ogScript Reference Guide

for DashBoard v6.2

www.rossvideo.com

ROSS
Production Technology Experts

Thank you for choosing Ross

You've made a great choice. We expect you will be very happy with your purchase of Ross Technology.

Our mission is to:

1. Provide a Superior Customer Experience
    - offer the best product quality and support
2. Make Cool Practical Technology
    - develop great products that customers love

Ross has become well known for the Ross Video Code of Ethics. It guides our interactions and empowers our employees. I hope you enjoy reading it below.

If anything at all with your Ross experience does not live up to your expectations be sure to reach out to us at *solutions@rossvideo.com*.

David Ross
CEO, Ross Video
dross@rossvideo.com

Ross Video Code of Ethics

Any company is the sum total of the people that make things happen. At Ross, our employees are a special group. Our employees truly care about doing a great job and delivering a high quality customer experience every day. This code of ethics hangs on the wall of all Ross Video locations to guide our behavior:

1. We will always act in our customers' best interest.
2. We will do our best to understand our customers' requirements.
3. We will not ship crap.
4. We will be great to work with.
5. We will do something extra for our customers, as an apology, when something big goes wrong and it's our fault.
6. We will keep our promises.
7. We will treat the competition with respect.
8. We will cooperate with and help other friendly companies.
9. We will go above and beyond in times of crisis. *If there's no one to authorize the required action in times of company or customer crisis - do what you know in your heart is right. (You may rent helicopters if necessary.)*

ogScript Reference Guide

- Ross Part Number: **8351DR-006-03**
- Release Date: September 2, 2014. Printed in Canada.

The information contained in this Guide is subject to change without notice or obligation.

Copyright

Patents

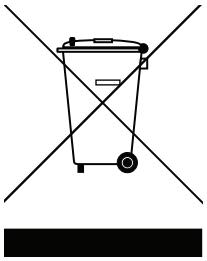
Ross Video products are protected by patent numbers US 7,034,886; US 7,508,455; US 7,602,446; US 7,802,802 B2; US 7,834,886; US 7,914,332; US 8,307,284; US 8,407,374 B2; US 8,499,019 B2; US 8,519,949 B2; US 8,743,292 B2; GB 2,419,119 B; GB 2,447,380 B. Other patents pending.

Environmental Information

The equipment that you purchased required the extraction and use of natural resources for its production. It may contain hazardous substances that could impact health and the environment.

To avoid the potential release of those substances into the environment and to diminish the need for the extraction of natural resources, Ross Video encourages you to use the appropriate take-back systems. These systems will reuse or recycle most of the materials from your end-of-life equipment in an environmentally friendly and health conscious manner.

The crossed-out wheeled bin symbol invites you to use these systems.



If you need more information on the collection, reuse, and recycling systems, please contact your local or regional waste administration.

You can also contact Ross Video for more information on the environmental performances of our products.

Company Address

Ross Video Limited
8 John Street
Iroquois, Ontario
Canada, K0E 1K0

Ross Video Incorporated
P.O. Box 880
Ogdensburg, New York
USA 13669-0880

General Business Office: (+1) 613 • 652 • 4886
Fax: (+1) 613 • 652 • 4425

Technical Support: (+1) 613 • 652 • 4886
After Hours Emergency: (+1) 613 • 349 • 0006

E-mail (Technical Support): techsupport@rossvideo.com
E-mail (General Information): solutions@rossvideo.com
Website: http://www.rossvideo.com

Contents

Introduction

Ross Video ogScript enables you to add functionality and logic to your custom panels. The information about ogScript contained in this guide applies to the following DashBoard version or greater:

- Version 6.2

To view your current DashBoard version, select **About DashBoard** from the **Help** menu in DashBoard.

JavaScript

Ross Video ogScript is a programming language developed by Ross Video to interact with Dashboard-enabled devices. It uses JavaScript functions, syntax, and primitive object types. To enable CustomPanel developers to interact with panels and devices, ogScript adds some new global objects to JavaScript. Most JavaScript works in ogScript scripts, although you might run across an occasional item that does not work.

For information about ogScript objects and functions, refer to the sections in this guide. For information about JavaScript commands and syntax, search for "JavaScript Reference" on the World Wide Web.

Commonly Used Functions

Ross Video recommends that you first learn the following commonly used functions:

- **ogscript**
  - "**debug**" on page 2–13
  - "**rename**" on page 2–31
- **params**
  - "**getValue**" on page 3–5
  - "**setValue**" on page 3–9

Functions Set in the User Interface

Functions in the following objects are typically set through a user interface:

- "**rosstalk Object**" on page 4–1
- "**robot Object**" on page 5–1
- "**vdcp Object**" on page 6–1
- "**nkscript Object**" on page 7–1

ogscript Object

In ogScript, use the **ogscript** object to access a library of general-purpose functions.

To call a general-purpose function, use:

```
ogscript.function name (parameters);
```

For example:

```
ogscript.debug ('This is a message');
```

ogscript Functions

The ogscript object contains the following functions:

Table 2.1 ogscript functions

Function	Parameters	Returns	Description
asyncExec	function	N/A	Executes a function outside of the UI current thread.
asyncFTP	host port username password destPath destName binary sourceFilePath callback	N/A	Sends a file to an FTP server.
asyncFTPGet	host port username password srcPath srcName binary destFilePath or null callback	N/A	Retrieves a file from FTP server.
asyncPost	String [URL], String [HTTP Post Data], Function [Callback Function]	N/A	Send an asynchronous post to the given URL.
cancelTimer	Timer ID	N/A	Cancel, stop and clean-up, a timer with the given ID.
copyByteArray	src offset length	byte array	Creates a full or partial copy of a byte array.
createByteArray	length	an empty byte array	Creates an empty byte array of a specified size.

Table 2.1 ogscript functions

Function	Parameters	Returns	Description
createMessageBuilder	N/A	Returns a MessageBuilder object used to build byte arrays (generally for creating network messages).	Creates a message builder, which enables you to construct a message.
createMessageParser	messageBytes	Returns a MessageParser object (generally used to parse the various pieces of messages received over the network).	Creates a message parser, which enables you to parse a message.
debug	String [Message]	N/A	Write a string to the openGear Debug Information View.
fireGPI	String [trigger] String [state] Boolean [global]	N/A	Sends Trigger GPI string [trigger] to execute component task lists. Sends optional [state] data string, which can be read by the script. When [global]' value is **'true'**, applies to all open panels. When [global] is '**false**', applies only to the current active panel.
getAllById	String [Object ID]	Object []	Get all Objects accessible in the current context that have the associated ID.
getAttribute	String [Attribute ID]	Object	Get an attribute registered in the context with the given ID.
getComponentsById	String [Object ID]	Component []	Get all Java Swing components accessible in the current context that have the associated ID.
getCurrentUser	N/A	String	Returns the username of the current DashBoard user.
getIncludeById	String [Include ID]	IncludeReloadableContainer	Returns the first **include** with the given ID.
getListenerById	ID	getListenerById returns an object representing the listener. This object has three public methods you can call: **start()**, **stop()**, and **isStarted()**. The return depends on which of the three methods is used: • If the **start()** method is used, return is **true** if the listener started successfully; otherwise **false**. • If the **stop()** method is used, return is **true** if the listener stopped successfully; otherwise **false**. • If the **isStarted()** method is used, return is **true** if the listener is started; otherwise **false**.	Starts or stops a listener. Can also check whether a listener is started.
getObject	String [Key]	String	Retrieves stored object

Table 2.1 ogscript functions

Function	Parameters	Returns	Description
getPrivateString	String [Lookup ID], String [Key]	String	Get a string defined in the lookup table with the specified lookup ID.
getScopedAttribute	String [Scope Name], String [Attribute ID]	Object	Get an attribute in the named scope that has the given ID. Scopes are often internally defined by DashBoard.
getString	String [Key]	String	Get a string defined in the global lookup table.
getTimerManager	N/A	ContextTimerManager	Get the timer manager for the context to access timers and perform operations on selected timers. This function includes several methods.
hide	String [ID]	N/A	Hide the popup with the specified ID.
installTimer	String [Timer ID], Boolean [Repeat], Long [Delay], Long [Repeat Rate], Function [Task]	N/A	Create a timer with the given ID and register it in the ContextTimerManager. Start the timer after the specified delay, repeat the timer if requested at the specified rate. When the timer fires, run the specified ogScript function.
isTimerRunning	String [Timer ID]	Boolean	Report whether or not a timer exists and is in the “running” state. **true** — a timer with the given ID exits and is in the “running” state. **false** — a timer with the give ID does not exist or is not in the “running” state.
parseXML	String	org.w3c.dom.Document	Parse and return an XML document using the org.w3c.dom.Document API.
putObject	String [Key] String [Value]	N/A	Defines a stored object.
putPrivateString	String [LookupID], String [Key], String [Value]	N/A	Add or replace a string in a private lookup table.
putString	String [key], String [value]	N/A	Add or replace a string in the global lookup table.
reload	String [ID]	N/A	Rebuild the UI element with the given ID.
rename	String [ID], String [Name]	N/A	Modify the text for a tab name, button, or label with the specified ID.
reposition	String [ID], Integer [x position], Integer [y position],	N/A	Moves object to specified XY pixel location

Table 2.1 ogscript functions

Function	Parameters	Returns	Description
repositionByPercent	String [ID], Integer [percent x], Integer [percent y], Boolean [center x], Boolean [center y]	N/A	Moves object to the specified location, as percentage of the container width or height. Center x and center y, when true, center the object at the location horizontally (x only), vertically (y only), or both (x and y).
reveal	String [ID]	N/A	Open a popup with the specified ID, or bring the tab with the specified ID to the foreground.
runXPath	String [XPath], XML Document or XML Element	NodeList	Execute the given XPath command on the given Document or Element and return the results as a NodeList.
saveToFile	path data overwrite	Returns **`true`**, if data is written successfully; otherwise **`false`**.	Saves data to a file. This function is typically used to save a byte array, string, or XML document to a file.
sendUDPAsBytes	String [Host], Integer [Port], Byte[] [Data]	N/A	Send the given Data bytes to the provided Host/Port through UDP.
sendUDPString	String [Host], Integer [Port], String [Data]	N/A	Convert the given Data string to UTF-8 bytes and send them to the provided Host/Port through UDP.
serializeXML	String [document]	Returns a string that is a serialized version of the original XML document.	Serializes an XML document and converts it to a string.
setAnchorPoints	String [ID], Boolean [top], Boolean [left], Boolean [bottom], Boolean [right]	N/A	Specifies how an object moves if the user interface is resized for different monitor and window sizes. Anchors or releases an object to/from the top, left, bottom, or right sides of its container.
setStyle	String [ID], String [Style]	N/A	Set Style parameters for the component with the given ID if it exists.
setXML	String [ID], String [new XML Content]	N/A	Dynamically generates UI components through ogscript. Replaces the contents of an element with a string of XML code.
toBottom	String [ID]	N/A	Displays the object below all others in the same container. Objects are layered. If they overlap, higher layers are drawn over lower layers.
toTop	String [ID]	N/A	Displays the object above all others in the same container. Objects are layered. If they overlap, higher layers are drawn over lower layers.
upload	File [Upload File]	N/A	Open the File Upload dialog with the specified file.

asyncExec

Executes a function outside of the UI current thread.

This is especially useful for operations that take time to complete. You can use **`asyncExec`** to run such operations while continuing to execute the rest of your tasks.

Syntax

```
ogscript.asyncExec (function);
```

Parameters

Parameter	Type	Required	Description
function	Function reference	Yes	Reference to the function to be run.

Returns

N/A

Example

Coming soon.

asyncFTP

Sends a file to an FTP server. If a callback is provided, asyncFTP calls it when the operation is complete.

Note: As the file is transferred, a progress attribute is updated. You can add an ogscipt handler to monitor changes to the attribute to show progress.

Syntax

```
ogscript.asyncFTP (host, port, username, password, destPath, destName, binary,
sourceFilePath, callback);
```

Parameters

Parameter	Type	Required	Description
host	String	Yes	The host name of the destination computer.
port	Integer	Yes	The port number to which the data is to be sent.
username	String	Yes	The username required to log onto the destination computer.
password	String	Yes	The password required to log onto the destination computer.
destPath	String	No	The directory path where the data is to be saved on the destination computer.
destName	String	No	The name of the destination file. Can be used to rename the existing file. If a file with the same name exists in the destination path, that file is overwritten.
binary	Boolean	Yes	Specifies the transfer mode. When true, binary transfer is used. When false, ASCII transfer is used.
sourceFilePath	String	Yes	The directory path to the source file. The path can be absolute or relative.
callback	function reference	No	The callback is called when the operation is complete, whether or not the operation is successful.

Returns

N/A

Example 1

The following example is a task. It uses variable to populate the parameters of the **`asyncFTP`** function. It also includes a callback to indicate success or failure of the transfer.

```
<task tasktype="ogscript">function callback(success, sourceFilePath,
exception)
   {
      if (success)
      {
         ogscript.rename('label.bytes', 'SUCCESS!');
      }
      else
      {
         ogscript.rename('label.bytes', 'FAIL!');
      }
   }

   ogscript.rename('label.bytes', 'TRYING TO SEND FILE');
```

```
  var host = params.getStrValue('params.host', 0);
  var port = params.getValue('params.port', 0);
  var user = params.getStrValue('params.username', 0);
  var password = params.getStrValue('params.password', 0);
  var file = params.getStrValue('params.file', 0);
  var destPath = params.getStrValue('params.destpath', 0);
  var destFileNameOverride = null;
  var isBinary = true;

  ogscript.asyncFTP(host, port, user, password, destPath,
  destFileNameOverride, isBinary, file, callback);
  ogscript.rename('label.bytes', 'Waiting...');
</task>
```

Example 2

The following is an example of an ogscript handler for monitoring and reporting the progress of the transfer.

```
<ogscript attribute="com.rossvideo.ftp.event"
handles="attributechange">var progressEvent = event.getNewValue();
if (progressEvent == null)
{
   ogscript.debug('No progress');
}
else
{
   ogscript.rename('label.bytes',
(progressEvent.getTotalBytesTransferred() / 1024) + 'kb');
}</ogscript>
```

asyncFTPGet

Retrieves a file from FTP server.

Syntax

```
ogscript.asyncFTPGet (host, port, username, password, srcPath, srcName, binary,
destFilePath or null, callback);
```

Parameters

Parameter	Type	Required	Description
host	String	Yes	The host name of the source computer, from which the file is to be retrieved
port	Integer	Yes	The port number required to access the source computer.
username	String	Yes	The username required to log onto the source computer.
password	String	Yes	The password required to log onto the source computer.
srcPath	String	No	The directory path where the source file is located.
srcName	String	Yes	The name of the file to be retrieved.
binary	Boolean	Yes	Specifies the transfer mode. When true, binary transfer is used. When false, ASCII transfer is used.
destFilePath or null	String	No	The directory path where the file is to be saved on the local computer. If null, the file is saved in the same directory as the panel.
callback	function reference	No	The callback is called when the operation is complete, whether or not the operation is successful.

Returns

N/A

Example

Coming soon.

asyncPost

Send an asynchronous post to the given URL. Call the given function when the post has completed. The data retrieved from the HTTP Post is passed as a string as the first variable in the method.

Syntax

```
ogscript.async (URL, HTTP Post Data, Callback Function);
```

Parameters

Parameter	Type	Required	Description
URL	String	Yes	URL to send a post.
HTTP Post Data	String	Yes	Post to send to the specified URL.
Callback Function	Function	Yes	Function to call after the post completes.

Returns

N/A

Example

Coming soon.

cancelTimer

Cancel, stop and clean up, a timer with the given ID.

Note: For information about creating a timer function, see "**installTimer**" on page 2–26.

Syntax

```
ogscript.cancelTimer (Timer ID);
```

Parameters

Parameter	Type	Required	Description
Timer ID	String	Yes	ID of the timer to stop and clean up.

Returns

N/A

Example

```
//Stop the timer that was created with installTimer
ogscript.cancelTimer('myTimer');
```

copyByteArray

Creates a full or partial copy of a byte array.

Syntax

```
ogscript.copyByteArray(src, offset, length)
```

Parameters

Parameter	Type	Required	Description
src	byte array	Yes	The byte array to be copied.
offset	Integer	Yes	Index of the first byte to be copied. Use 0 for the start of the array.
length	Integer	Yes	The number of bytes to copy. **Tip:** To copy the entire array, use **`src.length`**.

Returns

byte array

Example 1

In the following example, the contents of a byte array named **`srcArray`** are copied into a variable named **`myCopy`**.

```
var myCopy=ogscript.copyByteArray (srcArray,0,srcArray.length);
```

Example 2

In the following example, the 20 bytes of a byte array named **`srcArray`**, starting at byte **`4`**, are copied into a variable named **`myCopy`**.

```
var myCopy=ogscript.copyByteArray (srcArray,4,20);
```

createByteArray

Creates an empty byte array of a specified size.

Syntax

```
ogscript.createByteArray (length);
```

Parameters

Parameter	Type	Required	Description
length	Integer	Yes	The size of the new array, in bytes.

Returns

An empty byte array.

Example

```
var myNewByteArray = ogscript.createByteArray (12);
```

createMessageBuilder

Creates a message builder, which enables you to construct a message. The message is created as a byte array, and can contain multiple data types.

Syntax

```
ogscript.createMessageBuilder ();
```

Parameters

N/A

Returns

Returns a MessageBuilder object used to build byte arrays (generally for creating network messages).

Example

In the following example, a variable named **`myMessage`** is created to contain message content created by a message builder. Then data of various data types are added to the message. The variable **`messageArray`** is defined to contain the message content as a byte array.

Tip: You can use the **`createMessageParser`** function to parse messages.

```
var myMessage = ogscript.createMessageBuilder();

myMessage.writeBoolean(true);
myMessage.writeByte(255);
myMessage.writeByte(255);
myMessage.writeShort(65535);
myMessage.writeShort(65535);
myMessage.writeChar('a');
myMessage.writeInt(65536);
myMessage.writeLong(4294967296);
myMessage.writeFloat(0.000001);
myMessage.writeDouble(0.000002);
myMessage.writeString('abcd');
myMessage.writeUTF('Hello World'); //includes 2-byte length count

var messageArray = myMessage.toByteArray();
```

createMessageParser

Creates a message parser, which enables you to parse a message.

Syntax

Please provide syntax.

```
ogscript.createMessageParser (messageBytes);
```

Parameters

Parameter	Type	Required	Description
messageBytes	byte array	Yes	The source byte array.

Returns

Returns a MessageParser object (generally used to parse the various pieces of messages received over the network).

Example

In the following example, a variable named **`messageArray`** contains several pieces of data of various data types to be extracted by a message parser. A variable named **`parsedMessage`** is created to contain the extracted message content. Each element of the array is parsed and sent to the debug utility.

Tip: You can use the createMessageBuilder function to create messages.

```
var parsedMessage = ogscript.createMessageParser(messageArray);
ogscript.debug(parsedMessage.readBoolean());
ogscript.debug(parsedMessage.readByte());
ogscript.debug(parsedMessage.readUnsignedByte());
ogscript.debug(parsedMessage.readShort());
ogscript.debug(parsedMessage.readUnsignedShort());
ogscript.debug(parsedMessage.readChar());
ogscript.debug(parsedMessage.readInt());
ogscript.debug(parsedMessage.readLong());
ogscript.debug(parsedMessage.readFloat());
ogscript.debug(parsedMessage.readDouble());
ogscript.debug(parsedMessage.readString(4));
ogscript.debug(parsedMessage.readUTF());</task>
```

debug

Write a string to the **openGear Debug Information** view.

The openGear Debug Information view must be open to view debug messages. To open the openGear Debug Information view, select **openGear Debug Information** from the **Views** menu in DashBoard.

Syntax

```
ogscript.debug (Message);
```

Parameters

Parameter	Type	Required	Description
Message	String	Yes	Message to display in the openGear Debug Information View.

Returns

N/A

Example 1

```
ogscript.debug ('This is a message');
```

Example 2

```
var data = params.getValue(0x12,0);
ogscript.debug ('Parameter 0x12 (score): ' + data);
```

Example 3

```
ogscript.debug ('Parameter 0x12 (score): ' +
params.getValue(0x12,0));
```

fireGPI

Sends a Trigger GPI message to panels. When buttons, labels, and displayed parameters that have a matching GPI Trigger receive the message, their task lists are executed.

Tip: This function can be used for inter-panel communication, by triggering globally.

Syntax

```
ogscript.fireGPI (Trigger), (State), (Global);
```

Parameters

Parameter	Type	Required	Description
Trigger	String	Yes	GPI Trigger message.
State	String	No	Sends optional data string, which can be read by the script.
Global	Boolean	Yes	When **`true`**, applies to all open panels. When **`false`**, applies only the panel initiating the trigger.

Returns

N/A

Example

In this example, the GPI trigger message **`'startClock'`** and the state data **`'resetClock'`** are sent to all open panels.

```
ogscript.fireGPI ('startClock','resetClock',true);
```

getAllById

Get all Objects accessible in the current context that have the associated ID.

Syntax

```
ogscript.getAllById (Object ID);
```

Parameters

Parameter	Type	Required	Description
Object ID	String	Yes	ID of the objects in the current context to get.

Returns

Object []

Example

Coming soon.

getAttribute

Get an attribute registered in the context with the given ID.

Syntax

```
ogscript.getAttribute (Attribute ID);
```

Parameters

Parameter	Type	Required	Description
Attribute ID	String	Yes	ID from which to get a registered in context attribute.

Returns

Object

Example

Coming soon.

getComponentsById

Get all Java Swing components accessible in the current context that have the associated ID.

Syntax

```
ogscript.getComponentsById (Object ID);
```

Parameters

Parameter	Type	Required	Description
Object ID	String	Yes	ID from which to get all Java Swing components accessible in the current context.

Returns

Component []

Example

Coming soon.

getCurrentUser

Returns the username of the current DashBoard user.

- When a User Rights Management server is present, this function returns the username of the user signed-in to DashBoard.
- When no User Rights Management Server is found, this function returns the computer account name.

Syntax

```
ogscript.getCurrentUser ();
```

Parameters

Parameter	Type	Required	Description
N/A	N/A	N/A	N/A

Returns

String

Example

This example uses the getCurrentUser function to read the user name, and then uses the rename function to rename a label. For more information about the rename function, see “**rename**” on page 2–31.

The label is defined in the .grid file as follows:

```
<label height="49" id="Welcome Label" left="136" name="Welcome"
style="txt-align:west;" top="275" width="188"/>
```

The script to read the user name and then rename the label is as follows:

```
//read the login user name
var loginName = ogscript.getCurrentUser();

//display the user name in the Welcome label
var message = 'Welcome ' + loginName;
ogscript.rename('Welcome Label',message);
```

getIncludeById

Returns the first **include** with the given ID. The include must have been created using the `<include>` tag.

Syntax

```
ogscript.getIncludeById (Include ID);
```

Parameters

Parameter	Type	Required	Description
Include ID	String	Yes	ID of the include to find.

Returns

IncludeReloadableContainer

Example

Coming soon.

getListenerById

Starts or stops a listener. Can also check whether a listener is started.

Syntax

```
ogscript.getListenerById (ID);
```

Parameters

Parameter	Type	Required	Description
ID	String	Yes	ID of the listener.

Returns

getListenerById returns an object representing the listener.

This object has three public methods you can call: **`start()`**, **`stop()`**, and **`isStarted()`**.

The return depends on which of the three methods is used:

- If the **`start()`** method is used, return is **`true`** if the listener started successfully; otherwise **`false`**.
- If the **`stop()`** method is used, return is **`true`** if the listener stopped successfully; otherwise **`false`**.
- If the **`isStarted()`** method is used, return is **`true`** if the listener is started; otherwise **`false`**.

Example

```
var myListener = ogscript.getListenerById (myId);

myListener.start ();

myListener.stop ();

myListener.isStarted ();
```

getObject

You can create an object and reference it in other parts of the code.

Some possible uses include:

- Storing parsed XML data in an object so you don't have to re-parse it.
- Storing the results of an async HTTP post so you don't have to re-fetch it.
- Storing connection code so you cna reference it wherever your code needs to establish that connection.

The getObject function works in conjunction with the putObject function. The putObject function defines the object. The getObject function references the object. The scope of a defined object is global, so you can reference it from anywhere in your panel code.

For information about the putObject function, see "**putObject**" on page 2–28.

Syntax

```
ogscript.getObject(key);
```

Parameters

Parameter	Type	Required	Description
Key	String	Yes	The name used to reference what is being stored.

Returns

String.

Example

The following example parses and stores data from an XML file in a variable so it can be used globally without the need to re-parse the XML data each time you want to use it.

It defines a function named loadTheXML, which uses the parseXML function to retrieve XML data from a file and load it into a variable named myObject. It then uses the putObject function to copy the data into a variable named myXML. The readTheXML function loads the data into a variable named otherObject.

```
function loadTheXML()
   {
      var myObject = ogscript.parseXML('file:/c:/mydocument.xml');
      ogscript.putObject('myXML',myObject);
   }

function readTheXML()
   {
      var otherObject = ogscript.getObject('myXML');
      // Do anything you want with the data, now contained in the
      otherObject variable.
   }
```

getPrivateString

Get a string defined in a private lookup table that matches the specified lookup ID.

Note: Use the getPrivateString function if the lookup table has an ID. If the lookup table has no ID, use the getString function. For more information about the getString function, see "**getString**" on page 2–20.

Syntax

```
ogscript.getPrivateString (Lookup ID, Key);
```

Parameters

Parameter	Type	Required	Description
Lookup ID	String	Yes	ID of the string to find in the specified lookup table.
Key	String	Yes	Private lookup table in which to find the specified string.

Returns

String

Example

This example uses the getPrivateString function to read an IP address stored in a lookup table. The lookup table is defined at the beginning of the .grid file, and can be accessed by any script.

The lookup table definition for this example is as follows:

```
<lookup id="hosts">
   <entry key="XPression.host">10.0.2.210</entry>
   <entry key="XPression.port">7788</entry>
</lookup>
```

The script to read an entry from the lookup table is as follows:

```
//Get the IP Address associated with entry key XPression.host
var host = ogscript.getPrivateString('hosts',' XPression.host ');
```

getScopedAttribute

Get an attribute in the named scope that has the given ID. Scopes are often internally defined by DashBoard.

Syntax

```
ogscript.getScopedAttribute (Scope Name, Attribute ID);
```

Parameters

Parameter	Type	Required	Description
Scope Name	String	Yes	Name of the scope in which to get and attribute.
Attribute ID	String	Yes	ID of the attribute to get in the named scope.

Returns

Object

Example

Coming soon.

getString

Get a string defined in the global lookup table.

Note: Use the getString function if the lookup table has no ID. If the lookup table has an ID, use the getPrivateString function. For more information about the getPrivateString function, see “**getPrivateString**” on page 2–19.

Syntax

```
ogscript.getString (Key);
```

Parameters

Parameter	Type	Required	Description
Key	String	Yes	Private lookup table from which to get string.

Returns

Object

Example

This example uses the getString function to read an IP address stored in a lookup table.

The lookup table definition for this example is as follows:

```
<lookup>
  <entry key="Tom">television</entry>
</lookup>
```

The script to read an entry from the lookup table is as follows:

```
//Get the string associated with entry key Tom
ogscript.getString('Tom');
```

getTimerManager

Get the timer manager for the context to access timers and perform operations on selected timers.

Syntax

```
ogscript.getTimerManager ();
```

Parameters

Parameter	Type	Required	Description
N/A	N/A	N/A	N/A

Methods

The getTimerManager function is an object that has several methods. The following methods can be run on an existing timer. A timer can be created using the installTimer function or using the graphical editor. for more information about the installTimer function, see “installTimer” on page 1–12.

Method	Parameter Required	Description
isRunning()	N/A	Checks whether the time is running.
startTimer(Boolean reset)	Yes true or false	Starts the timer. If the boolean parameter is set to **true**, the timer resets to the starting time when the function is performed. If the boolean parameter is set to **false**, the function is performed at the timer's current time.
stopTimer(Boolean reset)	Yes true or false	Stops the timer. If the boolean parameter is set to **true**, the timer resets to the starting time when the function is performed. If the boolean parameter is set to **false**, the function is performed at the timer's current time.
resetTimer()	N/A	Resets the timer to the start time.
setStart(Long valueInMilliseconds)	Yes Milliseconds (Long)	Sets the start time of the timer.
setStop(Long valueInMilliseconds)	Yes Milliseconds (Long)	Sets the stop time of the timer.
setTime(Long valueInMilliseconds)	Yes Milliseconds (Long)	Sets the current time of the timer.

Method	Parameter Required	Description
getStart()	N/A	Returns the timer's start time in milliseconds (Long).
getStop()	N/A	Returns the timer's stop time in milliseconds (Long).
getCurrent()	N/A	Returns the timer's current value in milliseconds (Long).
incrementTime(Long difference)	Yes Milliseconds (Long)	Increments the timer value by the specified number of milliseconds
setPattern(String dateTimePattern)	Yes Time format definition	Sets the time format pattern for displaying time values.

Returns

ContextTimerManager

Example 1 — getTimerManager function using isRunning method

```
//verify if timer named 'selftimer' is currently running
if (ogscript.getTimerManager().getTimer('selftimer').isRunning())
   {
     ogscript.debug('running = true');
   }
else
   {
     ogscript.debug('running = false');
   }
```

Example 2 — getTimerManager function using startTimer method

```
//Starts a timer named 'selftimer'
ogscript.getTimerManager().getTimer('selftimer').startTimer(false);
```

Example 3 — getTimerManager function using stopTimer method

```
//Stops a timer named 'selftimer'
ogscript.getTimerManager().getTimer('selftimer').stopTimer(false);
```

Example 4 — getTimerManager function using resetTimer method

```
//Resets a timer named 'selftimer' to the start time
ogscript.getTimerManager().getTimer('selftimer').resetTimer();
```

Example 5 — getTimerManager function using setStart method

```
//Set the start time of a timer named  'selftimer' to 30 seconds
(30000ms)
ogscript.getTimerManager().getTimer('selftimer').setStart(30000);
```

Example 6 — getTimerManager function using setStop method

```
//Set the stop time of a timer named  'selftimer' to two minutes
(120000 ms)
ogscript.getTimerManager().getTimer('selftimer').setStop(120000);
```

Example 7 — getTimerManager function using setTime method

```
//Set the current time of a timer named  'selftimer' to 59 seconds
(59000 ms)
ogscript.getTimerManager().getTimer('selftimer').setTime(59000);
```

Example 8 — getTimerManager function using getStart method

```
// Get the start time of a timer named  'selftimer'
var startTime =
ogscript.getTimerManager().getTimer('selftimer').getStart();
```

Example 9 — getTimerManager function using getStop method

```
// Get the stop time of a timer named  'selftimer'
var stopTime =
ogscript.getTimerManager().getTimer('selftimer').getStop();
```

Example 10 — getTimerManager function using getCurrent method

```
// Get the current time of a timer named  'selftimer'
var currentTIme =
ogscript.getTimerManager().getTimer('selftimer').getCurrent();
```

Example 11 — getTimerManager function using incrementTime method

```
//increase the current time of a timer named  'selftimer' by 30
seconds
ogscript.getTimerManager().getTimer('selftimer').incrementTime(30000)
;


//decrease the current time of a timer named  'selftimer' by 5 seconds
ogscript.getTimerManager().getTimer('selftimer').incrementTime(-5000)
;
```

Example 12 — getTimerManager function using setPattern method

Table 2.2 on page 24 describes the syntax for setting the time format. For some formats, repeating the letter returns more digits or a variation of the format. For example, when specifying M for month, one M shows the month number with no leading zero, two Ms adds a leading zero for months 0 to 9, three Ms shows the three letter month (such as Jan), and four or more Ms shows the full month name (such as January).

Table 2.2 - Time Format Syntax

Letter	Date or Time Component	Presentation	Examples
D	Day	Number	189
H	Hour of the day (0-23)	Number	8
m	Minute of the hour	Number	30
s	Second of the minute	Number	55
S	Millisecond	Number	768
G	Era designator	Text	AD
Y	Year	Number	1969; 69
M	Month of the year	Text or number	September; Sep; 09
w	Week of the year	Number	27
W	Week of the month	Number	3
d	Day of the month	Number	12
F	Day of the week in the month	Number	1 If the day of the week is Tuesday, 1 would denote the first Tuesday of the month
E	Day of the week	Text	Friday; Fri
k	Hour of the day (1-24)	Number	22
K	Hour in AM/PM (0-11)	Number	0
h	Hour in AM/PM (1-12	Number	10
a	AM/PM marker	Text	PM
z	Time zone	General Time Zone	Pacific Standard Time, PST,
Z	Time zone	RFC 822 time zone	-0800

The following code example returns the date and time. An example of the date and time as returned by this example is Sep 30, 2013 2:35:34 PM.

```
//Sets the display format of a timer named 'simpleclock' to show full
date and time
ogscript.getTimerManager().getTimer('simpleclock').setPattern('MMM
dd, yyyy h:mm:ss a');
```

hide

Hide the popup associated with the specified ID.

Note: to use the hide function, a popup must already exist. Popups can be created only in the JavaScript source, not in DashBoard.

Syntax

```
ogscript.hide (Popup ID);
```

Parameters

Parameter	Type	Required	Description
Popup ID	String	Yes	ID of the popup to hide.

Returns

N/A

Example

This example includes two sections of XML code to be added to the .grid file. The first creates a button that opens a popup. The second creates a button that hides the popup.

```
//This example creates a button which, when clicked by a user, opens
the popup area.
<popup id="popup1" left="20" name="Click here to open the Popup"
top="25">
  <abs height="300" left="200" style="bdr:etched;" top="200"
  width="300">
  </abs>
</popup>
```

//This example creates a button which, when clicked by a user, hides the popup.

```
<button buttontype="push" height="50" left="50" name="Click here to
hide the Popup" top="500" width="200">
  <task tasktype="ogscript">ogscript.hide('popup1');</task>
</button>
```

installTimer

Create a timer with the given ID and register it in the ContextTimerManager. Start the timer after the specified delay. If requested, repeat the timer at the specified frequency. When the timer fires, run the specified ogScript function.

Syntax

```
ogscript.installTimer (Timer ID, Repeat, Delay, Repeat Delay, Task);
```

Parameters

Parameter	Type	Required	Description
Timer ID	String	Yes	ID of the timer to create and register in the ContextTimerManager.
Repeat	Boolean	Yes	**true** — repeat the timer using the specified Delay and Repeat Delay. **false** — only run the timer once, do not repeat the timer.
Delay	Long	Yes	Number of milliseconds to wait before starting the timer.
Repeat Delay	Long	Yes	How frequently the associated function runs, in milliseconds.
Task	Function	Yes	ogScript function to run when the timer fires.

Returns

N/A

Example

This example creates a label named "Time" and a button named "Install Timer". When a user clicks the "Install Timer" button, an associated task runs a function named "myFunction()", which creates a timer. It also retrieves the time value every 30 seconds, and loads it into a variable named "str" which is displayed on the "Time" label. The myFunction() function uses the "installTimer" function to create the timer and set the rate at which the time data is updated.

```
<label height="80" id="timeLabel" left="43" name="Time"
style="txt-align:west" top="26" width="275"/>
<button buttontype="push" height="57" left="48" name="Install Timer"
top="133" width="184">
  <task tasktype="ogscript">function myFunction()
    {
      var date = new Date();
      var str = date.getHours() + ':' + date.getMinutes() + ':' +
      date.getSeconds();
      ogscript.rename('timeLabel', 'Time: ' + str);
    }
    //create a timer that starts immediately and runs myFunction
    every 30 seconds (30000 milliseconds)
    ogscript.installTimer('myTimer', true, 0, 30000, myFunction);
  </task>
```

```
</button>
```

isTimerRunning

Report whether or not a timer exists and is in the “running” state.

Syntax

```
ogscript.isTimerRunning (Timer ID);
```

Parameters

Parameter	Type	Required	Description
Timer ID	String	Yes	**true** — a timer with the given ID exits and is in the “running” state. **false** — a timer with the give ID does not exist or is not in the “running” state.

Returns

Boolean

Example

```
//verify if the timer is currently running
var runtime = ogscript.isTimerRunning('selftimer');
```

parseXML

Parse and return an XML document using the org.w3c.dom.Document API. The XML document to parse can be provided in the following ways:

- Piece of well-formatted XML
- URL relative to a CustomPanel
- File URL (file:/c:/…)
- http URL (http://…)

The document is loaded via a blocking call that is run in the DashBoard User Interface thread.

✶ Calls to load documents over a network (for example, using http://) are strongly discouraged and can have undesired impacts on the UI performance.

Syntax

```
ogscript.parseXML (Document);
```

Parameters

Parameter	Type	Required	Description
Document	String	Yes	XML document to parse.

Returns

XML Document

For more information about returns, refer to the following URL:

- http://docs.oracle.com/javase/6/docs/api/org/w3c/dom/Document.html

Example

The following example loads an XML file from the web using an asynchronous http request. An XPath expression extracts data from the XML and displays it on a label.

```
function myFunc(pageContent)
  {
    var xmlPageContent = '&lt;?xml version="1.0"
    encoding="UTF-8"?&gt;\n' + pageContent;
    var document = ogscript.parseXML(xmlPageContent);
    var nodeList =
    ogscript.runXPath('/response/sports/sportsItem/leagues/leaguesIt
    em/teams/teamsItem/name', document);
    var teamList = '&lt;html&gt;';
    ogscript.debug(nodeList.getLength());
    for (var i = 0; i &lt; nodeList.getLength(); i++)
      {
        teamList = teamList + nodeList.item(i).getTextContent() +
        '&lt;br/&gt;';
      }
    ogscript.rename('resultLabel', teamList + '&lt;/html&gt;');
  }
ogscript.asyncPost('http://api.oursports.com/v1/sports/hockey/league/
teams/?_accept=text%6Axml&amp;apikey=ksjdur7euejru47fkbos85kg', null,
myFunc);
```

putObject

You can create an object and reference it in other parts of the code.

Some possible uses include:

- Storing parsed XML data in an object so you don't have to re-parse it.
- Storing the results of an async HTTP post so you don't have to re-fetch it.
- Storing connection code so you cna reference it wherever your code needs to establish that connection.

The putObject function works in conjunction with the getObject function. The putObject function defines the object. The getObject function references the object. The scope of a defined object is global, so you can reference it from anywhere in your panel code.

For information about the getObject function, see "**getObject**" on page 2–18.

Syntax

```
ogscript.putObject(Key, Value);
```

Parameters

Parameter	Type	Required	Description
Key	String	Yes	The name of the object in which the data is being stored.
Value	String	Yes	The value to be stored.

Returns

N/A.

Example

The following example parses and stores data from an XML file in a variable so it can be used globally without the need to re-parse the XML data each time you want to use it.

It defines a function named loadTheXML, which uses the parseXML function to retrieve XML data from a file and load it into a variable named myObject. It then uses the putObject function to copy the data into a variable named myXML. The readTheXML function loads the data into a variable named otherObject.

```
function loadTheXML()
   {
      var myObject = ogscript.parseXML('file:/c:/mydocument.xml');
      ogscript.putObject('myXML',myObject);
   }
function readTheXML()
   {
      var otherObject = ogscript.getObject('myXML');
      // Do anything you want with the data, now contained in the
      otherObject variable.
   }
```

putPrivateString

Add or replace a string in a private lookup table.

Note: Use the putPrivateString function if the lookup table has an ID. If the lookup table has no ID, use the putString function. For more information about the putString function, see “**putString**” on page 2–30.

Syntax

```
ogscript.putPrivateString (Lookup ID, Key, Value);
```

Parameters

Parameter	Type	Required	Description
Lookup ID	String	Yes	ID of the string to create or replace in the specified lookup table.
Key	String	Yes	Private lookup table in which to create or replace the specified string.
Value	String	Yes	New value for the specified string.

Returns

N/A

Example

This example uses the putPrivateString function to replace a datum in a lookup table.

The lookup table definition for this example is as follows:

```
<lookup id="hosts">
  <entry key="XPression.host">10.0.2.210</entry>
  <entry key="XPression.port">9999</entry>
</lookup>
```

The script to replace an entry in the lookup table is as follows:

```
//Replace the port number associated with entry key XPression.host
ogscript.putPrivateString('hosts',' XPression.port ', '7788');
```

putString

Add or replace a string in the global lookup table.

Note: Use the putPrivateString function if the lookup table has no ID. If the lookup table has an ID, use the putPrivateString function. For more information about the putPrivateString function, see "**putPrivateString**" on page 2–29.

Syntax

```
ogscript.putString (Lookup ID, Value);
```

Parameters

Parameter	Type	Required	Description
Lookup ID	String	Yes	ID of the string to create or replace in the global lookup table.
Value	String	Yes	New value for the specified string.

Returns

N/A

Example

This example uses the putString function to replace a datum in a lookup table.

The lookup table definition for this example is as follows:

```
<lookup>
  <entry key="Tom">television</entry>
</lookup>
```

The script to replace an entry in the lookup table is as follows:

```
//Replace the string associated with entry key Tom
ogscript.putString('Tom','telephone');
```

reload

Rebuild the user interface element with the specified ID. If the ID is for an `<include>` tag, re-fetch the included document before rebuilding the user interface.

Syntax

```
ogscript.reload (User Interface ID);
```

Parameters

Parameter	Type	Required	Description
User Interface ID	String	Yes	ID of the user interface element to rebuild.

Returns

N/A

Example

In this example, the reload function is used to rebuild a drop-down list to show new options.

```
//create a new array of colours
var color = new Array("Red","Green","Blue");

//populate the dropdown color_list with the color array
params.replaceIdentifiedConstraint('color_list',
params.createIntChoiceConstraint(color));

//reload the dropdown list to view the new options
ogscript.reload('color_list');
```

rename

Modify the text associated with a tab name, label, or button. Use the Component ID to specify the component to rename, do not use the Object ID (OID).

To view the ID of a component, double-click the component in **PanelBuilder** to open the **Edit Component** dialog box. The **ID** box displays the ID of the selected component.

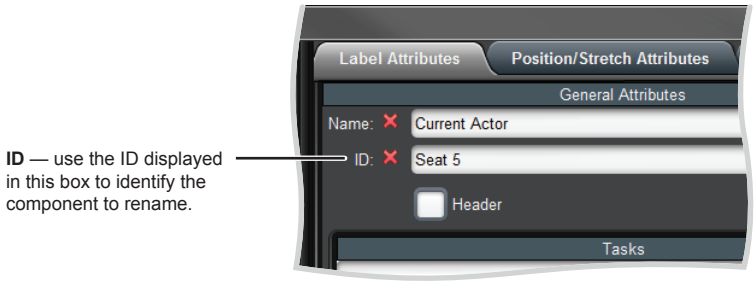


Figure 2.1 *Component ID in the Edit Component dialog box*

Syntax

```
ogscript.rename (Component ID, Name);
```

Parameters

Parameter	Type	Required	Description
Component ID	String	Yes	ID of the user interface component to rename.
Name	String	Yes	New text to display on the screen for the specified user interface component.

Returns

N/A

Example 1

```
// Set the item with ID='Seat 5' to have the text 'Mika Andersen'
ogscript.rename ('Seat 5','Mika Andersen');
```

Example 2

```
// Read the value of a parameter into a variable named data
var data = params.getValue(0x12,0);
// Use the variable named data to make a new ID and set the ID to have
the text 'Mika Andersen'
ogscript.rename('Seat ' + data,'Mika Andersen');
```

reposition

Moves a component to an absolute position, defined as an X - Y pixel position.

Alternatively, you can specify a component's position by percentage of the container's width and height. For more information, see "**repositionByPercent**" on page 2–33.

Syntax

```
ogscript.reposition (ID, x position, y position);
```

Parameters

Parameter	Type	Required	Description
ID	String	Yes	ID of the component you want to reposition
x position	Integer	Yes	Number of pixels from the left
y position	Integer	Yes	Number of pixels from the right

Returns

N/A

Example

In this example, the task associated with the "Top Left" button uses the ogscript.reposition function to reposition a label.

```
<label height="40" id="myLabel" left="160" name="myLabel"
style="txt-align:center" top="100" width="160"/>
<button buttontype="push" height="40" left="160" name="Top Left"
top="200" width="160">
```

```
    <task tasktype="ogscript">ogscript.reposition('myLabel', 0, 0);
    </task>
</button>
```

repositionByPercent

Moves a component to an absolute position, defined as a percentage of container width and height.

Alternatively, you can specify a component's position by pixel. For more information, see "**reposition**" on page 2–32.

Syntax

```
ogscript.repositionByPercent (ID, percent x, percent y);
```

Parameters

Parameter	Type	Required	Description
OID	String	Yes	OID of the component you want to reposition
x percent	Integer	Yes	Distance from the left, as a percentage of container width
y percent	Integer	Yes	Distance from the top, as a percentage of container height
center x	Boolean	Yes	**true** — Shows the full width of the object. **false** — Crops the object if it extends beyond the horizontal boundaries of the container.
center y	Boolean	Yes	**true** — Shows the full height of the object. **false** — Crops the object if it extends beyond the vertical boundaries of the container.

Returns

N/A

Example

In this example, the task associated with the **"One Quarter"** button uses the ogscript.reposition function to reposition a label 25% from the left, and 25% from the top. Centering is set to **false** in both the x and y axes, so if the label overhangs the edges of the container the overhanging portion is not shown.

```
<label height="41" id="myLabel" left="160" name="myLabel"
style="txt-align:center" top="101" width="160"/>
<button buttontype="push" height="40" left="160" name="One Quarter"
top="200" width="159">
    <task tasktype="ogscript">ogscript.repositionByPercent('myLabel',
    25, 25, false, false);
    </task>
</button>
```

reveal

Open a popup with the specified ID, or bring the tab with the specified ID to the foreground.

This function is especially useful for tab sets that have their placement set to the center, meaning that there are no tabs showing for users to click. Using the reveal function is the only way to display the specified tab.

Syntax

```
ogscript.reveal (User Interface ID);
```

Parameters

Parameter	Type	Required	Description
User Interface ID	String	Yes	ID of the popup to open or the tab to bring to the foreground.

Returns

N/A

Example

This example includes a definition for a set of tabs with its position set to center, and uses the reveal function to select a particular tab to be shown.

Tip: When tab position is set to center, tabposition="none" in the tab set's XML source code.

```
<tab height="91" left="580" tabposition="none" top="373" width="221">
   <abs id="page1" name="Tab 1"/>
   <abs id="page2" name="Tab 2"/>
   <abs id="page3" name="Tab 3"/>
</tab>

//Select Tab2
ogscript.reveal('page2');
```

runXPath

Execute the given XPath command on the given XML Document or XML Element and return the results as a NodeList.

```
ogscript.runXPath (XPath, Document);
```

or

```
ogscript.runXPath (XPath, Element);
```

Parameters

Parameter	Type	Required	Description
XPath	String	Yes	The XPath command to execute on the given XML Document or XML Element
Document	String	Yes	XML Document on which to execute the given XPath command.
Element		Yes	XML Element on which to execute the given XPath command.

For more information about the required parameters, refer to the following URLs:

- http://docs.oracle.com/javase/6/docs/api/org/w3c/dom/Document.html
- http://docs.oracle.com/javase/6/docs/api/org/w3c/dom/Element.html
- http://docs.oracle.com/javase/6/docs/api/org/w3c/dom/NodeList.html
- http://www.w3schools.com/xpath/

Returns

NodeList

Example

Coming soon.

saveToFile

Saves data to a file. This function is typically used to save a byte array, string, or XML document to a file.

Syntax

```
ogscript.saveToFile(path, data, overwrite);
```

Parameters

Parameter	Type	Required	Description
path	String	Yes	The directory path to the destination file.
data	String, byte[], or XML	Yes	The data to be saved to file.
overwrite	Boolean	Yes	When **`true`**, existing file of the same name is overwritten. When **`false`**, existing file of the same name is not overwritten.

Returns

Returns **`true`**, if data is written successfully; otherwise **`false`**.

Example

```
ogscript.saveToFile('files/my-new-file.txt','This is my data',true);
```

sendUDPAsBytes

Converts ASCII string data to a byte array, and sends it as bytes to the specified host/port through UDP.

The ASCII data is converted to Hexadecimal bytes, and can consist only of the following characters:

- 0 to 9
- A to F
- Spaces and commas (as delimiters)

Syntax

```
ogscript.sendUDPAsBytes (Host, Port, Data);
```

Parameters

Parameter	Type	Required	Description
Host	String	Yes	Host name to send the given data through UDP.
Port	Integer	Yes	Port number on the given host to be sent given data through UDP.
Data	ASCII string	Yes	Data to be converted to bytes and sent through UDP to the specified host/port.

Returns

N/A

Example

```
ogscript.sendUDPAsBytes (myComputer, 7788,'7A, 3C, FF');
```

sendUDPBytes

Send the given data bytes to the specified host/port through UDP.

Syntax

```
ogscript.sendUDPBytes (Host, Port, Data);
```

Parameters

Parameter	Type	Required	Description
Host	String	Yes	Host name to send the given Data bytes through UDP.
Port	Integer	Yes	Port number on the given Host to send given Data byte through UDP.
Data	Byte	Yes	Data bytes to send through UDP to the given Host and Port.

Returns

N/A

Example

Coming soon.

sendUDPString

Converts a string to UTF-8 bytes and send the bytes to the provided host/port through UDP.

Syntax

```
ogscript.sendUDPString (Host, Port, Data);
```

Parameters

Parameter	Type	Required	Description
Host	String	Yes	Host name to send the given Data string through UDP.
Port	Integer	Yes	Port number on the given Host to send given Data string through UDP.
Data	String	Yes	Data string to convert to bytes and send through UDP to the given Host and Port.

Returns

N/A

Example

This example uses the sendUDPString function to send a message to a particular host/port.

```
var host = ogscript.getPrivateString('hosts',' Panel.host ');
var port = parseInt(ogscript.getPrivateString('hosts',' Panel.port
'));
var message = "Hello, can you hear me?";

ogscript.sendUDPString(host,port,message);
```

serializeXML

Serializes an XML document and converts it to a string.

Syntax

```
ogscript.serializeXML([document]);
```

Parameters

Parameter	Type	Required	Description
document	XML object variable	Yes	XML document to be converted to a string.

Returns

Returns a string that is a serialized version of the original XML document.

Example

In this example, the variable `myXML` is loaded with the contents of the XML document referenced by the variable `sourceXML`.

```
var myXML = ogscript.serializeXML(sourceXML);
```

setAnchorPoints

Specifies how an object moves if the user interface is resized for different monitor and window sizes. Anchor points are relative to the container in which they are located (for example, a tab, a split pane, etc.).

The setAnchorPoints function allows you to anchor or release an object to/from the top, left, bottom, or right sides. By setting these values, you can effectively anchor an object to a corner, a side, or the center.

Syntax

```
ogscript.setAnchorPoints (ID, top, left, bottom, right);
```

Parameters

Parameter	Type	Required	Description
ID	String	Yes	ID of the object you want to anchor.
top	Boolean	Yes	**true** — object is anchored to the top **false** — object is not anchored to the top
left	Boolean	Yes	**true** — object is anchored to the left **false** — object is not anchored to the left
bottom	Boolean	Yes	**true** — object is anchored to the bottom **false** — object is not anchored to the bottom
right	Boolean	Yes	**true** — object is anchored to the right **false** — object is not anchored to the right

Returns

N/A

Example

The button in this example has a task that anchors an object (with ID `'dialog'`) to the top left.

```
<button buttontype="push" name="anchorTopLeft">
  <task tasktype="ogscript">ogscript.setAnchorPoints('dialog', true,
  true, false, false);
  </task>
</button>
```

setStyle

Set Style parameters for the component with the given ID if it exists. Style commands are additive. They can be added or modified, but not removed.

Tip: To view syntax examples for a particular styles, use the PanelBuilder user interface to add the style on the **Style** tab, and then view the resulting code in the **Source** tab.

For openGear Style Hints for the available style options, refer to the openGear documentation.

Syntax

```
ogscript.reveal (Component ID, Style);
```

Parameters

Parameter	Type	Required	Description
Component ID	String	Yes	ID of the Component to style with the given Style parameters.
Style	String	Yes	Style parameters with which to style the given Component.

Returns

N/A

Example 1

This example defines the style of a label, and then makes three style changes.

```
//label definition
<label height="45" id="label1" left="330" name="Change the style of
this label" style="txt-align:west;" top="100" width="325"/>

//first change - set the background to red
ogscript.setStyle('label1',"bg#FF0000");

//second change - set the text colour to black and text size to big
ogscript.setStyle('label1',"fg#000000;size:big");

//third change - modify the text alignment from left to right
ogscript.setStyle('label1',"txt-align:east");
```

Example 2

This example creates a pre-defined style, and applies it to a component. Pre-defined styles can add or replace a component's style settings, but not remove them.

```
//create a pre-defined style
<style id="Style1" name="Style1"
value="size:Big;bg#6F63FB;bdr:etched;"/>

//Add a predefined Style to a component
ogscript.setStyle('label1',"style:Style1")
```

setXML

Dynamically generates UI components through ogscript. Replaces the contents of an element with a string of XML code.

Syntax

`ogscript.Xml`*(ID, new XML content)*

Parameters

Parameter	Type	Required	Description
ID	String	Yes	ID for the component in which you want to replace XML
new XML content	String	Yes	The new XML content

Returns

N/A

Example

This example includes a label with text. When the user clicks the label, the associated task uses ogscript.setXml to replace the text.

```
<abs id="0x4">
  <label height="59" id="0x2" left="61" name="This Text Will Be
  Replaced" style="txt-align:center" top="40" width="238"/>
</abs>
<button buttontype="push" height="40" id="0x3" left="59"
name="replaceText" top="121" width="240">
  <task tasktype="ogscript">ogscript.setXml('0x4', '&lt;label
  height="59" id="0x2" left="61" name="This is the New Text"
  style="txt-align:center" top="40" width="238"/&gt;');
  </task>
</button>
```

toBottom

Displays the object below all others in the same container. Object display is layered. If objects overlap, higher layers are drawn over lower layers.

Syntax

```
ogscript.toBottom (ID);
```

Parameters

Parameter	Type	Required	Description
ID	String	Yes	ID object to be sent to the bottom

Returns

N/A

Example

This example includes two labels occupying the same position. LabelOne is defined second in the code, so it appears on top and is therefore visible. Button One runs a task that uses ogscript.toBottom to send Label One to the bottom of the stack. This makes Label Two visible. Button Two sends Label Two to the bottom.

```
<abs>
  <label height="317" id="labelTwo" left="100" name="Label Two"
  style="size:Biggest;bg#D92648;txt-align:center;" top="100"
  width="350"/>
```

```
  <label height="317" id="labelOne" left="100" name="Label One"
  style="size:Biggest;bg#selectbg;txt-align:center;" top="100"
  width="350"/>
  <button buttontype="push" height="40" id="oneBottom" left="150"
  name="Button One" style="bg#selectbg;txt-align:center;" top="450"
  width="100">
    <task tasktype="ogscript">ogscript.toBottom('labelOne');
    </task>
  </button>
  <button buttontype="push" height="40" id="twoBottomn" left="300"
  name="Button Two" style="bg#D92648;txt-align:center;" top="450"
  width="100">
    <task tasktype="ogscript">ogscript.toBottom('labelTwo');
    </task>
  </button>
</abs>
```

toTop

Displays the object above all others in the same container. Object display is layered. If objects overlap, higher layers are drawn over lower layers.

Syntax

```
ogscript.toTop (ID);
```

Parameters

Parameter	Type	Required	Description
ID	String	Yes	ID object to be sent to the top

Returns

N/A

Example

This example includes two labels occupying the same position. LabelTwo is defined second in the code, so it appears on top and is therefore visible. Button One runs a task that uses ogscript.toTop to send Label One to the top of the stack. This makes Label One visible. Button Two sends Label Two to the top.

```
<abs>
  <label height="317" id="labelOne" left="100" name="Label One"
  style="size:Biggest;bg#selectbg;txt-align:center;" top="100"
  width="350"/>
  <label height="317" id="labelTwo" left="100" name="Label Two"
  style="size:Biggest;bg#D92648;txt-align:center;" top="100"
  width="350"/>
  <button buttontype="push" height="40" id="oneTop" left="150"
  name="Button One" style="bg#selectbg;txt-align:center;" top="450"
  width="100">
    <task tasktype="ogscript">ogscript.toTop('labelOne');
    </task>
  </button>
```

```
    <button buttontype="push" height="40" id="twoTop" left="300"
    name="Button Two" style="bg#D92648;txt-align:center;" top="450"
    width="100">
      <task tasktype="ogscript">ogscript.toTop('labelTwo');
      </task>
    </button>
  </abs>
```

upload

Open the **File Upload** dialog with the specified file.

Syntax

```
ogscript.upload (Filename);
```

Parameters

Parameter	Type	Required	Description
Filename	String	Yes	Name of the file with which to open the File Upload dialog box.

Returns

N/A

Example

Coming soon.

params Object

In ogScript, use the **params** object to access functions to interact with openGear Device parameters and constraints. The params object is also used to manipulate parameters stored in the .grid file.

The params object is accessible when a CustomPanel is associated with an openGear device or XML data file (.grid file). Scripts referencing a device must follow beneath the referenced device in the XML hierarchy.

To call an openGear Device function, use:

```
params.function name  (parameters);
```

For example:

```
var data = params.getValue (0x12, 0);
```

params Functions

The params object contains the following functions:

Table 3.1 params functions

Function	Parameters	Returns	Description
createIntChoiceConstraint	[choices]		Creates a choice constraint (which is a set of key/value pairs) for use in toggle buttons, combo box, radio buttons, etc. The choice constraint you create here can be used to replace a constraint for a parameter.
getConstraint	String [OID]	Constraint	Get the constraint from the parameter with the specified OID.
getIdentifiedConstraint	String [ID]	Constraint	Get the constraint with the specified ID.
getParam	Integer [Context ID] String [OID], Integer [Index]	ParamScriptable	Get the information about an element in a parameter with the specified OID.
getParam (OID, Index).remove	N/A		Removes a parameter element. If the parameter is an array with more than one element, the element at the index location is removed.
getStrValue	String [OID], Integer [Index]	String	Get a string representation of an element in a parameter with the specified OID.
getValue	String [OID], Integer [Index]	String	Get the value of a parameter with the specified OID.
isPrivateParamContext	N/A	Boolean	Returns **true** if local OGLML-based parameters are operating disconnected from the real device.
replaceConstraint	String [OID], String [Constraint ID]	N/A	Replace the constraint for the parameter with the specified OID with the constraint with the specified constraint ID.
replaceIdentifiedConstraint	String [Destination ID] String [Source ID]	N/A	Replace (or create) the constraint identified by the specified destination ID with the constraint identified by the specified source ID.
setAccess	String [OID], Integer [Access]	N/A	Set the access level of the parameter with the provided OID.
setAllValues	OID Object[] Values		For an array parameter, replaces the current array with the new array.

Table 3.1 params functions

Function	Parameters	Returns	Description
setMenuState	Integer [Static Menu ID], Integer [Menu State]	N/A	Set the menu state of the menu with the specified static menu ID.
setPrivateParamContext	Boolean [Value]	N/A	**true** — disconnect parameters defined in the OGLML document from the device. **false** — re-connect parameters defined in the OGLML document from the device.
setValue	String [OID], Integer [Index], Object [Value]	N/A	Set the value of an element in a parameter with the provided OID to the provided value.
setValueRelative	String [OID], Integer [index], Integer [change in value]	N/A	Changes the value of a parameter. If the value is a string, it is replaced. If it is a float or int, the specified value is added to the current value.
toOid	String (OID)	N/A	Creates an OID object.

createIntChoiceConstraint

Syntax

```
params.createIntChoiceConstraint (Choices);
```

Parameters

Parameter	Type	Required	Description
Choices	String	Yes	Name of the array variable that contains the choices.

Returns

N/A

Example

Coming soon.

getConstraint

Get the constraint from the parameter with the specified Object ID.

Syntax

```
params.getConstraint (OID);
```

Parameters

Parameter	Type	Required	Description
OID	String	Yes	Object ID of the object of interest.

Returns

Constraint

Example

Coming soon.

getIdentifiedConstraint

Get the constraint with the specified ID. If the ID is an external object URL, get the constraint defined in the specified external object.

Syntax

```
params.getIdentifiedConstraint (ID);
```

Parameters

Parameter	Type	Required	Description
ID	String	Yes	ID of the constraint of interest.

Returns

String

Example

Coming soon.

getParam

Gets information about an element in the parameter with the specified Object ID.

Syntax

```
params.getParam (Context ID, OID, Index);
```

Parameters

Parameter	Type	Required	Description
Context ID	String	No	The context ID of the component that contains the parameter of interest.
OID	String	Yes	Object ID of the object of interest.
Index	Integer	Yes	Array parameter index. If the parameter is not an array parameter, use an *Index* of 0. In most cases, enter 0 as the *Index*.

Returns

ParamScriptable

Example

Coming soon.

getParam (OID, Index).remove

Removes a parameter element. If the parameter is an array with more than one element, the element at the index location is removed.

Syntax

```
params.getParam ([oid], [index]).remove();
```

Parameters

Parameter	Type	Required	Description
OID	String or Integer	Yes	OID can be a string or an integer, depending on how the parameter is defined.
Index	Integer	Yes	Array parameter index. If the parameter is not an array parameter, use an *Index* of 0.

Returns

N/A

Example

Coming soon.

getStrValue

Get a string representation of an element in a parameter with the specified Object ID.

Syntax

```
params.getStrValue (OID, Index);
```

Parameters

Parameter	Type	Required	Description
OID	String	Yes	Object ID of the object of interest.
Index	Integer	Yes	Array parameter index. If the parameter is not an array parameter, use an *Index* of 0. In most cases, enter 0 as the *Index*.

Returns

ParamScriptable

Example

Coming soon.

getValue

Get the value of a parameter with the specified Object ID.

Syntax

```
params.getValue (OID, Index);
```

Parameters

Parameter	Type	Required	Description
OID	String	Yes	Object ID of object of interest.
Index	Integer	Yes	Array parameter index. If the parameter is not an array parameter, use an *Index* of 0. In most cases, enter 0 as the *Index*.

Returns

String

Example

```
var data = params.getValue (0x12,0);
```

isPrivateParamContext

Returns **true** when the local OGLML-based parameters are operating disconnected from a real device. Changes and values are not sent to or fetched from the device if the parameter is defined in the OGLML document.

Syntax

```
params.isPrivateParamContext ();
```

Parameters

Parameter	Type	Required	Description
N/A	N/A	N/A	N/A

Returns

Boolean

Example

Coming soon.

replaceConstraint

Replace the constraint for the parameter with the specified Object ID with the constraint with the specified constraint ID. If the ID is an external object URL, replace the constraint with the constraint specified by the external object.

Syntax

```
params.replaceConstraint (OID, Constraint ID);
```

Parameters

Parameter	Type	Required	Description
OID	String	Yes	Object ID of object of interest.
Constraint ID	String	Yes	ID of the constraint with which to replace the constraint for the parameter with the specified Object ID.

Returns

N/A

Example

Coming soon.

replaceIdentifiedConstraint

Replace or create the constraint identified by the specified destination ID with the constraint specified by the provided source ID.

Syntax

```
params.replaceIdentifiedConstraint (Destination ID, Source ID);
```

Parameters

Parameter	Type	Required	Description
Destination ID	String	Yes	ID of the constraint to replace or create.
Source ID	String	Yes	ID of the constraint with which to replace or create the constraint specified by the Destination ID.

Returns

N/A

Example

Coming soon.

setAccess

Set the access level of the parameter with the specified Object ID.

Syntax

```
params.setAccess (OID, Access);
```

Parameters

Parameter	Type	Required	Description
OID	String	Yes	Object ID of object of interest.
Access	Integer	Yes	Access level to set for the specified OID. The available access levels are as follows: • **0** — Read Only • **1** — Read and Write

Returns

N/A

Example

Coming soon.

setAllValues

For an array parameter, replaces the current array with a new array.

Syntax

```
params.setAllValues([oid], [array]);
```

Parameters

Parameter	Type	Required	Description
OID	String	Yes	The OID of the parameter.
Array	String	Yes	The new array.

Returns

N/A

Example

Coming soon.

setMenuState

Set the menu state of the menu with the provided static menu ID.

Syntax

```
params.setMenuState (Static Menu ID, Menu State);
```

Parameters

Parameter	Type	Required	Description
Static Menu ID	Integer	Yes	ID of the menu of interest.
Menu State	Integer	Yes	Menu state to set for the specified Static Menu ID. The available menu states are as follows: • **0** — Hidden • **1** — Disabled • **2** — Normal

Returns

N/A

Example

Coming soon.

setPrivateParamContext

Control the context between the parameters defined in the OGLM document and a device. This function has no impact on parameters that are only defined on the device or only defined in the OGLML document.

Syntax

```
params.setPrivateParamContext (Value);
```

Parameters

Parameter	Type	Required	Description
Value	Boolean	Yes	The available contexts are as follows: • **true** — disconnect parameters defined in the OGLML document from the device. • **false** — re-connect parameters defined in the OGLML document from the device.

Returns

N/A

Example

Coming soon.

setValue

Set the value of a parameter for the provided Object ID.

Syntax

```
params.setValue (OID, Index, Value);
```

Parameters

Parameter	Type	Required	Description
OID	String	Yes	Object ID of object of interest.
Index	Integer	Yes	Array parameter index. If the parameter is not an array parameter, use an *Index* of 0. In most cases, enter 0 as the *Index*.
Value	Object	Yes	New value for the OID.

Returns

N/A

Example 1

```
// Set the parameter to 3:
params.setValue (0x12,0,3);
```

Example2

```
// Set the value to 3 greater than it was.
var data = getValue (0x12,0);
params.setValue (0x12,0,data + 3);
```

setValueRelative

Increments or decrements a numeric value by a specified amount.

Syntax

```
params.setValueRelative (OID, Index, Change in value);
```

Parameters

Parameter	Type	Required	Description
OID	String or Integer	Yes	The OID of the object of which you want to change the value.
Index	Integer	Yes	Position of data in the parameter.
Change in Value	Integer	Yes	Amount by which the value is incremented. To decrement the value, use a negative integer.

Returns

N/A

Example

Coming soon.

toOid

Creates an OID object.

Syntax

```
params.toOid (OID);
```

Parameters

Parameter	Type	Required	Description
OID	String	Yes	The value of the new OID object.

Returns

N/A

Example

This example is a function that uses the **`toOid`** function to create an OID with the value **`'my.special.oid'`**, then uses the **`getOid`** function to return the OID value.

```
function lookForSpecificOid(myParam)
   {
     var myOID = params.toOid('my.special.oid');
     return myParam.getOid() == myOID;
   }
```

rosstalk Object

In ogScript, use the **rosstalk** object to communicate over the network to other devices that speak RossTalk protocol. Functions in the rosstalk object are typically set through a user interface.

To call a general purpose function, use:

```
rosstalk.function name (parameters);
```

For example:

```
rosstalk.setHost (Server01);
```

Configuring DashBoard to listen for RossTalk GPI Commands

DashBoard can be configured to listen for RossTalk GPI commands. Scripts in DashBoard CustomPanels can interpret RossTalk GPI command messages and act on them. You can send GPI trigger text using the **`rosstalk.sendMessage`** function. When a panel component with a matching GPI trigger receives the message, that component's tasks are executed.

Tip: For more information about triggering tasks externally, see the ***DashBoard User Guide (8351DR-004)***.

To Configure DashBoard to Listen for RossTalk GPI Commands

1. In DashBoard, on the **Window** menu, click **Preferences**.

   The **Preferences** dialog appears.

2. In the menu list on the left side of the **Preferences** dialog, click **Smart GPI Listener**.

   The **Smart GPI Listener** settings appear.

3. Beside **Global GPI Listener**, select **Enable**.

4. In the **Listen Port** box, specify the port for receiving RossTalk GPI commands.

   Tip: The default port is **7788**.

5. Click **OK**.

rosstalk Functions

Table 4.1 rosstalk functions

Function	Parameters	Returns	Description
setHost	String [Host]	N/A	Set a default host to use for RossTalk commands where no host has been defined.
getHost	N/A	String	Get the default host previously defined.
setPort	Integer [Port]	N/A	Set a default port to use for RossTalk commands where no host has been defined.
getPort	N/A	Integer	Get the default port previously defined.
sendAsBytes	String [Host], Int [Port], String [Bytes as Hex String]	N/A	Equivalent of calling: sendAsBytes(host, port, bytes, null);

Table 4.1 rosstalk functions

Function	Parameters	Returns	Description
sendAsBytes	String [Host], Int [Port], String [Bytes as Hex String], Function [Callback]	N/A	Convert bytes from string (where string is formatted as ASCII representations of bytes e.g. "FDDFEAAE12F9…") and send them to the provided host at the provided port. Invoke the callback function when done.
sendBytes	String [Host], Int [Port], Byte[] [Data to Send], Function [Callback]	N/A	Send the provided bytes to the provided host at the provided port. Invoke the callback function when done.
sendMessage	String [RossTalk Command]	N/A	Equivalent of calling: sendMessage (getHost(), getPort(), RossTalk Command, null);
sendMessage	String [RossTalk Command], Function [Callback]	N/A	Equivalent of calling: sendMessage (getHost(), getPort(), RossTalk Command, Callback);
sendMessage	String [Host], Int [Port], String [RossTalk Command]	N/A	Equivalent of calling: sendMessage (Host, Port, RossTalk Command, null);
sendMessage	String [Host], Int [Port], String [RossTalk Command] Function [Callback]	N/A	Send the provided string as UTF-8 followed by CRLF bytes to the provided host at the provided port. Invoke the callback function when done.

robot Object

In ogScript, use the **robot** object to communicate with CamBot robotic cameras through the CamBot PC User Interface. Functions in the robot object are typically set through a user interface.

To call a general-purpose function, use:

```
robot.function name (parameters);
```

For example:

```
robot.setHost (Server01);
```

robot Functions

Table 5.1 robot functions

Function	Parameters	Returns	Description
setHost	String [Host]	N/A	Set a default host to use for CamBot commands where no host has been defined.
getHost	N/A	String	Get the default host previously defined.
setPort	Integer [Port]	N/A	Set a default port to use for CamBot commands where no host has been defined.
getPort	N/A	Integer	Get the default port previously defined.
sendCambot	String [CamBot Command]	N/A	Equivalent of calling: sendCambot (getHost(), getPort(), command, null)
sendCambot	String [CamBot Command] Function [Callback]	N/A	Equivalent of calling: sendCambot (getHost(), getPort(), CamBot Command, Callback);
sendCambot	String [Host], Int [Port], String [CamBot Command]	N/A	Equivalent of calling: sendCambot (Host, Port, CamBot Command, null);
sendCambot	String [Host], Int [Port], String [CamBot Command] Function [Callback]	N/A	Send the provided CamBot command to the provided host at the provided port. Invoke the callback function when done. Callback function signature: Function (Boolean success, String sentData, String receivedData, Exception javaException)

vdcp Object

In ogScript, use the **vdcp** object to communicate with BlackStorm video servers. Functions in the vdcp object are typically set through a user interface.

To call a general-purpose function, use:

```
vdcp.function name  (parameters);
```

For example:

```
vdcp.setHost (Server01);
```

vdcp Functions

Table 6.1 vdcp functions

setHost	String [Host]	N/A	Set a default host to use for VDCP commands where no host has been defined.
getHost	N/A	String	Get the default host previously defined.
setPort	Integer [Port]	N/A	Set a default port to use for VDCP commands where no host has been defined.
getPort	N/A	Integer	Get the default port previously defined.
activeClip	String [Host], Int [Port], Int [Channel], Function [Callback]	N/A	Fetch the active clip ID for the provided channel from the server at the provided host/port. Invoke the callback with the active clip ID when done. Callback function signature: Function (Boolean success, String sentCommand, String resultString, Exception javaException)
clipDuration	String [Host] Int [Port] String [ClipID] Function [Callback]	N/A	Fetch the duration [HH:MM:SS:FF] of the clip with the given ID. Invoke the callback with the clip duration when done. Callback function signature: Function (Boolean success, String sentCommand, String resultString, Exception javaException)
cueClip	String [Host], Int [Port], Int [Channel]	N/A	
cueClip	String [Host], Int [Port], Int [Channel], Function [Callback]	N/A	

Table 6.1 vdcp functions

fastForward	String [Host], Int [Port], Int [Channel]	N/A	
fastForward	String [Host], Int [Port], Int [Channel], Function [Callback]	N/A	
listClips	String [Host], Int [Port], Function [Callback]	N/A	
pause	String [Host], Int [Port], Int [Channel]	N/A	
pause	String [Host], Int [Port], Int [Channel], Function [Callback]	N/A	
rewind	String [Host], Int [Port], Int [Channel]	N/A	
rewind	String [Host], Int [Port], Int [Channel], Function [Callback]	N/A	
stop	String [Host], Int [Port], Int [Channel]	N/A	
stop	String [Host], Int [Port], Int [Channel], Function [Callback]	N/A	

nkscript Object

In ogScript, use the **nkscript** object to control NK Router OGLML tags used in Switchboard virtual control panels. Functions in the nkscript object are typically set through a user interface.

- The nkscript global object is only accessible in OGLML contexts that have been declared as having the **NK Router** context type or are beneath such a context in the OGML document hierarchy.

To call a general-purpose function, use:

```
nkscript.function name (parameters);
```

For example:

```
nkscript.setHost (Server01);
```

nkscript Functions

Table 7.1 nkScript functions

Function	Parameters	Returns	Description
doSwitch	N/A	Boolean	Equivalent of calling: doSwitch(getActiveDst(), getActiveSrc(), getLevelMask());
doSwitch	Int [Dst], Int [Src], Long [Levels]	Boolean	Do a switch on the active IPS to route the given dst to the given src on the given levels.
getActiveDst	N/A	Int	Get the active dst number (0-indexed). Returns -1 if there is no active destination.
getActiveDstName	N/A	String	Get the name of the active dst (from the switchboard configuration). Returns null if there is no active destination.
getActiveIPS	N/A	String	Get the serial number of the active IPS
getActiveIPSName	N/A	String	Get the name of the active IPS
getActiveSrc	N/A	Int	Get the active src number (0-indexed). Returns -1 if there is no active source.
getActiveSrcName	N/A	String	Get the name of the active src (from the switchboard configuration). Returns null if there is no active source.
getLevelMask	N/A	Long	get the current level mask (as a bit field)
isLevelActive	Int [Level Num]	Boolean	Is the current level active. Equivalent to asking: levelMask & (1 << levelNum) != 0;
isMCFlag	N/A	Boolean	Is the Machine Control flag set.
isProtected	N/A	Boolean	Is the active destination protected.
isProtectedByMe	N/A	Boolean	Is the active destination protected by this virtual panel.

Table 7.1 nkScript functions

Function	Parameters	Returns	Description
isSrcActive	Int [Src]	Boolean	Is the given source active on the active destination any level.
isSrcActive	Int [Dst], Int [Src], Long [Levels]	Boolean	Is the given source active on the given destination on the given level mask.
isVirtual	N/A	boolean	Is virtual routing in use (for switch commands and status requests).
setActiveDst	Int [Dst]	Boolean	Set the active destination (0-indexed).
setActiveIPS	String [Serial]	boolean	Set the IPS with the given serial number as the active IPS to receive commands and send status. Deactivate any currently active IPS.
setActiveSrc	Int [Src]	N/A	Set the active source (0-indexed).
setLevelActive	Int [Level Num]	boolean	Set the given level as active.
setLevelMask	Long [Level Mask]	N/A	Set the complete level mask bitfield.
setMCFlag	Boolean	Boolean	Set the Machine Control flag to true or false.
setProtected	Boolean	Boolean	Request the router to protect the active destination.
setVirtual	Boolean	Boolean	Set virtual routing on/off for switch commands and status requests.
verifyConfiguration	N/A	Boolean	Re-activate the current IPS.