

# Data Feed API

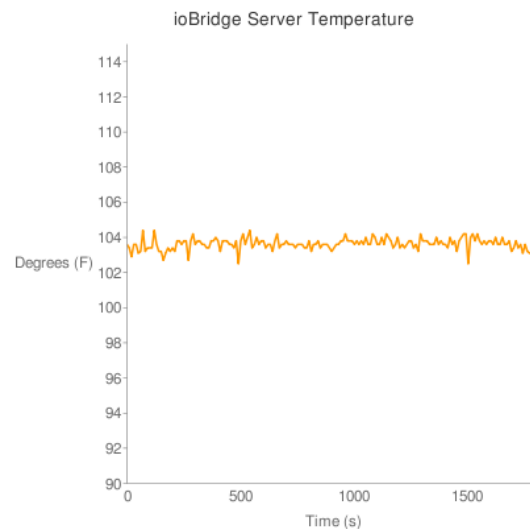
For the ioBridge IO-204 Monitor and Control Module

## Introduction

Each module that you have linked and registered to your account has a unique Feed URL. When requested, the URL returns a JSON formatted snapshot of your module's input and output states as well as any META information you have supplied such as Location, Latitude, Longitude, Labeling, and Scaling. The returned data allows for the creation of data logging, charting, custom scaling, mash-ups, and importing into server and client-side scripting. The API is lightweight and allows you to easily extend the ioBridge system data to remote systems, servers, clients, desktops, and applications.

The Feed URL is linked on your Module List under the Actions heading.

We are excited to see how you will extend the data from your module to your applications and services. Visit [www.iobridge.com/projects](http://www.iobridge.com/projects) for ideas and to submit your own ideas, comments, and project developments.



*Data Charting with Google Charts and the ioBridge Data Feed API*

## Extending the API

You may extend the data to server-side scripts, desktop applications, and client-side browser-based applications.

Here are some ways you can extend the API into your application:

- Google Gadgets and Desktop Widgets
- Vista Widgets
- Custom Analog Sensor Scaling
- Data Charting
- Data Logging
- Google Maps
- Weather Widgets

## Limitations

The feed requests are cached for 10 seconds, which means your application will only return new data every 10 seconds. ioBridge acts as a proxy between the world and your module, so you don't have to worry about your project going viral. We will only link to your module once every 10 seconds regardless of the number of people viewing your application, widget, or website based on the Data Feed API.

## Server-side JSON

You can safely request your Feed URL from any server-side scripting language like Perl, Python, or PHP that supports JSON. The Feed URL returns the HTTP header of “Content-type: application/json”.

### JSON Feed URL Example

[http://www.iobridge.com/interface/feed/FD\\_YWDFXHHs5Ur4PSQ6VK5](http://www.iobridge.com/interface/feed/FD_YWDFXHHs5Ur4PSQ6VK5)

### JSON Object

```
{
  "module": {
    "serial": "24000XXX",
    "label": "ioServer Monitor",
    "lat": "",
    "long": "",
    "location": "Gainesville, FL",
    "datetime": "Dec-24-2008 00:35:22 AM",
    "status": "Online",
    "channels": [
      {"channel": "1", "label": "Channel 1", "AnalogInput":
"384", "AnalogInputRaw": "384", "AnalogInputScale": "Raw",
"DigitalInput": "Off", "DigitalInputState": "0", "DigitalOutput": "On",
"DigitalOutputState": "1"},
      {"channel": "2", "label": "Channel 2", "AnalogInput": "308",
"AnalogInputRaw": "308", "AnalogInputScale": "Raw", "DigitalInput":
"Off", "DigitalInputState": "0", "DigitalOutput": "Off",
"DigitalOutputState": "0"},
      {"channel": "3", "label": "Channel 3", "AnalogInput": "0",
"AnalogInputRaw": "0", "AnalogInputScale": "Raw", "DigitalInput": "0",
"DigitalInputState": "0", "DigitalOutput": "0", "DigitalOutputState":
"0"},
      {"channel": "4", "label": "Channel 4", "AnalogInput": "2.737",
"AnalogInputRaw": "560", "AnalogInputScale": "Voltage", "DigitalInput":
"Off", "DigitalInputState": "0", "DigitalOutput": "Down",
"DigitalOutputState": "0"}]
    }
  }
}
```

## Client-side JSON

If you are using jQuery or other client-scripting libraries to parse JSON, you have the ability to use JSONP (JSON with Padding) to get JSON from two different domains using Callback.

JSONP Callback on the Feed URL

[http://www.iobridge.com/interface/feed/FD\\_YWDFXHHs5Ur4PSQ6VK5&callback=?](http://www.iobridge.com/interface/feed/FD_YWDFXHHs5Ur4PSQ6VK5&callback=?)

When callback is used, the Feed URL returns the following JSON Object:

```
jsonp1230097032945({
  "module": {
    "serial": "24000XXX",
    "label": "ioServer Monitor",
    "lat": "",
    "long": "",
    "location": "Gainesville, FL",
    "datetime": "Dec-24-2008 00:36:40 AM",
    "status": "Online",
    "channels": [
      {"channel": "1", "label": "Temperature", "AnalogInput":
"103.6", "AnalogInputRaw": "665"
, "AnalogInputScale": "Temp F", "DigitalInput": "On",
"DigitalInputState": "1", "DigitalOutput": "Off"
, "DigitalOutputState": "0"},
      {"channel": "2", "label": "Channel Tester", "AnalogInput": "337",
"AnalogInputRaw": "337", "AnalogInputScale"
: "Raw", "DigitalInput": "Hi", "DigitalInputState": "1",
"DigitalOutput": "Off", "DigitalOutputState"
: "0"},
      {"channel": "3", "label": "Servo", "AnalogInput": "269",
"AnalogInputRaw": "269", "AnalogInputScale"
: "Raw", "DigitalInput": "On", "DigitalInputState": "1",
"DigitalOutput": "On", "DigitalOutputState"
: "1"},
      {"channel": "4", "label": "IR", "AnalogInput": "0", "AnalogInputRaw":
"0", "AnalogInputScale": "Raw"
, "DigitalInput": "0", "DigitalInputState": "0", "DigitalOutput": "0",
"DigitalOutputState": "0"}]
    }
  })
```

## JSON Keys and Meta Information

You set the keys and meta information on the Modules tab. Data and configuration entered on the Modules tab are the returned labels and scaling in the JSON object. Only I/O Channel modes return data in the feed.

The following keys are available to parse in your JSON feed:

Key	Description
<b>serial</b>	Module Serial Number
<b>label</b>	Entered Module Label
<b>lat</b>	Entered Latitude (Decimal Degrees)
<b>long</b>	Entered Longitude (Decimal Degrees)
<b>location</b>	Entered Location Description
<b>datetime</b>	Date and Time of Module Snapshot
<b>status</b>	Status of Module (Offline, Online)
<b>channels</b>	Array of Four Module Channel (0,1,2,3)
<b>channel</b>	Channel Number
<b>label</b>	Entered Channel Label
<b>AnalogInput</b>	Scaled Analog Input value (Voltage, Raw, Temperature)
<b>AnalogInputRaw</b>	Relative Analog Input Value (0-1023)
<b>AnalogInputScale</b>	Selected Scale of Analog Input
<b>DigitalInput</b>	Labeled Digital Input State
<b>DigitalInputState</b>	Digital Input State (1,0)
<b>DigitalOutput</b>	Labeled Digital Output State
<b>DigitalOutputState</b>	Digital Output State (1,0)

## Parsing JSON

You can get access to any of the data in the JSON Object by parsing out the desired keys.

For example here is a jQuery request and data selection from feed:

```
function checkTemp() {  
    $.getJSON("http://www.iobridge.com/interface/feed/FD_0Bo  
sKdymNbB3LyiFYUVI&callback=?",  
    function (data) {  
        storeTemp(data.module.channels[0].AnalogInput);  
updateGraph();  
    });  
}
```

“data.module.channels[0].AnalogInput” references the scaled analog input value on channel 1 of the IO-204 Module.

If you would want the modules serial number, the reference would be “data.module.serial” using the serial key under module.

## Client-side Application Example

From soup to nuts, we wanted to show you a complete application extension of the Data Feed API and the steps involved. Share your projects at [www.iobridge.com/projects](http://www.iobridge.com/projects).

The following projects takes your Feed URL extracts the Scaled Analog Input from Channel 1 and feeds Google Charts to return a Google-o-Meter indicating the current temperature using jQuery.

**Step 1:** Create a new HTML file called ServerTempNeedle.html

**Step 2:** Create the HTML frame work – head and body

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>ioBridge Server Temperature</title>
</head>
<body>
<div id="container">
  <div id="data_graph"></div>
</div>
</body>
</html>
```

**Step 3:** Load jQuery from Google JSAPI to the document header

```
<script type="text/javascript" src="http://www.google.com/jsapi"></script>
<script type="text/javascript">
  google.load("jquery", "1");
</script>
```

**Step 4:** Add a ready function when the HTML fully loads to check the temperature from the feed and set a timer that will check the temperature every 10 seconds

```
$(document).ready(function() {
  checkTemp(); // Gets initial temperature
  setInterval("checkTemp()", 10000); // Sets a 10 Second timer
});
```

**Step 5:** Add function that checks the temperature and updates the graph when the data is returned (use your feed URL to get your data or use our public feed to get our server temperature)

```
function checkTemp() {
    $.getJSON("http://www.iobridge.com/interface/feed/FD_0BosKdymNbB3LyiFYUVI&callback=?",
        function (data) {

            // Channel 1 is the first element in the array
            updateGraph(data.module.channels[0].AnalogInput);

        });
}
```

**Step 6:** Add function to get the Google chart and display it in the “data\_graph” division in the document body

```
function updateGraph(temp) {

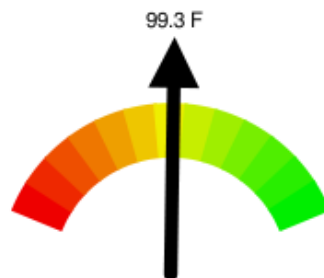
    var newTemp = temp-50;
    newTemp = 100 - newTemp;

    var chart_url = "http://chart.apis.google.com/chart?"
    var chart_type = "cht=gom";
        var chart_size = "&chs=300x200";
    var chart_title = "&chtt=ioBridge Server Temperature";
    var chart_data = "&chd=t:" + newTemp + "&chl=" + temp + " F";
    var url = chart_url + chart_type + chart_size + chart_title + chart_data;

    var i = $("<img>").attr("src", url);
    $("#data_graph").empty().append(i);
}
```

**Step 7:** Save file and open in web browser such as Safari, Chrome, Firefox, Opera, or Internet Explorer

ioBridge Server Temperature



Check out the live version at: [www.iobridge.com/ServerTempNeedle.html](http://www.iobridge.com/ServerTempNeedle.html)

## ServerTempNeedle.html Source Code

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>ioBridge Server Temperature</title>

<script type="text/javascript" src="http://www.google.com/jsapi"></script>
<script type="text/javascript">
    google.load("jquery", "1");
</script>
<script type="text/javascript">

function updateGraph(temp) {

    var newTemp = temp-50;
    newTemp = 100 - newTemp;

    var chart_url = "http://chart.apis.google.com/chart?"
    var chart_type = "cht=gom";
    var chart_size = "&chs=300x200";
    var chart_title = "&chtt=ioBridge Server Temperature";
    var chart_data = "&chd=t:" + newTemp + "&chl=" + temp + " F";
    var url = chart_url + chart_type + chart_size + chart_title + chart_data;

    var i = $("<img>").attr("src", url);
    $("#data_graph").empty().append(i);
}

function checkTemp() {
    $.getJSON("http://www.iobridge.com/interface/feed/FD_0BosKdymNbB3LyiFYUUI&call
back=?",
    function (data) {
        updateGraph(data.module.channels[0].AnalogInput);
    });
}

$(document).ready(function() {
    checkTemp(); // Gets initial temperature
    setInterval("checkTemp()", 10000); // Sets a 10 Second timer
});

</script>
</head>
<body>

<div id="container">
<div id="data_graph"> </div>
</div>
</body>
</html>

```