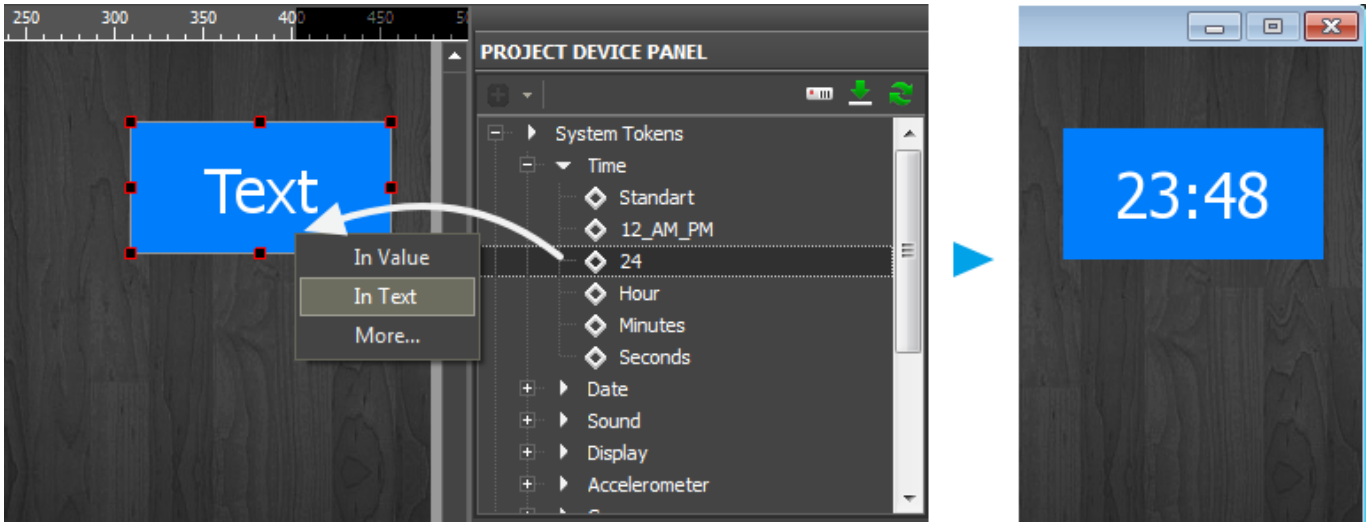
Contents

System Tokens

DOWNLOAD: Example of a project

System Token – it returns system parameters of the control panel (time, date, GPS coordinates, etc.). You can assign tokens to any item of your graphic interface. They will output information received from the system on the item. You cannot write interface values in system tokens. They can be changed by the system only.

In order to assign a system token to a graphic item, use the Drag&Drop method. There is also an alternative way of assigning tokens – using scripts.



Functions of system tokens with examples of their activation using scripts:

Get Time

- **"System.Time.Seconds"** – system time, seconds

```
IR.GetVariable("System.Time.Seconds")
```

- **"System.Time.Minutes"** - system time, minutes

```
IR.GetVariable("System.Time.Minutes")
```

- **"System.Time.Hour"** - system time, hours

```
IR.GetVariable("System.Time.Hour")
```

- **"System.Time.24"** - system time in the 24-hour format

```
IR.GetVariable("System.Time.24")
```

- **"System.Time.12_AM_PM"** - system time in the 12-hour format

```
IR.GetVariable("System.Time.12_AM_PM")
```

- **"System.Time.Standart"** - system time in the format for OS

```
IR.GetVariable("System.Time.Standart")
```

```
IR.AddListener(IR.EVENT_START, 0, function()
{
var time = IR.GetVariable("System.Time.Seconds");
IR.Log(time);
```

```
});
```

Get Date

- **"System.Date.DayOfYear"** – system date, the day of the year

```
IR.GetVariable("System.Date.DayOfYear")
```

- **"System.Date.Year"** - system date, the current year

```
IR.GetVariable("System.Date.Year")
```

- **"System.Date.DayOfWeek"** - system date, the day of the week

```
IR.GetVariable("System.Date.DayOfWeek")
```

- **"System.Date.Month"** - system date, the current month

```
IR.GetVariable("System.Date.Month")
```

- **"System.Date.Day"** - system date, the current day of the month

```
IR.GetVariable("System.Date.Day")
```

- **"System.Date.YYYY_MM_DD"** - system date in the format YYYY_MM_DD (numeric)

```
IR.GetVariable("System.Date.YYYY_MM_DD")
```

- **"System.Date.DD_MONTH_YYYY"** - system date in the format DD_MONTH_YYYY (alphanumeric)

```
IR.GetVariable("System.Date.DD_MONTH_YYYY"")
```

- **"System.Date.MONTH_DD_YYYY"** - system date in the format MONTH_DD_YYYY (alphanumeric)

```
IR.GetVariable("System.Date.MONTH_DD_YYYY")
```

- **"System.Date.DD_MM_YYYY"** - system date in the format DD_MM_YYYY (numeric)

```
IR.GetVariable("System.Date.DD_MM_YYYY")
```

- **"System.Date.MM_DD_YYYY"** - system date in the format MM_DD_YYYY (numeric)

```
IR.GetVariable("System.Date.MM_DD_YYYY")
```

- **"System.Date.DD_MM"** - system date in the format DD_MM (numeric)

```
IR.GetVariable("System.Date.DD_MM")
```

- **"System.Date.MM_DD"** - system date in the format MM_DD (numeric)

```
IR.GetVariable("System.Date.MM_DD")
```

- **"System.Date.Weekday"** - system date, the day of the week (numeric)

```
IR.GetVariable("System.Date.Weekday")
```

```
IR.AddListener(IR.EVENT_START, 0, function()
{
var date = IR.GetVariable("System.Date.Year");
IR.Log(date);
});
```

Get GPS Coordinates and Parameters

- **"System.Location.Course"** - course

```
IR.GetVariable("System.Location.Course")
```

- **"System.Location.Speed"** - speed

```
IR.GetVariable("System.Location.Speed")
```

- **"System.Location.Altitude"** - altitude

```
IR.GetVariable("System.Location.Altitude")
```

- **"System.Location.Longitude"** - longtitude

```
IR.GetVariable("System.Location.Longitude")
```

- **"System.Location.Latitude"** - latitude

```
IR.GetVariable("System.Location.Latitude")
```

```
IR.AddListener(IR.EVENT_START, 0, function()
{
var GPS = IR.GetVariable("System.Location.Course");
IR.Log(GPS);
});
```

Get Data from Compass

- **"System.Magnetic.Accuracy"** - accuracy of readings (intensity of magnetization)

```
IR.GetVariable("System.Magnetic.Accuarcy")
```

- **"System.Magnetic.True"** - true North

```
IR.GetVariable("System.Magnetic.True")
```

- **"System.Magnetic.Heading"** - magnetic North

```
IR.GetVariable("System.Magnetic.Heading")
```

- **"System.Magnetic.Z"** - Z-coordinate

```
IR.GetVariable("System.Magnetic.Z")
```

- **"System.Magnetic.Y"** - Y-coordinate

```
IR.GetVariable("System.Magnetic.Y")
```

- **"System.Magnetic.X"** - X-coordinate

```
IR.GetVariable("System.Magnetic.X")
```

```
IR.AddListener(IR.EVENT_START, 0, function()
{
var compass = IR.GetVariable("System.Magnetic.Accuarcy");
IR.Log(compass);
});
```

Get Data from Accelerometer

- **"System.Accelerometer.Shake"** - shake

```
IR.GetVariable("System.Accelerometer.Shake")
```

- **"System.Accelerometer.Z"** - Z-coordinate

```
IR.GetVariable("System.Accelerometer.Z")
```

- **"System.Accelerometer.Y"** - Y-coordinate

```
IR.GetVariable("System.Accelerometer.Y")
```

- **"System.Accelerometer.X"** - X-coordinate

```
IR.GetVariable("System.Accelerometer.X")
```

```
IR.AddListener(IR.EVENT_START, 0, function()
{
var Accelerometer = IR.GetVariable("System.Accelerometer.Shake");
IR.Log(Accelerometer);
});
```

Get Data from Gyroscope

- **"System.Gyroscope.Z"** - Z-coordinate

```
IR.GetVariable("System.Gyroscope.Z")
```

- **"System.Gyroscope.Y"** - Y-coordinate

```
IR.GetVariable("System.Gyroscope.Y")
```

- **"System.Gyroscope.X"** - X-coordinate

```
IR.GetVariable("System.Gyroscope.X")
```

```
IR.AddListener(IR.EVENT_START, 0, function()
{
var Gyroscope = IR.GetVariable("System.Gyroscope.Z");
IR.Log(Gyroscope);
});
```

Get Data about Device Display

- **"System.Display.FullScreen"** – full screen mode

```
IR.GetVariable("System.Display.FullScreen")
```

- **"System.Display.Height"** - display height

```
IR.GetVariable("System.Display.Height")
```

- **"System.Display.Width"** - display width

```
IR.GetVariable("System.Display.Width")
```

- **"System.Display.Bright"** - display brightness

```
IR.GetVariable("System.Display.Bright")
IR.SetVariable("System.Display.Bright", 80)  // (for iOS only)
```

- **"System.Orientation"** - display orientation

```
IR.GetVariable("System.Orientation")
```

```
IR.AddListener(IR.EVENT_START, 0, function()
{
var Display = IR.GetVariable("System.Display.FullScreen");
IR.Log(Display);
});
```

Get Information about Network

Variable name:

- **"System.Net.Cell"** – status of connection to 3G / 4G networks
- **"System.Net.WiFi"** – status of connection to Wi-Fi

Reading syntax:

- IR.GetVariable("System.Net.Cell")
- IR.GetVariable("System.Net.WiFi")

Example:

```
var Cell = IR.GetVariable("System.Net.Cell");
var WiFi= IR.GetVariable("System.Net.WiFi");

IR.Log("Cell:" + Cell + " / Wi-Fi:" + WiFi);
```

Values on outputs:

- **true** – there is a connection
- **false** – there is no connection

Use: To connect to local or external IP-address of your equipment automatically.

```
var Cell = IR.GetVariable("System.Net.Cell");

if(Cell) // if yes, then connect to the external address

   IR.GetDevice("MyDevice").SetParameters({Host: "23.165.123.21", Port:
```

```
2332});

else     // if no, then connect to the local address
   IR.GetDevice("MyDevice").SetParameters({Host: "192.168.0.32", Port:
9090});
```

See more about SetParametters

Get Information about Battery

- **"System.Battery.Level"** - level of battery charge

```
IR.GetVariable("System.Battery.Level")
```

- **"System.Battery.Status"** - battery status ()

```
IR.GetVariable("System.Battery.Status")
```

```
IR.AddListener(IR.EVENT_START, 0, function()
{
var Battery = IR.GetVariable("System.Battery.Level");
IR.Log(Battery);
});
```

Volume on Device

- **"System.Sound.Volume"** - volume value

```
IR.GetVariable("System.Sound.Volume")
```

- **"System.Sound.Mute"** - mute mode

```
IR.GetVariable("System.Sound.Mute")
```

```
IR.AddListener(IR.EVENT_START, 0, function()
{
var Sound = IR.GetVariable("System.Sound.Volume");
IR.Log(Sound);
});
```

Activate Vibration Mode

- **"System.Vibro"** - activate vibration mode

```
IR.SetVariable("System.Vibro")
```

```
IR.AddListener(IR.EVENT_START, 0, function()
{
var Vibro = IR.SetVariable("System.Vibro");
IR.Log(Vibro);
});
```

System.Update (Get)

"System.Update" is a system variable storing the current state of the project update mode. When using the function of receiving values **IR.GetVariable**, it returns **false** (0) or **true** (1).

- **false** (0) – the project update mode is off
- **true** (1) - the project update mode is on

The **System.Update** system variable is available for writing and you can use the **IR.SetVariable** function for changing the project update mode. You can find information about it here.

```
IR.AddListener(IR.EVENT_START, 0, function()
{
// Receive the project update mode
var UpdateState = IR.GetVariable("System.Update");
// Output in the log console
IR.Log(UpdateState);
});
```

System.Update (Set)

"System.Update" is a system variable storing the current state of the project update mode.

The variable is available for writing when using the function for writing values in the variable **IR.SetVariable**. Due to this function you are able to turn the project update mode on or off at any time of working with the project.

- To turn the project update mode off there should be written 0 or the logic value **false** in the system variable **System.Update**, example: **IR.SetVariable("System.Update", false);**
- To turn the project update mode on there should be written 1 or the logic value **true** in the system variable **System.Update**, example: **IR.SetVariable("System.Update", true);**

Also the **"System.Update"** variable is available for reading. You can find information about it here.

```
IR.AddListener(IR.EVENT_START, 0, function()
{
// Turn the update mode on
IR.SetVariable("System.Update", true);
// Turn the update mode off
IR.SetVariable("System.Update", false);
});
```

System.OS.Name

The system variable returns the name of the OS where the **iRidium** client is running.

Syntax:
IR.GetVariable("System.OS.Name");

Outgoing values:

- Windows
- MacOSX
- iOS
- Android

Example:

```
IR.AddListener(IR.EVENT_START, 0, function()
{
   // Writing the OS value in the OS variable
   var OS = IR.GetVariable("System.OS.Name");
   // Output of the OS variable in the log
   IR.Log(OS);
});
```

System.OS.Device

The system variable returns the special identifier of the OS version where the **iRidium** client is running.

Syntax:
IR.GetVariable("System.OS.Device");

Outgoing values:

- When using on Windows OS
  - 1 - Windows OSx86 32 / 64 bit
- When using on Mac OS X
  - 1 - Mac OS X PPC
  - 2 - Mac OS X X86 32 bit
  - 3 - Mac OS X X86 64 bit
- When using on iOS
  - 1 - iPhone
  - 2 - iPad
- When using on Android
  - 1 - Phone
  - 2 - Tablet

Example:

```
IR.AddListener(IR.EVENT_START, 0, function()
{
   // Writing of the OS version identifier in the Device variable
   var Device = IR.GetVariable("System.OS.Device");
   // Output of the Device variable in the log
   IR.Log(Device);
});
```

System.OS.Version

The system variable returns the OS version. It is not implemented at the moment.

Syntax:
IR.GetVariable("System.OS.Version");

↑ Back

System.Device.Name

The system variable returns network device name which launcing i2 Control app.

Syntax:
IR.GetVariable("System.Device.Name");

↑ Back

System.Device.Orientation

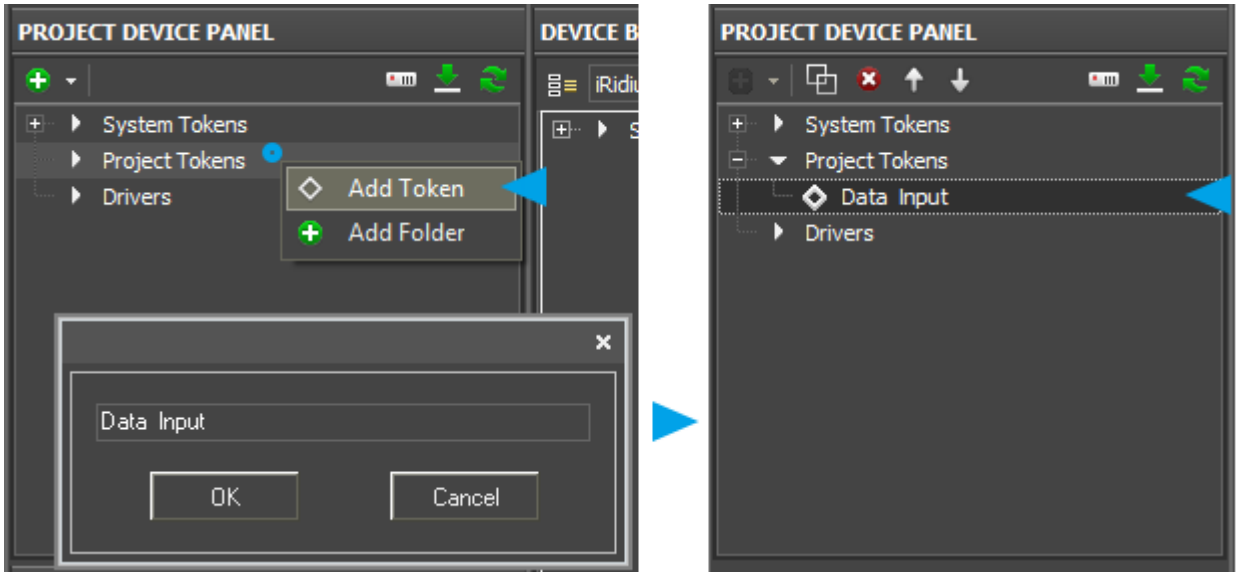
Syntax:
IR.GetVariable("System.Device.Orientation");

Orientation can have the following values:
1 portrait - Home button at the bottom
2 portrait - Home button at the top of the screen
3 landscape - Home button on the right
4 landscape - Home button on the left
5 when flat on the table with the touchscreen facing up
6 when flat and the touchscreen is facing down
0 when being moved between orientations

↑ Back

Project Tokens

Project Tokens – they can store data which are available for reading and writing. A graphic item, a driver or a script can write data to the tokens and read them.



- The variables are stored in Project Device Panel
- You can write a number or a string (DEC or ASCII) in the variables.
- The variables can be created, deleted, copied, sorted or grouped in folders.
- Data can be saved (or not saved) in the variables after closing iRidium.
- You can write data to the variables using a command from an item or iRidium Script

Writing Values in Project Tokens

The **IR.SetVariable** function is used for writing any value in any Project Token.

IR.SetVariable("Tokens.TokenName",Value), where

- **TokenName** – the name of any token in the Project Tokens folder of Project Device Panel
- **Value** – any value (number, string, array of numbers, array of strings).

Example:

```
IR.AddListener(IR.EVENT_START, 0, function()
{
// Writing number 23 in project token with the name  test
IR.SetVariable("Tokens.test",23);
// Output the value of the project token with the name test in the debug
console
IR.Log(IR.GetVariable("Tokens.test"));
});
```

Reading Values from Project Tokens

The **IR. GetVariable** function is used for reading of any value in any Project Token.

IR.GetVariable("Tokens.TokenName"), где

- **TokenName** – the name of any token in the Project Tokens folder of Project Device Panel

Example:

```
IR.Log(IR.GetVariable("Tokens.test_token"));
```

Writing of Arrays in Projects Tokens and Their Extraction

DOWNLOAD: Projects

One-dimensional Array

```
var Value_Buttons = [0,0];

function LoadData(){
   Value_Buttons = IR.GetVariable("Global.Data").split(",");
   IR.Log(Value_Buttons)
}


function SaveData(){
   IR.SetVariable("Global.Data", Value_Buttons);
   IR.Log(Value_Buttons);
}

SaveData();
LoadData();
```

Two-dimensional Array

```
var Value_Buttons = [[0,0],[1,1]];

function LoadData(){
   var data = IR.GetVariable("Global.Data");
   data = data.split(";");

   for(var i = 0; i < data.length; i++)
     Value_Buttons[i] = data[i].split(",");

   IR.Log(Value_Buttons)
}

function SaveData(){
   var data = "";

   for(var i = 0; i < Value_Buttons.length; i++)
```

```
     data += Value_Buttons[i] + ";";

   IR.SetVariable("Global.Data", data);

   IR.Log(data);
}

SaveData();
LoadData();
```

Three-dimensional Array

```
var Value_Buttons = [[[0,0],[1,1]],[[2,2],[3,3]]];

function LoadData(){
   var data_1 = IR.GetVariable("Global.Data");
   var data_2 = [];

   data_1 = data_1.split(":");

   for(var i = 0; i < data_1.length; i++){
      data_2[i] = data_1[i].split(";");

      for(var j = 0; j < data_2[i].length; j++)
         Value_Buttons[i][j] = data_2[i][j].split(",");
   }
   IR.Log(Value_Buttons);
}

function SaveData(){
   var data = "";

   for(var i = 0; i < Value_Buttons.length; i++){
      for(var j = 0; j < Value_Buttons[i].length; j++){
         data += Value_Buttons[i][j];

         if(j < Value_Buttons[i].length -1)
            data += ";"
      }
      if(i < Value_Buttons.length -1)
         data += ":";
   }
   IR.SetVariable("Global.Data", data);
   IR.Log(data);
}

SaveData();
LoadData();
```

DOWNLOAD: Example of a project