received if the event was generated via *roHttpRequest.AddPostToString()*. An empty string is returned with any other event.

GetRequestBodyFile() As String

Returns the name of the temporary file created if the event is generated via *roHttpRequest.AddGetFromEvent()*. This call is ignored with any other event.

GetRequestHeader(header_name As String) As String

Returns the value of the specified HTTP request header. If the header does not exist, an empty string is returned.

GetRequestHeaders() As Object

Returns an *roAssociativeArray* containing all the HTTP request headers.

GetRequestParam(uri_parameter As String) As String

Returns the value of the specified URI parameter. If the parameter does not exist, an empty string is returned.

GetRequestParams() As Object

Returns an *roAssociativeArray* containing all the URI parameters.

AddResponseHeader(header As String, value As String) As Boolean

Adds the specified HTTP header and value to the response. This function returns *True* upon success.

AddResponseHeaders(a As Object) As Boolean

Adds the specified HTTP header/value pairs to the response. This method expects an *roAssociativeArray* of header names mapped to header values, which can be of type *roString*, *roInt*, or *roFloat*. Any other value types will cause the request to fail, though a subset of headers to might be set before the failure occurs. This function returns *True* upon success.

SendResponse(http_status_code As Integer) As Boolean

Sends the HTTP response using the specified HTTP status code. To ensure that the response is sent, this function needs to be called once the script has finished handling the event. This function returns *False* upon failure.

GetFormData() As Object

Returns an *roAssociativeArray* containing all the form data. See the entry on *roHttpServer.AddPostToFormData()* for more information.

GetUrl() As String

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via *SetUserData()*. It will return *Invalid* if no data has been set.

roKeyStore

ON THIS PAGE

- [ifKeyStore](#)
 - [GetFailureReason\(\) As String](#)
 - [AddCACertificate\(certificat_file As String\) As Boolean](#)
 - [AddClientCertificate\(parameters As roAssociativeArray\) As Boolean](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object allows you to register client certificates with the player. These certificates can be used by *roHtmlWidget* objects when communicating with websites. If there are multiple *roHtmlWidget* instances, they will share the same certificate database.

Client certificates are not persistent on a BrightSign player; they must be registered with the certificate database after each reboot.

ifKeyStore

GetFailureReason() As String

Returns additional useful information if an *ifKeyStore* method returns *False*.

AddCACertificate(certificat_file As String) As Boolean

Registers the specified CA certificate with the certificate database. Client certificates can be either self-signed or signed using a 3rd-party certificate issuer (Versign, DigiCert, etc.).

AddClientCertificate(parameters As roAssociativeArray) As Boolean

Registers a .p12 client certificate with the certificate database. This method accepts an associative array with the following parameters:

- *certificat_file*: The file name and path of the .p12 client certificate.
- *passphrase*: A passphrase for the .p12 client certificate.
- *obfuscated_passphrase*: An obfuscated passphrase for the .p12 client certificate.

Important

Provide the passphrase using either the "passphrase" or "obfuscated_passphrase" parameter (not both). We recommend using the "obfuscated_passphrase" in production environments, while the "passphrase" should be used for testing purposes only. Contact support@brightsign.biz to learn more about generating a key for obfuscation and storing it on the player.

BrightSign players use the "nickname" of a .p12 client certificate to match it with a website. The "nickname" consists of the `host:port` of the web address you wish to match: For example, to use a client certificate for `https://brightsign.biz`, you would specify a "nickname" of `"brightsign.biz:443"`.

The following example uses an openssl terminal to generate a .p12 client certificate to use with `https://brightsign.biz`.

```
openssl pkcs12 -export -clcerts -in client.crt -inkey client.key -out client.p12 -name "brightsign.biz:443"
```

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifMessagePort`

SetPort(port As roMessagePort)

Posts messages to the attached message port.

Example

```
k=createobject("rokeystore")
k.addcacertificate("ssd:/apache.crt")
aa = CreateObject("roAssociativeArray")
aa.AddReplace("certificate_file", "ssd:/client.p12")
aa.AddReplace("passphrase", "1q2w3e4r")
k.addclientcertificate(aa)
```

roMediaServer

ON THIS PAGE

- [ifMediaServer](#)
 - [GetFailureReason\(\) As String](#)
 - [Start\(a As String\) As Boolean](#)
 - [Stop\(\) As Boolean](#)
 - [Terminate\(\) As Boolean](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)
- [ifIdentity](#)

- [GetIdentity\(\) As Integer](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

The *roMediaServer* object waits for client requests, deals with negotiation, and ultimately generates an *roMediaStreamer* pipeline to fulfill the request. For more information, see the [BrightSign Media Server](#) section. This object currently supports RTSP and HTTP requests. Requests from the client must take the following form:

```
protocol://IP_address:port/media_streamer_pipeline
```

- `protocol`: Either `rtsp` or `http`
- `IP_address:port`: The IP address of the BrightSign player and the port number on which the media server is running.
- `media_streamer_pipeline`: A media streamer pipeline, but without the final destination component (as the destination is implicit in the request from the client).

Object Creation: The *roMediaServer* object is created with no parameters

```
CreateObject("roMediaServer")
```

ifMediaServer

GetFailureReason() As String

Returns useful information if the `Start()`, `Stop()`, or `Terminate()` methods return `False`.

Start(a As String) As Boolean

Begins a media server instance. This method can be passed a string that specifies the streaming protocol and the port number of the server:

```
s = CreateObject("roMediaServer")
s.Start("http:port=8080")
```

A number of optional parameters can be added after the port parameter using an "&" (ampersand):

- `trace`: Displays a trace of messages in the negotiation with the client. This parameter is useful particularly for debugging RTSP sessions. For example: `"rtsp:port=554&trace"`
- `maxbitrate`: Sets the maximum instantaneous bitrate (in Kbps) of the RTP transfer initiated by RTSP. This parameter has no effect for HTTP. The parameter value 80000 (i.e. 80Mbps) has been found to work well. The default behavior (also achieved by passing a zero value) is to not limit the bitrate at all. For example: `"rtsp:port=554&trace&maxbitrate=80000"`
- `threads`: Sets the maximum number of threads the server is prepared to have running. Each thread handles a single client request. The default value is "5". For example: `"http:port=8080&threads=10"`

Stop() As Boolean

Stops the media server. This method signals all threads to stop, but does not wait for this to happen before destroying the server instance.

Terminate() As Boolean

Stops the media server. This method waits for all threads to stop before destroying the server instance.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifMessagePort`

SetPort(port As roMessagePort)

Posts messages to the attached message port.

`ifIdentity`

GetIdentity() As Integer

Note

The `ifIdentity` interface has been deprecated. We recommend using the `ifUserData` interface instead.

roMediaStreamer

ON THIS PAGE

- [ifMediaStreamer](#)
 - [GetFailureReason\(\) As String](#)
 - [SetPipeline\(pipeline As String\) As Boolean](#)
 - [Initialize\(\) As Boolean](#)
 - [Connect\(\) As Boolean](#)
 - [Start\(\) As Boolean](#)
 - [Stop\(\) As Boolean](#)
 - [Disconnect\(\) As Boolean](#)
 - [Reset\(\) As Boolean](#)
 - [Inject\(a As Integer\) As Boolean](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)
- [Source Specifications](#)

The current implementation of this object allows a player to stream `.ts` files over UDP and RTP. For more information, see the [BrightSign Media Server](#) section.

Object Creation: The `roMediaStreamer` object is created with no parameters.

```
CreateObject("roMediaStreamer")
```

`ifMediaStreamer`

GetFailureReason() As String

SetPipeline(pipeline As String) As Boolean

Specifies a streaming pipeline. The source (a file URI) and destination (an IP address) of the stream are specified in the passed stream. This method replaces the `SetSource()` and `SetDestination()` methods from firmware version 4.7. To stream media as before, use the `filesimple` source designation and the `udpsimple`/`rtpsimple` destination designations:

Example

```
m = CreateObject("roMediaStreamer")
m.SetPipeline("filesimple:///data/clip.ts, udpsimple://239.192.0.0:1234/")
m.Start()
```

Initialize() As Boolean

Progresses the pipeline into the INITIALIZED state. This allocates some resources for the pipeline, but does not begin a stream.

Connect() As Boolean

Progresses the pipeline into the CONNECTED state. This allows the script to create a memory stream without starting it.

Start() As Boolean

Begins streaming.

Stop() As Boolean

Stops the pipeline stream. Some internal pipeline stages may continue running.

Disconnect() As Boolean

Regresses the steam back to the CONNECTED state.

Reset() As Boolean

Resets the pipeline stream. All internal pipeline stages are terminated.

Inject(a As Integer) As Boolean

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return Invalid if no data has been set.

ifMessagePort

SetPort(port As roMessagePort)

Posts messages of type `roMediaStreamerEvent` to the attached message port.

Source Specifications

The string passed to the `roMediaStreamer.SetPepline()` method can have unique parameters that determine the source type and playback behavior.

- **Looping:** By default, a stream from a media file will not loop when it ends. You can specify a looping parameter at the end of the source string as follows: `""filesimple:///data/example.mp4?loop"`. It is also possible to loop the stream using end-of-stream messages from `roMediaStreamerEvent`. However, the slightly longer restart gap that results from using BrightScript may cause problems with the streaming client. This is especially true if you attempt to set a new media file source upon looping the function.

▼ Firmware Version 7.0

- **Version 7.0**

- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

roMediaStreamerEvent

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifMediaStreamerEvent](#)
 - [GetEvent\(\) As Integer](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is sent by instances of *roMediaStreamer*. It provides information about the current state of an IP stream being sent by the player.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifMediaStreamerEvent`

GetEvent() As Integer

Returns an integer describing the status of an *roMediaStreamer* instance:

- 0 - `EOS_NORMAL`: The end of the stream has been reached without any errors being detected. This signal is not sent if the loop parameter is specified using the *roMediaStreamer.SetSource()* method.
- 1 - `EOS_ERROR`: The stream has been aborted prematurely because of an error condition.

roMimeStream

ON THIS PAGE

- [ifPictureStream](#)
 - [GetUrl\(\) As String](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)

- [Version 6.1](#)
- [Previous Versions](#)

This object passes an MJPEG stream in MIME format to the *roVideoPlayer.PlayFile()* method. There are some limitations to what MJPEG streams this object will play correctly. *roMimeStream* has been optimized to play streaming video from a local source with the smallest possible delay. The result is a short buffering window that is not appropriate for playing MJPEG streams from URLs outside of a local network. We are currently optimizing *roMimeStream* to work with different IP camera brands: see the [IP Camera FAQ](#) for more details.

Object Creation: To play an RTSP stream, first instantiate an *roUrlTransfer* object. Then wrap it in an *roMimeStream* object and pass the *PictureStream* to *PlayFile*, as shown in the following example.

```
u=createobject("roUrlTransfer")
u.seturl("http://mycamera/video.mjpg")
r=createobject("roMimeStream", u)
p=createobject("roVideoPlayer")
p.PlayFile({ PictureStream: r })
```

ifPictureStream

***GetUrl()* As String**

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

***GetUserData()* As Object**

Returns the user data that has previously been set via *SetUserData()*. It will return *Invalid* if no data has been set.

ifMessagePort

SetPort(port As roMessagePort)

Posts event messages to the attached message port. The event messages are of the type *roMimeStreamEvent*. There are currently two possible event messages:

- *PICTURE_STREAM_FIRST_PICTURE_AVAILABLE* = 0: The first picture is now available for decoding.
- *PICTURE_STREAM_CONNECT_FAILED*: The object is unable to connect to the specified URL.

roMimeStreamEvent

ON THIS PAGE

- [ifInt](#)
 - [GetInt\(\)](#) As Integer
 - [SetInt\(value As Integer\) As Void](#)
 -

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is generated by the *roMimeStream* object. It will return an integer corresponding to the event on the *roMimeStream* object:

ifInt

GetInt() As Integer

Returns the event ID.

SetInt(value As Integer) As Void

Sets the value of the event.

roNetworkAdvertisement

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is used to advertise services running on a BrightSign player to other devices on the network. The current implementation supports advertising via mDNS (which is part of [zeroconf](#) via [Bonjour™](#)).

Tip

To discover a player advertising on the local network, use the [roNetworkDiscovery](#) object.

Object creation: The *roNetworkAdvertisement* object is created with an associative array of network parameters and arbitrary text information.

```
CreateObject("roNetworkAdvertisement", advertisement As roAssociativeArray) As Object
```

The *roAssociativeArray* can contain the following keys:

- **name**: The service name. This should be a readable string such as "Remote BrightSign Widget Service."
- **type**: The service type. This should be a service from the definitive list, formatted in the following manner: "_service._protocol" (for example, "_http._tcp").
- **port**: The port number on which the service runs.
- **_*<name>***: An arbitrary text key. The value can be used to provide additional, optional data to clients (serial number, IP address, etc.). The key must be preceded by an underscore to avoid name conflicts within the associative array; the underscore will be removed before the record is registered with mDNS.

Once the object is created, advertising starts immediately and continues until the object is destroyed (i.e. when it becomes unreferenced).

There are no interfaces on the *roNetworkAdvertisement* object.

Example

```
mp = CreateObject("roMessagePort")
di = CreateObject("roDeviceInfo")
props = { name: "My Hoopy Service", type: "_http._tcp", port: 8080, _serial: di.
GetDeviceUniqueId() }
advert = CreateObject("roNetworkAdvertisement", props)

wait(600000, mp)
advert = invalid ' Stop advertising
```

roNetworkConfiguration

ON THIS PAGE

- ifNetworkConfiguration
 - Apply() As Boolean
 - SetupDWS(settings As roAssociativeArray) As Boolean
 - EnableLEDs(enable As Boolean) As Boolean
 - SetClientIdentifier(a As String) As Boolean
 - GetClientIdentifier() As String
 - SetLoginPassword(password As String) As Boolean
 - SetObfuscatedLoginPassword(password As String) As Boolean
 - SetInboundShaperRate(rate As Integer) As Boolean
 - SetMTU(mtu As Integer) As Boolean
 - SetRoutingMetric(a As Integer) As Boolean
 - SetDHCP() As Boolean (interface)
 - SetIP4Address(ip As String) As Boolean (interface)
 - SetIP4Netmask(netmask As String) As Boolean (interface)
 - SetIP4Broadcast(broadcast As String) As Boolean (interface)
 - SetIP4Gateway(gateway As String) As Boolean (interface)
 - SetVlanIds(vlan_ids As roArray) As Boolean
 - SetWiFiESSID(essid as String) As Boolean
 - GetWiFiESSID() As String
 - SetWiFiPassphrase(passphrase as String) As Boolean
 - SetWiFiPassphraseAndObfuscate(a As String) As String
 - SetObfuscatedWiFiPassphrase(password As String) As Boolean
 - SetWiFiAccessPointMode(enable as Boolean) As Boolean
 - SetAccessPointFrequencyMHz(frequency As Integer) As Boolean
 - SetWiFiIdentity(identity As String) As Boolean
 - SetWiFiEapTlsOptions(options As String) As Boolean
 - SetWiFiCACertificates(certificates As Dynamic) As Boolean
 - SetWiFiClientCertificate(certificate As Dynamic) As Boolean
 - SetWiFiPrivateKey(key As Dynamic) As Boolean
 - SetWiFiSecurityMode(mode As String) As Boolean
 - ConfigureDHCPServer(config As roAssociativeArray) As Boolean
 - SetDomain(domain As String) As Boolean (host)
 - SetDNSServers(servers As roArray) As Boolean (host)
 - AddDNSServer(server As String) As Boolean (host)
 - SetTimeServer(time_server As String) As Boolean (host)
 - GetTimeServer() As String
 - SetTimeServerIntervalSeconds(interval_in_seconds As Integer) As Boolean
 - GetTimeServerIntervalSeconds() As Integer
 - SetHostName(name as String) As Boolean (host)
 - GetHostName() As String (host)
 - SetProxy(proxy as String) As Boolean (host)
 - GetProxy() As String (host)
 - SetProxyBypass(hostnames As Array) As Boolean
 - GetProxyBypass() As roArray
 - GetRecoveryUrl() As String
 - ResetInterfaceSettings() As Boolean
 - ResetHostSettings() As Boolean
 - GetFailureReason() As String
 - GetCurrentConfig() As Object
 - TestInterface() As Object
 - TestInternetConnectivity() As Object
 - GetNeighborInformation() As roAssociativeArray
- ifWiFiConfiguration
 - ScanWiFi() As roArray
- Network Authentication
 - EAP-TLS
 - PEAP/MSCHAP
 - Common Variants
 - Additional Certificates
 - Obfuscated WiFi Passphrase
 - TKIP/CCMP Encryption

- [Special Cases](#)
 - [MD5 Support](#)
 - [Anonymous Identity](#)
 - [Validity Dates](#)
- [Examples](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object provides various methods for configuring the network interfaces on a BrightSign player.

Object Creation: The *roNetworkConfiguration* object is created with a single parameter.

```
CreateObject("roNetworkConfiguration", network_interface as Integer)
```

The *network_interface* parameter can be any of the following:

- 0 or "eth0": The Ethernet port on the BrightSign player
- 1 or "wlan0": The internal WiFi
- 2 or "ppp0": A connected modem

The network interface can be configured as a VLAN using the following string format: "[parent_interface].[vlan_id]". Once the VLAN interface(s) are configured, they must be enabled on the parent network interface (e.g. "eth0") using the *SetVlanIds()* method. VLAN interfaces use DHCP by default. They are supported on Series 4 (XTx44, XDx34) and Series 3 (XTx43, XDx33, HDx23, LS423, HO523) players only.

Example

```
n6 = CreateObject("roNetworkConfiguration", "eth0.6")
```

Some of the settings below are specific to the network interface, while others are used by the BrightSign host for all network interfaces.

ifNetworkConfiguration

***Apply()* As Boolean**

Applies the requested changes to the network interface. This method returns *false* if the changes could not be applied to the network interface, though changes made using "Set" and "Reset" methods will be preserved through future reboots even if *Apply()* fails. Configuration may take several seconds to complete.

Note

"Set" and "Reset" methods do not take effect until *Apply()* is called.

SetupDWS(settings As roAssociativeArray) As Boolean

Configures the Diagnostic Web Server (DWS). This method returns *true* if a restart is required for the changes to take effect. It will return *false* if a restart is not required or if the method failed—the *GetFailureReason()* method returns a non-empty string in event of failure.

By default, the Diagnostic Web Server is enabled on port 80, with the player serial number as password. Settings for the DWS are specified in an associative array. These properties are written to the registry and persist after reboot:

- `port`: The port number of the Diagnostic Web Server, located at the IP address of the player. Setting this value to 0 will disable the DWS, while setting it to "default" will make the DWS accessible on the default port (80). Specifying *only* this parameter in the associative array is equivalent to enabling the DWS without password protection.
- `password`: An obfuscated password for the DWS. This method uses digest access authentication. Specifying this parameter without setting a `port` number will make the DWS accessible on the default port.
- `open`: An unobfuscated password for the DWS. This method uses digest access authentication. Specifying this parameter without setting a `port` number will make the DWS accessible on the default port.
- `basic`: A flag indicating whether basic authentication should be used or not. Setting this parameter to `true` allows the password set with the `open` parameter to be validated using basic authentication, rather than digest access authentication. This option allows for backwards compatibility with older platforms; most, if not all, modern browsers require basic authentication to be disabled in order to communicate with the DWS.

The user name is "admin" for all authentication configurations.

EnableLEDs(enable As Boolean) As Boolean

Enables or disables the Ethernet activity LED (i.e. flashing during link and activity behavior). The Ethernet LED is enabled by default. Changes to this setting do not persist across reboots. This method returns `true` upon success and `false` upon failure. Note that this method is not available on HDx10, HDx20, and LSx22 models.

SetClientIdentifier(a As String) As Boolean

Sets the DHCP client identifier.

GetClientIdentifier() As String

Returns the DHCP client identifier.

SetLoginPassword(password As String) As Boolean

Specifies a login password for the SSH connection (if SSH has been enabled in the registry). This method accepts a plain-text password.

SetObfuscatedLoginPassword(password As String) As Boolean

Specifies a login password for the SSH connection (if SSH has been enabled in the registry). This method accepts a password that has been obfuscated using a shared secret.

Note

Contact support@brightsign.biz to learn more about generating a key for obfuscation and storing it on the player.

SetInboundShaperRate(rate As Integer) As Boolean

Sets the bandwidth limit for inbound traffic in bits per second. For the default bandwidth limit, pass -1 to the method; for no bandwidth limit, pass 0 (though these two settings are functionally the same). You will need to call `Apply()` for this setting to take effect, and changing this setting at any time will cause the network interface to be taken down and reinitialized.

Note

Because of overhead on the shaping algorithm, attempting to limit the bandwidth at rates greater than approximately 2Mbit/s will reduce speeds to less than the specified rate.

SetMTU(mtu As Integer) As Boolean

Sets the maximum transmission unit (MTU) for the network interface in bytes. Currently, the MTU setting is not returned when the `GetCurrentConfig()` method is called.

SetRoutingMetric(a As Integer) As Boolean

Configures the metric for the default gateway on the current network interface. Routes with lower metrics are preferred over routes with higher metrics. This function returns `true` upon success.

SetDHCP() As Boolean (interface)

Enables DHCP and disables all other settings. This function returns `true` if successful.

SetIP4Address(ip As String) As Boolean (interface)

SetIP4Netmask(netmask As String) As Boolean (interface)

SetIP4Broadcast(broadcast As String) As Boolean (interface)

SetIP4Gateway(gateway As String) As Boolean (interface)

Sets the IPv4 interface configuration. All values must be specified explicitly. Unlike the `ifconfig` shell command, there is no automatic inference. The parameter is a string dotted decimal quad (i.e. "192.168.1.2" or similar). It returns `true` upon success.

Example

```
nc = CreateObject("roNetworkConfiguration", 0)
nc.SetIP4Address("192.168.1.42")
nc.SetIP4Netmask("255.255.255.0")
nc.SetIP4Broadcast("192.168.1.255")
nc.SetIP4Gateway("192.168.1.1")
```

SetVlanIds(vlan_ids As roArray) As Boolean

Enables the specified VLAN IDs (and only those IDs) on the network interface (the VLAN interfaces must be configured first). This method accepts an array of integers.

Example

```
' Configure the VLANs first. Use defaults for VLAN6.
n6=CreateObject("roNetworkConfiguration", "eth0.6")
print n6.ResetInterfaceSettings()
print n6.Apply()

' Use defaults for VLAN11.
n11=CreateObject("roNetworkConfiguration", "eth0.11")
print n11.ResetInterfaceSettings()
print n11.Apply()

' Use static IP for VLAN15.
n15=CreateObject("roNetworkConfiguration", "eth0.15")
print n15.SetIP4Address("192.168.15.100")
print n15.SetIP4Netmask("255.255.255.0")
print n15.Apply()

' Now enable VLANs 6, 11 and 15 on eth0.
n0=CreateObject("roNetworkConfiguration", "eth0")
print n0.SetVlanIds([6, 11, 15])
print n0.Apply()
```

Note

VLAN configuration is persistent, even when VLAN is not enabled. If you use the `SetVlanIds()` method to remove and then add VLAN IDs, they will return with the previous configuration by default. You can use the `ResetInterfaceSettings()` method to remove any previous configuration.

SetWiFiESSID(essid as String) As Boolean

Configures the WiFi ESSID of the wireless network that the player will connect to. This method returns `true` on success.

GetWiFiESSID() As String

Retrieves the configured WiFi ESSID, even if the player is not currently connected to that wireless network. Use the `GetCurrentConfig()`. `wifi_essid` value to retrieve the ESSID of the wireless network that the player is currently connected to.

SetWiFiPassphrase(passphrase As String) As Boolean

Configures the passphrase or key for the wireless network. This method accepts a plain-text passphrase. It returns `true` if the passphrase is successfully set.

SetWiFiPassphraseAndObfuscate(a As String) As String

Configures the passphrase or key for the wireless network. This method accepts a plain-text passphrase and returns the obfuscated result. If the passphrase is not set, an empty string is returned instead.

SetObfuscatedWiFiPassphrase(password As String) As Boolean

Configures the passphrase or key for the wireless network. This method accepts a passphrase that has been obfuscated using a shared secret. It returns `true` if the password is successfully set.

Note

Contact support@brightsign.biz to learn more about generating a key for obfuscation and storing it on the player.

SetWiFiAccessPointMode(enable As Boolean) As Boolean

Enables or disables WiFi access-point mode: Passing `true` to this method instructs the player to act as a wireless access point, which uses the ESSID and passphrase set with the `SetWiFiESSID()` and `SetWiFiPassphrase()` (or `SetObfuscatedWiFiPassphrase()`) methods. If a passphrase has been set, the wireless access point will use WPA2 authentication—otherwise, it will use no authentication. This method returns `true` on success and `false` on failure; use the `GetFailureReason()` method to return more information in case of failure.

WiFi access-point mode should be used for configuration only. It offers a minimal DHCP server, but does not provide DNS servers or routing to the Ethernet. We recommend assigning a static IP address to the wireless interface, though this is not enforced by the firmware.

Note

WiFi access-point mode is not available on Series 2 (4Kx42, XDx32, HDx22) or earlier models.

SetAccessPointFrequencyMHz(frequency As Integer) As Boolean

Sets the `frequency` of the wireless access point (in MHz). This method returns `true` on success and `false` on failure; use the `GetFailureReason()` method to return more information in case of failure.

Note

The following `SetWifi*` methods are used to configure WPA Enterprise with EAP-TLS.

SetWiFiIdentity(identity As String) As Boolean

Sets the RADIUS identity. In EAP-TLS mode, the identity will be taken from the client certificate if this method is not called ("subjectAltName" will be used if present; otherwise, the "commonName" is used). In EAP-PEAP/MSCHAP mode, this method must be called to set the identity.

SetWiFiEapTlsOptions(options As String) As Boolean

Sets EAP-specific options. Currently, this method can be used to enable or disable MD5 support ("md5=enable" or "md5=disable") and to set an anonymous identity for EAP-PEAP/MSCHAP configurations ("anonymous_identity=anon@brightsign.biz"). Multiple parameters can be set with a space-separated list. To clear previous settings, pass an empty string to this method.

SetWiFiCACertificates(certificates As Dynamic) As Boolean

Sets the CA certificate file for EAP-TLS (certificates can also be sent from an EAP peer). The certificate data can be specified as an `roByteArray` of Base64 data (for `.cer`, `.pfx`, or `.p12` file formats) or an `roString` (for the `.pem` file format). The supported binary formats are ASN.1 DER (for separate certificates and keys) and PKCS#12 (for a single file that may contain both certificates and keys). The supported text format is PEM.

SetWiFiClientCertificate(certificate As Dynamic) As Boolean

Sets the client certificate for EAP-TLS. The certificate data can be specified as an `roByteArray` of Base64 data or an `roString`.

SetWiFiPrivateKey(key As Dynamic) As Boolean

Sets the private key for authentication. The certificate data can be specified as an *roByteArray* of Base64 data or an *roString*. If the private key is password protected, use the `SetWiFiPassphrase()` method to set the password.

Note

If the client certificate and associated private key are in the same PKCS#12 file, the file contents should be specified using the `SetWiFiPrivateKey()` method and the `SetWiFiClientCertificate()` value should be left empty.

SetWiFiSecurityMode(mode As String) As Boolean

Sets the WiFi encryption method. By default, both WPA (TKIP) and WPA2 (CCMP) encryption are permitted. This method accepts a space-separated, case-insensitive list that can include either "tkip" or "ccmp" values. Passing an empty string sets the default mode. If both CCMP and TKIP are allowed, CCMP always has priority.

ConfigureDHCPServer(config As roAssociativeArray) As Boolean

Configures a DHCP server on the Ethernet port or wireless access point. The associative array should consist of two properties: `ip4_start`, which specifies the beginning of the range of offered IP addresses, and `ip4_end`, which specifies the end of the range. To disable the DHCP server, pass an empty associative array (`{}`) to this method. This method returns `true` on success and `false` on failure; use the `GetFailureReason()` method to return information in case of failure.

SetDomain(domain As String) As Boolean (host)

Sets the device domain name. This will be appended to names to fully qualify them, though it is not necessary to call this. This method returns `true` on success.

Example

```
nc.SetDomain("brightsign.biz")
```

SetDNSServers(servers As roArray) As Boolean (host)

Sets the list of DNS servers if there were no DNS servers when the object was created. This method accepts an array of strings, each containing the dotted-quad IP address of a DNS server, and it returns `true` on success. There is currently a maximum of three servers; the method will return `false` if you attempt to add more than three servers.

Tip

To clear the list of DNS servers, call `SetDNSServers({})`.

AddDNSServer(server As String) As Boolean (host)

Adds another DNS server to the list if there were no DNS servers when the object was created. This method accepts the dotted-quad IP address of a DNS server as a string, and it returns `true` on success. There is currently a maximum of three servers; the method will return `false` if you attempt to add more than three servers.

SetTimeServer(time_server As String) As Boolean (host)

Sets the default time server, which is "time.brightsignnetwork.com". You can disable the use of NTP by calling `SetTimeServer("")`. You can use URL syntax to specify that the player use an HTTP or HTTPS server to synchronize the clock. The following are valid time server addresses:

- `http://time.brightsignnetwork.com/`
- `https://time.brightsignnetwork.com/`
- `ntp://time.brightsignnetwork.com/`
- `time.brightsignnetwork.com`

Note

The last two addresses are equivalent.

GetTimeServer() As String

Retrieves the (host) time server currently in use.

SetTimeServerIntervalSeconds(interval_in_seconds As Integer) As Boolean

Specifies how often the player should communicate with the time server and adjust its clock. The default interval is 12 hours; passing a value of 0 specifies the default interval. The minimum interval allowed is 120 seconds.

GetTimeServerIntervalSeconds() As Integer

Returns the current interval for time-server renewal (in seconds).

SetHostName(name as String) As Boolean (host)

Sets the device host name. If no host name has been explicitly set, then a host name is automatically generated based on the device serial number. Passing an empty string to this method resets the device host name to its automatically generated value.

GetHostName() As String (host)

Retrieves the host name currently in use.

SetProxy(proxy as String) As Boolean (host)

Sets the name or address of the proxy server used for HTTP and FTP requests. The proxy string should be formatted as "http://user:password@hostname:port". The hostname can contain up to four "*" characters; each "*" character can be used to replace one octet from the current IP address. For example, if the IP address is currently 192.168.1.2, and the proxy is set to "proxy-*-**", then the player will attempt to use a proxy named "proxy-192.168".

GetProxy() As String (host)

Returns the name or address of the proxy server.

SetProxyBypass(hostnames As Array) As Boolean

Exempts the specified hosts from the proxy setting. The passed array should consist of one or more hostnames. The player will attempt to reach the specified hosts directly rather than using the proxy that has been specified with the SetProxy() method. For example, the hostname "example.com" would exempt "example.com", "example.com:80", and "www.example.com" from the proxy setting.

GetProxyBypass() As roArray

Returns an array of hostnames that have been exempted from the proxy setting using the SetProxyBypass() method.

GetRecoveryUrl() As String

Returns the current recovery URL, which is stored in the "ru" entry of the "networking" registry section. If the recovery URL is set via DHCP Option 43, it will take precedence over the "ru" registry entry.

ResetInterfaceSettings() As Boolean

Clears all interface-specific settings (e.g. IP address, default gateway). This method is ideal for resetting an interface to a known, empty state before configuring specific settings.

ResetHostSettings() As Boolean

Clears all host settings (i.e. settings that don't take the interface number into account and that don't affect the specified network interface; e.g. the time server).

GetFailureReason() As String

Returns additional information when a member function returns false.

GetCurrentConfig() As Object

Retrieves the entire current configuration as an associative array containing the following members:

Value	Type	Host/Interface	Description
metric	Integer	Interface	

			Returns the current routing metric for the interface. See the <code>SetRoutingMetric()</code> entry for more details.
dhcp	Boolean	Interface	Returns <code>true</code> if the system is currently configured to use DHCP. Returns <code>false</code> otherwise.
hostname	String	Host	The currently configured host name
mdns_hostname	String	Host	The Zeroconf host name currently in use. This may be longer than the host name if there is a collision on the current network.
ethernet_mac	String	Interface	The Ethernet MAC address
ip4_address	String	Interface	The current IPv4 address. If none is currently set, the string will be empty.
ip4_netmask	String	Interface	The current IPv4 network mask. If none is currently set, the string will be empty.
ip4_broadcast	String	Interface	The current IPv4 broadcast address. If none is currently set, the string will be empty.
ip4_gateway	String	Interface	The current IPv4 gateway address. If none is currently set, the string will be empty.
domain	String	Host	The current domain suffix
dns_servers	<i>roArray</i> of Strings	Host	The currently active DNS servers
time_server	String	Host	The current time server
configured_proxy	String	Host	The currently configured proxy. This may contain magic characters as explained under <code>SetProxy()</code> above.
current_proxy	String	Host	The currently active proxy. Any magic characters will have been replaced as explained under <code>SetProxy()</code> above.
shape_inbound	Integer	Interface	The current bandwidth shaping for inbound traffic determined by the <code>SetInboundShaperRate()</code> method.
type	String	Interface	Either "wired" or "wifi"
link	Boolean	Interface	Indicates whether the network interface is currently connected.
wifi_essid	String	Interface	The name of the current Wi-Fi network (if any)
wifi_signal	Integer	Interface	An indication of the received signal strength. The absolute value of this field is usually not meaningful, but it can be compared with the reported value on other networks or in different locations.

TestInterface() As Object

Performs various tests on the network interface to determine whether it appears to be working correctly. It reports the results via an associative array containing the following members:

Value	Type	Description
ok	Boolean	This value is <code>true</code> if the tests find no problems, or <code>false</code> if at least one problem was identified.
diagnosis	String	A single-line diagnosis of the first problem identified in the network interface.
log	<i>roArray</i> of strings	

A complete log of all the tests performed and their results.

***TestInternetConnectivity()* As Object**

Performs various tests on the Internet connection (via any available network interface, not necessarily the one specified when the *roNetworkConfiguration* object was created) to determine whether it appears to be working correctly. It reports the results via an associative array containing the following members:

Value	Type	Description
ok	Boolean	This value is <code>true</code> if the tests find no problems, or <code>false</code> if at least one problem was identified.
diagnosis	String	A single line diagnosis of the first problem identified with the Internet connection.
log	<i>roArray</i> of strings	A complete log of all the tests performed and their results.

***GetNeighborInformation()* As roAssociativeArray**

Retrieves location information from the network infrastructure using the LLDP-MED protocol. The information is returned as an associative array of strings corresponding to civic-address types, which are defined as follows according to the LLDP-MED specification:

CAtype	Label	Description
1	A1	national subdivisions (state, region, province, prefecture)
2	A2	county, parish, gun(JP), district(IN)
3	A3	city, township, shi(JP)
4	A4	city division, borough, city district, ward, chou(JP)
5	A5	neighborhood, block
6	A6	street

CAtype	NENA	PIDF	Description	Examples
0			language	i-default [3]
16	PRD	PRD	leading street direction	N
17	POD	POD	trailing street suffix	SW
18	STS	STS	street suffix	Ave, Platz
19	HNO	HNO	house number	123
20	HNS	HNS	house number suffix	A, 1/2
21	LMK	LMK	landmark or vanity address	Columbia University
22	LOC	LOC	additional location information	South Wing
23	NAM	NAM	name (residence and office occupant)	Joe's Barbershop
24	ZIP	PC	postal/ZIP code	10027-1234
25			building (structure)	Low Library
26			unit (apartment, suite)	Apt 42
27		FLR	floor	4
28			room number	450F
29			placetype	office
30	PCN		postal community name	Leonia
31			post office box (P.O Box)	12345
32			additional code	13203000003

128		script	Latn
255		reserved	

ifWiFiConfiguration

ScanWiFi() As roArray

Scans for available wireless networks. The results are reported as an *roArray* containing one or more associative arrays with the following members:

Value	Type	Description
ssid	String	Network name
bssid	String	Access point BSSID
signal	Integer	Received signal strength indication. The absolute value of this field is not usually relevant, but it can be compared with the reported value on other networks or in different locations.

Network Authentication

BrightSign players support most commonly used wireless encryption formats: WEP (64 & 128), WPA (TKIP), and WPA2 (AES).

WPA Enterprise is supported using EAP-TLS (with DER, PEM, or PKCS#12 certificates) and PEAPv0/MSCHAPv2 (with a username and passphrase). Wired authentication via 802.1x is also supported. The configuration instructions below apply to both WiFi and wired authentication via 802.1x; the only difference is the wired/wireless parameter (0/1) passed during *roNetworkConfiguration* initialization.

EAP-TLS

EAP-TLS authentication requires a client certificate and private key. There are two ways to accomplish this:

- Packaging the client certificate and private key in a single file in PKCS#12 format (using PEM or DER encoding), usually with a *.p12* file extension. To do this, set the PKCS#12 file using the `SetWiFiPrivateKey()` method and pass a blank string to the `SetWiFiClientCertificate()` method.
- Packaging the client certificate as a X.509 certificate (using PEM or DER encoding) and optionally securing the key with a passphrase (instead of the usual WiFi passphrase). To do this, set the client certificate using the `SetWiFiClientCertificate()` method and the key using the `SetWiFiPrivateKey()` method. If the key is protected with a passphrase, you should then call `SetWiFiPassphrase()`.

Certificate keys may contain binary data if not PEM formatted. In this case, they must be provided as an *roByteArray* object. Text formats may be passed using a string or *roByteArray*.

Example: Setting EAP-TLS with a .p12 file

```
nc = CreateObject("roNetworkConfiguration", 1)

p12 = CreateObject("roByteArray")
p12.ReadFile("client.p12")

nc.SetWiFiClientCertificate("")
nc.SetWiFiPrivateKey(p12)
nc.SetWiFiPassphrase("passwordgoeshere")
```

During authentication, the Radius server is passed an identity. By default, the identity will be taken from the client certificate. If the certificate has a "subjectAltName", it will be used; otherwise, the "CommonName" is used. If neither of these are correct, the identity may be overridden. This default behavior can be overridden by calling `SetWiFiIdentity()`, or it can be specified by passing a blank string to `SetWiFiIdentity()`.

PEAP/MSCHAP

This mode requires an identity (username) and passphrase, rather than a client certificate. There may also be a second, "outer", identity—see the Special Cases section below for configuration details.

```
nc = CreateObject("roNetworkConfiguration",1)
nc.SetWifiIdentity("user@brightsign-example.com")
nc.SetWifiPassphrase("passwordgoeshere")
```

Common Variants

Additional Certificates

Additional CA certificates are often required (and using a custom CA for authentication is potentially more secure). Also, some CA hierarchies require intermediate certificates, which are sometimes supplied via EAP. If the server does not supply them, they may be added to the CA file using the `SetWifiCACertificates()` method. This method supports PEM and DER certificates.

```
nc = CreateObject("roNetworkConfiguration",1)
ca = CreateObject("roByteArray")
ca.ReadFile("ca.pem")
nc.SetWifiCACertificates(ca)
```

Obfuscated WiFi Passphrase

If an obfuscated WiFi passphrase is required, you can substitute the `SetObfuscatedWifiPassphrase()` method for the `SetWifiPassphrase()` method. Contact support@brightsign.biz to learn more about generating a key for obfuscation and storing it on the player.

TKIP/CCMP Encryption

By default, both WPA (TKIP) and WPA2 (CCMP) encryption is permitted. You can alter this behavior using the `SetSecurityMode()` method, which accepts a space-separated, case-insensitive list of allowed modes ("ccmp" and "tkip"). If both CCMP and TKIP are allowed, CCMP always has priority.

```
nc = CreateObject("roNetworkConfiguration",1)
nc.SetWifiSecurityMode("") 'Sets the default mode
nc.SetWifiSecurityMode("ccmp tkip") 'Explicitly allows both modes (same as the default mode)
nc.SetWifiSecurityMode("ccmp") 'Requires the use of CCMP
```

Special Cases

MD5 Support

MD5 support is enabled by default for backwards-compatibility reasons. This behavior can be modified by passing the "md5=enable" or "md5=disable" string to the `SetWifiEapTlsOptions()` method.

```
nc = CreateObject("roNetworkConfiguration",1)
nc.SetWifiEapTlsOptions("md5=disable")
```

Anonymous Identity

Some EAP-PEAP/MSCHAP configurations require an anonymous identity. By default, the BrightSign player uses the same inner and outer identity. An anonymous identity can be configured with the `SetWifiEapTlsOptions()` method:

```
nc = CreateObject("roNetworkConfiguration",1)
nc.SetWifiEapTlsOptions("anonymous_identity=anon@brightsign.biz")
```

Tip

If neither MD5 or identity options are being used, the setting should be cleared by passing a blank string to the `SetWiFiEapTlsOptions()` method.

Validity Dates

Validity dates are required for both EAP-TLS and PEAP/MSCHAP. The current time is required to check the certificate. If the player clock has not been set, it will typically set its time using the network, but this requires EAP authentication first. To avoid this problem, there are two special exceptions:

- EAP-TLS: When the clock is not set, the time is set to ten seconds after the client certificate becomes valid, which is usually sufficient to authenticate (though this may not be sufficient for servers with newer certificates than the client).
- PEAP/MSCHAP: There is no client certificate, and the server certificate time is not checked when the date is not set.

The clock can also be set via the [Diagnostic Web Server](#), which prevents either of the above exceptions from being used. Typically, the clock is also set from the network once it is available, so these exceptions are only used on first boot or when the clock battery has been completely emptied.

Examples

The following script disables WPA Enterprise configuration (which is otherwise persistent):

```
nc = CreateObject("roNetworkConfiguration", 1)
nc.SetWiFiSecurityMode("")
nc.SetWiFiESSID("")
nc.SetWiFiIdentity("")
nc.SetWiFiCaCertificates("")
nc.SetWiFiClientCertificate("")
nc.SetWiFiPrivateKey("")
nc.SetWiFiPassphrase("")
nc.SetWiFiEapTlsOptions("")
ok = nc.Apply()
```

The following script configures the WiFi for EAP-TLS using a PKCS#12 (.p12) file, without additional CA certificates:

```
ssid = "brightwifi"
pkcs12file = "client.p12"
pkcspass = "passwordgoeshere"

nc = CreateObject("roNetworkConfiguration", 1)
nc.SetWiFiESSID(ssid)

p12 = CreateObject("roByteArray")
ok = p12.ReadFile(pkcs12file)

if ok then
    nc.SetWiFiClientCertificate("")
    nc.SetWiFiPrivateKey(p12)
    nc.SetWiFiPassphrase(pkcspass)

    nc.SetWiFiCaCertificates("")
    nc.SetWiFiSecurityMode("")
    nc.SetWiFiIdentity("")
    nc.SetWiFiEapTlsOptions("")

    ok = nc.Apply()
end if
```

The following script configures the WiFi for EAP-TLS with additional certificates:

```

ssid = "brightwifi"
cafile = "cacert.pem"
pkcs12file = "client.p12"
pkcspass = "passwordgoeshere"

nc = CreateObject("roNetworkConfiguration", 1)
nc.SetWiFiESSID(ssid)

p12 = CreateObject("roByteArray")
ok = p12.ReadFile(pkcs12file)

ca = CreateObject("roByteArray")
ok = ca.ReadFile(cafile) and ok

if ok then
    nc.SetWiFiClientCertificate("")
    nc.SetWiFiPrivateKey(p12)
    nc.SetWiFiPassphrase(pkcspass)

    nc.SetWiFiCaCertificates("")
    nc.SetWiFiSecurityMode("")
    nc.SetWiFiIdentity("")
    nc.SetWiFiEapTlsOptions("")

    ok = nc.Apply()
end if

```

The following script configures the WiFi for PEAP/MSCHAP with additional CA files:

```

ssid = "brightwifi"
cafile = "ca.pem"
user = "someuser@brightsign.biz"
pass = "whatever"
ca = CreateObject("roByteArray")
ok = ca.ReadFile(cafile)

if ok then
    nc = CreateObject("roNetworkConfiguration", 1)
    nc.SetWiFiESSID(ssid)
    nc.SetWiFiIdentity(user)
    nc.SetWiFiPassphrase(pass)
    nc.SetWiFiCaCertificates(ca)
    nc.SetWiFiClientCertificate("")
    nc.SetWiFiPrivateKey("")
    'Use this if an anonymous outer identity is required
    'nc.SetWiFiEapTlsOptions("anonymous_identity=anon@brightsign.biz")
    nc.SetWiFiEapTlsOptions("")

    ok = nc.Apply()
    if not ok then
        fail = "Unable to set configuration: " + nc.GetFailureReason()
    end if
else
    fail = "Unable to read file " + cafile
end if

' if not ok, do something with the failure message

```

The following script performs various VLAN configurations on the Ethernet interface:

```

' Configure the VLANs first. Use defaults for VLAN6
n6=CreateObject("roNetworkConfiguration", "eth0.6")
print n6.ResetInterfaceSettings()
print n6.Apply()

' Use defaults for VLAN11
n11=CreateObject("roNetworkConfiguration", "eth0.11")
print n11.ResetInterfaceSettings()
print n11.Apply()

' Use static IP for VLAN15
n15=CreateObject("roNetworkConfiguration", "eth0.15")
print n15.SetIP4Address("192.168.15.100")
print n15.SetIP4Netmask("255.255.255.0")
print n15.Apply()

' Now enable VLANs 6, 11 and 15 on eth0.
n0=CreateObject("roNetworkConfiguration", "eth0")
print n0.SetVlanIds([6, 11, 15])
print n0.Apply()

```

roNetworkAttached

ON THIS PAGE

- [ifInt](#)
 - [GetInt\(\) As Integer](#)
 - [SetInt\(value As Integer\) As Void](#)
- [ifString](#)
 - [GetString\(\) As String](#)
 - [SetString\(value As String\) As Void](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object implements the *ifInt* and *ifString* interfaces to report the index of an attached [network interface](#). Instances of this object are posted by *roNetworkHotplug* when a configured network connection becomes available. For Ethernet, it may take some time after the cable is inserted for this to take place.

`ifInt`

GetInt() As Integer

Returns the interface ID of the event. This method returns -1 VLAN interfaces; use `GetString()` to retrieve the interface name.

SetInt(value As Integer) As Void

Sets the value that will be returned by `GetInt()`.

`ifString`

GetString() As String

Returns the interface name ("eth0", "wlan0", "ppp0") of the *roNetworkConfiguration* instance associated with the event.

SetString(value As String) As Void

Sets the value that will be returned by *GetString()*.

ifUserData

SetUserData(user_data As Object)

Sets the user data.

GetUserData() As Object

Returns the user data that has previously been set via *SetUserData()* (either on the source or event object). This method will return *Invalid* if no data has been set.

roNetworkDetached

ON THIS PAGE

- [ifInt](#)
 - [GetInt\(\) As Integer](#)
 - [SetInt\(value As Integer\) As Void](#)
- [ifString](#)
 - [GetString\(\) As String](#)
 - [SetString\(value As String\) As Void](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object implements the *ifInt* interface to report the index of a detached **network interface**. Instances of this object are posted by *roNetworkHotplug* when a configured network connection becomes unavailable.

ifInt

GetInt() As Integer

Returns the interface ID of the event. This method returns -1 VLAN interfaces; use *GetString()* to retrieve the interface name.

SetInt(value As Integer) As Void

Sets the value that will be returned by *GetInt()*.

ifString

GetString() As String

Returns the interface name ("eth0", "wlan0", "ppp0") of the *roNetworkConfiguration* instance associated with the event.

SetString(value As String) As Void

Sets the value that will be returned by *GetString()*.

ifUserData

SetUserData(user_data As Object)

Sets the user data.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()` (either on the source or event object). This method will return `Invalid` if no data has been set.

roNetworkDiscovery

ON THIS PAGE

- [ifNetworkDiscovery](#)
 - [Search\(parameters As roAssociativeArray\) As Boolean](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [Event Types](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object allows for zeroconf discovery of devices on the local network (including other BrightSign players) using mDNS. Other BrightSign players must be running an instance of *roNetworkAdvertisement* to be discovered using the *roNetworkDiscovery* object.

Object Creation: The *roNetworkDiscovery* object is created with no parameters.

```
CreateObject("roNetworkDiscovery")
```

ifNetworkDiscovery

Search(parameters As roAssociativeArray) As Boolean

Searches for BrightSign players on the local network. A player will only respond to the search if it is currently running an instance of *roNetworkAdvertisement*. Search results will be posted to the attached message port. Search parameters are passed to this method as an associative array containing the following values:

- `type`: The service type. If this entry is omitted, the search will default to `"_http._tcp"`.
- `protocol`: The IP protocol. Acceptable values are `"IPv4"` and `"IPv6"`. You can also omit this entry if you wish to search for players using either protocol.

ifMessagePort

SetPort(port As roMessagePort)

Posts events to the attached message port. See the Event Types section below for more details.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

Event Types

The `roNetworkDiscovery` object can post three event object types to the attached message port:

- `roNetworkDiscoveryResolvedEvent`. Raised when a host is fully resolved.
- `roNetworkDiscoveryCompletedEvent`. Raised when the search is complete.
- `roNetworkDiscoveryGeneralEvent`. Raised for events other than the above two. This event rarely occurs.

All three event objects offer the `SetUserData()`, `GetUserData()`, and `GetData()` methods. Use `GetData()` to retrieve an associative array of results. For the `roNetworkDiscoveryResolvedEvent` object, calling `GetData()` will return the following entries:

- `protocol`: Either "IPv4" or "IPv6"
- `host_name`: The hostname of the player
- `name`: The service name
- `txt`: An associative array containing arbitrary text entries specified during instantiation of the `roNetworkAdvertisement` instance.
- `domain`: The domain of the player
- `type`: The service type
- `address`: The IPv4 or IPv6 address

The following script searches for a player running an `roNetworkAdvertisement` instance and prints the results of the discovery.

```
rn = CreateObject("roNetworkDiscovery")
mp = CreateObject("roMessagePort")
rn.SetPort(mp)
params = {}
rn.Search(params)

complete = false
while not complete
    ev = mp.WaitMessage(10000)
    if ev = invalid then
        stop
    end if

    if type(ev) = "roNetworkDiscoveryCompletedEvent" then
        print "roNetworkDiscoveryCompletedEvent"
    end if

    if type(ev) = "roNetworkDiscoveryGeneralEvent" then
        print "roNetworkDiscoveryGeneralEvent"
    end if

    if type(ev) = "roNetworkDiscoveryResolvedEvent" then
        complete = true
        data = ev.GetData()
        print "DATA: "; data
        print "DONE"
        txtdata = data[ "txt" ]
        if txtdata <> invalid then
            print "TEXT: "; txtdata
        endif
    end if
end while
```

roNetworkHotplug

ON THIS PAGE

- [ifMessagePort](#)
 - [SetPort\(a As Object\)](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object can be used to generate events when a [network interface](#) becomes available or unavailable. It will post events of type [roNetworkAttached](#) and [roNetworkDetached](#) to the associated message port.

To determine which network was attached or detached, the script needs to call `GetInt()` or `GetString()` on the [roNetworkAttached](#) [roNetworkDetached](#) event object. These methods provide an index of the network interface that was attached or detached.

Note

Reconfiguring a network interface using [roNetworkConfiguration](#) may cause it to detach and attach again.

`ifMessagePort`

SetPort(a As Object)

Posts messages of type [roNetworkAttached](#) [roNetworkDetached](#) to the attached message port.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

roNetworkStatistics

ON THIS PAGE

- [ifNetworkStatistics](#)
 - [GetTotals\(\) As roAssociativeArray](#)
 - [GetIncremental\(\) As roAssociativeArray](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object allows you to monitor and post how much bandwidth the player is using.

Object Creation: The *roNetworkStatistics* object is created with a single parameter.

```
CreateObject("roNetworkStatistics", network_interface as Integer)
```

The `network_interface` parameter is used to distinguish between the following:

- 0: The Ethernet port on the BrightSign player.
- 1: The optional internal Wi-Fi.

`ifNetworkStatistics`

GetTotals() As roAssociativeArray

Yields the total network figures since booting up.

GetIncremental() As roAssociativeArray

Yields the total network figures since booting up. Then, every subsequent time this method is called, it will yield the amount each figure has changed since the previous call.

Note

If multiple instances of `roNetworkStatistics` are created, `GetIncremental()` calls for each instance will track changes independently.

Both methods return the following statistics as floating point values:

- tx_carrier_errors
- tx_packets
- rx_packets
- tx_errors
- rx_frame_errors
- tx_bytes
- rx_errors
- tx_collisions
- rx_dropped
- tx_compressed
- rx_multicast
- tx_dropped
- rx_fifo_errors
- rx_bytes
- tx_fifo_errors
- rx_compressed

roPtp

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)
- [ifPtp](#)
 - [GetPtpStatus\(\) As roAssociativeArray](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object can be used to retrieve information about the network PTP state of the player.

Object Creation: This object is created with no additional parameters.

```
ptp = CreateObject("roPtp")
```

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return Invalid if no data has been set.

ifMessagePort

SetPort(port As roMessagePort)

Posts messages of the type *roPtpEvent* to the attached message port.

ifPtp

GetPtpStatus() As roAssociativeArray

Returns an associative array containing information about the network PTP state of the player:

- `state`: A string indicating the current PTP state of the player. Values can be "MASTER", "SLAVE", or "UNCALIBRATED".
- `timestamp`: A value indicating when the PTP state was last changed. This value is measured in seconds since the player booted. This value can be compared against the total uptime of the player, which is retrieved by calling `Uptime()`.

roPtpEvent

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifPtpEvent](#)
 - [GetPtpStatus\(\) As roAssociativeArray](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This event object is generated by the *roPtp* object whenever the PTP status of the player changes.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifPtpEvent`

GetPtpStatus() As roAssociativeArray

Returns an associative array containing information about the network PTP state of the player:

- `state`: A string indicating the current PTP state of the player. Values can be "MASTER", "SLAVE", or "UNCALIBRATED".
- `timestamp`: A value indicating when the PTP state was last changed. This value is measured in seconds since the player booted. This value can be compared against the total uptime of the player, which is retrieved by calling `Uptime()`.

roRssArticle

ON THIS PAGE

- [ifRssArticle](#)
 - [GetTitle\(\) As String](#)
 - [GetDescription\(\) As String](#)
 - [GetTimestampInSeconds\(a As Integer\) As Boolean](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

Objects of type `roRssArticle` are returned by the `roRssParser.GetNextArticle()` method. These instances can be passed to the `roTextWidget` object to display the feed on-screen.

`ifRssArticle`

GetTitle() As String

Returns the title of the RSS item.

GetDescription() As String

Returns the content of the RSS item.

GetTimestampInSeconds(a As Integer) As Boolean

Returns in seconds the difference in publication date between this RSS item and the most recent item in the feed. The user can utilize this to decide if an article is too old to display.

`SetTitle(a As String) As Boolean`

`SetDescription(a As String) As Boolean`

`SetTimestampInSeconds(a As Integer) As Boolean`

Important

For firmware versions 4.7.x and above, if no alpha value is specified when `roTextWidget.SetForegroundColor()` is called, the text widget area will appear blank.

Example

```
u=CreateObject("roUrlTransfer")
u.SetUrl("http://www.lemonde.fr/rss/sequence/0,2-3208,1-0,0.xml")
u.GetToFile("tmp:/rss.xml")

r=CreateObject("roRssParser")
r.ParseFile("tmp:/rss.xml")

EnableZoneSupport(1)
b=CreateObject("roRectangle", 0, 668, 1024, 100)
t=CreateObject("roTextWidget", b, 3, 2, 2)
t.SetForegroundColor(&hFFD0D0D0)
t.Show()

a = r.GetNextArticle()
while type(a) = "roRssArticle"
    t.PushString(a.GetDescription())
    sleep(1000)
    a = r.GetNextArticle()
end while

while true
    sleep(1000)
end while
```

roRssParser

ON THIS PAGE

- [ifRssParser](#)
 - [ParseFile\(filename As String\) As Boolean](#)
 - [ParseString\(filename As String\) As Boolean](#)
 - [GetNextArticle\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

The `roRssParser` object is used to parse an RSS feed before displaying it. Each item in an RSS feed is represented by an `roRssArticle` object.

`ifRssParser`

ParseFile(filename As String) As Boolean

Parses an RSS feed from a file.

ParseString(filename As String) As Boolean

Parses an RSS feed from a string.

GetNextArticle() As Object

Gets the next article parsed by the RSS parser. The articles are sorted by publication date, with the most recent article first. This returns an *roRss Article* object if there is one. Otherwise, an integer is returned.

roRtspStream

ON THIS PAGE

- [ifRtspStream](#)
 - [GetUrl\(\) As String](#)
 - [AddHeader\(header As String, text As String\)](#)
 - [ClearHeaders\(\)](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This is a simple media-streaming object that is passed to the *roVideoPlayer.PlayFile()* method. Use this object to play UDP, RTP, HLS, and HTTP streams. See the [Video Streaming](#) and [IP Camera](#) FAQs for more details.

Object Creation: To play a stream, instantiate an *roRtspStream* object with a URL as its argument. Then pass it to the `PlayFile()` method as shown in the following example:

```
v = createobject("rovideoplayer")
r = createobject("rortspstream", "http://172.30.1.37/alldigital/1080p/playlist.m3u8")
v.playfile({rtsp:r})
```

Note

The key in the passed associative array will always be "rtsp", no matter which streaming protocol is used.

`ifRtspStream`

GetUrl() As String

Retrieves the currently configured URL.

AddHeader(header As String, text As String)

Adds the specified header and header text to the streaming request. The ":" after the header and the "\r\n" after the header text are supplied automatically by the method. Headers only take effect when a stream is played; you cannot add more headers when a stream is playing (though these headers will be applied if the stream is played again).

ClearHeaders()

Removes headers that have been added using the `AddHeader()` method.

`ifMessagePort`

SetPort(port As roMessagePort)

Posts event messages to the attached message port. The event messages are of the type *roRtspStreamEvent* and will implement the *ifInt* interface.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via *SetUserData()*. It will return *Invalid* if no data has been set.

roSnmAgent

ON THIS PAGE

- [ifSnmAgent](#)
 - [AddOidHandler\(oid_string As String, writable_flag As Boolean, initial_value As Object\) As Boolean](#)
 - [GetOidValue\(oid_string As String\) As Object](#)
 - [SetOidValue\(oid_string As String, new_value As Object\) As Boolean](#)
 - [Start\(\) As Boolean](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

When this object is created, it starts an SNMP process that handles some standard SNMP MIBs such as system uptime. Prior to starting the SNMP agent, you can register other OIDs for handling. You can set and retrieve these by both an SNMP client and the script.

OID values are retrieved by an SNMP client without script interaction. Setting OID values will generate an *roSnmEvent* object stating that they have been changed. The script event handler can then retrieve new values and take appropriate action.

Setting an OID value from an SNMP client doesn't block the client waiting on any script action; there is a short, but indeterminate, time delay for scripts to act on a value change. This isn't a problem generally because of the way SNMP MIBs are designed. If you want to provide constantly updating OID values, you can update them using either a timer or state changes.

ifSnmAgent

AddOidHandler(oid_string As String, writable_flag As Boolean, initial_value As Object) As Boolean

Adds an OID handler with the following parameters to the SNMP agent:

- *oid_string*: The OID string (e.g. "1.3.6.1.4.1.26095.1.1.1.4.4.0"). All OID strings should be numerical.
- *writable_flag*: A Boolean value indicating whether the value can be changed by an SNMP client.
- *initial_value*: The initial value, which can be either an *roString* or *roInt*. The OID will reflect the type chosen here.

GetOidValue(oid_string As String) As Object

Returns the current value (as either *roString* or *roInt*) for a given OID.

SetOidValue(oid_string As String, new_value As Object) As Boolean

Changes the current value for a given OID. The passed value can be either an *roString* or *roInt*.

Start() As Boolean

Starts the SNMP agent. Call this method once all OID handlers have been registered.

Example

```
agent = CreateObject("roSnmpAgent")
agent.AddOidHandler("1.3.6.1.4.1.26095.1.1.1.4.4.0", false, "ValueOfOid")
agent.AddOidHandler("1.3.6.1.4.1.26095.1.1.1.4.5.0", true, 10)
agent.Start()
```

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return Invalid if no data has been set.

ifMessagePort

SetPort(port As roMessagePort)

Posts messages of type `roSnmpEvent` to the attached message port.

roSnmpEvent

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifString](#)
 - [GetString\(\) As String](#)
 - [SetString\(a As String\)](#)

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return Invalid if no data has been set.

ifString

GetString() As String

SetString(a As String)

roStreamByteEvent

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

ON THIS PAGE

- [ifInt](#)
 - [GetInt\(\) As Integer](#)
 - [SetInt\(value As Integer\) As Void](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

`ifInt`

GetInt() As Integer

Returns the event ID.

SetInt(value As Integer) As Void

Sets the value of the event.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

roStreamConnectResultEvent

ON THIS PAGE

- [ifInt](#)
 - [GetInt\(\) As Integer](#)
 - [SetInt\(value As Integer\) As Void](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\) As Void](#)
 - [GetUserData\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)

- [Previous Versions](#)

This event is sent to a message port associated with an *roTCPStream* object when an `AsyncConnectTo()` request has been completed or has failed.

`ifInt`

GetInt() As Integer

Returns the result code of the event. If the connection was successfully established, then this method will return 0. If connection failed for any reason, this method will return a non-zero integer.

SetInt(value As Integer) As Void

Sets the value of the event.

`ifUserData`

SetUserData(user_data As Object) As Void

Sets the user data.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()` (on either the event or source object). It will return `Invalid` if no data has been set.

`roStreamEndEvent`

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

ON THIS PAGE

- [ifInt](#)
 - [GetInt\(\) As Integer](#)
 - [SetInt\(value As Integer\) As Void](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

`ifInt`

GetInt() As Integer

Returns the event ID.

SetInt(value As Integer) As Void

Sets the value of the event.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

roStreamLineEvent

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifString](#)
 - [GetString\(\) As String](#)
 - [SetString\(a As String\)](#)

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifString`

GetString() As String

SetString(a As String)

roSyncManager

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

ON THIS PAGE

- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)
- [ifSyncManager](#)
 - [SetMasterMode\(master_mode As Boolean\) As Boolean](#)
 - [Synchronize\(identifier As String, ms_delay As Integer\) As Object](#)
- [Examples](#)

This object provides PTP synchronization capabilities for video walls and other deployments that require closely calibrated interaction among players. *roSyncManager* handles all network traffic for master/slave synchronization, including the network clock. Multiple synchronization groups are allowed on the same local network and even within the same video wall.

The *roSyncManager* object currently supports video, image, and scrolling ticker synchronization. The *roVideoPlayer* and *roImagePlayer* objects rely on values generated by the *ifSyncManager* and *roSyncManagerEvent* methods to synchronize playback, while the *roTextWidget* object simply requires the domain name of the *roSyncManager* instance.

As of firmware version 7.0, *roSyncManager* also supports Genlock synchronization (frame-accurate VSync) over Ethernet. To enable Genlock on an *roSyncManager* domain, pass the domain name to the *roVideoMode.SetSyncDomain()* method. Otherwise, the process for synchronizing video, image, and ticker playback is the same.

Note

The *roSyncManager* object supports synchronization over Ethernet networks only.

Before using *roSyncManager*, you will need to instantiate a synchronization group by setting all players within the group to the same PTP domain value. To do this, use the *roRegistrySection.Write()* method to set the `ptp_domain` key of the "networking" section to a value between 0 and 127. In general, changes to the registry only take effect after a reboot, so the PTP synchronization service will start on each player after it is rebooted.

Example

```
regSec = CreateObject("roRegistrySection", "networking")
regSec.Write("ptp_domain", "0")
regSec.Flush()

RebootSystem()
```

Object Creation: The *roSyncManager* object is created with an associative array representing a set of parameters.

```
CreateObject("roSyncManager", parameters as roAssociativeArray)
```

The associative array can have the following parameters:

- [string] `Domain`: An identifier that is used to distinguish among different *roSyncManager* instances within the same synchronization group (i.e. PTP domain). The default string is "BrightSign". This parameter allows multiple *roSyncManager* instances to operate at the same time.
- [string] `MulticastAddress`: The multicast address to which synchronization messages are communicated. The default address is "224.0.126.10".
- [string] `MulticastPort`: The multicast port to which synchronization messages are communicated. The default port is "1539".

`ifMessagePort`

SetPort(port As roMessagePort)

Posts messages of type *roSyncManagerEvent* to the attached message port. Use these messages to synchronize *roVideoPlayer* and *roImagePlayer* playback.

`ifSyncManager`

SetMasterMode(master_mode As Boolean) As Boolean

Specifies whether the unit is running the master instance of *roSyncManager*.

Synchronize(identifier As String, ms_delay As Integer) As Object

Broadcasts a time-stamped message to other players. This method is used on the master unit only. The message will be rebroadcasted every second to allow slave units that are powered on late to catch up. The network message contains the sync ID, as well as the domain and a timestamp. The timestamp is created at the point when this method is called; however, it can be offset by passing a non-zero `ms_delay` value, allowing synchronization points to be set slightly in the future and giving the client enough time to switch video files and perform other actions. The `identifier` parameter allows the script on the master unit to pass a filename, or some other useful marker, to the slave units as part of the synchronization message. This method returns the message that is sent out so that the master can access the timestamp.

The synchronization message is sent over all available networks (including WiFi), but slave units will use only the first message received. The PTP messages, which are used by the firmware during synchronization, are sent over Ethernet only.

Note

Because synchronization can involve slave units seeking to catch up with the playback of a master unit, we recommend using the more efficient MOV/MP4 container format when synchronizing video files. Transport Stream files (MPEG-TS) are also supported, but they must begin with a presentation timestamp (PTS) of 0. Program Stream files (MPEG-PS) are not supported.

Currently, there are two objects that can accept synchronization parameters: The *roVideoPlayer.PlayFile()* call accepts the parameters provided by *SyncManagerEvent* messages, while the *roImagePlayer.DisplayFile()* and *roImagePlayer.PreloadFile()* calls accept *SyncIsoTimestamp* in an associative array. To synchronize image playback, an *roImagePlayer* object will simply delay the transition thread prior to running the transition. If there is a separate call for *DisplayFile()*, then the transition will be cancelled and the image will be displayed immediately (as with non-synchronized *DisplayFile()* calls).

Examples

The following script synchronizes video playback on the master and slave units.

Example

```
' Create a sync manager with default address and port.
aal=CreateObject("roAssociativeArray")
aal.Domain = "BS1"
s=CreateObject("roSyncManager", aal)
p=CreateObject("roMessagePort")
s.SetPort(p)

' Create a video player - we're going to play a seamlessly looped file
v=CreateObject("roVideoPlayer")
v.SetLoopMode(True)

' THIS SECTION IS ONLY DONE BY THE MASTER
' We're the master unit - send out a synchronize event saying that we're starting.
' playback 1000ms from now
s.SetMasterMode(True)
msg = s.Synchronize("Blah1", 1000)

' THIS SECTION IS ONLY DONE BY THE SLAVE
' We're a slave unit, and we're sitting waiting for a sync message.
msg=Wait(4000, p)

' EVERYONE DOES THE REST
aa=CreateObject("roAssociativeArray")
aa.FileName = "Text_1.mov"
aa.SyncDomain = msg.GetDomain()
aa.SyncId = msg.GetId()
aa.SyncIsoTimestamp = msg.GetIsoTimestamp()

v.PlayFile(aa)
```

roSyncManagerEvent

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifSyncManagerEvent](#)
 - [GetDomain\(\) As String](#)
 - [GetId\(\) As String](#)
 - [GetIsoTimestamp\(\) As String](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

These events are generated on slave units in response to *roSyncManager.Synchronize()* calls from the master unit. The *roSyncManager* on each slave unit will handle message duplicates, so the script will receive the sync message only once during normal operations.

If the slave unit is already booted up, then the event will arrive from the first network event generated by *roSyncManager.Synchronize()*. On the other hand, if the slave unit is booted up while the master is in the middle of playing a video file or displaying an image file, then one of the message resends (generated at one second intervals by the master unit) will trigger the event. The script passes on the data from the event to the `PlayFile()` command of the video player or the `DisplayFile()` command of the image player, which will then determine how far forward in the file it needs to seek.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifSyncManagerEvent`

GetDomain() As String

Returns the domain of the sync group, which is specified during creation of the *roSyncManager* object on the master unit.

GetId() As String

Returns the identifier of the event.

GetIsoTimestamp() As String

Returns the timestamp of the event in ISO format.

`roTCPServer`

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

ON THIS PAGE

- [ifTCPServerInstance](#)
 - [GetFailureReason\(\) As String](#)
 - [SetPort\(port As Object\)](#)
 - [BindToPort\(port As Dynamic\) As Boolean](#)
- [ifUserData](#)
 - [SetUserData\(a As Object\)](#)
 - [GetUserData\(\) As Object](#)

`ifTCPServerInstance`

GetFailureReason() As String

Yields additional useful information if an *roTCPServer* method fails.

SetPort(port As Object)

Sets the message port that will receive events from an *roTCP*Server instance.

BindToPort(port As Dynamic) As Boolean

Prepares to accept incoming TCP connections on the specified port. Passing an integer to this method will specify a standard port number. This method can also accept an index of integer interfaces contained within an associative array, which can contain the following members:

- -1: Any (this is the default value)
- 0: Ethernet
- 1: WiFi
- 2: Modem
- 32767: Loopback (i.e. TCP connections can only be established by internal sources)

`ifUserData`

SetUserData(a As Object)

Supplies an object that will be provided by every event called by an *roTCP*Server instance.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

roTCPConnectEvent

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifSocketInfo](#)
 - [GetSourceAddress\(\) As String](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

The event is posted when a new connection is made to an *roTCP*Server port. The normal response to receiving such an event is to create a new *roTCPStream* object and pass the event to its `AcceptFrom` call.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifSocketInfo`

GetSourceAddress() As String

Returns the IP address of the remote end of the TCP connection.

roUPnPActionResult

ON THIS PAGE

- **ifUPnPActionResult**
 - **GetType() As Integer**
 - **GetID() As Integer**
 - **GetResult() As Boolean**
 - **GetValues() As roAssociativeArray**
- **ifUserData**
 - **SetUserData(user_data As Object)**
 - **GetUserData() As Object**

▼ Firmware Version 7.0

- **Version 7.0**
- **Version 6.2**
- **Version 6.1**
- **Previous Versions**

This object contains the results of an *roUPnPService.Invoke()* call. It is important to match the transaction ID of this object with the value returned by the *Invoke()* method.

ifUPnPActionResult

GetType() As Integer

Returns the result type, which can be one of the following:

- 1 – Invoke result
- 2 – Subscribe result

GetID() As Integer

Returns the transaction ID of the result as an integer. Use it to match the *roUPnPActionResult* instance with the *roUPnPService.Invoke()* call that generated it.

GetResult() As Boolean

Returns True if the originating *Invoke()* call was successful.

GetValues() As roAssociativeArray

Returns an associative array containing the "out" values (if any) of the originating *Invoke()* call.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via *SetUserData()*. It will return *Invalid* if no data has been set.

roUPnPController

ON THIS PAGE

- **ifUPnPController**
 - **SetDebug(debug As Boolean) As Void**
 - **Search(searchTarget As String, mx As Integer) As Boolean**
- **ifMessagePort**
 - **SetPort(port As roMessagePort)**
- **ifUserData**
 - **SetUserData(user_data As Object)**

- [GetUserData\(\) As Object](#)
- [UPnP Controller Operation](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object establishes and maintains a UPnP Control Point. It must exist for the entirety of UPnP discovery operations. Refer to the [UPnP Device Architecture](#) document for more information about UPnP discovery protocols.

Object Creation: The *roUPnPController* object is created without any parameters.

```
CreateObject("roUPnPController")
```

ifUPnPController

SetDebug(debug As Boolean) As Void

Enables detailed debugging in the UPnP engine.

Search(searchTarget As String, mx As Integer) As Boolean

Issues a Search request for a UPnP device. The parameters correspond to the ST (Search Target) and MX (Maximum wait time) header values that are sent with a UPnP M-SEARCH command. Responses to the Search request will generate messages in the form of *roUPnPSearchEvent* objects.

Note

The most common value for the searchTarget parameter is "upnp:rootdevice". This allows you to search all root devices, identify which devices you wish to interact with, and then get the *roUPnPDevice* and *UPnPService* instances for these embedded devices and services.

RemoveDevice(udn As String) As Boolean

Forces removal of the specified device from the Control Point device list. The passed string must include "uuid: " prepended to the UDN value.

ifMessagePort

SetPort(port As roMessagePort)

Specifies the port that will receive events generated by the *roUPnPController* instance.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via *SetUserData()*. It will return Invalid if no data has been set.

UPnP Controller Operation

The *roUPnPController* object maintains a list of all currently detected UPnP devices accessible via the local network. To maintain this list, the *roUPnPController* object follows these generally accepted control-point practices:

- If an `ssdp:alive` multicast notification is received from a device that is not part of the list, it is queried for its device information and added to the list. However, the `ssdp:alive` message is not intended as the primary means for device discovery; rather, this behavior is intended to keep the list up-to-date and remove devices that disappear without an `ssdp:byebye` notification.
- If an `ssdp:byebye` multicast notification is received from a device that is part of the list, it will be removed from the list.
- The UPnP Controller allows a client to issue a Search request for UPnP devices. All devices on the network are expected to respond directly to the requesting device. If a response is received from a device that is not part of the list, it is queried for its device information and added to the list.
- UPnP devices report a "time-to-live" for notifications. For UPnP NOTIFY and search-response messages, this is contained in the "Cache-Control: max-age" header. Typically, this "time-to-live" is 20 or 30 minutes, though some devices have much shorter time values. Every device is configured to expire after its "time-to-live" is reached, at which point it is removed from the device list. The counter is reset (i.e. the device is renewed) after each receipt of an `ssdp:alive` message.

BrightScript does not allow direct access to its internal DeviceList. Rather, it raises events in the form of `roUPnPSearchEvent` objects when devices are added or removed from the list. These objects can, in turn, be used to retrieve `roUPnPDevice` objects containing all device information.

The controller also raises events whenever it receives a NOTIFY multicast message or a response to an M-SEARCH message (i.e. a response to a controller search request). These events return associative arrays containing headers from the NOTIFY multicast message or from the HTTP response to the M-SEARCH message.

The associative arrays may also contain additional non-header items. For an SSDP multicast message notification (type 0), the associative array will contain an "ssdpType" key, the value of which designates whether it is a NOTIFY or M-SEARCH message. In most cases, it is best to ignore M-SEARCH messages, unless you are implementing a UPnP device (the UPnP controller object does allow this).

During an M-SEARCH request, a "new device" notification (type 2) will only be sent when a device is added to the controller's internal list. Once a device is part of the device list, subsequent M-SEARCH requests will only return type 1 (search response) values for that device. This type 1 response returns an associative array with message headers, but not an `roUPnPDevice` object (which is used to contain a complete set of device information).

See the `roUPnPSearchEvent` page for more information about the messages sent by the UPnP Controller.

roUPnPDevice

ON THIS PAGE

- [ifUPnPDevice](#)
 - [GetUUID\(\) As String](#)
 - [GetHeaders\(\) As roAssociativeArray](#)
 - [GetDeviceInfo\(\) As roAssociativeArray](#)
 - [GetEmbeddedDevices\(\) As roAssociativeArray](#)
 - [GetEmbeddedDevice\(deviceType As String\) As roUPnPDevice](#)
 - [GetServices\(\) As roAssociativeArray](#)
 - [GetService\(serviceType As String\) As roUPnPService](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is returned by the `roUPnPSearchEvent.GetObject()` method under certain conditions.

`ifUPnPDevice`

`GetUUID() As String`

`GetHeaders() As roAssociativeArray`

Returns an associative array of headers (including vendor-specific extensions) associated with the advertisement or search.

GetDeviceInfo() As roAssociativeArray

Returns an associative array of device metadata from the device XML (applicable to root items only).

GetEmbeddedDevices() As roAssociativeArray

Returns an associative array of embedded *roUPnPDevice* object instances, keyed by device type.

GetEmbeddedDevice(deviceType As String) As roUPnPDevice

Returns an *roUPnPDevice* instance, using the specified deviceType as a unique identifier. The deviceType parameter must use one of the following formats:

- `urn:schemas-upnp:device:deviceType:v`: Search for any device of this type. The deviceType and version (v) are defined by the UPnP Forum working committee.
- `urn:domain-name:device:deviceType:v`: Search for any device of this type. The domain-name, deviceType, and version (v) are defined by the UPnP vender. Period characters in the domain name must be replaced with hyphens in accordance with RFC2141.

GetServices() As roAssociativeArray

Returns an associative array of embedded *roPnPService* object instances, keyed by type.

GetService(serviceType As String) As roUPnPService

Returns an *roUPnPService* instance of the specified type.

roUPnPSearchEvent

ON THIS PAGE

- [ifUPnPSearchEvent](#)
 - [GetObject\(\) As Object](#)
 - [GetType\(\) As Integer](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This event object is returned when there is a response to a *roUPnPController.Search()* operation. It is also returned when the *roUPnPController* object receives multicast UDP SSDP messages.

ifUPnPSearchEvent

GetObject() As Object

Returns either an *roUPnPDevice* or an *roAssociativeArray* instance, depending on the value returned from the *GetType()* method.

GetType() As Integer

Returns an integer value indicating the type of response:

- 0 (Advertisement) – Indicates the receipt of an SSDP multicast message, which can be either a NOTIFY message or an M-SEARCH message. The *GetObject()* method will return an associative array with all SSDP headers and an "ssdpType" key, which can have a value of either "m-search" or "notify".

- 1 (Search response) – Indicates that the message is a response to an *roUPnPController.Search()* request. The *GetObject()* method will return an associative array with all headers from the HTTP M-SEARCH response.
- 2 (New device added to the device list) – Indicates that the *roUPnPController* object has detected a new device and added it to the device list, which is maintained internally. Device detection can result from receiving either an *ssdp:alive* message or a response to an M-SEARCH message. The *GetObject()* method will return an *roUPnPDevice* instance containing information about the added device. This message type is only delivered once per new device. Once a device is part of the device list, subsequent M-SEARCH requests will only return type 1 (Search response) values for that device.
- 3 (Device will be deleted from the device list) – Indicates that a device will be deleted from the device list, which is maintained internally. A device is deleted from the list when the player receives an *ssdp:byebye* message from the device, when the device does not send an *ssdp:alive* message within the defined "max-age" interval, or when the device is forcibly removed using the *roUPnPController.RemoveDevice()* method. The *GetObject()* method will return an *roUPnPDevice* instance containing information about the device to be removed.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via *SetUserData()*. It will return *Invalid* if no data has been set.

roUPnPService

ON THIS PAGE

- [ifUPnPService](#)
 - [Invoke\(actionName As String, params As Object\) As Integer](#)
 - [Subscribe\(\) As Integer](#)
 - [RenewSubscription\(\) As Integer](#)
 - [GetSID\(\) As String](#)
 - [GetTimeout\(\) As Integer](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is returned by the *GetServices()* and *GetService()* methods on the *roUPnPDevice* object.

ifUPnPService

Invoke(actionName As String, params As Object) As Integer

Invokes an action asynchronously on the UPnP service. This method returns a transaction ID that can be used to match it against the associated *roUPnPActionResult* instance.

Subscribe() As Integer

Subscribes to events on the UPnP service asynchronously.

RenewSubscription() As Integer

Resubscribes to events on the UPnP service asynchronously. This method should only be called after calling `Subscribe()`.

GetSID() As String

Returns the subscription ID. This method should only be called after calling `Subscribe()`.

GetTimeout() As Integer

Returns the service timeout period. This method should only be called after calling `Subscribe()`.

`ifMessagePort`

SetPort(port As roMessagePort)

Specifies the port that will receive events generated by the `roUPnPService` instance.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

roUPnPServiceEvent

ON THIS PAGE

- [ifUPnPServiceEvent](#)
 - [GetUUID\(\) As String](#)
 - [GetVariable\(\) As String](#)
 - [GetValue\(\) As String](#)
 - [GetSequence\(\) As Integer](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This event object is returned whenever a UPnP event message is received (for example, from a `Subscribe()` operation on the `roUPnPService` object). If a UPnP event message contains multiple state variables, separate event objects will be returned for each state variable.

`ifUPnPServiceEvent`

GetUUID() As String

Returns the subscription ID of the subscription service sending the event. This string matches the value returned by the `GetSID()` method of the `roUPnPService` instance that generated the event.

GetVariable() As String

Returns the name of the UPnP state variable to which the value corresponds.

GetValue() As String

Returns the value of the variable.

GetSequence() As Integer

Returns the incrementing sequence number, which denotes the UPnP message from which the update originated. The sequence number will be the same for multiple variable updates reported in a single UPnP event.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via SetUserData(). It will return Invalid if no data has been set.

roTCPStream

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

ON THIS PAGE

- [ifStreamReceive](#)
 - [SetLineEventPort\(a As Object\)](#)
 - [SetByteEventPort\(a As Object\)](#)
 - [SetReceiveEol\(a As String\)](#)
 - [SetMatcher\(matcher As Object\) As Boolean](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifStreamSend](#)
 - [SetSendEol\(eol_sequence As String\) As Void](#)
 - [SendByte\(byte As Integer\) As Void](#)
 - [SendLine\(string As String\) As Void](#)
 - [SendBlock\(a As Dynamic\) As Void](#)
 - [Flush\(\)](#)
- [ifTCPStream](#)
 - [GetFailureReason\(\) As String](#)
 - [ConnectTo\(a As String, b As Integer\) As Boolean](#)
 - [Accept\(a As Object\) As Boolean](#)
 - [AsyncConnectTo\(a As String, b As Integer\) As Boolean](#)

ifStreamReceive

SetLineEventPort(a As Object)

SetByteEventPort(a As Object)

SetReceiveEol(a As String)

SetMatcher(matcher As Object) As Boolean

Instructs the stream to use the specified matcher. This object returns True if successful. Pass Invalid to this method to stop using the specified matcher.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via SetUserData(). It will return Invalid if no data has been set.

`ifStreamSend`

SetSendEol(eol_sequence As String) As Void

Sets the EOL sequence when writing to the stream. The default value is CR (ASCII value 13). If you need to set this value to a non-printing character, use the `chr()` global function.

SendByte(byte As Integer) As Void

Writes the specified byte to the stream.

SendLine(string As String) As Void

Writes the specified characters to the stream followed by the current EOL sequence.

SendBlock(a As Dynamic) As Void

Writes the specified characters to the stream. This method can support either a string or an *roByteArray*. If the block is a string, any null bytes will terminate the block.

Flush()

`ifTCPStream`

GetFailureReason() As String

Yields additional useful information if an *roTCPStream* method fails.

ConnectTo(a As String, b As Integer) As Boolean

Connects the stream to the specified host (designated using a dotted quad) and port. The function returns True upon success.

Accept(a As Object) As Boolean

Accepts an incoming connection event. The function returns True upon success.

AsyncConnectTo(a As String, b As Integer) As Boolean

Attempts to connect the stream to the specified host (designated using a dotted quad) and port. The function returns False if this action is immediately impossible (for example, when the specified host is not in the correct format). Otherwise, the function returns True upon success. The connect proceeds in the background, and an *roStreamConnectResultEvent* is posted to the associated message port when the connect attempt succeeds or fails.

`roUrlTransfer`

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifIdentity](#)
 - [GetIdentity\(\) As Integer](#)

- ifMessagePort
 - SetPort(port As roMessagePort) As Void
- ifUrlTransfer
 - SetUrl(URL As String) As Boolean
 - AddHeader(name As String, value As String) As Boolean
 - GetToString() As String
 - GetToFile(filename As String) As Integer
 - AsyncGetToString() As Boolean
 - AsyncGetToFile(filename As String) As Boolean
 - EnableResume(enable As Boolean) As Boolean
 - Head() As Object
 - AsyncHead() As Boolean
 - PostFromString(request As String) As Integer
 - PostFromFile(filename As String) As Integer
 - AsyncPostFromString(request As String) As Boolean
 - AsyncPostFromFile(filename As String) As Boolean
 - SetUserAndPassword(user As String, password As String) As Boolean
 - SetMaximumReceiveBytesPerSecond(bytes_per_second as Double) As Boolean
 - SetMaximumSendBytesPerSecond(bytes_per_second as Double) As Boolean
 - SetMinimumTransferRate(bytes_per_second As Integer, period_in_seconds As Integer) As Boolean
 - GetFailureReason() As String
 - SetHeaders(a As Object) As Boolean
 - AsyncGetObject(type As String) As Boolean
 - AsyncCancel() As Boolean
 - EnableUnsafeAuthentication(enable As Boolean) As Boolean
 - EnableUnsafeProxyAuthentication(enable As Boolean) As Boolean
 - EnablePeerVerification(verification As Boolean) As Boolean
 - EnableHostVerification(verification As Boolean) As Boolean
 - SetCertificatesFile(filename As String) As Boolean
 - SetClientCertificate(parameters As roAssociativeArray) As Boolean
 - SetCookie(cookie As String) As Boolean
 - SetCookieFile(filename As String) As Boolean
 - GetCookies() As roList
 - EnableEncodings(enable As Boolean) As Boolean
 - SetUserAndPassword(a As String, b As String) As Boolean
 - Head() As Object
 - Escape(unescaped As String) As String
 - Unescape(a As String) As String
 - GetUrl() As String
 - SetProxy(proxy As String) As Boolean
 - SetProxyBypass(hostnames As Array) As Boolean
 - PutFromString(a As String) As Integer
 - SetTimeout(milliseconds As Integer) As Boolean
 - SetUserAgent(a As String) As Boolean
 - PutFromFile(a As String) As Integer
 - AsyncPutFromString(a As String) As Boolean
 - AsyncPutFromFile(a As String) As Boolean
 - Delete() As Object
 - AsyncDelete() As Boolean
 - ClearHeaders() As Void
 - AddHeaders(a As Object) As Boolean
 - SyncMethod(a As Object) As Object
 - SetRelativeLinkPrefix(prefix As String) As Boolean
 - BindToInterface(interface As Integer) As Boolean
 - AsyncMethod(parameters As roAssociativeArray) As Boolean

▼ Firmware Version 7.0

- Version 7.0

- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is used for reading from and writing to remote servers through URLs. It reports transfer status using the *roUrlEvent* object.

Object Creation: This object is created with no parameters.

```
CreateObject("roUrlTransfer")
```

Note

You must create a separate *roUrlTransfer* instance for each asset you wish to read/write.

```
ifUserData
```

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via *SetUserData()*. It will return *Invalid* if no data has been set.

```
ifIdentity
```

GetIdentity() As Integer

Returns a unique number that can be used to identify when events originate from this object.

Note

The *ifIdentity* interface has been deprecated. We recommend using the *ifUserData* interface instead.

```
ifMessagePort
```

SetPort(port As roMessagePort) As Void

Sets the message port to which events will be posted for asynchronous requests.

```
ifUrlTransfer
```

SetUrl(URL As String) As Boolean

Sets the URL for the transfer request. This function returns *False* on failure. Use *GetFailureReason* to learn the reason for the failure.

When using *SetUrl* to retrieve content from local storage, you do not need to specify the full file path: `SetUrl("file:/example.html")`. If the content is located somewhere other than the current storage device, you can specify it within the string itself. For example, you can use the following syntax to retrieve content from a storage device inserted into the USB port when the current device is an SD card: `SetUrl("file:///USB1:/example.html")`.

AddHeader(name As String, value As String) As Boolean

Adds the specified HTTP header. This is only valid for HTTP URLs. This function returns *False* on failure. Use *GetFailureReason()* to learn the reason for the failure.

GetToString() As String

Connects to the remote service as specified in the URL and returns the response body as a string. This function cannot return until the exchange is complete, and it may block for a long time. Having a single string return means that much of the information (headers, response codes) has been discarded. If you need this information, you can use *AsyncGetToString()* instead.

The size of the returned string is limited to 65,536 characters.

GetToFile(filename As String) As Integer

Connects to the remote service as specified in the URL and writes the response body to the specified file. This function does not return until the exchange is complete and may block for a long time. The response code from the server is returned. It is not possible to access any of the response headers. If you need this information, use `AsyncGetToFile()` instead.

AsyncGetString() As Boolean

Begins a GET request to a string asynchronously. Events will be sent to the message port associated with the object. If `False` is returned, then the request could not be issued and no events will be delivered.

AsyncGetToFile(filename As String) As Boolean

Begins a GET request to a file asynchronously. Events will be sent to the message port associated with the object. If `False` is returned, then the request could not be issued and no events will be delivered.

EnableResume(enable As Boolean) As Boolean

Specifies the file-creation behavior of the `GetToFile()` and `AsyncGetToFile()` methods. If this method is set to `False` (the default setting), each download will generate a temporary file: if the download is successful, the temporary file will be renamed to the specified filename; if the download fails, the temporary file will be deleted. If this method is set to `True`, the file with the specified filename will be created regardless of whether the download is successful or not—this allows the download to be resumed by a subsequent `GetToFile()` or `AsyncGetToFile()` call.

Head() As Object

Synchronously perform an HTTP HEAD request and return the resulting response code and headers through an `roUrlEvent` object. In the event of catastrophic failure (e.g. an asynchronous operation is already active), a null object is returned.

AsyncHead() As Boolean

Begins an asynchronous HTTP HEAD request. Events will be sent to the message port associated with the object. If the request could not be issued, the method will return `False` and will not deliver any events.

PostFromString(request As String) As Integer

Uses the HTTP POST method to post the supplied string to the current URL and return the response code. Any response body is discarded.

PostFromFile(filename As String) As Integer

Uses the HTTP POST method to post the contents of the file specified to the current URL and then return the response code. Any response body is discarded.

AsyncPostFromString(request As String) As Boolean

Uses the HTTP POST method to post the supplied string to the current URL. Events of type `roUrlEvent` will be sent to the message port associated with the object. A `False` return indicates that the request could not be issued and no events will be delivered.

AsyncPostFromFile(filename As String) As Boolean

Uses the HTTP POST method to post the contents of the specified file to the current URL. Events of the type `roUrlEvent` will be sent to the message port associated with the object. A `False` return indicates that the request could not be issued and no events will be delivered.

SetUserAndPassword(user As String, password As String) As Boolean

Enables HTTP authentication using the specified user name and password. Note that HTTP basic authentication is deliberately disabled due to it being inherently insecure. HTTP digest authentication is supported.

SetMaximumReceiveBytesPerSecond(bytes_per_second as Double) As Boolean

Limits the rate at which downloads are performed by the `roUrlTransfer` instance. The source data rate isn't under the direct control of the BrightSign player, but download rates should average below the specified value over time.

This method returns `true` on success and `false` on failure. In event of failure, the `GetFailureReason()` method may provide more information.

SetMaximumSendBytesPerSecond(bytes_per_second as Double) As Boolean

Limits the rate at which uploads are performed by the `roUrlTransfer` instance.

This method returns `true` on success and `false` on failure. In event of failure, the `GetFailureReason()` method may provide more information.

SetMinimumTransferRate(bytes_per_second As Integer, period_in_seconds As Integer) As Boolean

Causes the transfer to be terminated if the rate drops below `bytes_per_second` when averaged over `period_in_seconds`. Note that if the transfer is over the Internet, you may not want to set `period_in_seconds` to a small number in case network problems cause temporary drops in performance. For large file transfers and a small `bytes_per_second` limit, averaging fifteen minutes or more might be appropriate.

GetFailureReason() As String

May provide additional information if any of the `roUrlTransfer` methods indicate failure.

SetHeaders(a As Object) As Boolean

AsyncGetObject(type As String) As Boolean

Begins an asynchronous GET request and uses the contents to create an object of the specified type. Events will be sent to the message port associated with the object. If this method returns `False`, the request could not be issued and no events will be delivered.

AsyncCancel() As Boolean

EnableUnsafeAuthentication(enable As Boolean) As Boolean

Supports basic HTTP authentication if `True`. HTTP authentication uses an insecure protocol, which might allow others to easily determine the password. The `roUrlTransfer` object will still prefer the stronger digest HTTP if it is supported by the server. If this method is `False` (which is the default setting), it will refuse to provide passwords via basic HTTP authentication, and any requests requiring this authentication will fail.

EnableUnsafeProxyAuthentication(enable As Boolean) As Boolean

Supports basic HTTP authentication against proxies if `True` (which, unlike `EnableUnsafeAuthentication()`, is the default setting). HTTP authentication uses an insecure protocol, which might allow others to easily determine the password. If this method is `False`, it will refuse to provide passwords via basic HTTP authentication, and any requests requiring this authentication type will fail.

EnablePeerVerification(verification As Boolean) As Boolean

Enables checking of TLS/SSL certificates. This method is set to `true` by default. Disabling peer verification allows you to bypass an expired certificate check.

EnableHostVerification(verification As Boolean) As Boolean

Enables checking of the TLS/SSL certificate for the correct hostname. This method is set to `true` by default. Disabling host verification allows you to accept a certificate being sent for the wrong hostname.

Important

Peer verification and host verification are important security checks that prevent "man-in-the-middle" attacks. These features should only be disabled after careful consideration of the security implications.

SetCertificatesFile(filename As String) As Boolean

Configures an alternative set of CA certificates for the connection. This method is useful if the connection certificates are signed by a CA that is not on the the default trusted list (for example, if your organization uses a private CA hierarchy that is not signed by a well known root CA). This method replaces the default list, so the passed certificate file must contain all acceptable CA certificates required for the connection.

SetClientCertificate(parameters As roAssociativeArray) As Boolean

Specifies an HTTPS Client Certificate for use with server authentication. PKCS#12 certificates are not supported. This method accepts an associative array with the following parameters:

- `certificate_file`: The filename of the Client Certificate file
- `key_file`: The filename of the key file. You do not need to specify a key file if the key is embedded in the Client Certificate file (which might be the case when using PEM format).
- `type`: Either "PEM" or "DER"
- `passphrase`: The string passphrase to use if the key is encrypted

- `obfuscated_passphrase`: The obfuscated string passphrase to use if the key is encrypted.

Note

If any of the parameters are set incorrectly, you'll likely get a -35 error when making a request.

SetCookie(cookie As String) As Boolean

Adds the specified cookie to the player storage and enables the cookie parsing/sending engine. The cookie can be either a single line in Netscape/Mozilla format or a standard HTTP-style header (i.e. `Set-Cookie:`). You can also carry out commands by passing these exact strings to the method:

- "ALL": Erases all cookies held in memory.
- "SESS": Erases all session cookies held in memory.
- "RELOAD": Loads all cookies from files specified by `SetCookieFile()` calls.

SetCookieFile(filename As String) As Boolean

Adds cookies to the player storage using the specified cookie file and enables the cookie parsing/sending engine. The cookie data can be in either Netscape/Mozilla format or standard HTTP format.

GetCookies() As roList

Returns a string list of all cookies (including expired cookies).

EnableEncodings(enable As Boolean) As Boolean

Enables HTTP compression, which communicates to the server that the system can accept any encoding that the `roUrlTransfer` object is capable of decoding by itself. This currently includes "deflate" and "gzip", which allow for transparent compression of responses. Clients of the `roUrlTransfer` instance see only the decoded data and are unaware of the encoding being used.

Note

HTTP compression is enabled by default in firmware versions 6.0.x and later.

SetUserAndPassword(a As String, b As String) As Boolean

Head() As Object

Performs a synchronous HTTP HEAD request and returns the resulting response code and headers through an `roURLEvent` object. In the event of catastrophic failure (e.g. an asynchronous operation is already active), a null object is returned.

Escape(unescaped As String) As String

Converts the provided string to a URL-encoded string. All characters that could be misinterpreted in a URL context are converted to the `%XX` form.

Unescape(a As String) As String

GetUrl() As String

SetProxy(proxy As String) As Boolean

Sets the name or address of the proxy server that will be used by the `roUrlTransfer` instance. The proxy string should be formatted as follows: "http://user:password@hostname:port". It can contain up to four "*" characters; each "*" character can be used to replace one octet from the current IP address. For example, if the IP address is currently 192.168.1.2, and the proxy is set to "proxy-*-**", then the player will attempt to use a proxy named "proxy-192.168".

SetProxyBypass(hostnames As Array) As Boolean

Exempts the specified hosts from the proxy setting. The passed array should consist of one or more hostnames. The player will attempt to reach the specified hosts directly rather than using the proxy that has been specified with the `SetProxy()` method. For example, the hostname "example.com" would exempt "example.com", "example.com:80", and "www.example.com" from the proxy setting.

PutFromString(a As String) As Integer

Uses the HTTP PUT method to write the supplied string to the current URL and return the response code. Any response body is discarded; use *roUrlTransfer.SyncMethod* to retrieve the response body.

SetTimeout(milliseconds As Integer) As Boolean

Terminates the transfer if the request takes longer than the specified number milliseconds. Note that this includes the time taken by any name lookups, so setting this value too low will cause undesirable results. Passing 0 to the method disables the timeout. This method returns True upon success and False upon failure. In the event of failure, using the *GetFailureReason()* method may provide more information. If the operation times out, the status return is -28.

SetUserAgent(a As String) As Boolean

PutFromFile(a As String) As Integer

Uses the HTTP PUT method to write the contents of the specified file to the current URL and return the response code. Any response body is discarded; use *roUrlTransfer.SyncMethod* to retrieve the response body.

AsyncPutFromString(a As String) As Boolean

Uses the HTTP PUT method to write the supplied string to the current URL. Events of type *roUrlEvent* will be sent to the message port associated with the object. A False return indicates that the request could not be issued and no events will be delivered. Any response body is discarded; use *roUrlTransfer.AsyncMethod* to retrieve the response body.

AsyncPutFromFile(a As String) As Boolean

Uses the HTTP PUT method to write the contents of the specified file to the current URL. Events of type *roUrlEvent* will be sent to the message port associated with the object. A False return indicates that the request could not be issued and no events will be delivered. Any response body is discarded; use *roUrlTransfer.AsyncMethod* to retrieve the response body.

Delete() As Object

Uses the HTTP DELETE method to delete the resource at the current URL and return the response code. Any response body is discarded; use *roUrlTransfer.SyncMethod* to retrieve the response body.

AsyncDelete() As Boolean

Uses the HTTP DELETE method to delete the resource at the current URL. Events of type *roUrlEvent* will be sent to the message port associated with the object. A False return indicates that the request could not be issued and no events will be delivered. Any response body is discarded; use *roUrlTransfer.AsyncMethod* to retrieve the response body.

ClearHeaders() As Void

Removes all headers that would be supplied with an HTTP request.

AddHeaders(a As Object) As Boolean

Adds one or more headers to HTTP requests. Pass headers to this object as an *roAssociativeArray* of name/value pairs. This method returns True upon success and False upon failure. All headers that are added with this method will continue to be sent with HTTP requests until *ClearHeaders()* is called.

SyncMethod(a As Object) As Object

Performs a synchronous HTTP method request using the specified parameters. If the request is started successfully, then the method returns an *roUrlEvent* object containing the results of the request. This method returns Invalid if the the request could not be started. In this case, the *GetFailureReason()* method may provide more information.

SetRelativeLinkPrefix(prefix As String) As Boolean

Places the specified prefix in front of the URL if the URL is relative. Use this method to easily make *file:///* URLs drive agnostic.

BindToInterface(interface As Integer) As Boolean

Ensures that the request only goes out over the specified network interface. By default, the request goes out over the most appropriate network interface (which may depend on the routing metric configured via *roNetworkConfiguration*). Note that if both interfaces are on the same layer 2 network, this method may not always work as expected due to the Linux weak host model. The default behavior can be selected by passing -1 to the method. This method returns False upon failure. In this case, the *GetFailureReason()* method may provide more information.

AsyncMethod(parameters As roAssociativeArray) As Boolean

Begins an asynchronous HTTP method request using the specified parameters (see below). If the request is started successfully, the method returns True and will deliver an event. If the request could not be started, then the method returns False and will not deliver an event. If this occurs, you may be able to use the `GetFailureReason()` method to get more information.

The parameters are specified using an *roAssociativeArray* instance that may contain the following members:

Name	Type	Description
method	String	An HTTP method. Normal values include "HEAD", "GET", "POST", "PUT", and "DELETE". Other values are supported; however, depending on server behavior, they may not work as expected.
request_body_string	String	A string containing the request body.
request_body_file	String	The name of a file that contains the request body
response_body_string	Boolean	If specified and set to True, the response will be stored in a string and provided via the <i>roUrlEvent.GetString()</i> method.
response_body_file	String	The name of the file that will contain the response body. The body is written to a temporary file and then renamed to the specified filename if successful.
response_body_resume_file	String	The name of the file that will contain the response body. For a GET request, a RANGE header is sent based on the current size of the file, which is written in place rather than using a temporary file.
response_body_object	String	Uses the response body to create an object of the specified type. See the entry for <code>AsyncGetToObject()</code> for supported object types.
response_pipe	<i>roArray</i>	Use a pipeline of handlers to process the response body as it is received. See below for more details.

The *roArray* response for `response_pipe` consists of one or more *roAssociativeArray* instances containing a filter description (see below). The last associative array is usually an output filter.

Name	Type	Description
hash	String	Calculate a hash (digest) of the data using the specified algorithm as it passes through the pipeline. Supported hashes include the following: "CRC32", "MD5", "SHA1", "SHA256", "SHA384", "SHA512". The resulting hash can be retrieved as a hexadecimal string using the <i>roUrlEvent.GetHash()</i> method.
decompress	String	Decompress the response body using the specified algorithm. Currently, the only supported algorithm is "gzip". It is often easier to use an HTTP Content-Encoding rather than explicitly decompressing the body.
prefix_capture	Integer	Capture the specified number of bytes (between 1 and 16384) from the start of the stream and store them separately. The bytes can be retrieved using the <i>roUrlEvent.GetPrefix()</i> method, but they cannot be passed on to subsequent filters.

output_file	String	Output the pipeline to the specified file. The output is written to a temporary file and then renamed to the specified filename if successful.
output_string	Boolean	If specified and set to True, the response will be stored in a string and provided via the <i>roUrlEvent.GetString()</i> method.

The following example code specifies an array of handlers to filter the response body of an HTTP request.

```
url = CreateObject("roUrlTransfer")
pipe = [ { decompress: "gzip"}, { hash: "MD5" }, { output_file: "test.txt" } ]
result = url.AsyncMethod({ method: "GET", response_pipe: pipe })
```

roUrlEvent

ON THIS PAGE

- [ifInt](#)
 - [GetInt\(\) As Integer](#)
- [ifString](#)
 - [GetString\(\) As String](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\) As Void](#)
 - [GetUserData\(\) As Object](#)
- [ifSourceIdentity](#)
 - [GetSourceIdentity\(\) As Integer](#)
- [ifUrlEvent](#)
 - [GetResponseCode\(\) As Integer](#)
 - [GetObject\(\) As Object](#)
 - [GetFailureReason As String](#)
 - [GetSourceIdentity As Integer](#)
 - [GetResponseHeaders\(\) As roAssociativeArray](#)
 - [GetHash\(\) As String](#)
 - [GetPrefix\(\) As String](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This event is generated by the *roUrlTransfer* object.

ifInt

GetInt() As Integer

Returns the type of event. The following event types are currently defined:

- 1: transfer complete
- 2: transfer started

ifString

GetString() As String

Returns the string associated with the event. For transfer-complete `AsyncGetToString()`, `AsyncPostFromString()`, and `AsyncPostFromFile()` requests, this will be the actual response body from the server, truncated to 65,536 characters.

`ifUserData`

SetUserData(user_data As Object) As Void

Sets the user data.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()` (on either the event or source object). It will return `Invalid` if no data has been set.

`ifSourceIdentity`

GetSourceIdentity() As Integer

Returns a unique number that can be matched with the value returned by `roUrlTransfer.GetIdentity()` to determine where this event originated.

Note

The `ifSourceIdentity` interface has been deprecated. We recommend using the `ifUserData` interface instead.

`ifUrlEvent`

GetResponseCode() As Integer

Returns the protocol response code associated with an event. The following codes indicate success:

- 200: Successful HTTP transfer
- 226: Successful FTP transfer
- 0: Successful local file transfer

For unexpected errors, the return value is negative. There are many possible negative errors from the CURL library, but it is often best to look at the text version by calling `GetFailureReason()`.

Here are some potential errors. Not all of them can be generated by a BrightSign player:

Status	Name	Description
-1	CURLE_UNSUPPORTED_PROTOCOL	
-2	CURLE_FAILED_INIT	
-3	CURLE_URL_MALFORMAT	
-5	CURLE_COULDNT_RESOLVE_PROXY	
-6	CURLE_COULDNT_RESOLVE_HOST	
-7	CURLE_COULDNT_CONNECT	
-8	CURLE_FTP_WEIRD_SERVER_REPLY	
-9	CURLE_REMOTE_ACCESS_DENIED	A service was denied by the server due to lack of access. When login fails, this is not returned.
-11	CURLE_FTP_WEIRD_PASS_REPLY	
-13	CURLE_FTP_WEIRD_PASV_REPLY	
-14	CURLE_FTP_WEIRD_227_FORMAT	

-15	CURLE_FTP_CANT_GET_HOST	
-17	CURLE_FTP_COULDNT_SET_TYPE	
-18	CURLE_PARTIAL_FILE	
-19	CURLE_FTP_COULDNT_RETR_FILE	
-21	CURLE_QUOTE_ERROR	Failed quote command
-22	CURLE_HTTP_RETURNED_ERROR	
-23	CURLE_WRITE_ERROR	
-25	CURLE_UPLOAD_FAILED	Failed upload command.
-26	CURLE_READ_ERROR	Could not open/read from file.
-27	CURLE_OUT_OF_MEMORY	
-28	CURLE_OPERATION_TIMEDOUT	The timeout time was reached.
-30	CURLE_FTP_PORT_FAILED	FTP PORT operation failed.
-31	CURLE_FTP_COULDNT_USE_REST	REST command failed.
-33	CURLE_RANGE_ERROR	RANGE command did not work.
-34	CURLE_HTTP_POST_ERROR	
-35	CURLE_SSL_CONNECT_ERROR	Wrong when connecting with SSL.
-36	CURLE_BAD_DOWNLOAD_RESUME	Could not resume download.
-37	CURLE_FILE_COULDNT_READ_FILE	
-38	CURLE_LDAP_CANNOT_BIND	
-39	CURLE_LDAP_SEARCH_FAILED	
-41	CURLE_FUNCTION_NOT_FOUND	
-42	CURLE_ABORTED_BY_CALLBACK	
-43	CURLE_BAD_FUNCTION_ARGUMENT	
-45	CURLE_INTERFACE_FAILED	CURLOPT_INTERFACE failed.
-47	CURLE_TOO_MANY_REDIRECTS	Catch endless re-direct loops.
-48	CURLE_UNKNOWN_TELNET_OPTION	User specified an unknown option.
-49	CURLE_TELNET_OPTION_SYNTAX	Malformed telnet option.
-51	CURLE_PEER_FAILED_VERIFICATION	Peer's certificate or fingerprint wasn't verified correctly.
-52	CURLE_GOT_NOTHING	When this is a specific error.
-53	CURLE_SSL_ENGINE_NOTFOUND	SSL crypto engine not found.
-54	CURLE_SSL_ENGINE_SETFAILED	Cannot set SSL crypto engine as default.
-55	CURLE_SEND_ERROR,	Failed sending network data.
-56	CURLE_RECV_ERROR	Failure in receiving network data.
-58	CURLE_SSL_CERTPROBLEM	Problem with the local certificate.
-59	CURLE_SSL_CIPHER	Could not use specified cipher.
-60	CURLE_SSL_CACERT	Problem with the CA cert (path?)
-61	CURLE_BAD_CONTENT_ENCODING	Unrecognized transfer encoding.
-62	CURLE_LDAP_INVALID_URL	Invalid LDAP URL.

-63	CURLE_FILESIZE_EXCEEDED,	Maximum file size exceeded.
-64	CURLE_USE_SSL_FAILED,	Requested FTP SSL level failed.
-65	CURLE_SEND_FAIL_REWIND,	Sending the data requires a rewind that failed.
-66	CURLE_SSL_ENGINE_INITFAILED	Failed to initialize ENGINE.
-67	CURLE_LOGIN_DENIED	User, password, or similar field was not accepted and login failed .
-68	CURLE_TFTP_NOTFOUND	File not found on server.
-69	CURLE_TFTP_PERM	Permission problem on server.
-70	CURLE_REMOTE_DISK_FULL	Out of disk space on server.
-71	CURLE_TFTP_ILLEGAL	Illegal TFTP operation.
-72	CURLE_TFTP_UNKNOWNID	Unknown transfer ID.
-73	CURLE_REMOTE_FILE_EXISTS	File already exists.
-74	CURLE_TFTP_NOSUCHUSER	No such user.
-75	CURLE_CONV_FAILED	Conversion failed.
-76	CURLE_CONV_REQD	Caller must register conversion callbacks using the following URL_easy_setopt options: CURLOPT_CONV_FROM_NETWORK_FUNCTION CURLOPT_CONV_TO_NETWORK_FUNCTION CURLOPT_CONV_FROM_UTF8_FUNCTION
-77	CURLE_SSL_CACERT_BADFILE	Could not load CACERT file, missing or wrong format.
-78	CURLE_REMOTE_FILE_NOT_FOUND	Remote file not found.
-79	CURLE_SSH	Error from the SSH layer (this is somewhat generic, so the error message will be important when this occurs).
-80	CURLE_SSL_SHUTDOWN_FAILED	Failed to shut down the SSL connection.

The following error codes are generated by the system software, and are outside the range of CURL events:

Status	Name	Description
-10001	ERROR_CANCELLED	The transfer request has been cancelled because the <i>roUrlTransfer</i> instance is out of scope.
-10002	ERROR_EXCEPTION	The callback threw an exception.

GetObject() As Object

Returns the object associated with the event. Currently, this method can only return an object created in response to an *roUrlTransfer.AsyncGetToObject* request.

GetFailureReason As String

Returns a description of the failure that occurred.

GetSourceIdentity As Integer

Returns a unique number that can be matched with the value returned by *roUrlTransfer.GetIdentity()* to determine where the event came from.

GetResponseHeaders() As roAssociativeArray

Returns an associative array containing all the headers returned by the server for appropriate protocols (such as HTTP).

GetHash() As String

The hash (digest) of the response body, as specified by the `response_pipe{hash:}` parameter of the `roUrlTransfer.AsyncMethod()` method.

GetPrefix() As String

A number of bytes from the start of the response body. The amount of bytes is specified with the `response_pipe{prefix_capture:}` parameter of the `roUrlTransfer.AsyncMethod()` method.

Input/Output Objects

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This section describes objects that enable input/output functions on hardware interfaces.

- [roBtManager](#)
- [roBtClientManager](#)
- [roBtClientManagerEvent](#)
- [roBtClient](#)
- [roBtClientEvent](#)
- [roCecInterface](#)
- [roCecRxFrameEvent](#)
- [roCecTxCompleteEvent](#)
- [roChannelManager](#)
- [roControlPort](#)
- [roControlUp](#), [roControlDown](#)
- [roGpioButton](#)
- [roGpioControlPort](#)
- [roIRReceiver](#)
- [roIRDownEvent](#), [roIRRepeatEvent](#), [roIRUpEvent](#)
- [roIRTransmitter](#)
- [roIRTransmitCompleteEvent](#)
- [roIRRemote](#)
- [roIRRemotePress](#)
- [roKeyboard](#)
- [roKeyboardPress](#)
- [roSequenceMatcher](#)
- [roSequenceMatchEvent](#)
- [roSerialPort](#)

roBtManager

ON THIS PAGE

- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifIdentity](#)
 - [GetIdentity\(\) As Integer](#)
- [ifBtPeripheralManager](#)
 - [GetFailureReason\(\) As String](#)
 - [GetAdapterList\(\) As roArray](#)

- [StopAdvertising\(\) As Boolean](#)
- [GetAdvertisingList\(\) As roArray](#)
- [StartAdvertising\(data As roAssociativeArray\) As Boolean](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object facilitates BLE one-way communication (i.e. "beaconing"): Use the *roBtManager* object to discover whether any BLE adapters are present and to send BLE advertisements using the adapters.

ifMessagePort

SetPort(port As roMessagePort)

Posts messages of type *roBtEvent* to the attached message port. Use these messages to detect insertion or removal of Bluetooth adapters.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Retrieves an arbitrary object set via the *SetUserData()* method.

ifIdentity

GetIdentity() As Integer

Returns a unique number that can be used to identify events that originate from the object instance.

Note

The *ifIdentity* interface has been deprecated. We recommend using the *ifUserData* interface instead.

ifBtPeripheralManager

GetFailureReason() As String

Returns additional diagnostic information if a method returns False.

GetAdapterList() As roArray

Returns an array describing all available Bluetooth adapters. Each entry in the array consists of an associative array containing adapter properties. At present, each associative array contains a single *name* property that describes the adapter name. Use this method to determine if Bluetooth adapters are connected to the player.

StopAdvertising() As Boolean

Stops all BLE advertisements. This method returns True on success and False on failure.

GetAdvertisingList() As roArray

Returns an array describing all active Bluetooth advertisements. Each entry in the array consists of an associative array describing a single advertisement. The associative-array values correspond to the properties set using the *StartAdvertising()* method, but can also include default parameter values that were not set explicitly. Note that all UUID values will be returned in lowercase.

StartAdvertising(data As roAssociativeArray) As Boolean

Begins transmitting a BLE "beacon" message. This method returns True on success and False on failure. Each message can contain data in a standard format or arbitrary custom values. The message format is specified using the `mode` parameter, and other required values in the associative array will depend on the value of this parameter:

- `mode: "beacon"`: This mode uses a simple beaconing format.
 - `beacon_uuid`: A string representation of a UUID, which can be in 16-bit, 32-bit, or 128-bit format. A 16-bit UUID must be exactly four hex digits with no punctuation; a 32-bit UUID must be exactly eight hex digits with no punctuation; and a 128-bit UUID must be punctuated exactly as follows: "cd7b6f81-f738-4cad-aebf-d2a2ea36d996".
 - `beacon_major`: An integer specifying the 2-byte Major value (0 to 65535)
 - `beacon_minor`: An integer specifying the 2-byte Minor value (0 to 65535)
 - `beacon_level`:(optional) An 8-bit signed integer (-127 to 128) that corresponds to the measurement of the Tx power level (in dBm) at 1 meter. The default level is -60.
 - `beacon_manufacturer`:(optional) A 2-byte integer value (0 to 65535) specifying the beacon manufacturer. The default value is 76 (&H4C).
 - `connectable`:(optional) A Boolean value indicating whether the advertisement should be connectable or not. Advertisements are non-connectable by default.
 - `persistent`:(optional) A Boolean value indicating whether the advertisement should persist after every reboot. Beacon advertisements are persistent by default.
- `mode: "eddystone-url"`: This mode uses the [Eddystone-URL](#) format.
 - `url`: The URL to encapsulate in the advertisement packet. If the URL is too long to fit in the packet, the `StartAdvertising()` call will return False and `GetFailureReason()` will report "Compressed URL is too long".
 - `tx_power`:(optional) An integer value that corresponds to the measurement of the Tx power level (in dBm) at 0 meters. The default value is -19, which corresponds to a level of -60dBm at 1 meter. The recommended calibration practice is to measure the RSSI in a circle at 1 meter from the beacon, then add 41 to the average measured signal strength to get the `tx_power` value: For example, measuring a -65dBm RSSI at 1 meter yields a **TxPower** value of -24.

Note

The `tx_power` value does not modify the power level of the Bluetooth transmitter (this requires [physical attenuation](#)). Rather, the `tx_power` value is transmitted to Bluetooth clients in the BLE advertisement. Clients can then compare this value to the current RSSI of the signal to determine their approximate distance from the beacon.

- `connectable`:(optional) A Boolean value indicating whether the advertisement should be connectable or not. Advertisements are non-connectable by default.
- `persistent`:(optional) A Boolean value indicating whether the advertisement should persist after every reboot. Eddystone-URL advertisements are persistent by default.
- `mode: "custom"`: This mode supports arbitrary custom data in a vendor-specific field.
 - `custom_manufacturer_data`:(optional) An associative array containing two keys:
 - `manufacturer`: A 2-byte integer value (0 to 65535)
 - `data`: An *roByteArray* instance containing data
 - `service_uuid`:(optional) A set of ServiceUUID elements, which can be specified as either an array of UUID strings or an array of associative arrays containing a `uuid` key/value pair. Each associative array can also contain a `data` key, which specifies *ServiceData* as an *roByteArray* instance.
 - `connectable`:(optional) A Boolean value indicating whether the advertisement should be connectable or not. Advertisements are non-connectable by default.
 - `persistent`:(optional) A Boolean value indicating whether the advertisement should persist after every reboot. Custom advertisements are not persistent by default.

To specify multiple advertisements, pass an array of associative arrays to the `StartAdvertising()` method. Advertisements will be sent in rotation. The method will fail if one or more advertisement is incorrect—try each advertisement individually and call `GetFailureReason()` to determine which advertisement is causing the error.

Calling the `StartAdvertising()` method will replace any previous advertisements. You can change a list of advertisements by modifying the array and calling `StartAdvertising()` again.

This script uses `roBtManager.GetAdapterList()` to determine if there are any Bluetooth adapters available:

```
btm = CreateObject("roBtManager")
if btm.GetAdapterList().Count() > 0 then print "Bluetooth available"
```

This script constructs two associative arrays for advertising with the "beacon" format and then broadcasts them both:

```
adv1 = { mode : "beacon", beacon_uuid : "41fac2b21-c8cb-41e7-b011-12d1016dd39e", beacon_major : 400, beacon_minor : 22 }
adv2 = { mode : "beacon", beacon_manufacturer: &H4C, beacon_uuid : "41fac221-c8cb-41e7-b011-12d1016dd39e", beacon_major : &H1234, beacon_minor : &HFF01, beacon_level : -50 }

advlist = [adv1, adv2]
bm.StartAdvertising(advlist)
```

The associative array can also be constructed in parts:

```
adv1 = { mode: "beacon" }
adv1.Append({ beacon_uuid : "41fac221-c8cb-41e7-b011-12d1016dd39e" })
adv1.Append({ beacon_major : 32000, beacon_minor : 100 })
```

This script constructs an advertisement with the "Eddystone-URL" format. It uses the optional `tx_power` parameter as well:

```
adv1 = { mode: "eddytone-url", url: "http://www.brightsign.biz", tx_power: -24}
```

This script constructs a custom-formatted advertisement:

```
custom_adv = CreateObject("roByteArray")
custom_adv.FromHexString("0215434B2EB8C28F40898E7A1E644BB13B9FA000B001C5")
adv2 = { mode : "custom", custom_manufacturer_data : { manufacturer: &H4C, data : custom_adv } }
```

roBtClientManager

ON THIS PAGE

- [ifBtClientManager](#)
 - [Start\(params As roAssociativeArray\) As Boolean](#)
 - [Stop\(\) As Boolean](#)
 - [SetDeviceInfo\(a As Object\) As Boolean](#)
 - [SetDeviceData\(a As Object\) As Boolean](#)
- [ifMessagePort](#)
 - [SetPort\(a As Object\)](#)
- [ifIdentity](#)
 - [GetIdentity\(\) As Integer](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifFailureReason](#)
 - [GetFailureReason\(\) As String](#)

This object represents a BLE GATT service, where the player acts as a peripheral and another device (phone, tablet, etc.) acts as the client/central. Unlike *roBtManager*, which allows for unidirectional, non-connectable communication (i.e. beaconing), the *roBtClientManager* object is used to facilitate two-way, connectable communication. It provides an event to an attached message port when a new client connects. It also manages the client state, provides client update events, and times out the client after a period of inactivity.

The BLE clients themselves are represented by the *roBtClient* object.

`ifBtClientManager`

Start(params As roAssociativeArray) As Boolean

Starts BLE communication. This method accepts the following parameters:

- `client_timeout`: The amount of time (in seconds) that a client can be inactive before being disconnected
- `service_uuid`: A UUID that identifies the service
- `client_uuid`: The client-identifier attribute. Once a client device connects to the player, it should write its unique client identifier value to this attribute.
- `user_variable_uuid`: The user variables attribute. This attribute is intended to be writable by both the player and the client, providing the primary means for exchanging state data. User variables can be written as a JSON string.
- `command_uuid`: The command attribute. When a client device writes to this attribute, an event containing the command string is generated by the associated *roBtClient* instance.
- `device_info_uuid`: The attribute for device information. This attribute is read-only for clients.
- `device_data_uuid`: The attribute for device data. This attribute can be used to communicate available commands to clients. It is read-only for clients.

Note

There is a limit 512 bytes on each attribute.

Stop() As Boolean

Stops BLE communication.

SetDeviceInfo(a As Object) As Boolean

SetDeviceData(a As Object) As Boolean

`ifMessagePort`

SetPort(a As Object)

Posts messages of type *roBtClientManagerEvent* to the attached message port.

`ifIdentity`

GetIdentity() As Integer

Returns a unique number that can be used to identify *roBtClientManagerEvent* objects that originate from this object.

Note

The *ifIdentity* interface has been deprecated. We recommend using the *ifUserData* interface instead.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifFailureReason`

GetFailureReason() As String

Returns additional useful information when a method on the *ifBtClientManager* interface returns `False`.

roBtClientManagerEvent

ON THIS PAGE

- [ifBtEvent](#)
 - [GetEvent\(\) As String](#)
 - [GetClient\(\) As roBtClient](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\) As Void](#)
 - [GetUserData\(\) As Object](#)

This event object is generated by the [roBtClientManager](#) object when a BLE client connects to or disconnects from the player.

`ifBtEvent`

GetEvent() As String

Returns the name of the event, which can be one of the following:

- "client-new": This value is returned the first time a client connects only.
- "client-delete": This value is returned when a client reaches the `client-timeout` interval and is disconnected.

GetClient() As roBtClient

Returns the [roBtClient](#) object for the client associated with the event.

`ifUserData`

SetUserData(user_data As Object) As Void

Sets the user data.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()` (on either the event or source object). It will return `Invalid` if no data has been set.

roBtClient

ON THIS PAGE

- [ifBtClient](#)
 - [GetClientId\(\) As String](#)
 - [SetUserVars\(data As String\) As Boolean](#)
 - [GetUserVars\(\) As String](#)
 - [Disconnect\(\) As Boolean](#)
- [ifMessagePort](#)
 - [SetPort\(a As Object\)](#)
- [ifUserData](#)
 - [SetUserData\(a As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifIdentity](#)
 - [GetIdentity\(\) As Integer](#)
- [ifFailureReason](#)
 - [GetFailureReason\(\) As String](#)

Each instance of this object represents a single BLE client. This object is not instantiated by the script; instead, when a BLE client connects to the player, the [roBtClientManager](#) object sends an event to its attached message port and creates a [roBtClient](#) instance.

The primary purpose of the *roBtClient* object is to transfer data between the player and the client via user variables (i.e. characteristics). When a client updates user-variables or sends a command, an *roBtClientEvent* is sent to the attached message port. The state can be retrieved using either the *roBtClient.GetUserVars()* method or the *roBtClientEvent.GetParam()* method: Updates from the *roBtClientEvent* object are queued until the script processes them, while the *roBtClient* object returns the most current state.

`ifBtClient`

GetClientId() As String

Retrieves the UUID of the client associated with the object instance.

SetUserVars(data As String) As Boolean

Specifies new data for the client to retrieve.

GetUserVars() As String

Retrieves the latest data from the *roBtClient* state.

Disconnect() As Boolean

Forces the player to disconnect from the client.

`ifMessagePort`

SetPort(a As Object)

Posts messages of type *roBtClientEvent* to the attached message port. An event is raised whenever the end of the queue is reached.

`ifUserData`

SetUserData(a As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifIdentity`

GetIdentity() As Integer

Returns a unique number that can be used to identify *roBtClientEvent* objects that originate from this object.

Note

The *ifIdentity* interface has been deprecated. We recommend using the *ifUserData* interface instead.

`ifFailureReason`

GetFailureReason() As String

Returns additional useful information if a method on the *ifBtClient* interface returns `False`.

roBtClientEvent

ON THIS PAGE

- [ifBtClientEvent](#)
 - [GetEvent\(\) As String](#)
 - [GetParam\(\) As String](#)
- [ifUserData](#)

- [SetUserData\(user_data As Object\) As Void](#)
- [GetUserData\(\) As Object](#)

This event object is generated by an *roBtClient* object when a BLE client updates the *roBtClient* state or sends a command.

`ifBtClientEvent`

GetEvent() As String

Returns the event name, which can currently be "client-update" or "client-command".

GetParam() As String

Returns the event data as a string.

`ifUserData`

SetUserData(user_data As Object) As Void

Sets the user data.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()` (on either the event or source object). It will return `Invalid` if no data has been set.

roCecInterface

ON THIS PAGE

- [IfCecInterface](#)
 - [SendRawMessage\(packet As roByteArray\) As Void](#)
 - [UseInitiatorAddressFromPacket\(enable As Boolean\) As Boolean](#)
 - [GetLogicalAddress\(\) As Integer](#)
 - [EnableCecDebug\(filename As String\) As Void](#)
 - [GetPhysicalAddress\(\) As roByteArray](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object provides access to the HDMI CEC channel.

Object Creation: The *roCecInterface* object is created with no parameters.

```
CreateObject("roCecInterface")
```

`IfCecInterface`

SendRawMessage(packet As roByteArray) As Void

Sends a message on the CEC bus. The frame data should be provided as an *roByteArray*, with the destination address in the low 4 bits of the first octet. The high 4 bits of the first octet will be replaced with the source address unless source bit replacement is disabled using the `UseInitiatorAddressFromPacket()` method.

Note

The physical address of the player is copied into the following messages automatically prior to transmission: `ActiveSource`, `InactiveSource`, `SystemAudioModeRequest`.

UseInitiatorAddressFromPacket(enable As Boolean) As Boolean

Removes the source address included by system software if passed `True`. This method allows you to transmit unmodified bytes via CEC.

GetLogicalAddress() As Integer

Returns the current logical address. This method will return the unregistered address (`0x0f`) to indicate that the logical address has not been negotiated. In this case, messages can still be sent, but the receivers will behave differently and may not process all messages.

EnableCecDebug(filename As String) As Void

Enables CEC debugging. This method will write a log of all CEC packets to the specified file.

GetPhysicalAddress() As roByteArray

Returns the CEC physical address, which is sometimes required when processing CEC messages.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifMessagePort`

SetPort(port As roMessagePort)

Posts messages of type *roCecRxFrameEvent* and *roCecTxCompleteEvent* to the attached message port.

roCecRxFrameEvent

ON THIS PAGE

- [ifCecRxFrameEvent](#)
 - [GetByteArray\(\) As Object](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

If an *roMessagePort* is attached to an *roCecInterface* instance, it will receive events of type *roCecRxFrameEvent*.

`ifCecRxFrameEvent`

GetByteArray() As Object

Returns the message data as an *roByteArray*.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned by `GetUserData()`.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

roCecTxCompleteEvent

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifCecRxFrameEvent](#)
 - [GetByteArray\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

If an *roMessagePort* is attached to an *roCecInterface* instance, it will receive events of type *roCecTxCompleteEvent*.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned by `GetUserData()`.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifCecRxFrameEvent`

GetByteArray() As Object

Returns the message data as an *roByteArray*.

0x00	Transmission successful
0x80	Unable to send, CEC hardware powered down
0x81	Internal CEC error
0x82	Unable to send, CEC line jammed
0x83	Arbitration error
0x84	Bit-timing error

0x85	Destination address not acknowledged
0x86	Data byte not acknowledged

roChannelManager

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)
- [ifChannelManager](#)
 - [Synchronous API](#)
 - [Scan\(parameters As roAssociativeArray\) As Boolean](#)
 - [GetChannelCount\(\) As Integer](#)
 - [ClearChannelData\(\) As Boolean](#)
 - [GetCurrentSnr\(\) As Integer](#)
 - [ExporttoXML\(\) As String](#)
 - [ImportFromXML\(a As String\) As Boolean](#)
 - [EnableScanDebug\(filename As String\) As Boolean](#)
 - [CreateChannelDescriptor\(a As Object\) As Object](#)
 - [Asynchronous API](#)
 - [AsyncScan\(parameters As roAssociativeArray\) As Boolean](#)
 - [CancelScan\(\) As Boolean](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

You can use this object to manage RF channel scanning and tuning. The *roVideoPlayer* method also has channel scanning capabilities.

Object Creation: The *roChannelManager* object is created with no parameters.

```
CreateObject("roChannelManager")
```

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via *SetUserData()*. It will return Invalid if no data has been set.

ifMessagePort

SetPort(port As roMessagePort)

ifChannelManager

The *ifChannelManager* interface provides both a Synchronous and Asynchronous API:

Synchronous API

Scan(parameters As roAssociativeArray) As Boolean

Performs a channel scan on the RF input for both ATSC and QAM frequencies and builds a channel map based on what it finds. The *roChannelManager* object stores a list of all channels that are obtained using the `CreateChannelDescriptor()` method (described below). The list is cleared on each call to `Scan()` by default, but this behavior can be overridden.

Each channel takes approximately one second to scan; you can limit the scope of the channel scan with the following parameters:

- ["ChannelMap"] = "ATSC" or "QAM": Limits the frequency scan to either QAM or ATSC.
- ["ModulationType"] = "QAM64" or "QAM256": Limits the modulation type of the scan to QAM64 or QAM256.
- ["FirstRfChannel"] = Integer and/or ["LastRfChannel"] = Integer: Limits the scan to the specified range of channels. The high end of the channel range is an optional parameter.
- ["ChannelStore"] = "DISCARD ALL" or "MERGE": Controls how the script handles previous channel scan information. The default setting is DISCARD ALL, which clears all channel data prior to scanning. On the other hand, MERGE overwrites the data only for channels specified in the scan.

GetChannelCount() As Integer

Returns the number of found channels.

ClearChannelData() As Boolean

Clears all stored channel scanning data, including that which persists in the registry. This method also cancels any `AsyncScan()` calls that are currently running.

GetCurrentSnr() As Integer

Returns the SNR (in centibels) of the currently tuned channel.

ExporttoXML() As String

Serializes the contents of RF channels into XML. You can write the XML to a file that can be used at a later point on the same or other units. See below for an example of XML output.

ImportFromXML(a As String) As Boolean

Retrieves the RF channel contents stored as XML. The formatting of the XML is controlled using version tags.

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE boost_serialization>
<boost_serialization signature="serialization::archive" version="7">
<ChannelList class_id="0" tracking_level="0" version="0">
<ChannelCount>2</ChannelCount>
<Channel class_id="1" tracking_level="0" version="0">
<RfChannel>42</RfChannel>
<ModulationType>7</ModulationType>
<SpectralInversion>0</SpectralInversion>
<MajorChannelNumber>1</MajorChannelNumber>
<MinorChannelNumber>1</MinorChannelNumber>
</Channel>
<Channel>
<RfChannel>42</RfChannel>
<ModulationType>7</ModulationType>
<SpectralInversion>0</SpectralInversion>
<MajorChannelNumber>1</MajorChannelNumber>
<MinorChannelNumber>2</MinorChannelNumber>
</Channel>
</ChannelList>
```

EnableScanDebug(filename As String) As Boolean

Allows all scan debugging to be written to a text file. By default, there is no debug output from a scan. You can close the debug file by passing an empty string.

Example

```
c=CreateObject("roChannelManager")
c.EnableScanDebug("tmp:/scandebug.txt")

v = CreateObject("roVideoPlayer")
aa = CreateObject("roAssociativeArray")
aa["RfChannel"] = 12
aa["VirtualChannel"] = "24.1"
print v.PlayFile(aa)

c.EnableScanDebug("")
```

CreateChannelDescriptor(a As Object) As Object

Creates an associative array that can either be passed to the *roVideoPlayer.PlayFile()* method (to tune to a channel) or parsed for metadata. The generated channel object can be based on one of the following:

- Index:

```
["ChannelIndex"] = 0
```

- Virtual channel number as a string in an associative array:

```
["VirtualChannel"] = "12.1"
```

- Channel name as a string:

```
["ChannelName"] = "KCBS"
```

Note

Channels are sorted internally by virtual channel, so you could use a Channel Index script to implement standard channel up/down behavior.

These are the entries generated in the array:

- VirtualChannel
- ChannelName
- CentreFrequency
- ModulationType
- VideoPid
- VideoCodec
- AudioPid
- AudioCodec
- SpectralInversion
- ChannelMap
- FirstRfChannel
- LastRfChannel

The last three entries in this array allow you to use the same *roArray* as a parameter for *Scan()* and *PlayFile()*. The first and last RF channel values are set to the same value so that only one RF channel will be scanned. This kind of scan can be performed at the same time as playing the channel because it doesn't require retuning.

Example


```

c=CreateObject("roChannelManager")
aa=CreateObject("roAssociativeArray")
aa["ChannelMap"] = "QAM"
aa["FirstRfChannel"] = 10
aa["LastRfChannel"] = 15
c.Scan(aa)

cinfo = CreateObject("roAssociativeArray")
cinfo["ChannelIndex"] = 0
desc = c.CreateChannelDescriptor(cinfo)
print desc

v = CreateObject("roVideoPlayer")
v.PlayFile(desc)
c.Scan(desc)

```

Asynchronous API

AsyncScan(parameters As roAssociativeArray) As Boolean

Begins a channel scan on the RF input and returns the results immediately. Otherwise, the behavior and parameters of this method are identical to `Scan()`. When completed or cancelled, `AsyncScan()` generates an *roChannelManagerEvent*, which supports *UserData* and outputs two types of event:

- 0 – Scan Complete: Generated upon the completion of a scan. No extra data is supplied.
- 1 – Scan Progress: Generated upon every tune that is performed during the scan. `GetData()` returns the percentage complete of the scan.

CancelScan() As Boolean

Cancels any asynchronous scans that are currently running. This method does not generate an *roChannelManagerEvent*.

Synchronous Example

```

c = CreateObject("roChannelManager")

' Scan the channels
aa = CreateObject("roAssociativeArray")
aa["ChannelMap"] = "ATSC"
aa["FirstRfChannel"] = 12
aa["LastRfChannel"] = 50
c.Scan(aa)

' Start at the first channel
index = 0
cinfo = CreateObject("roAssociativeArray")
cinfo["ChannelIndex"] = index
desc = c.CreateChannelDescriptor(cinfo)

' Play the first channel
v = CreateObject("roVideoPlayer")
v.PlayFile(desc)

' Play the second channel
index = index + 1
cinfo["ChannelIndex"] = index
desc = c.CreateChannelDescriptor(cinfo)
v.PlayFile(desc)

```

Asynchronous Example

```

    c = CreateObject("roChannelManager")
    p = CreateObject("roMessagePort")
    c.SetPort(p)

    ' Scan the channels
    aa = CreateObject("roAssociativeArray")
    aa["ChannelMap"] = "ATSC"
    aa["FirstRfChannel"] = 12
    aa["LastRfChannel"] = 50
    c.AsyncScan(aa)

loop:
    msg = Wait(2000,p)
    if msg = 0 then goto scan_complete
    goto loop

scan_complete:
    ' Start at the first channel
    index = 0
    cinfo = CreateObject("roAssociativeArray")
    cinfo["ChannelIndex"] = index
    desc = c.CreateChannelDescriptor(cinfo)

    ' Play the first channel
    v = CreateObject("roVideoPlayer")
    v.PlayFile(desc)

    ' Rescan the current channel, and update the
    desc["ChannelStore"] = MERGE
    c.Scan(desc)

```

roControlPort

ON THIS PAGE

- [ifControlPort](#)
 - [GetVersion\(\) As String](#)
 - [EnableOutput\(button As Integer\) As Boolean](#)
 - [EnableInput\(button As Integer\) As Boolean](#)
 - [EnableAlternateFunction\(button As Integer, pin_function As String\) As Boolean](#)
 - [GetWholeState\(\) As Integer](#)
 - [IsInputActive\(button As Integer\) As Boolean](#)
 - [SetWholeState\(state As Integer\) As Boolean](#)
 - [SetOutputState\(button As Integer, level As Boolean\) As Boolean](#)
 - [SetOutputValue\(offset As Integer, bit-mask As Integer\) As Boolean](#)
 - [SetOutputValues\(values As roAssociativeArray\) As Boolean](#)
 - [GetProperties\(\) As roAssociativeArray](#)
 - [SetPulseParams\(parameters As roAssociativeArray\) As Boolean](#)
 - [SetPulse\(button As Integer, bit-field As Integer\) As Boolean](#)
 - [RemovePulse\(button As Integer\) As Boolean](#)
- [ifMessagePort](#)
 - [SetPort\(port As Object\)](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifIdentity](#)
 - [GetIdentity\(\) As Integer](#)
- [BP200/BP900 Setup](#)
- [BP200/BP900 LED Output](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object provides support for the BP200/BP900 USB button boards, GPIO ports, and side buttons on the BrightSign player. Button presses are returned as *roControlUp* and *roControlDown* events. The object is used to configure output levels on the I/O connector and monitor inputs. Typically, LEDs and buttons are attached to the GPIO connector on the BrightSign player or the BrightSign Expansion Module.

Object Creation: The *roControlPort* object is created with a single parameter that specifies the port being used.

```
CreateObject("roControlPort", port As String)
```

The port parameter can be one of the following:

- **BrightSign**: Specifies the onboard GPIO connector (including the SVC (GPIO12) button).
- **Expander-GPIO**: Specifies the DB-25 connector on the BrightSign Expansion Module. If no BrightSign Expansion module is attached, then object creation will fail and *Invalid* will be returned.
- **Expander-<n>-GPIO**: Specifies a **USB-to-GPIO device** connected to the player. Multiple USB-to-GPIO devices can be controlled using separate *roControlPort* instances: The first device corresponds to "Expander-0-GPIO", the second to "Expander-1-GPIO", etc.
- **Expander-DIP**: Specifies the eight DIP switches on the BrightSign Expansion Module. If no BrightSign Expansion module is attached, then object creation will fail and *Invalid* will be returned.

Note

Hot-plugging the BrightSign Expansion Module is not supported.

- **Touchboard-<n>-GPIO**: Retrieves events from the specified BP200/BP900 button board. Events are handled in the same manner as events from the BrightSign port.
- **Touchboard-<n>-LED-SETUP**: Sets various LED output options for the specified BP200/BP900 button board.
- **Touchboard-<n>-LED**: Sets the bits for each button on the specified BP200/BP900 button board. The bits indicate whether the associated LED should be on or off.

Note

Since multiple BP200/BP900 button boards can be connected to a player simultaneously, the <n> value specifies the port enumeration of each board. This value corresponds to the <raw> or <fid> value returned by the *roDeviceInfo.GetUSBTTopology()* method. An unspecified enumeration value is synonymous with a button board with an enumeration value of 0 (e.g. Touchboard-GPIO and Touchboard-0-GPIO are identical).

ifControlPort

Note

The "button" numbers described below are not the same as GPIO "pin" numbers: Some pins act as power supply or ground, so they are not included in the button numbering scheme. See the [hardware manual](#) associated with your player model to view a mapping of buttons to pins.

GetVersion() As String

Returns the version number of the firmware (either the main BrightSign firmware or the BrightSign Expansion Module firmware) responsible for the control port.

EnableOutput(button As Integer) As Boolean

Marks the specified button as an output. If an invalid button number is passed, *false* will be returned. If successful, the function returns *true*. The output will be driven high or low depending on the current output state of the pin.

Tip

See [here](#) for a table of pins and corresponding buttons for the onboard GPIO connector.

EnableInput(button As Integer) As Boolean

Marks the specified button as an input. If an invalid button number is passed, `false` will be returned. If successful, the function returns `true`. The button will be tri-stated and can be driven high or low externally.

EnableAlternateFunction(button As Integer, pin_function As String) As Boolean

Enables an alternate function on a GPIO button. This method applies to the onboard GPIO connector and is currently supported on the XTx44, XTx43, XDx34, XDx33, HDx23, and HO523 models.

The first argument specifies the GPIO button number (between 0 and 7). The second argument specifies the alternate function setting; the following table outlines the possible alternate setting for each pin:

GPIO Pin	Button Number	Alternate Function
3	0	"serial1" (Rx)
4	1	"irin1"
5	2	"irout" (HDx23, HO523 only)
6	3	N/A
9	4	"serial0" (Rx - console port)*
10	5	"serial0" (Tx)*
11	6	"serial1" (Tx)
12	7	N/A

*Models that do not have a 3.5mm serial port (e.g. HD223, XD233) do not support serial port 0.

Note

To revert a GPIO button to its primary function, specify the `pin_function` as "gpio".

GetWholeState() As Integer

Returns the state of all the inputs attached to the control port as bits in an integer. Individual buttons can be checked using binary operations, although it is normally easier to call `IsInputActive()` instead.

IsInputActive(button As Integer) As Boolean

Returns the state of the specified input. If the button is not configured as an input, then the result is undefined.

SetWholeState(state As Integer) As Boolean

Specifies the desired state of all outputs attached to the control port as bits in an integer. The individual buttons can be set using binary operations, although it is normally easier to call `SetOutputState()` instead.

Example

```
port = CreateObject("roControlPort", "BrightSign")
gpio1 = 2 '2^1
gpio3 = 8 '2^3
gpio5 = 32 '2^5
gpio7 = 128 '2^7
port.SetWholeState(gpio1 + gpio2 + gpio5 + gpio7) 'turns on ports 1, 3, 5, and 7
```

SetOutputState(button As Integer, level As Boolean) As Boolean

Configures the output of the specified pin, which can be either "off" (0) or "on" (1). If the button is not configured as an output, the resulting level is undefined. This method can also be used to configure LED output behavior on BP200/B900 button boards; see the [BP200/BP900 Setup](#) section below for more details.

SetOutputValue(offset As Integer, bit-mask As Integer) As Boolean

Configures a button on a BP200/BP900 button board. This method can only be used when the *roControlPort* object is instantiated with the `Touchboard-<n>-LED-SETUP` or `Touchboard-<n>-LED` parameter. See the [BP200/BP900 Setup](#) section below for more details.

SetOutputValues(values As roAssociativeArray) As Boolean

Configures buttons on a BP200/BP900 button board. This method can only be used when the *roControlPort* object is instantiated with the `Touchboard-<n>-LED-SETUP` or `Touchboard-<n>-LED` parameter. See the [BP200/BP900 Setup](#) section below for more details.

GetProperties() As roAssociativeArray

Returns an associative array of values related to the attached BP200/BP900 button board, including hardware, header, and revision. This method can only be used with an *roControlPort* instantiated with the `Touchboard-<n>-GPIO` parameter.

SetPulseParams(parameters As roAssociativeArray) As Boolean

Specifies a period of time, as well as the time slices within that period, for pulsing GPIO LEDs. These properties are applied to all GPIO outputs. This method is passed an associative array with the following parameters:

- `milliseconds`: An integer specifying the time period (in ms) for pulsing
- `slices`: An integer specifying the number of divisions within the milliseconds time period: For example, a 500ms time period with slices: 2 is divided into two 250ms slices.

SetPulse(button As Integer, bit-field As Integer) As Boolean

Sets the off/on bit field for a particular GPIO. Use the `slices` parameter of the `SetPulseParams()` method to determine the number of bits in the bit field. For example, specifying `milliseconds:500`, `slices:2`, and a bit field of 10 will cause the button to turn on every other 250 millisecond period.

RemovePulse(button As Integer) As Boolean

Removes the specified GPIO from the set affected by the pulse.

`ifMessagePort`

SetPort(port As Object)

Posts messages of type *roControlUp* and *roControlDown* to the attached message port.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifIdentity`

GetIdentity() As Integer

Returns an identity value that can be used to associate *roControlUp* and *roControlDown* events with this control port.

Note

The *ifIdentity* interface has been deprecated. We recommend using the *ifUserData* interface instead.

This example script applies timed pulses to a set of GPIOs:

```
' set up button 2 and 3 to flash at 2Hz (i.e. on & off twice in a second) in an alternating '
fashion.

gpioPort = CreateObject("roControlPort", "BrightSign")

gpioPort.EnableOutput(2)
gpioPort.SetOutputState(2, true)

gpioPort.EnableOutput(3)
gpioPort.SetOutputState(3, true)

' set up pulse to have two time slices of 250ms each.
gpioPort.SetPulseParams({ milliseconds: 500, slices: 2 })

' button 2 will have slice 1 on and slice 2 off.
gpioPort.SetPulse(2, &h01)

' button 3 will have the reverse of button 2.
gpioPort.SetPulse(3, &h02)

' wait for a bit.
sleep(10000)

' stop pulsing on button 2.
gpioPort.RemovePulse(2)
```

This example script enables various alternate functions on the GPIO:

```
c = CreateObject("roControlPort", "BrightSign")

'Enable serial port 1 on the GPIO.

c.EnableAlternateFunction(0, "serial1")
c.EnableAlternateFunction(6, "serial1")

s1 = CreateObject("roSerialPort", 1, 115200)
s1.SendLine("This is serial port 1")

mp = CreateObject("roMessagePort")
s1.SetLineEventPort(mp)
? wait(10000, mp)

'Switch serial port 0 from the 3.5mm serial port to the GPIO.
'[Note: it is advised use telnet/ssh or a script when testing this]

c.EnableAlternateFunction(4, "serial0")
c.EnableAlternateFunction(5, "serial0")

s = CreateObject("roSerialPort", 0, 115200)
s.SendLine("Hello on the console?")

mp = CreateObject("roMessagePort")
s.SetLineEventPort(mp)
? wait(10000, mp)

'Restore normal operation on serial port 0.
```

```

c.EnableAlternateFunction(4, "gpio")
c.EnableAlternateFunction(5, "gpio")

'Enable IR input on the GPIO.

c = CreateObject("roControlPort", "brightsign")
? c.EnableAlternateFunction(1, "irin1")

nexus_encodings = [ "NEC", "NEC32" ]
ir_gpio = CreateObject("roIRReceiver", { source: "GPIO", encodings: nexus_encodings })

mp = CreateObject("roMessagePort")
ir_gpio.SetPort(mp)

m = wait(10000, mp)

'Enable IR output on the GPIO (HDx23, HO523 only--the XT/XD models have a dedicated 3.5mm IR
socket)

c.EnableAlternateFunction(2, "irout")

ir = CreateObject("roIRTransmitter", { destination: "GPIO" } )
ir.Send("NEC32", &H12345)

```

BP200/BP900 Setup

To send a configuration to the BP200/BP900 button board, instantiate *roControlPort* with the `Touchboard-<n>-LED-SETUP` parameter and call the `SetOutputValue()` method. This method accepts two integers: the first integer specifies one of three command types (offsets); the second integer is a bit field consisting of 32 bits.

- **Offset 0:** Configures the button board using a bit field that is split into four bytes of eight bits each. Each byte is a separate part of the configuration. In the script, these bytes need to be listed from right to left in hex value (i.e. Byte 1 + Byte 2 + Byte 3 + Byte 4).
 - **Byte 1:** Specifies the configuration type for the button board. Currently, the only configuration type is for LED output, which is specified with the value `&hA0`.
 - **Byte 2:** The button number(s) that will be configured. Buttons are numbered beginning from 1. The value is set to 0 (`&h00`) if this command is not required.
 - **Byte 3:** The LED bit-field configuration. This value specifies how many on/off bits should be used (up to 32 bits) when `SetOutputValue()` is called on a `Touchboard-<n>-LED` instance (see the **BP200/BP900 LED Output** section below for details). Set the value to 0 (`&h00`) if this command is not required (the bit field will be set to eight bits by default).
 - **Byte 4:** This value is currently always set to 0 (`&h00`).
- **Offset 1:** Disables buttons on the button board according to values in the bit field. Each button is disabled individually by setting bits 0-10: For example, passing the hex value `&h00000008` will disable button 4 only.
- **Offset 2:** Disables LEDs on the button board according to values in the bit field. Each LED is disabled individually by setting bits 0-10: For example, passing the hex value `&h00000080` will disable the LED on button 8 only.

Note

Disabling a button LED will not automatically disable the button itself (and vice-versa). To disable both the button and the LED, make separate `SetOutputValue()` calls for Offset 1 and Offset 2.

BP200/BP900 LED Output

To control the behavior of individual button LEDs, instantiate *roControlPort* with the `Touchboard-<n>-LED` parameter, then pass per-LED bit fields to the `SetOutputValue()` method. This method accepts two integers: the first integer specifies the button number (0-11), while the second integer uses a bit field to specify the on/off behavior of the button LED. The size of the bit field (up to 32 bits) is determined with the Offset 0 – Byte 3 value described in the section above.

Each bit specifies the on/off behavior of a single cycle, and the BP200/BP900 button boards run at approximately 11Hz. For example, if you want an LED to cycle on every other second, you would set the Offset 0 – Byte 3 value to `&h16` (22 bits) and the bit field itself to `&h3FF800` (00000000000111111111).

This example script sets a BP900 to “twinkle” by turning off each button LED at a different point in the cycle:

```

led=CreateObject("roControlPort", "TouchBoard-0-LED")
led_setup=CreateObject("roControlPort", "TouchBoard-0-LED-SETUP")
led_setup.SetOutputValue(0, &h000B00A0)
led.SetOutputValue(0, &h07fe)
led.SetOutputValue(1, &h07fd)
led.SetOutputValue(2, &h07fb)
led.SetOutputValue(3, &h07f7)
led.SetOutputValue(4, &h07ef)
led.SetOutputValue(5, &h07df)
led.SetOutputValue(6, &h07bf)
led.SetOutputValue(7, &h077f)
led.SetOutputValue(8, &h06ff)
led.SetOutputValue(9, &h05ff)
led.SetOutputValue(10, &h03ff)

```

roControlUp, roControlDown

ON THIS PAGE

- [ifInt](#)
 - [GetInt\(\) As Integer](#)
 - [SetInt\(value As Integer\) As Void](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\) As Void](#)
 - [GetUserData\(\) As Object](#)
- [ifSourceIdentity](#)
 - [GetSourceIdentity\(\) As Integer](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

These objects are posted by an *roControlPort* instance to the configured message port when inputs change state. An *roControlDown* event is posted when the input level goes from high to low. An *roControlUp* event is posted when the input level goes from low to high.

ifInt

GetInt() As Integer

Retrieves the pin number associated with the event.

SetInt(value As Integer) As Void

Sets the value of the event.

ifUserData

SetUserData(user_data As Object) As Void

Sets the user data.

GetUserData() As Object

Returns the user data that has previously been set via *SetUserData()* (on either the event or source object). It will return *Invalid* if no data has been set.

`ifSourceIdentity`

GetSourceIdentity() As Integer

Retrieves the identity value that can be used to associate events with the source *roControlPort* instance.

Note

The *ifSourceIdentity* interface has been deprecated. We recommend using the *ifUserData* interface instead.

roGpioButton

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object contains the input ID from *roGpioControlPort* instances:

`ifInt`

GetInt() As Integer

Returns the input ID of the event.

SetInt(value As Integer) As Void

Sets the input ID of the event.

roGpioControlPort

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

ON THIS PAGE

- [ifMessagePort](#)
 - [SetPort\(obj As Object\) As Void](#)
- [ifGpioControlPort](#)
 - [IsInputActive\(input_id As Integer\) As Boolean](#)
 - [GetWholeState\(\) As Integer](#)
 - [SetOutputState\(output_id As Integer, onState As Boolean\) As Void](#)
 - [SetWholeState\(on_state As Integer\) As Void](#)
 - [EnableInput\(input_id As Integer\) As Boolean](#)
 - [EnableOutput\(output_id As Integer\) As Boolean](#)

Important

New scripts should use *roControlPort* instead of *roGpioControlPort*.

This object is used to control and wait for events on the BrightSign generic GPIO control port. Typically, LEDs or buttons are connected to the GPIO port. Turning on a GPIO output changes the voltage on the GPIO port to 3.3V. Turning off a GPIO output changes the voltage on the GPIO port to 0V.

The GPIO ports are bidirectional and must be programmed as either inputs or outputs. The IDs range from 0–7. The `SetWholeState()` method will overwrite any prior output settings. The `SetOutputState()` takes an output ID (1, 2, or 6, for example). The `SetWholeState()` method takes a mask (for example, `SetWholeState(SetWholeState(2^1 + 2^2))` will set IDs 1 and 2).

`ifMessagePort`

SetPort(obj As Object) As Void

`ifGpioControlPort`

IsInputActive(input_id As Integer) As Boolean

GetWholeState() As Integer

SetOutputState(output_id As Integer, onState As Boolean) As Void

SetWholeState(on_state As Integer) As Void

EnableInput(input_id As Integer) As Boolean

EnableOutput(output_id As Integer) As Boolean

roIRReceiver

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object supports receiving arbitrary Infrared remote control codes using the NEC and RC5 protocols.

Object Creation: The `roIRReceiver` object is created with an associative array.

```
CreateObject("roIRReceiver", config As roAssociativeArray)
```

The associative array can contain the following parameters:

- `source`: A string value indicating the source of the input.
 - "IR-in": The 3.5mm IR input/output connector (available on 4Kx42 and XDx32 models)

- "GPIO": Pin 1 of the GPIO connector
- "Iguana": The [Iguanaworks](#) IR transceiver. This source can support both NEC and RC5 encodings simultaneously.
- `encodings`: An array indicating the required encodings.
 - "NEC"
 - "RC5" (supported on the Iguanaworks IR transceiver only)

NEC codes are expressed in 24 bits:

- Bits 0-7: Button code
- Bits 8-23: Manufacturer code

Note

If the manufacturer code is zero, then the code is considered to be intended for the Roku SoundBridge remote control.

The `roIRReceiver` object can generate the following events:

- `roIRDownEvent`. Generates when a button is pressed.
- `roIRRepeatEvent`. Generates when a button repeats.
- `roIRUpEvent` (Iguanaworks IR transceiver only): Generates when a button is released.

`ifUserData`

`SetUserData(user_data As Object)`

Sets the user data that will be returned when events are raised.

`GetUserData() As Object`

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifMessagePort`

`SetPort(port As roMessagePort)`

Specifies the port that will receive events generated by the `roIRReceiver` instance.

`roIRDownEvent`, `roIRRepeatEvent`, `roIRUpEvent`

ON THIS PAGE

- [ifInt](#)
 - [GetCode\(\) As Integer](#)
 - [SetCode\(a As Integer\)](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifReceivedEvent](#)
 - [GetEncoding\(\) As String](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

An IR event object is generated when an IR button input (button press, button repeat, button release) is received by the `roIRReceiver` object. Use these objects to retrieve the message body of the IR input.

Note

The *roIRUpEvent* object is generated with the Iguanaworks IR transceiver only.

ifInt

GetCode() As Integer

Returns the IR code received by the *roIRReceiver* instance.

SetCode(a As Integer)

Overrides the IR code received by the *roIRReceiver* instance, replacing it with the specified binary code.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned by *GetUserData()*.

GetUserData() As Object

Returns the user data that has previously been set via *SetUserData()*. It will return *Invalid* if no data has been set.

ifReceivedEvent

GetEncoding() As String

Returns the encodings setting of the *roIRReceiver* instance. This setting can be one of the following strings:

- "NEC"
- "RC5" (supported on the Iguanaworks IR transceiver only)

roIRTransmitter

ON THIS PAGE

- [ifMessagePort](#)
 - [SetPort\(message_port_object As Object\) As Void](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifIRTransmitter](#)
 - [GetFailureReason\(\) As String](#)
 - [Send\(protocol As String, code As Dynamic\) As Boolean](#)
 - [AsyncSend\(protocol As String, code As Dynamic\) As Boolean](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object supports sending arbitrary remote control Infrared remote control codes using the NEC, RC5, or PHC (Pronto Hex Controls) protocols.

Object Creation: The *roIRTransmitter* is created with an associate array.

```
CreateObject("roIRTransmitter", config as roAssociativeArray)
```

The associative array can contain the following parameter:

- **destination**: A string value indicating the connector that will be used to output the signal.
 - "IR-out": The 3.5mm IR output connector (available on XDx30 models) or 3.5mm IR input/output connector (available on 4Kx42 and XDx32 models)
 - "Iguana": The [Iguanaworks](#) IR transceiver

Note

System software will not prevent you from generating both an *roIRTransmitter* instance set to "IR-out" and an *roIRReceiver* instance set to "IR-in" (i.e. configuring the 3.5mm IR connector for input and output at the same time). However, input/output performance will not be reliable.

`ifMessagePort`

SetPort(message_port_object As Object) As Void

Posts event messages to the attached message port.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

`ifIRTransmitter`

GetFailureReason() As String

Send(protocol As String, code As Dynamic) As Boolean

Sends the specified code using the output destination set during object creation. The system currently supports two IR transmission protocols: "NEC" and "PHC" ([Pronto Hex Code](#)). This method returns `True` if the code was successfully transmitted, but there is no way to determine from `BrightScript` if the controlled device actually received it.

AsyncSend(protocol As String, code As Dynamic) As Boolean

Sends the specified code and generates an *roIRTransmitCompleteEvent* object upon completion. The system currently supports two IR transmission protocols: "NEC" and "PHC" ([Pronto Hex Code](#)). This method is only supported using the IR-out destination.

roIRTransmitCompleteEvent

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This event object is generated by the `roIRTransmitter.ASyncSend()` method. It does not return any information other than user data.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned by `GetUserData()`.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

roIRRemote

ON THIS PAGE

- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)
- [ifIRRemote](#)
 - [Send\(protocol as String, code as Dynamic\) As Boolean](#)
- [Pronto Hex Format](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object supports receiving and transmitting arbitrary Infrared remote control codes using the NEC protocol. You can also use this object to send PHC (Pronto Hex Code) commands. The best way to determine the required send values is to capture the codes received by `roIRRemote` when the remote buttons of the device are pressed and then send the same codes.

Important

The `roIRRemote` object cannot be used to receive input over the 3.5mm IR port on the 4Kx42 and XDx32 series—use the `roIRReceiver` object instead.

NEC codes are expressed in 24 bits:

- Bits 0-7: Button code
- Bits 8-23: Manufacturer code

Note

If the manufacturer code is zero, then the code is considered to be intended for the Roku SoundBridge remote control.

`ifMessagePort`

SetPort(port As roMessagePort)

Posts messages of type `roIRRemotePress` to the attached message port.

`ifIRRemote`

Send(protocol as String, code as Dynamic) As Boolean

Sends the specified code using the IR blaster. The system currently supports two IR transmission protocols: "NEC" and "PHC" (Pronto Hex Code). This method returns True if the code was successfully transmitted, but there is no way to determine from BrightScript if the controlled device actually received it.

Pronto Hex Format

Raw captures of Pronto Hex commands will likely not work with the inbuilt IR blaster, though they should work with [Iguanaworks](#) IR transceivers. This is a result of the trailing *off* periods, which are too long to be encoded properly. Changing the *off* periods to all zeros ("0000") will fix this issue.

The following example sends an "ON" command to a Panasonic television using a single string of Pronto Hex Code. You can also provide Pronto Hex Code as an *roArray* of hex values, which results in less work for the script engine.

```
ir = CreateObject("roIRRemote")

pronto_hex_Panasonic_on_str = " 0000 0071 0000 0032 0080 003F 0010 0010 0010 0030 0010 0010
0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010
0010 0010 0010 0030 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010
0010 0010 0010 0010 0010 0030 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010 0010
0010 0010 0010 0010 0010 0010 0010 0030 0010 0030 0010 0030 0010 0030 0010 0030 0010 0030 0010
0010 0000"

ir.Send("PHC", pronto_hex_lg_on_str)
```

roIRRemotePress

ON THIS PAGE

- [ifInt](#)
 - [GetInt\(\) As Integer](#)
 - [SetInt\(value As Integer\) As Void](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

Messages of the type *roIRRemotePress* are generated upon key presses from a Roku Soundbridge remote.

ifInt

GetInt() As Integer

Returns the event ID.

SetInt(value As Integer) As Void

Sets the value of the event.

For the Roku SoundBridge remote control, the Integer returned can have one of the following values:

Integer	Command	Integer	Command
0	West	8	Search
1	East	9	Play
2	North	10	Next
3	South	11	Previous

4	Select	12	Pause
5	Exit	13	Add
6	Power	14	Shuffle
7	Menu	15	Repeat
8	Search	16	Volume up
9	Play	17	Volume down
10	Next	18	Brightness

roKeyboard

ON THIS PAGE

- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifKeyboardConfig](#)
 - [IsPresent\(\) As Boolean](#)
 - [SetLayout\(layout As String\) As Boolean](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is used to wait for events from a USB keyboard. It can also be used to configure the localization of the keyboard.

Object Creation: The *roKeyboard* object is created with no parameters.

```
CreateObject("roKeyboard")
```

ifMessagePort

SetPort(port As roMessagePort)

Posts messages of type *roKeyboardPress* to the attached message port.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via *SetUserData()*. It will return *Invalid* if no data has been set.

ifKeyboardConfig

IsPresent() As Boolean

Returns *True* if a USB keyboard is connected to the player. This method counts a connected device as a keyboard if it reports having the following keys: "A", "Z", "0", "9", ".", and Enter.

setLayout(layout As String) As Boolean

Specifies the localized layout for the attached USB keyboard. This setting takes effect immediately and persists in the registry after a reboot. The following table lists valid keymap parameters (players are set to "us" by default):

af	Afghanistan	es	Spain	kh	Cambodia	pk	Pakistan
al	Albania	et	Ethiopia	kr	Korea, Republic of	pl	Poland
am	Armenia	fi	Finland	kz	Kazakhstan	pt	Portugal
at	Austria	fo	Faroe Islands	la	Laos	ro	Romania
az	Azerbaijan	fr	France	lk	Sri Lanka	rs	Serbia
ba	Bosnia and Herzegovina	gb	United Kingdom	lt	Lithuania	ru	Russia
bd	Bangladesh	ge	Georgia	lv	Latvia	se	Sweden
be	Belgium	gh	Ghana	ma	Morocco	si	Slovenia
bg	Bulgaria	gn	Guinea	md	Moldova	sk	Slovakia
br	Brazil	gr	Greece	me	Montenegro	sn	Senegal
bt	Bhutan	hr	Croatia	mk	Macedonia	sy	Syria
bw	Botswana	hu	Hungary	ml	Mali	th	Thailand
by	Belarus	ie	Ireland	mm	Myanmar	tj	Tajikistan
ca	Canada	il	Israel	mn	Mongolia	tm	Turkmenistan
cd	Congo (DRC)	in	India	mt	Macao	tr	Turkey
ch	Switzerland	iq	Iraq	mv	Maldives	tw	Taiwan
cm	Cameroon	ir	Iran	ng	Nigeria	tz	Tanzania
cn	China	is	Iceland	nl	Netherlands	ua	Ukraine
cz	Czech Republic	it	Italy	no	Norway	us*	United States
de	Germany	jp	Japan	np	Nepal	uz	Uzbekistan
dk	Denmark	ke	Kenya	pc	Pitcairn	vn	Vietnam
ee	Estonia	kg	Kyrgyzstan	ph	Philippines	za	South Africa

*The default setting.

roKeyboardPress

ON THIS PAGE

- [ifInt](#)
 - [GetInt\(\) As Integer](#)
 - [SetInt\(a As Integer\)](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This is an event object resulting from the user pressing a key on a USB keyboard. The returned integer value is equivalent to the ASCII code of the key that was pressed.

`ifInt`

GetInt() As Integer

Returns the ASCII value of the key press.

SetInt(a As Integer)

Sets the value returned by the `GetInt()` method.

The returned `roInt32` can have one of the following values:

Letter Keys		Number Keys	Function Keys	Misc Keys	Special Keys		
A - 97	R - 114	0 - 48	F1 - 32826	Del - 127	"-" 45	:	58
B - 98	S - 115	1 - 49	F2 - 32827	Backspace - 8	"=" 61	"	34
C - 99	T - 116	2 - 50	F3 - 32828	Tab - 9	\ 92	<	60
D - 100	U - 117	3 - 51	F4 - 32829	Enter - 13	` 96	>	62
E - 101	V - 118	4 - 52	F5 - 32830	Print Scrn - 32838	[91	?	63
F - 102	W - 119	5 - 53	F6 - 32831	Scrl Lock - 32839] 93	!	33
G - 103	X - 120	6 - 54	F7 - 32832	Pause/Brk - 32840	; 59	@	64
H - 104	Y - 121	7 - 55	F8 - 32833	INS - 32841	"'" 39	#	35
I - 105	Z - 122	8 - 56	F9 - 32834	Home - 32842	, 44	\$	36
J - 106		9 - 57	F11 - 32836	Page Up - 32843	.	%	37
K - 107			F12 - 32837	Page Down - 32846	/ 47	^	94
L - 108				End - 32845	_ 95	&	38
M - 109				Caps - 32811	"+" 43	*	42
N - 110				Left Arrow - 32848	124	(40
O - 111				Right Arrow - 32847	~ 126)	41
P - 112				Up Arrow - 32850	{ 123		
Q - 113				Down Arrow - 32849	} 125		

roSequenceMatcher

ON THIS PAGE

- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)
- [ifSequenceMatcher](#)
 - [SetPort\(a As Object\)](#)
 - [Add\(pattern As Object, user_data As Object\) As Boolean](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is used to send `roSequenceMatchEvent` events when the specified byte sequence patterns are matched. Once a pattern has been matched and the event has been posted, all contributing bytes are discarded. As a result, if one pattern is a prefix of another pattern, then the second, longer pattern will never be matched by the object.

This object provides both a standard interface and an overloaded interface for sending events to a message port.

`ifMessagePort`

`SetPort(port As roMessagePort)`

Posts messages of type *roSequenceMatchEvent* to the attached message port.

ifSequenceMatcher

SetPort(a As Object)

Specifies the message port where *roSequenceMatchEvent* objects will be posted.

Add(pattern As Object, user_data As Object) As Boolean

Adds a pattern to be matched by the *roSequenceMatcher* object instance. The pattern should be specified as an object that is convertible to a byte sequence (e.g. *roByteArray*, *roString*). For the user data, pass the object that should be returned if the specified pattern is matched.

Example

```
Function FromHex(hex as String) as Object
    bytes = CreateObject("roByteArray")
    bytes.FromHexString(hex)
    return bytes
End Function

Sub Main()
    serial = CreateObject("roSerialPort", 1, 115200)
    mp = CreateObject("roMessagePort")

    button1_seq = FromHex("080a01040001e000")
    button2_seq = FromHex("080e01040001e000")

    matcher = CreateObject("roSequenceMatcher")
    matcher.SetMessagePort(mp)
    matcher.Add(button1_seq, { name: "button1" })
    matcher.Add(button2_seq, { name: "button2" })
    matcher.Add("flibbet", { name: "flibbet" })
    matcher.Add("flobbet", { name: "flobbet" })

    if not serial.SetMatcher(matcher) then
        stop
    end if

    finished = false
    while not finished
        ev = mp.WaitMessage(10000)
        if ev = invalid then
            finished = true
        else if type(ev) = "roSequenceMatchEvent" then
            print "Got button: "; ev.GetUserData().name
        else
            print "Unexpected event: "; type(ev)
        end if
    end while
End Sub
```

roSequenceMatchEvent

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is generated whenever *roSequenceMatcher* matches a specified byte sequence pattern.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned by `GetUserData()`.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

roSerialPort

ON THIS PAGE

- [ifStreamSend](#)
 - [SetSendEol\(eol_sequence As String\) As Void](#)
 - [SendByte\(byte As Integer\) As Void](#)
 - [SendLine\(string As String\) As Void](#)
 - [SendBlock\(a As Dynamic\) As Void](#)
 - [Flush\(\)](#)
- [ifStreamReceive](#)
 - [SetLineEventPort\(port As Object\) As Void](#)
 - [SetByteEventPort\(port As Object\) As Void](#)
 - [SetReceiveEol\(eol_sequence As String\)](#)
- [ifSerialControl](#)
 - [SetBaudRate\(baud_rate As Integer\) As Boolean](#)
 - [SetMode\(mode As String\) As Boolean](#)
 - [SetEcho\(enable As Boolean\) As Boolean](#)
 - [SetEol\(a As String\)](#)
 - [SetFlowControl\(enable As Boolean\) As Boolean](#)
 - [SetInverted\(inverted As Boolean\) As Boolean](#)
 - [SendBreak\(duration_in_ms As Integer\) As Boolean](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object controls the serial port on the player, allowing you to transmit and receive serial data.

Object Creation: The *roSerialPort* object is created with two parameters.

```
CreateObject("roSerialPort", port As Dynamic, baud_rate As Integer)
```

- `port` As Integer: The port enumeration of the serial device:
 - Most standard serial devices enumerate on port 0.
 - To communicate with the serial port of an OPS display (i.e. with the HO523), use port 1.
 - To communicate with a USB-serial device (such as a GPS receiver), use port 2.
- `port` As String: If multiple USB-serial devices are connected to the player, the device can be specified with a raw value ("RAW:<raw_enumeration>") or a friendly name ("USB:<friendly_name>"). These values correspond to the <raw> and <fid> values returned by the *roDeviceInfo.GetUSBTTopology()* method.
- `baud_rate` As Integer: The baud rate for serial communication. The serial port supports the following baud rates: 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400.

Example

```
serial1 = CreateObject("roSerialPort", 0, 115200)
serial2 = CreateObject("roSerialPort", "RAW:4-1:1.0", 57600)
```

The *roSerialPort* object sends the following event types:

- *roStreamLineEvent*. The line event is generated whenever the end of line string set using *SetEol* is found and contains a string for the whole line. This object implements the *ifString* and *ifUserData* interfaces.
- *roStreamByteEvent*. The byte event is generated on every byte received. This object implements the *ifInt* and *ifUserData* interfaces.

ifStreamSend

SetSendEol(eol_sequence As String) As Void

Sets the EOL sequence when writing to the stream. The default value is CR (ASCII value 13). If you need to set this value to a non-printing character, use the `chr()` [global function](#).

SendByte(byte As Integer) As Void

Writes the specified byte to the stream.

SendLine(string As String) As Void

Writes the specified characters to the stream followed by the current EOL sequence.

SendBlock(a As Dynamic) As Void

Writes the specified characters to the stream. This method can support either a string or an *roByteArray*. If the block is a string, any null bytes will terminate the block.

Flush()

ifStreamReceive

SetLineEventPort(port As Object) As Void

SetByteEventPort(port As Object) As Void

SetReceiveEol(eol_sequence As String)

Sets the EOL sequence to detect when receiving the stream. The default value is CR (ASCII value 13). If you need to set this value to a non-printing character, use the `chr()` [global function](#).

SetMatcher(matcher As Object) As Boolean

Instructs the stream to use the specified matcher. This method returns True if successful. Pass Invalid to this method to stop using the specified matcher.

ifSerialControl

SetBaudRate(baud_rate As Integer) As Boolean

Sets the baud rate of the device. The supported baud rates are as follows: 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400.

SetMode(mode As String) As Boolean

Sets the serial mode in "8N1" syntax. The first character is the number of data bits. It can be either 5, 6, 7, or 8. The second number is the parity. It can be "N"one, "O"dd, or "E"ven. The third is the number of stop bits. It can be 1 or 2.

SetEcho(enable As Boolean) As Boolean

Enables or disables serial echo. It returns True on success and False on failure.

SetEol(a As String)***SetFlowControl(enable As Boolean) As Boolean***

Enables or disable RTS/CTS handshaking over the serial port. This feature is currently only available on 4Kx42, XDx32, and HDx22 models.

SetInverted(inverted As Boolean) As Boolean

Inverts the TX/RX signal levels on the serial port. This allows the player to communicate with devices that use -12V to 12V signaling. Inversion is supported on the DE9 and USB ports only. Passing True to the method enables inversion, whereas passing False disables it.

SendBreak(duration_in_ms As Integer) as Boolean

Sends a serial break or sets the serial break condition. This method returns True upon success and False upon failure.

- `duration_in_ms = -1`: Sends a continuous break.
- `duration_in_ms = 0`: Clears the break state.
- `duration_in_ms >= 100`: Sets the break condition for the specified period of milliseconds (note that this integer is only accurate to the tenth of a second).

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

This example script waits for a serial event and echoes the input received on the serial port to the shell:

```
serial = CreateObject("roSerialPort", 0, 9600)
p = CreateObject("roMessagePort")
serial.SetLineEventPort(p)

serial_only:
msg = Wait(0,p) ' Wait forever for a message.
if(type(msg) <> "roStreamLineEvent") goto serial_only 'Accept serial messages only.
serial.SendLine(msg) ' Echo the message back to serial.
```

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This section describes objects that interact with system software.

- [roDeviceCustomization](#)
- [roDeviceInfo](#)
- [roResourceManager](#)
- [roSystemLog](#)

roDeviceCustomization

ON THIS PAGE

- [ifFailureReason](#)
 - [GetFailureReason\(\) As String](#)
- [ifDeviceCustomization](#)
 - [WriteSplashScreen\(filename As String\) As Boolean](#)
 - [FactoryReset\(confirm As String\) As Boolean](#)
 - [EncryptStorage\(device As String, params As roAssociativeArray\) As Boolean](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object provides miscellaneous device configuration and customization methods.

`ifFailureReason`

GetFailureReason() As String

Returns helpful information if one of the *ifDeviceCustomization* methods fail.

`ifDeviceCustomization`

WriteSplashScreen(filename As String) As Boolean

Removes the default splash screen (or a previously set splash screen) and replaces it with the specified image file. The image file must use a [supported format](#). This method returns True upon success and False upon failure.

FactoryReset(confirm As String) As Boolean

Applies a factory reset to the player. This method must be passed the string "confirm" to work; otherwise, it will return False and do nothing. If successful, this method will reboot without a return value. The following steps will be carried out during a factory reset:

1. All files are wiped from the `BOOT:` drive (including custom splash screens and autorun scripts).
2. All values are wiped from the registry.
3. The RTC is reset (if the player has an RTC).
4. The `FLASH:` drive is wiped.

EncryptStorage(device As String, params As roAssociativeArray) As Boolean

Encrypts the contents of a storage device using an encryption key stored in the private section of the player registry. This prevents other devices from reading the names or contents of files on the storage device (though file metadata will still be readable). The storage device can still be mounted on other devices, which can list, delete, and copy files (even though they cannot read them).

Use the `device` parameter to specify which storage drive to encrypt (see [here](#) for a list of valid drive names). Encryption parameters are passed as an associative array, which can contain the following:

- `method` As String:(Required) The encryption method, which can be one of the following:
 - "none": Encryption is disabled and any encryption keys for the storage device are deleted from the registry.
 - "passphrase": The storage device is encrypted using an un-obfuscated passphrase.
 - "obfuscated_passphrase": The storage device is encrypted using an obfuscated passphrase. Contact support@brightsign.biz to learn more about generating a key for obfuscation and storing it on the player.
 - "brightsign_key": The storage device is encrypted using an encryption key that is common to all BrightSign players, but is not shared with outside parties.
 - "generate_key": The storage device is encrypted using an encryption key that is generated on-demand using a secure random generator. The private registry of the player contains the only copy of the key; if the value in the registry is erased or becomes corrupted, there will be no way to recover data from that storage device.

Important

The private registry does not distinguish between different physical volumes of the same drive. For example, if you encrypt two SD cards, the key for the first SD card will be overwritten when the second SD card is encrypted; in the case of a randomly generated key, the data from the first SD card will be irretrievable.

- `passphrase` As String: The passphrase for encrypting the storage device. This parameter must be included if the method is specified as "passphrase" or "obfuscated_passphrase".
- `format` As Boolean: Setting this parameter to `true` will cause the specified storage device to be reformatted—all existing files will be deleted (but are not guaranteed to be securely erased) before the device is encrypted; however, if the device cannot be reformatted, the operation will fail. If this parameter is set to `false`, the storage device will be mounted as if it has been previously encrypted using the supplied key. If the storage device hasn't been previously encrypted, it will be encrypted while leaving any preexisting files unencrypted. If the storage device has been encrypted using a different key, the operation will fail.

Note

If the `format` parameter is not specified, the storage device will be mounted as if it has been previously encrypted using the supplied key. If the storage device hasn't been previously encrypted, the operation will fail.

roDeviceInfo

ON THIS PAGE

- [ifDeviceInfo](#)
 - [GetModel\(\) As String](#)
 - [GetVersion\(\) As String](#)
 - [GetVersionNumber\(\) As Integer](#)
 - [FirmwaresAtLeast\(version As String\) As Boolean](#)
 - [GetBootVersion\(\) As String](#)
 - [GetBootVersionNumber\(\) As Integer](#)
 - [BootFirmwaresAtLeast\(version As String\) As Boolean](#)
 - [GetDeviceUptime\(\) As Integer](#)
 - [GetDeviceLifetime\(\) As Integer](#)
 - [GetLoadStatistics\(parameters As roAssociativeArray\) As String](#)
 - [GetUSBTopology\(return As roAssociativeArray\) As Dynamic](#)
 - [GetDeviceUniqueId\(\) As String](#)
 - [GetFamily\(\) As String](#)
 - [HasFeature\(feature As String\) As Boolean](#)

▼ Firmware Version 7.0

- [Version 7.0](#)

- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object provides information about the device hardware, firmware, and features.

`ifDeviceInfo`

GetModel() As String

Returns the model name for the BrightSign device running the script as a string (for example, "HD1020" or "XD230").

GetVersion() As String

Returns the version number of BrightSign firmware running on the device (for example, "4.0.13").

GetVersionNumber() As Integer

Returns the three most significant version numbers of the BrightSign firmware running on the device in comparable numeric form: `major*65536 + minor*256 + build`

Important

The return value for the `GetVersionNumber()` method does not include any additional version numbers after the first three: For example, firmware version 4.0.13.4 will be returned as 262157 (as if it were "4.0.13"). For this reason, we recommend using the `GetVersion()` or `FirmwareIsAtLeast()` methods to determine the current firmware version.

FirmwareIsAtLeast(version As String) As Boolean

Returns True if the BrightSign firmware version on the device is less than or equal to the version number represented by the passed string (e.g. "4.0.13").

GetBootVersion() As String

Returns the version number of the BrightSign boot firmware, also known as "safe mode", as a string (for example, "1.0.4").

GetBootVersionNumber() As Integer

Returns the three most significant version numbers of the BrightSign boot firmware, also known as "safe mode," in comparable numeric form: `major*65536 + minor*256 + build`.

Important

The return value for the `GetBootVersionNumber()` method does not include any additional version numbers after the first three: For example, firmware version 1.0.4.1 will be returned as 65540 (as if it were "1.0.4"). For this reason, we recommend using the `GetBootVersion()` or `BootFirmwareIsAtLeast()` methods to determine the current boot firmware version.

BootFirmwareIsAtLeast(version As String) As Boolean

Returns True if the BrightSign boot firmware version on the device is less than or equal to the version number represented by the passed string (e.g. "4.4.22").

GetDeviceUptime() As Integer

Returns the number of seconds that the device has been running since the last power cycle or reboot.

GetDeviceLifetime() As Integer

GetLoadStatistics(parameters As roAssociativeArray) As String

Provides current performance information related to the Linux kernel. This method accepts an associative array with a single key/value pair formatted as `item:<parameter>`; it will then return a string containing information associated with that parameter. The following are recognized parameters:

- "loadavg": Provides information about system performance. The first three columns measure CPU and I/O utilization over the past 1, 5, and 10 minutes, respectively. The fourth column displays the number of currently running processes and the total number of processes. The last column displays the ID of the most recently used process.
- "meminfo": Displays physical and swap memory usage.
- "slabinfo": Provides information about memory usage at the slab level.
- "stat": Provides overall statistics about the system (e.g. the number of page faults since the system booted).
- "vmstat": Displays detailed virtual memory statistics from the kernel.
- "zoneinfo": Provides overall statistics about the system, broken down by system Node.
- "interrupts": Displays which interrupts are in use and how many of each type there have been.
- "version": Provides the kernel version.

Example

```
stat = CreateObject("roDeviceInfo")
print stat.GetLoadStatistics({item: "interrupts"})
```

GetUSBTopology(return As roAssociativeArray) As Dynamic

Returns the USB topology of the player, including the port enumeration, PID, and VID of each USB device. Use this method to determine whether certain USB devices are connected to certain ports. The passed associative array determines the return value:

- {}: Returns the USB topology as a string list.
- {xml: true}: Returns the USB topology as an XML-formatted string.
- {array: true}: Returns an array of associative arrays. Each associative array contains information about a single USB port.

String list

```
BrightScript Debugger> di = CreateObject("roDeviceInfo")
BrightScript Debugger> topo = di.GetUSBTopology({})
BrightScript Debugger> print topo
>|usb1,4,1d6b,0002||usb2,4,1d6b,0002||usb3,4,1d6b,0002||usb4,4,1d6b,0002||usb5,4,1d6b,0001||usb6,4,1d6b,0001||usb7,4,1d6b,0001||usb8,4,1d6b,0001||5-1:1.0,1,084f,ee03|<
```

XML string

```
BrightScript Debugger> di = CreateObject("roDeviceInfo")
BrightScript Debugger> topo = di.GetUSBTopology({xml: true})
BrightScript Debugger> print topo
<topology>
  <device>
    <raw>8-1:1.0</raw>
    <fid>C.0</fid>
    <category>HID</category>
    <vid>084f</vid>
    <pid>ee03</pid>
    <type>BP900</type>
  </device>
  <devices raw="2-1.4" category="HUB" vid="05e3" pid="0610" >
    <device>
      <raw>2-1.4.1:1.0</raw>
      <fid>A/4.0</fid>
      <category>AUDIO</category>
      <vid>05a7</vid>
      <pid>40fa</pid>
    </device>
    <device>
      <raw>2-1.4.1:1.2</raw>
      <fid>A/4.2</fid>
      <category>HID</category>
      <vid>05a7</vid>
      <pid>40fa</pid>
```

```
</device>
```

```
...
```

Array

```
BrightScript Debugger> di = CreateObject("roDeviceInfo")
BrightScript Debugger> topo = di.GetUSBTTopology({array: true})
BrightScript Debugger> print topo[0]
vid: 05a7
pid: 40fa
fid: A.0
category: AUDIO
raw: 2-1.4.1:1.0
```

Ports that have USB devices connected to them will include a `fid` (Friendly ID) value (only array and XML-string returns include this value). Use the `fid` to determine which physical port on the player the USB device is connected to. The following table matches `fid` values to the ports on different model lines:

Model Family	Friendly ID (fid)	Location
LS423	A	Front (USB-C)
HDx23	A	Front (USB-A)
HS123	A	N/A (M.2 connector)
HO523	A	Upper Front (USB-A)
	B	Lower Front (USB-A)
	C	Back (JAE TX-25)
XDx33/XDx34	A	Front (USB-C)
	B	Back (USB-A)
XTx43/XTx44	A	Back (USB-A)
	B	Front (USB-C)

A `fid` value of `<port>.<int>` (e.g. "B.0") indicates that the USB device is directly connected to the player, while a value of `<port>/<int>.<int>` (e.g. "A/4.2") indicates a device connected through a USB hub. The last integer specifies the endpoint: For example, if a USB device presents as both an HID and audio device, they will be listed as separate devices with different endpoint numbers. Also note that there can be more than one layer of port numbers if multiple USB hubs are connected together (e.g. "A/4.0.0").

GetDeviceUniqueld() As String

Returns an identifier that, if not an empty string, is unique to the unit running the script.

GetFamily() As String

Returns a single string that indicates the family to which the device belongs. A device family is a set of models that are all capable of running the same firmware.

HasFeature(feature As String) As Boolean

Returns True if the player feature, which is passed as a case-insensitive string parameter, is present on the current device and firmware. The possible features that can be queried from the script are listed below:

Important

If you pass a parameter other than one of those listed below, it may return False even if the feature is available on the hardware and firmware.

- "5v serial": A 5V serial port

- "audio1": The first audio output
- "audio2": A second audio output
- "audio3": A third audio output
- "brightscript1": BrightScript Version 1
- "brightscript2": BrightScript Version 2
- "component video": A component video output
- "ethernet": An Ethernet interface
- "gpio connector": A DA15 or Phoenix-style GPIO port
- "hdmi": An HDMI output
- "hdmi input": An HDMI input
- "hevc_decode": An H.265 video decoder
- "media_decryption": The ability to decrypt AES-encrypted media, including video, image, and audio files.
- "nand storage": NAND storage for the boot loader and firmware
- "networking": Any form of networking capability. A False return may indicate that no network is currently available.
- "reset button": A reset button
- "registry": On-board persistent storage
- "rtc": A real-time clock (RTC)
- "sd": SD- or SDHC-compatible storage
- "sdhc": SDHC-compatible storage only
- "serial port 0": The first serial port
- "serial port 1": A second serial port
- "serial port 2": A third serial port
- "svc button": A service ("SVC") button. Passing the legacy term "gpio12 button" will yield the same result.
- "usb": One or more USB interfaces
- "vga": A VGA output
- "video_encoder": A video encoder/transcoder

Example

```
di = CreateObject("roDeviceInfo")
print di.GetModel()
print di.GetVersion(), di.GetVersionNumber()
print di.GetBootVersion(), di.GetBootVersionNumber()
print di.GetDeviceUptime(), di.GetDeviceBootCount()
```

On a particular system, this will generate the following:

```
HD1010
3.2.41      197161
3.2.28      197148
14353       3129
```

roResourceManager

ON THIS PAGE

- [ifResourceManager](#)
 - [SetLanguage\(language_identifier As String\) As Boolean](#)
 - [GetResource\(resource_identifier As String\) As String](#)
 - [GetFailureReason\(\) As String](#)
 - [GetLanguage\(\) As String](#)
- [Usage](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

The *roResourceManager* object is used to manage strings in multiple languages.

Object creation: The *roResourceManager* object is created with a single filename parameter that specifies the name of the file that contains all of the localized resource strings required by the user. This file must be in UTF-8 format.

```
CreateObject("roResourceManager", filename As String)
```

ifResourceManager

SetLanguage(language_identifier As String) As Boolean

Instructs the *roResourceManager* object to use the specified language. False is returned if there are no resources associated with the specified language.

GetResource(resource_identifier As String) As String

Returns the resource string in the current language for a given resource identifier.

GetFailureReason() As String

Yields additional useful information if a function return indicates an error.

GetLanguage() As String

Usage

At present, *roResourceManager* is primarily used for localizing the *roClockWidget*. The resource file passed in during creation has the following format for each string entry:

```
[RESOURCE_IDENTIFIER_NAME_GOES_HERE]
eng "Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec"
ger "Jan|Feb|Mär|Apr|Mai|Jun|Jul|Aug|Sep|Okt|Nov|Dez"
spa "Ene|Feb|Mar|Abr|May|Jun|Jul|Ago|Sep|Oct|Nov|Dic"
fre "Jan|Fév|Mar|Avr|Mai|Jun|Jul|Aou|Sep|Oct|Nov|Déc"
ita "Gen|Feb|Mar|Apr|Mag|Giu|Lug|Ago|Set|Ott|Nov|Dic"
dut "Jan|Feb|Mar|Apr|Mei|Jun|Jul|Aug|Sep|Okt|Nov|Dec"
swe "Jan|Feb|Mar|Apr|Maj|Jun|Jul|Aug|Sep|Okt|Nov|Dec"
```

The name in square brackets is the resource identifier. Each line after it is a language identifier followed by the resource string. Multiple *roResourceManager* objects can be created. A default "resources.txt" file, which contains a range of internationalization values for the clock widget, is available from the [BrightSign website](#).

roSystemLog

ON THIS PAGE

- [ifSystemLog](#)
 - [ReadLog\(\) As roArray](#)
- [ifStreamSend](#)
 - [SetSendEol\(eol_sequence As String\) As Void](#)
 - [SendByte\(byte As Integer\) As Void](#)
 - [SendLine\(string As String\) As Void](#)
 - [SendBlock\(a As Dynamic\) As Void](#)
 - [Flush\(\)](#)

- [Writing to a Remote Syslog Server](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object enables recording of the system log on a BrightSign player. Note that BrightScript `print` values are not written to the system log; use the `ifStreamSend` methods to write to the log instead.

Object Creation: This object is created with no parameters:

```
CreateObject("roSystemLog")
```

`ifSystemLog`

ReadLog() As roArray

Returns an array of strings. The strings correspond to lines in the system log, spanning from bootup to the most recent entry.

`ifStreamSend`

SetSendEol(eol_sequence As String) As Void

Sets the EOL sequence when writing to the stream. The default value is CR+LF. If you need to set this value to a non-printing character, use the `chr()` [global function](#).

SendByte(byte As Integer) As Void

Writes the specified byte to the stream.

SendLine(string As String) As Void

Writes the specified characters to the stream followed by the current EOL sequence.

SendBlock(a As Dynamic) As Void

Writes the specified characters to the stream. This method can support either a string or an `roByteArray`. If the block is a string, any null bytes will terminate the block.

Flush()

Writing to a Remote Syslog Server

BrightSign players can write to a remote syslog server using the standard syslog protocol. To configure remote logging, write the "syslog" entry to the "networking" section of the [player registry](#):

Example

```
registry = CreateObject("roRegistrySection", "networking")
registry.Write("syslog", "my-syslog-server")
registry.Flush()
```

You can specify the syslog using either its IP address or hostname.

[Date and Time Objects](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This section describes objects that manipulate date and time settings on the player.

- [roDateTime](#)
- [roNetworkTimeEvent](#)
- [roSystemTime](#)
- [roTimer](#)
- [roTimerEvent](#)
- [roTimeSpan](#)

roDateTime

ON THIS PAGE

- [ifDateTime](#)
 - [GetDayOfWeek\(\) As Integer](#)
 - [GetDay\(\) As Integer](#)
 - [GetMonth\(\) As Integer](#)
 - [GetYear\(\) As Integer](#)
 - [GetHour\(\) As Integer](#)
 - [GetMinute\(\) As Integer](#)
 - [GetSecond\(\) As Integer](#)
 - [GetMillisecond\(\) As Integer](#)
 - [SetDay\(day As Integer\) As Void](#)
 - [SetMonth\(month As Integer\) As Void](#)
 - [SetYear\(year As Integer\) As Void](#)
 - [SetHour\(hour As Integer\) As Void](#)
 - [SetMinute\(minute As Integer\) As Void](#)
 - [SetSecond\(second As Integer\) As Void](#)
 - [SetMillisecond\(millisecond As Integer\) As Void](#)
 - [AddSeconds\(seconds As Integer\) As Void](#)
 - [SubtractSeconds\(seconds As Integer\) As Void](#)
 - [AddMilliseconds\(milliseconds As Integer\) As Void](#)
 - [SubtractMilliseconds\(milliseconds As Integer\) As Void](#)
 - [Normalize\(\) As Boolean](#)
 - [ToIsoString\(\) As String](#)
 - [FromIsoString\(date-time As String\) As Boolean](#)
 - [ToSecondsSinceEpoch\(\) As Integer](#)
 - [FromSecondsSinceEpoch\(seconds As Integer\) As Boolean](#)
 - [GetString\(\) As String](#)
- [ifString](#)
 - [GetString\(\) As String](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is used to represent an instant in time.

At the time of its creation, a new object represents zero seconds.

`ifDateTime`

GetDayOfWeek() As Integer

GetDay() As Integer

GetMonth() As Integer

GetYear() As Integer

GetHour() As Integer

GetMinute() As Integer

GetSecond() As Integer

GetMillisecond() As Integer

SetDay(day As Integer) As Void

SetMonth(month As Integer) As Void

SetYear(year As Integer) As Void

SetHour(hour As Integer) As Void

SetMinute(minute As Integer) As Void

SetSecond(second As Integer) As Void

SetMillisecond(millisecond As Integer) As Void

AddSeconds(seconds As Integer) As Void

SubtractSeconds(seconds As Integer) As Void

AddMilliseconds(milliseconds As Integer) As Void

SubtractMilliseconds(milliseconds As Integer) As Void

Normalize() As Boolean

Checks that all the fields supplied are correct. This function fails if the values are out of bounds.

ToIsoString() As String

Returns the current *roDateTime* value as an ISO-8601 basic formatted string. Hyphens for date and colons for time are omitted, and a comma is used to separate seconds from milliseconds: For example, the ISO-8601 standard "2014-05-29T12:30:00.100" would be formatted as "20140529T123000,100".

FromIsoString(date-time As String) As Boolean

Sets the value of the *roDateTime* object using an ISO-8601 basic formatted string. Hyphens for date and colons for time are omitted, and either a period or comma can be used to separate seconds from milliseconds: The ISO-8601 standard "2014-05-29T12:30:00.100" could, for example, be formatted as either "20140529T123000,100" or "20140529T123000.100". This method will return *False* (indicating that it has not affected changes to the *roDateTime* object) if the string is formatted incorrectly or if the date passed is outside the range of January 1, 1970 and December 31, 2100.

ToSecondsSinceEpoch() As Integer

Returns the number of seconds that have elapsed since midnight on January 1, 1970, as represented by the *roDateTime* instance (not the system time).

FromSecondsSinceEpoch(seconds As Integer) As Boolean

Populates the *roDateTime* instance with the specified number of seconds since midnight on January 1, 1970.

GetString() As String

`ifString`

GetString() As String

Returns the current date using a sortable date format: "YYYY/MM/DD hh:mm:ss.sss".

roNetworkTimeEvent

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifNetworkTimeEvent](#)
 - [WasSuccessful\(\) As Boolean](#)
 - [GetFailureReason\(\) As String](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This event object is generated by the *roSystemTime* object.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data that will be returned by `GetUserData()`.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return *Invalid* if no data has been set.

`ifNetworkTimeEvent`

WasSuccessful() As Boolean

Returns True if the last attempt to set the clock via the network (i.e. NTP or HTTP) was successful.

GetFailureReason() As String

Returns a description of the error if the last attempt to set the clock via the network failed.

roSystemTime

ON THIS PAGE

- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\)](#)
- [ifSystemTime](#)
 - [GetLocalDateTime\(\) As roDateTime](#)
 - [GetUtcDateTime\(\) As roDateTime](#)
 - [GetZoneDateTime\(timezone_name As String\) As Object](#)
 - [SetLocalDateTime\(local_DateTime As roDateTime\) As Boolean](#)
 - [SetUtcDateTime\(utc_DateTime As roDateTime\) As Boolean](#)
 - [GetTimeZone\(\) As String](#)
 - [SetTimeZone\(zone_name As String\) As Boolean](#)
 - [IsValid\(\) As Boolean](#)
 - [GetLastNetworkTimeResult\(\) As roAssociativeArray](#)
- [Supported Time Zones](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object provides the ability to read and write the time stored in the real-time clock (RTC). It can also be used to read and write the time-zone setting.

Note

Dates up to January 1, 2038 are supported.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

ifMessagePort

SetPort(port As roMessagePort)

Posts messages of the type `roNetworkTimeEvent` to the attached message port.

ifSystemTime

GetLocalDateTime() As roDateTime

Returns the current time from the RTC (modulated using the current time zone) as an *roDateTime* instance.

GetUtcDateTime() As roDateTime

Returns the current time from the RTC (modulated using the UTC/GMT time zone) as an *roDateTime* instance.

GetZoneDateTime(timezone_name As String) As Object

Returns the current time from the RTC (modulated using the specified time zone) as an *roDateTime* instance. Supported time zones are listed below.

SetLocalDateTime(local_DateTime As roDateTime) As Boolean

Specifies a new time for the RTC using the current time zone.

SetUtcDateTime(utc_DateTime As roDateTime) As Boolean

Specifies a new time for the RTC using the UTC/GMT time zone.

GetTimeZone() As String

Returns the current time-zone setting of the player. A `POSIX:` value is appended to the beginning of the string if the time zone has been set using the POSIX format.

SetTimeZone(zone_name As String) As Boolean

Specifies a new time-zone setting for the player (supported time zones are listed below). Alternatively, a POSIX formatted time zone can be applied by appending a `POSIX:` value to the beginning of the string.

The following code specifies a POSIX-formatted time zone:

```
t = CreateObject("roSystemTime")
t.SetTimeZone("POSIX:GMT-0BST-1,M3.5.0/1:00,M10.5.0/2:00")
```

IsValid() As Boolean

Returns True if the system time is set to a valid value. The time can be set from the RTC or with NTP.

GetLastNetworkTimeResult() As roAssociativeArray

Returns an associative array containing information about the last attempt to set the time via the network:

- `success_timestamp`: A value indicating when the clock was last set successfully via the network. This value is zero if the clock has never been set successfully via the network.
- `attempt_timestamp`: A value indicating when the last attempt was made to set the clock via the network. This value is zero if no attempt has been made yet.
- `failure_reason`: If the last attempt to set the clock via the network failed, this string will contain an error message. If the last attempt was successful, this string will be empty.

In this associative array, "timestamp" refers to the number of seconds since the player booted. This value can be compared against the total uptime of the player, which is retrieved by calling `UpTime()`.

Supported Time Zones

The following are supported system time zones (this list does not apply to POSIX-formatted time zones):

- EST: US Eastern Time
- CST: US Central Time
- MST: US Mountain Time
- PST: US Pacific Time
- AKST: Alaska Time
- HST: Hawaii-Aleutian Time with no Daylight Savings (Hawaii)
- HST1: Hawaii-Aleutian Time with Daylight Saving
- MST1: US MT without Daylight Saving Time (Arizona)
- EST1: US ET without Daylight Saving Time (East Indiana)
- AST: Atlantic Time

- CST2: Mexico (Mexico City)
- MST2: Mexico (Chihuahua)
- PST2: Mexico (Tijuana)
- BRT: Brazil Time (Sao Paulo)
- NST: Newfoundland Time
- AZOT: Azores Time
- GMTBST: London/Dublin Time
- WET: Western European Time
- CET: Central European Time
- EET: Eastern European Time
- MSK: Moscow Time
- SAMT: Delta Time Zone (Samara)
- YEKT: Echo Time Zone (Yekaterinburg)
- IST: Indian Standard Time
- NPT: Nepal Time
- OMST: Foxtrot Time Zone (Omsk)
- JST: Japanese Standard Time
- CXT: Christmas Island Time (Australia)
- AWST: Australian Western Time
- AWST1: Australian Western Time without Daylight Saving Time
- ACST: Australian Central Standard Time (CST) with Daylight Saving Time
- ACST1: Darwin, Australia/Darwin, and Australian Central Standard Time (CST) without Daylight Saving Time
- AEST: Australian Eastern Time with Daylight Saving Time
- AEST1: Australian Eastern Time without Daylight Saving Time (Brisbane)
- NFT: Norfolk (Island) Time (Australia)
- NZST: New Zealand Time (Auckland)
- CHAST: , Fiji Time, , Fiji, Pacific/Fiji, Yankee Time Zone (Fiji)
- SST: X-ray Time Zone (Pago Pago)
- GMT: Greenwich Mean Time
- GMT-1: 1 hour behind Greenwich Mean Time
- GMT-2: 2 hours behind Greenwich Mean Time
- GMT-3: 3 hours behind Greenwich Mean Time
- GMT-3:30: 3.5 hours behind Greenwich Mean Time
- GMT-4: 4 hours behind Greenwich Mean Time
- GMT-4:30: 4.5 hours behind Greenwich Mean Time
- GMT-5: 5 hours behind Greenwich Mean Time
- GMT-6: 6 hours behind Greenwich Mean Time
- GMT-7: 7 hours behind Greenwich Mean Time
- GMT-8: 8 hours behind Greenwich Mean Time
- GMT-9: 9 hours behind Greenwich Mean Time
- GMT-9:30: 9.5 hours behind Greenwich Mean Time
- GMT-10: 10 hours behind Greenwich Mean Time
- GMT-11: 11 hours behind Greenwich Mean Time
- GMT-12: 12 hours behind Greenwich Mean Time
- GMT-13: 13 hours behind Greenwich Mean Time
- GMT-14: 14 hours behind Greenwich Mean Time
- GMT+1: 1 hour ahead of Greenwich Mean Time
- GMT+2: 2 hours ahead of Greenwich Mean Time
- GMT+3: 3 hours ahead of Greenwich Mean Time
- GMT+3:30: 3.5 hours ahead of Greenwich Mean Time
- GMT+4: 4 hours ahead of Greenwich Mean Time
- GMT+4:30: 4.5 hours ahead of Greenwich Mean Time
- GMT+5: 5 hours ahead of Greenwich Mean Time
- GMT+5:30: 5.5 hours ahead of Greenwich Mean Time
- GMT+6: 6 hours ahead of Greenwich Mean Time
- GMT+6:30: 6.5 hours ahead of Greenwich Mean Time
- GMT+7: 7 hours ahead of Greenwich Mean Time
- GMT+7:30: 7.5 hours ahead of Greenwich Mean Time
- GMT+8: 8 hours ahead of Greenwich Mean Time
- GMT+8:30: 8.5 hours ahead of Greenwich Mean Time
- GMT+9: 9 hours ahead of Greenwich Mean Time

- GMT+9:30: 9.5 hours ahead of Greenwich Mean Time
- GMT+10: 10 hours ahead of Greenwich Mean Time
- GMT+10:30: 10.5 hours ahead of Greenwich Mean Time
- GMT+11: 11 hours ahead of Greenwich Mean Time
- GMT+11:30: 11.5 hours ahead of Greenwich Mean Time
- GMT+12: 12 hours ahead of Greenwich Mean Time
- GMT+12:30: 12.5 hours ahead of Greenwich Mean Time
- GMT+13: 13 hours ahead of Greenwich Mean Time
- GMT+14: 14 hours ahead of Greenwich Mean Time

roTimer

ON THIS PAGE

- [ifTimer](#)
 - [SetTime\(hour As Integer, minute As Integer, second As Integer, millisecond As Integer\) As Void](#)
 - [SetTime\(a As Integer, b As Integer, c As Integer\)](#)
 - [SetDate\(year As Integer, month As Integer, day As Integer\) As Void](#)
 - [SetDayOfWeek\(day_of_week As Integer\) As Void](#)
 - [SetDateTime\(date_time As roDateTime\) As Void](#)
 - [Start\(\) As Boolean](#)
 - [Stop\(\) As Void](#)
 - [SetElapsed\(seconds As Integer, milliseconds As Integer\)](#)
- [ifMessagePort](#)
 - [SetPort\(port As roMessagePort\) As Void](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)
- [ifIdentity](#)
 - [GetIdentity\(\) As Integer](#)
- [Examples](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object allows the script to trigger events at a specific date/time or during specified intervals. Events are triggered by delivering *roTimerEvent* objects to the specified message port.

`ifTimer`

SetTime(hour As Integer, minute As Integer, second As Integer, millisecond As Integer) As Void

Sets the time for the event to trigger. In general, if a value is -1, then it is a wildcard and will cause the event to trigger every time the rest of the specification matches. If there are no wildcards, then the timer will trigger only once when the specified time occurs.

SetTime(a As Integer, b As Integer, c As Integer)

SetDate(year As Integer, month As Integer, day As Integer) As Void

Sets the date for the event to trigger. In general, if a value is -1, then it is a wildcard and will cause the event to trigger every time the rest of the specification matches. If there are no wildcards, then the timer will trigger only once when the specified date/time occurs.

SetDayOfWeek(day_of_week As Integer) As Void

Sets the day of week for the event to trigger. In general, if a value is -1, then it is a wildcard and will cause the event to trigger every time the rest of the specification matches. If there are no wildcards, then the timer will trigger only once when the specified date/time occurs.

It is possible, using a combination of `SetDate()` and `SetDayOfWeek()` calls, to specify invalid combinations that will never occur. If specifications include any wildcard, then the second and millisecond specifications must be zero; events will be raised at most once per minute near the whole minute.

SetDateTime(date_time As roDateTime) As Void

Sets the time when you wish the event to trigger from an *roDateTime* object. It is not possible to set wildcards using this method.

Start() As Boolean

Starts the timer based on the current values specified using the above functions.

Stop() As Void

Stops the timer.

SetElapsed(seconds As Integer, milliseconds As Integer)

Configures a timer to trigger once the specified time period has elapsed. Unlike the absolute timer methods above, changes to the system clock will not affect the period of the `SetElapsed()` timer.

ifMessagePort

SetPort(port As roMessagePort) As Void

Posts messages of type *roTimerEvent* to the attached message port.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

ifIdentity

GetIdentity() As Integer

Returns a unique number that can be used to identify when events originate from this object.

Note

The *ifIdentity* interface has been deprecated. We recommend using the *ifUserData* interface instead.

Examples

This script uses the `SetElapsed()` method to create a timer that triggers every 30 seconds:

```
Sub Main()  
    mp = CreateObject("roMessagePort")  
    timer = CreateObject("roTimer")  
    timer.SetPort(mp)  
  
    timer.SetElapsed(30, 0)  
  
    print "Start at "; Uptime(0)  
    timer.Start()  
  
    while true  
        ev = mp.WaitMessage(0)  
        if type(ev) = "roTimerEvent" then
```

```

        print "Timer event received at "; Uptime(0)
        timer.Start()
    else
        print "Another event arrived: "; type(ev)
    end if
end while
End Sub

```

This script creates a timer that triggers every minute using wildcards in the timer spec:

```

st=CreateObject("roSystemTime")
timer=CreateObject("roTimer")
mp=CreateObject("roMessagePort")
timer.SetPort(mp)

timer.SetDate(-1, -1, -1)
timer.SetTime(-1, -1, 0, 0)
timer.Start()

while true
    ev = Wait(0, mp)
    if (type(ev) = "roTimerEvent") then
        print "timer event received"
    else
        print "unexpected event received"
    endif
endwhile

```

This script creates a timer that triggers once at a specific date/time.

```

timer=CreateObject("roTimer")
mp=CreateObject("roMessagePort")
timer.SetPort(mp)

timer.SetDate(2008, 11, 1)
timer.SetTime(0, 0, 0, 0)

timer.Start()

while true
    ev = Wait(0, mp)
    if (type(ev) = "roTimerEvent") then
        print "timer event received"
    else
        print "unexpected event received"
    endif
endwhile

```

roTimerEvent

ON THIS PAGE

- [ifUserData](#)

- [SetUserData\(user_data As Object\)](#)
- [GetUserData\(\) As Object](#)
- [ifSourceIdentity](#)
 - [GetSourceIdentity\(\) As Integer](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This event object is generated by the *roTimer* object.

`ifUserData`

SetUserData(user_data As Object)

Sets the user data.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()` (either on the source or event object). It will return `Invalid` if no data has been set.

`ifSourceIdentity`

GetSourceIdentity() As Integer

Retrieves the identity value that can be used to associate events with the source *roTimer* instance.

Note

The *ifSourceIdentity* interface has been deprecated. We recommend using the *ifUserData* interface instead.

roTimeSpan

ON THIS PAGE

- [ifTimeSpan](#)
 - [Mark\(\)](#)
 - [TotalMilliseconds\(\) As Integer](#)
 - [TotalSeconds\(\) As Integer](#)
 - [GetSecondsToISO8601Date\(a As String\) As Integer](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object provides an interface to a simple timer for tracking the duration of activities. It is useful for tracking how long an action has taken or whether a specified time has elapsed from a starting event.

`ifTimeSpan`

Mark()

TotalMilliseconds() As Integer

TotalSeconds() As Integer

GetSecondsToISO8601Date(a As String) As Integer

Legacy Objects

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This section describes objects that are still offered by BrightScript, even though their primary functionality has been replaced by more modern objects.

- [roRtspStreamEvent](#)
- [roSyncPool](#)
- [roSyncPoolEvent](#)
- [roSyncPoolFiles](#)
- [roSyncPoolProgressEvent](#)

roRtspStreamEvent

ON THIS PAGE

- [ifInt](#)
 - [GetInt\(\) As Integer](#)
 - [SetInt\(value As Integer\) As Void](#)
- [ifUserData](#)
 - [SetUserData\(user_data As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

This object is no longer used to return events related to RTSP streams. The *roVideoPlayer* object now returns events related to an associated *roRtspStream*.

ifInt

GetInt() As Integer

Returns the event ID.

SetInt(value As Integer) As Void

Sets the value of the event.

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned by `GetUserData()`.

***GetUserData()* As Object**

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

roSyncPool

ON THIS PAGE

- **ifSyncPool**
 - `ValidateFiles(sync_spec As roSyncSpec, directory As String, options_array As roAssociativeArray) As Object`
 - `GetFailureReason() As String`
 - `AsyncDownload(a As Object) As Boolean`
 - `AsyncCancel() As Boolean`
 - `Realize(a As Object, b As String) As Object`
 - `ProtectFiles(a As Object, b As Integer) As Boolean`
 - `ReserveMegabytes(a As Integer) As Boolean`
 - `GetPoolSizeInMegabytes() As Integer`
 - `EstimateRealizedSizeInMegabytes(a As Object, b As String) As Integer`
 - `IsReady(a As Object) As Boolean`
 - `Validate(a As Object, b As Object) As Boolean`
 - `EnablePeerVerification(a As Boolean)`
 - `EnableHostVerification(a As Boolean)`
 - `SetCertificatesFile(a As String)`
 - `SetUserAndPassword(a As String, b As String) As Boolean`
 - `AddHeader(a As String, b As String)`
 - `SetHeaders(a As Object) As Boolean`
 - `SetMinimumTransferRate(a As Integer, b As Integer) As Boolean`
 - `AsyncSuggestCache(a As Object) As Boolean`
 - `SetProxy(a As String) As Boolean`
 - `SetProxyBypass(a As Array) As Boolean`
 - `SetFileProgressIntervalSeconds(a As Integer) As Boolean`
 - `QueryFiles(a As Object) As Object`
 - `SetFileRetryCount(a As Integer) As Boolean`
 - `SetRelativeLinkPrefix(prefix As String) As Boolean`
 - `BindToInterface(interface As Integer) As Boolean`
 - `EnableUnsafeAuthentication(a As Boolean)`
 - `EnableUnsafeProxyAuthentication(enable As Boolean) As Boolean`
 - `EnableEncodings(enable As Boolean) As Boolean`
 - `SetMaximumReceiveBytesPerSecond(bytes_per_second As Double) As Boolean`
 - `SetMaximumPoolSizeMegabytes(maximum_size As Integer) As Boolean`
- **ifIdentity**
 - `GetIdentity() As Integer`
- **ifMessagePort**
 - `SetPort(a As Object)`
- **ifUserData**
 - `SetUserData(user_data As Object)`
 - `GetUserData() As Object`

▼ Firmware Version 7.0

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

We recommend using `roAssetPool` instead.

Object Creation: The *roSyncPool* object is created with a single parameter that specifies the file path where the pool is located.

```
CreateObject("roSyncPool", pool_path As String)
```

Example

```
pool = CreateObject ("roSyncPool", "SD:/pool")
```

ifSyncPool

ValidateFiles(sync_spec As roSyncSpec, directory As String, options_array As roAssociativeArray) As Object

Validates the files in the specified directory against the hashes in the specified sync spec. Files that are not in the sync spec are ignored. The options array can currently contain the following optional parameters:

- ***DeleteCorrupt*** (Boolean): Automatically delete files that do not match the sync spec. The method will return an associative array that maps each filename to an explanation of why it is corrupt. The array only contains corrupt files, so the success is reported by the method returning an empty associative array.

GetFailureReason() As String

AsyncDownload(a As Object) As Boolean

AsyncCancel() As Boolean

Realize(a As Object, b As String) As Object

ProtectFiles(a As Object, b As Integer) As Boolean

ReserveMegabytes(a As Integer) As Boolean

GetPoolSizeInMegabytes() As Integer

EstimateRealizedSizeInMegabytes(a As Object, b As String) As Integer

IsReady(a As Object) As Boolean

Validate(a As Object, b As Object) As Boolean

EnablePeerVerification(a As Boolean)

EnableHostVerification(a As Boolean)

SetCertificatesFile(a As String)

SetUserAndPassword(a As String, b As String) As Boolean

AddHeader(a As String, b As String)

SetHeaders(a As Object) As Boolean

SetMinimumTransferRate(a As Integer, b As Integer) As Boolean

AsyncSuggestCache(a As Object) As Boolean

SetProxy(a As String) As Boolean

SetProxyBypass(a As Array) As Boolean

SetFileProgressIntervalSeconds(a As Integer) As Boolean

QueryFiles(a As Object) As Object

SetFileRetryCount(a As Integer) As Boolean

SetRelativeLinkPrefix(prefix As String) As Boolean

BindToInterface(interface As Integer) As Boolean

EnableUnsafeAuthentication(a As Boolean)

EnableUnsafeProxyAuthentication(enable As Boolean) As Boolean

EnableEncodings(enable As Boolean) As Boolean

SetMaximumReceiveBytesPerSecond(bytes_per_second as Double) As Boolean

SetMaximumPoolSizeMegabytes(maximum_size As Integer) As Boolean

ifIdentity

GetIdentity() As Integer

Note

The *ifSourceIdentity* interface has been deprecated. We recommend using the *ifUserData* interface instead.

ifMessagePort

SetPort(a As Object)

ifUserData

SetUserData(user_data As Object)

Sets the user data that will be returned when events are raised.

GetUserData() As Object

Returns the user data that has previously been set via `SetUserData()`. It will return `Invalid` if no data has been set.

roSyncPoolEvent

ON THIS PAGE

- [ifSourceIdentity](#)
 - [GetSourceIdentity\(\) As Integer](#)
- [ifSyncPoolEvent](#)
 - [GetEvent\(\) As Integer](#)
 - [GetName\(\) As String](#)
 - [GetResponseCode\(\) As Integer](#)
 - [GetFailureReason\(\) As String](#)
 - [GetFileIndex\(\) As Integer](#)
- [ifUserData](#)
 - [SetUserData\(a As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

We recommend using [roAssetFetcherEvent](#) instead.

`ifSourceIdentity`

GetSourceIdentity() As Integer

`ifSyncPoolEvent`

GetEvent() As Integer

GetName() As String

GetResponseCode() As Integer

GetFailureReason() As String

GetFileIndex() As Integer

`ifUserData`

SetUserData(a As Object)

GetUserData() As Object

roSyncPoolFiles

ON THIS PAGE

- [ifSyncPoolFiles](#)
 - [GetFailureReason\(\) As String](#)
 - [GetPoolFilePath\(a As String\) As String](#)
 - [GetPoolFileInfo\(a As String\) As Object](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

We recommend using [roAssetPoolFiles](#) instead.

`ifSyncPoolFiles`

[GetFailureReason\(\) As String](#)

[GetPoolFilePath\(a As String\) As String](#)

[GetPoolFileInfo\(a As String\) As Object](#)

`roSyncPoolProgressEvent`

ON THIS PAGE

- [ifSourceIdentity](#)
 - [GetSourceIdentity\(\) As Integer](#)
- [ifSyncPoolProgressEvent](#)
 - [GetFileName\(\) As String](#)
 - [GetFileIndex\(\) As Integer](#)
 - [GetFileCount\(\) As Integer](#)
 - [GetCurrentFileTransferredMegabytes\(\) As Integer](#)
 - [GetCurrentFileSizeMegabytes\(\) As Integer](#)
 - [GetCurrentFilePercentage\(\) As Float](#)
- [ifUserData](#)
 - [SetUserData\(a As Object\)](#)
 - [GetUserData\(\) As Object](#)

▼ [Firmware Version 7.0](#)

- [Version 7.0](#)
- [Version 6.2](#)
- [Version 6.1](#)
- [Previous Versions](#)

We recommend using [roAssetFetcherProgressEvent](#) instead.

`ifSourceIdentity`

[GetSourceIdentity\(\) As Integer](#)

`ifSyncPoolProgressEvent`

[GetFileName\(\) As String](#)

GetFileIndex() As Integer

GetFileCount() As Integer

GetCurrentFileTransferredMegabytes() As Integer

GetCurrentFileSizeMegabytes() As Integer

GetCurrentFilePercentage() As Float

ifUserData

SetUserData(a As Object)

GetUserData() As Object