Viper Data Export Options

January 2023

Contents

Non Web-Service Options

Option 1 – Survey Controller Laptop Exports

Data is temporarily stored in the local instance of SQL server and stored data is automatically published to Viper.Net every minute. After each publish, Survey Controller saves both "raw" and "compiled" .CSV files locally on the Survey Controller computer. All .CSV files are created in 500,000 record batches. Raw & Compiled .CSVs can be accessed through the Survey Controller interface under the Data Menu. The default file location for .CSVs = Program Files \Viper\Viper Field Survey Controller. Stored data is purged from SQL after publishing to ensure the SQL database does not exceed limitations (10GB for SQL Express).

Raw Data

The Raw Data contains all data observed by Survey Controller. Regardless of any reading interval(s) set in Survey Controlle, the Raw Data will contain every reading sent by the instrument, including alarms. The Raw Data will contain the latitude and longitude reported by the LINC GPS along with the reading. By default, the Raw Data will be written to local .csv files even if an internet connection is not available for publishing to Viper.NET (options exist to disable this feature). The .csv file can contain up to 500,000 records. If it exceeds that number of records, Viper Survey Controller will automatically create another file.

Compiled Data

The Compiled Data contains all of the data published to Viper.NET. If reading (polling) intervals other than the default 'all readings' are configured in Survey Controller, Compiled Data will contain only the readings at those intervals. If alarms are configured in the MeterApp, during an alarm event, the Compiled Data will contain all sensor readings. If GPS Mode is set to Fixed, Compiled Data will contain the Latitude and Logitude values entered during configuration. A second set of columns will also contain the GPS values reported by the LINC. If the GPS Mode is set to Mobile and the PDOP falls too low (>20), Compiled Data will re-use the last quality Latitude and Longitude coordinate. If using a Clustered Configuration, Compiled Data will contain the same Latitude and Longitude across all instruments for each reading.

.CSV File Details

- The default storage folders for Raw & Compiled files can be accessed or modified from Survey Controller's Data menu
- .CSV files are stored in *folders* organized *by date*
- .CSV *files* are named *by* the *time* of the first reading they contain
- .CSV files might need to be combined and manipulated to report readings for a desired date/time range

.CSV File Details – UTC Time

- **UTC Time**
- **All dates and times including folder and file names are reported in UTC**
    - UTC is 5 hours ahead of Eastern Standard Time
    - UTC is 4 hours ahead of Eastern Daylight Time.
    - UTC is 6 hours ahead of Central Standard Time
    - UTC is 7 hours ahead of Mountain Standard Time
    - UTC is 8 hours ahead of Pacific Standard Time
- Data (.CSV) files for a particular calendar day or time period might be spread across multiple files and/or *folders* since UTC date will increment before local calendar date

  - For example, data collected July 4th at 9pm EDT will be in the folder for July 5th and will be reported as 1am

Option 2 - Deployment User Exports

Instrument specific data can be exported from Deployment Manager. The benefit of exporting from Deployment Manager is that the times will be Local versus UTC and any TWA's, alarms, etc. configured in Deployment Manager will be included with the export. For specific instructions on exporting instrument data from Deployment Manager, please reference Section 3 (Sensor Data Graphs) of the "Viper Guide to Deployment Manager.PDF". This guide can be downloaded using the following link:
https://response.epa.gov/sites/5033/files/Viper%20Guide%20to%20Deployment%20Manager.pdf

If you do not have sufficient permissions to download the file above, please request access by e-mailing ertsupport@epa.gov.

Option 3 - Automated CSV Exports

ERT Support can create automated .CSV exports for a deployment. These exports can be configured to run automatically at specific times to support the data reporting requirements of a deployment or as a one-time request. When a continuous automated data export is configured, the requestor will receive a website address along with a username and password for logging in to the secure website to download the exported files. Automated Data exports can include "all data" or a subset of data using specific arguments.

By default, automated exports will be posted as .ZIP files. Within the .ZIP file, data will be organized in folders by Survey Controller and then by Date.

Automated Data Export Options/Arguments

When requesting automated data exports, the following arguments can be used to refine the data set that is exported. If no arguments are specified, the data export will contain "all data". These arguments can be used alone or in combination with each other to achieve the desired output. Not all arguments are required and some arguments may contradict others. Readings which triggered one or more alarms are ALWAYS exported.

Combining the arguments with task scheduling (discussed later) allows for a fairly robust amount of data export options.

- **Runs:** One or many specific Run Identifiers whose data is to be exported.
  - Using this argument will limit the data set to include the specified Runs only

- **Instrument:** One or many Instrument Identifiers whose data is to be exported.
  - Using this argument will limit the data set to include the specified instruments only

- **Sensor:** One or many Sensor Names whose data is to be exported (e.g. "VOC")
  - Using this argument will limit the data set to include the specified Sensors only

- **StartDateUTC:** Data recorded after this UTC Date and Time will be exported

- **StopDateUTC:** Data recorded before this UTC Date and Time will be exported

- **LastNHours:** Data recorded this number of hours ago, thru till now, will be exported
  - Common examples are “last 24 hours”, “last 8 hours”, etc.

- **LastReadingIntervalInSeconds:** For each instrument, only 1 set of readings per this number of seconds will be exported

- **Alarms Only:** Only readings which triggered alarms

Automated Export Examples:

- Export all readings from Deployment 15, recorded within the past 24 hours, which triggered one or more alarms
- For Deployment 15, export one set of readings for each instrument for each 5-minute period.
  NOTE: Any and all readings which triggered one or more alarms will also be exported
- For run 1301-1, export all data recorded after 1/10/2012 5:00AM UTC (12:00AM EST)
- For runs 1301-1 and 1450-10, export all data recorded after 1/10/2012 5:00AM UTC (12:00AM EST)
- For run 1301-1, only export data from instrument "(.3001) AreaRAE"
- For Deployment 15, export all of the VOC readings

Scheduling Automated Data Exports

Automated CSV exports can be scheduled to run at a designated time/s to support data reporting requirements. Data will be available for download from the secure website shortly after the requested run. When requesting automated data exports, ERT Support will also need to know the required times for running the export.

Scheduling Examples

- A request for all data for the prior 24 hours can run at any specified time (midnight, noon, etc.) and a dataset for the prior 24 hours will be available shortly after the run time.
- A request for one set of readings for each 5-minute period for the prior 4 hours can run at 8am, noon, 4pm, 8pm, and midnight to produce 5 datasets of 5-minute readings for the 4 hours prior to each run time.
- If a 1 hour Rolling TWA has been configured in Deployment Manager and the reporting requirement is for a single Rolling TWA reading per hour, the request for one set of hourly TWAs can be configured to run once an hour to produce datasets containing one single TWA for the prior hour.

Web-Service Options

All of the web service options require a Deployment Manager *system* account, SSL/TLSv1.2, and HTTP Basic Authentication:
https://en.wikipedia.org/wiki/Basic_access_authentication
https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication

Attachment 1 and 2 PowerPoint presentations illustrate how the deployments, runs, instruments, and sensors are organized within the OData feed, and how to retrieve a sensor's readings using the JSON ReadingArray service.

Option 1 – CSV File

https://viper.ert.org/DeploymentManager/svc/GetCSV.ashx
Returns: Comma-Separated Values (CSV) File

Supported Parameters:

- **DeploymentID** (integer)
- **RunIdentifier** (string; "1234-12")
- **InstrumentID** (string; "(.123) AreaRAE")
- **Sensor** (string; "SO2")
- **StartAt** (datetime; "6/1/2018 18:00"; assumed UTC, local also supported)
- **StopAt** (datetime; "6/1/2018 19:00"; assumed UTC, local also supported)
- **LocalTime** (any value; only specify if datetimes are local to the deployment)
- **LastNHours** (integer; return data for the last "this many" hours)
- **ReadingIntervalInSeconds** (integer; for each instruments, returns one set of readings for every "this number" of seconds)
- **AlarmsOnly** (boolean; "true"; only return readings that exceeded an alarm threshold)

Notes:

- **DeploymentID** is required
- Not all parameters are required. For instance, if you want data from multiple runs within the deployment, the RunIdentifier can be omitted. Similarly, the InstrumentID and Sensor parameters can be omitted if you want all data from all instruments.
- If neither Start/StopAt are provided, nor LastNHours, the service returns the last hour of data, by default.

Example:

https://viper.ert.org/DeploymentManager/svc/GetCSV.ashx?DeploymentID=xxx&LastNHours=72

Option 2 – OData

https://viper.ert.org/DeploymentManager/OData
Returns: Current Readings via OData Protocol (Version 2)
See: Attached PowerPoint Slides
http://www.odata.org/getting-started/basic-tutorial/

Option 3 – JSON ReadingArray

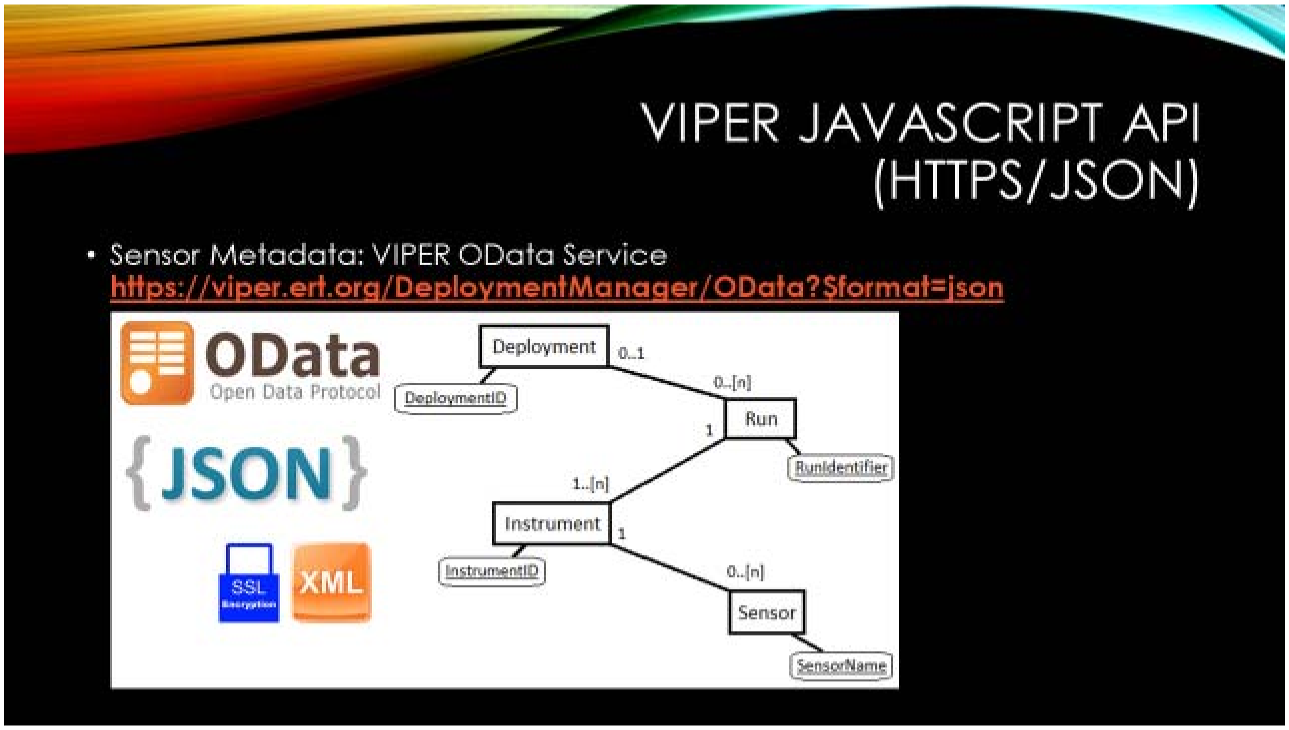
https://viper.ert.org/DeploymentManager/GetPrivateReadingArray.ashx
Returns: For a single Sensor, returns a 2-Dimensional JSON Array of datetime and reading values
See: Attached PowerPoint Slides
https://en.wikipedia.org/wiki/JSON
https://json.org/

Attachment1

VIPER - JavaScript-JSON API (Reading Arrays)

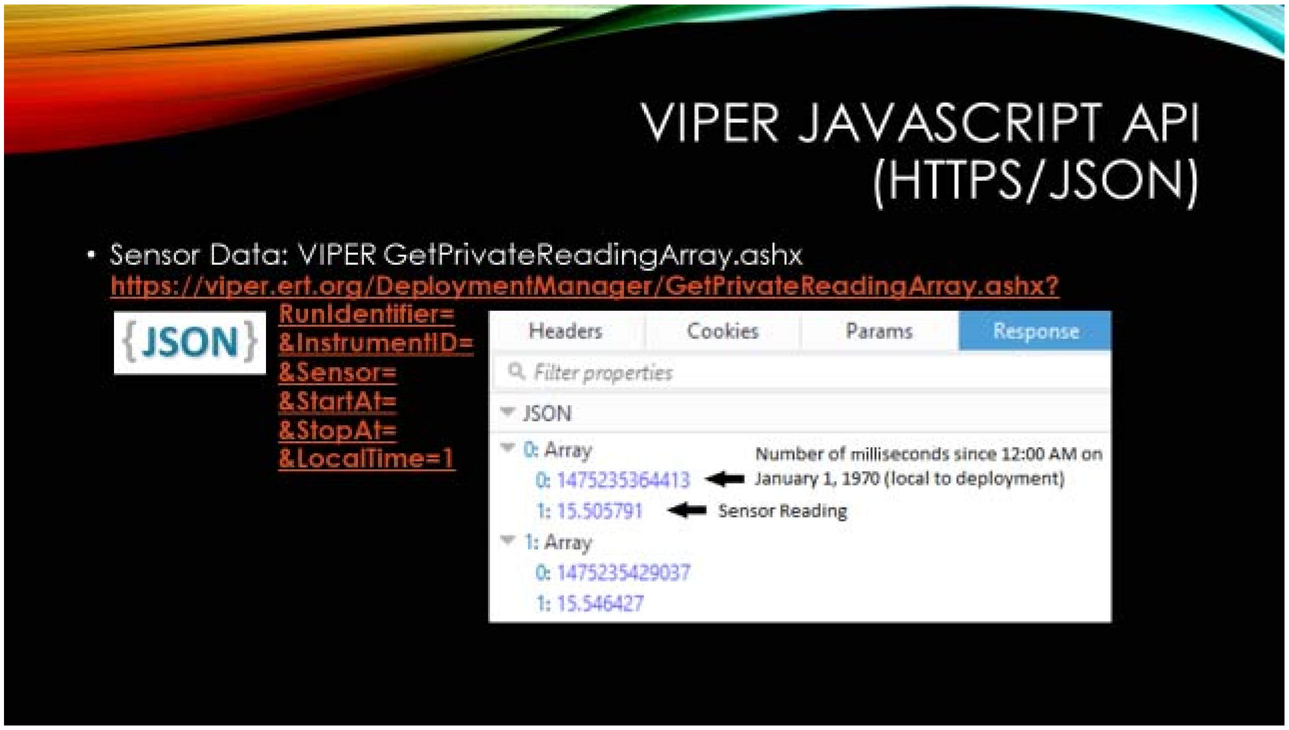
VIPER:
JAVASCRIPT/JSON API

Accessing VIPER Data as JSON via REST services over HTTPS/TLS with BASIC HTTP Authentication



VIPER JAVASCRIPT API (HTTPS/JSON)

- Sensor Metadata: VIPER OData Service
  https://viper.ert.org/DeploymentManager/OData?$format=json



VIPER JAVASCRIPT API (HTTPS/JSON)

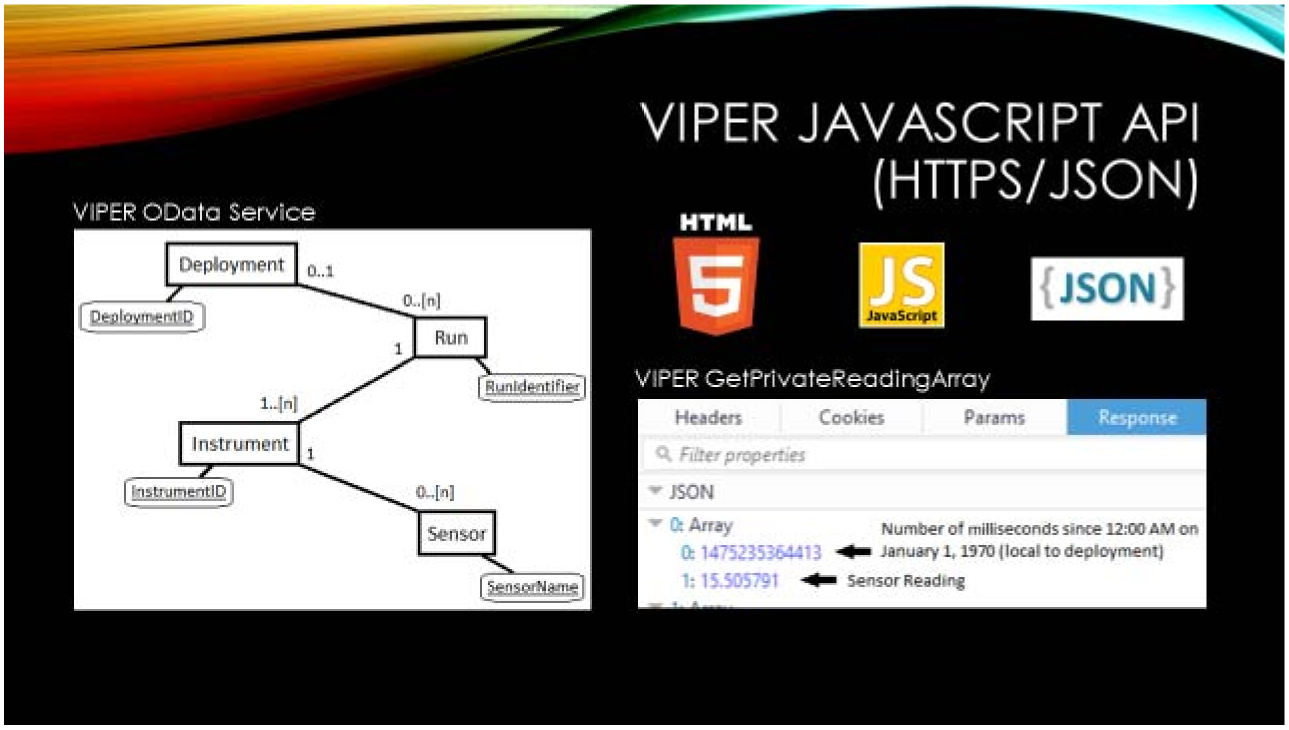
- Sensor Data: VIPER GetPrivateReadingArray.ashx
  https://viper.ert.org/DeploymentManager/GetPrivateReadingArray.ashx?RunIdentifier=&InstrumentID=&Sensor=&StartAt=&StopAt=&LocalTime=1

{JSON}

Headers | Cookies | Params | Response

Filter properties

```
▼ JSON
▼ 0: Array
    0: 1475235364413  ← Number of milliseconds since 12:00 AM on January 1, 1970 (local to deployment)
    1: 15.505791      ← Sensor Reading
▼ 1: Array
    0: 1475235429037
    1: 15.546427
```



VIPER JAVASCRIPT API (HTTPS/JSON)

VIPER OData Service

VIPER GetPrivateReadingArray

Headers | Cookies | Params | Response

Filter properties

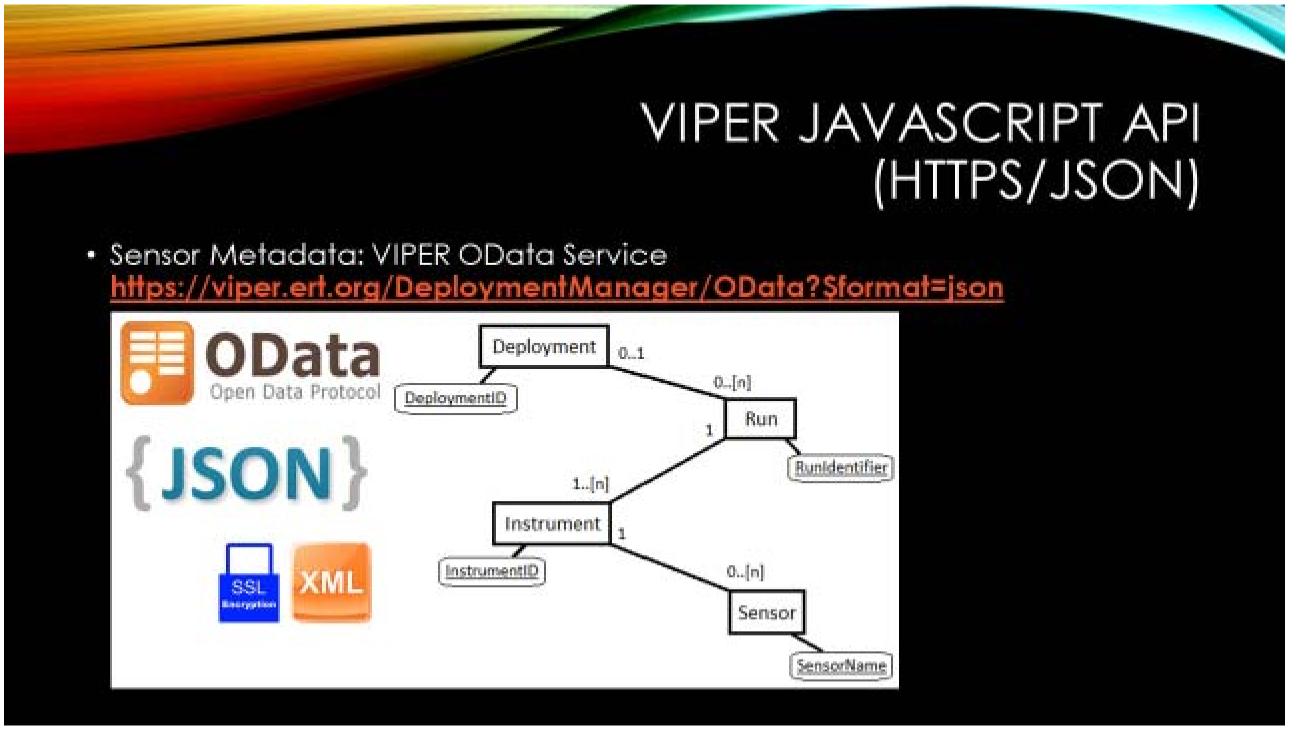
```
▼ JSON
▼ 0: Array
    0: 1475235364413  ← Number of milliseconds since 12:00 AM on January 1, 1970 (local to deployment)
    1: 15.505791      ← Sensor Reading
▼ 1: Array
```

Attachment 2

VIPER - JavaScript-JSON API (Reading Arrays - for Developers)

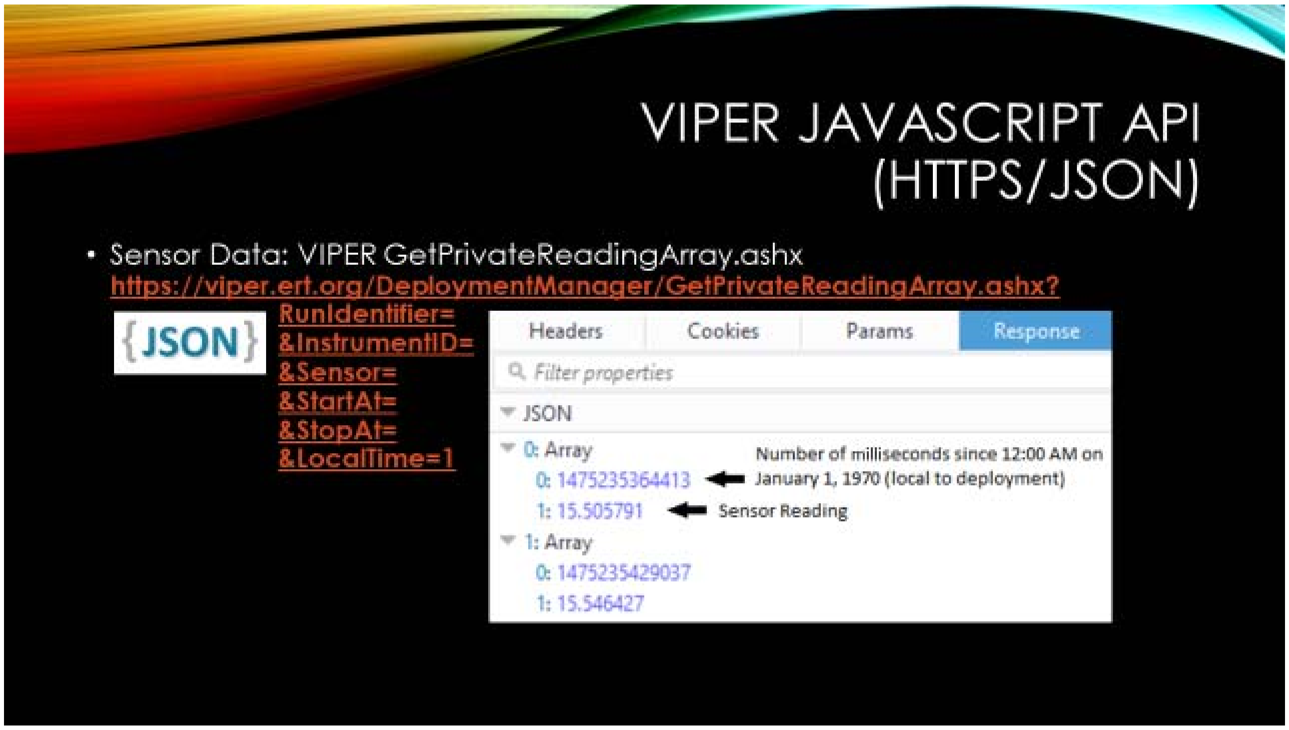
VIPER:
JAVASCRIPT/JSON API

Accessing VIPER Data as JSON via REST services
using HTTPS/TLS with BASIC HTTP Authentication



VIPER JAVASCRIPT API
(HTTPS/JSON)

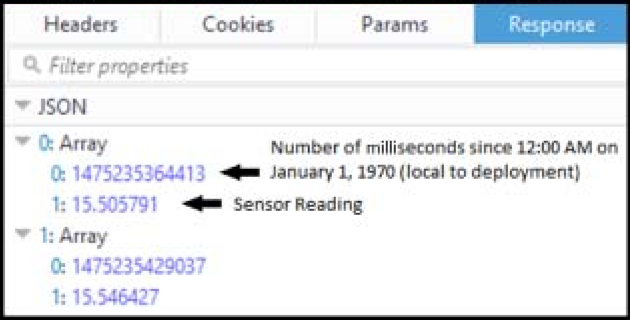
- Sensor Metadata: VIPER OData Service
https://viper.ert.org/DeploymentManager/OData?$format=json



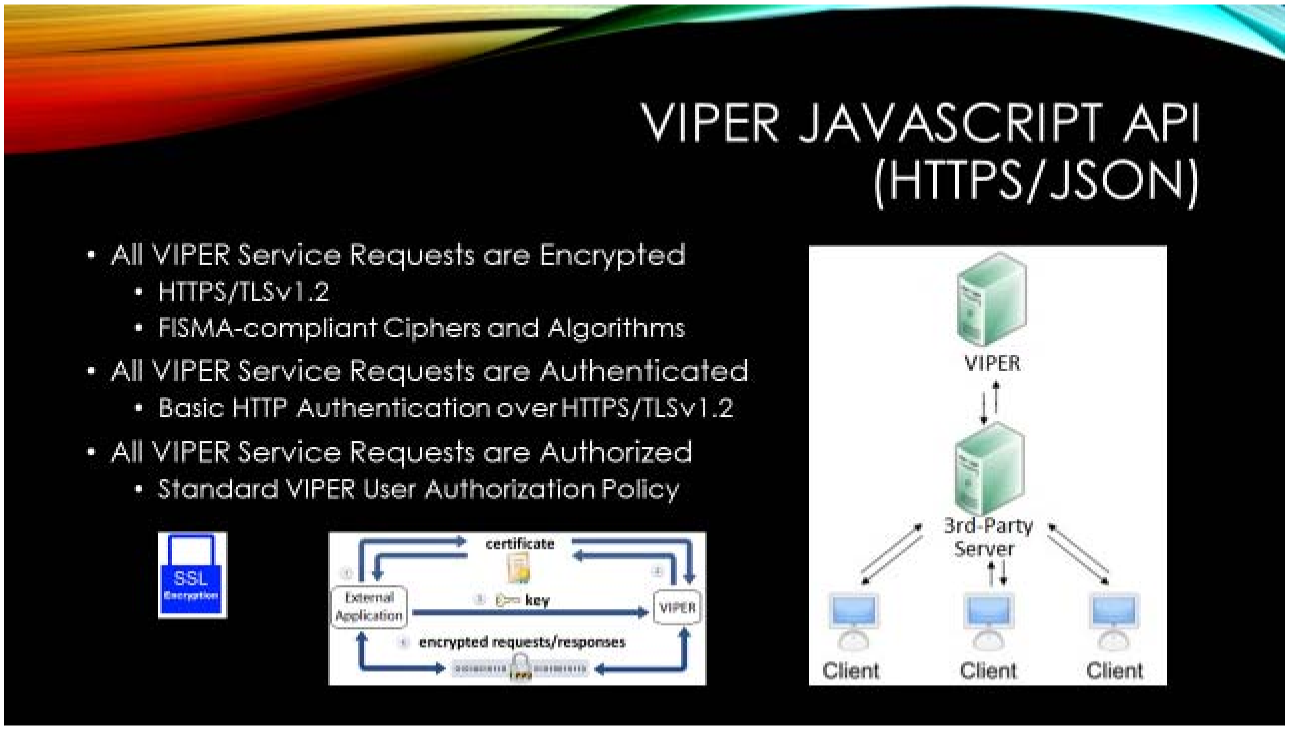
VIPER JAVASCRIPT API (HTTPS/JSON)

- Sensor Data: VIPER GetPrivateReadingArray.ashx
  https://viper.ert.org/DeploymentManager/GetPrivateReadingArray.ashx?
  RunIdentifier=
  &InstrumentID=
  &Sensor=
  &StartAt=
  &StopAt=
  &LocalTime=1

{JSON}



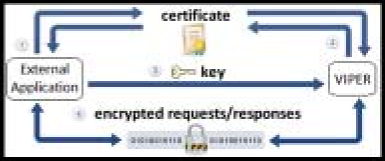
```
JSON
  0: Array
    0: 1475235364413  ← Number of milliseconds since 12:00 AM on January 1, 1970 (local to deployment)
    1: 15.505791      ← Sensor Reading
  1: Array
    0: 1475235429037
    1: 15.546427
```

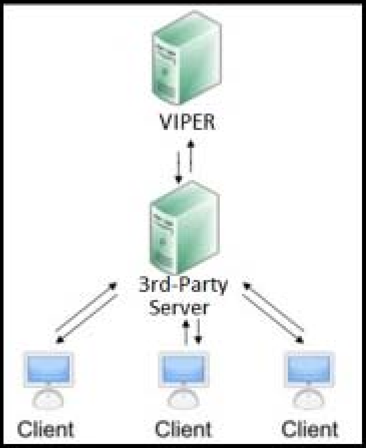


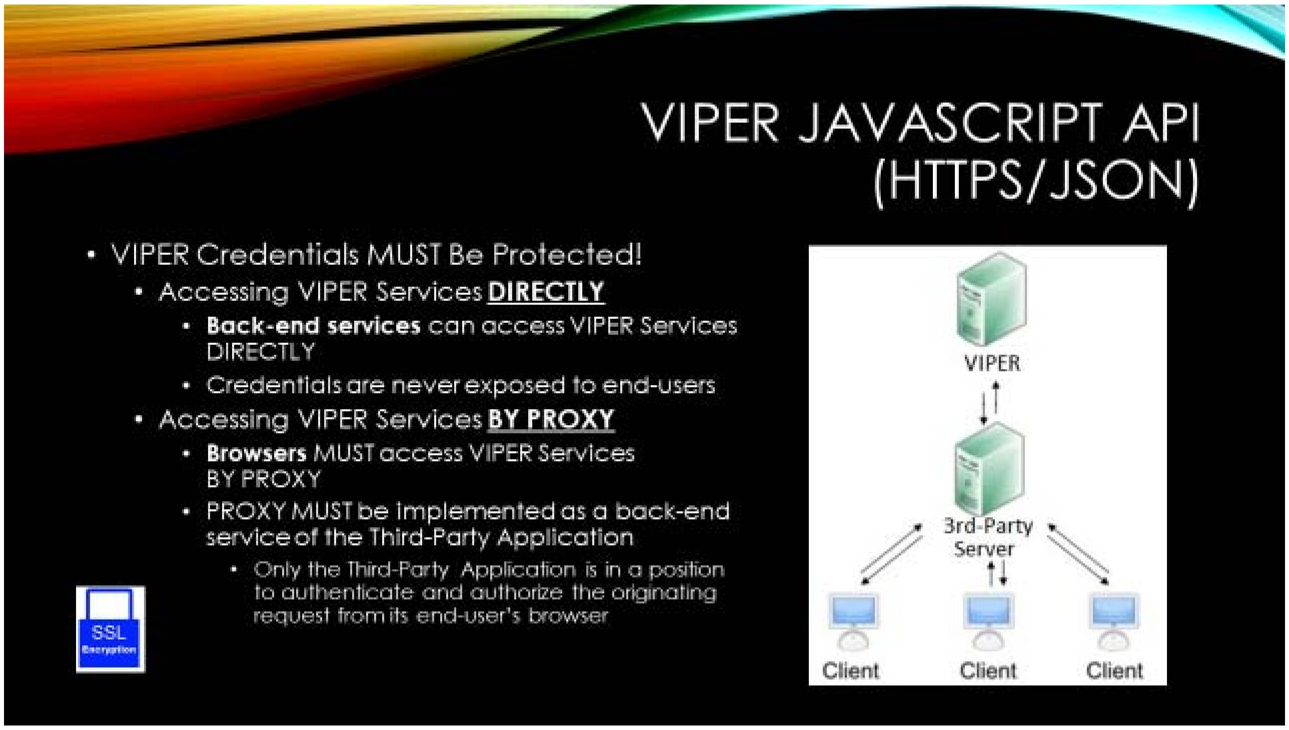
VIPER JAVASCRIPT API (HTTPS/JSON)

- All VIPER Service Requests are Encrypted
  - HTTPS/TLSv1.2
  - FISMA-compliant Ciphers and Algorithms
- All VIPER Service Requests are Authenticated
  - Basic HTTP Authentication over HTTPS/TLSv1.2
- All VIPER Service Requests are Authorized
  - Standard VIPER User Authorization Policy

SSL Encryption



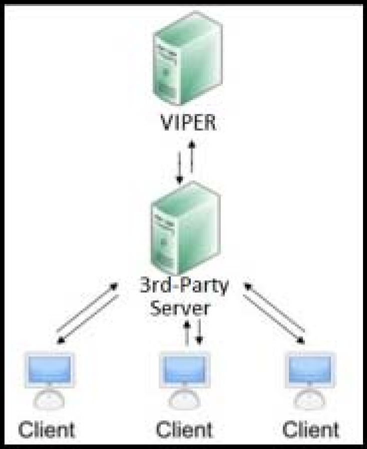


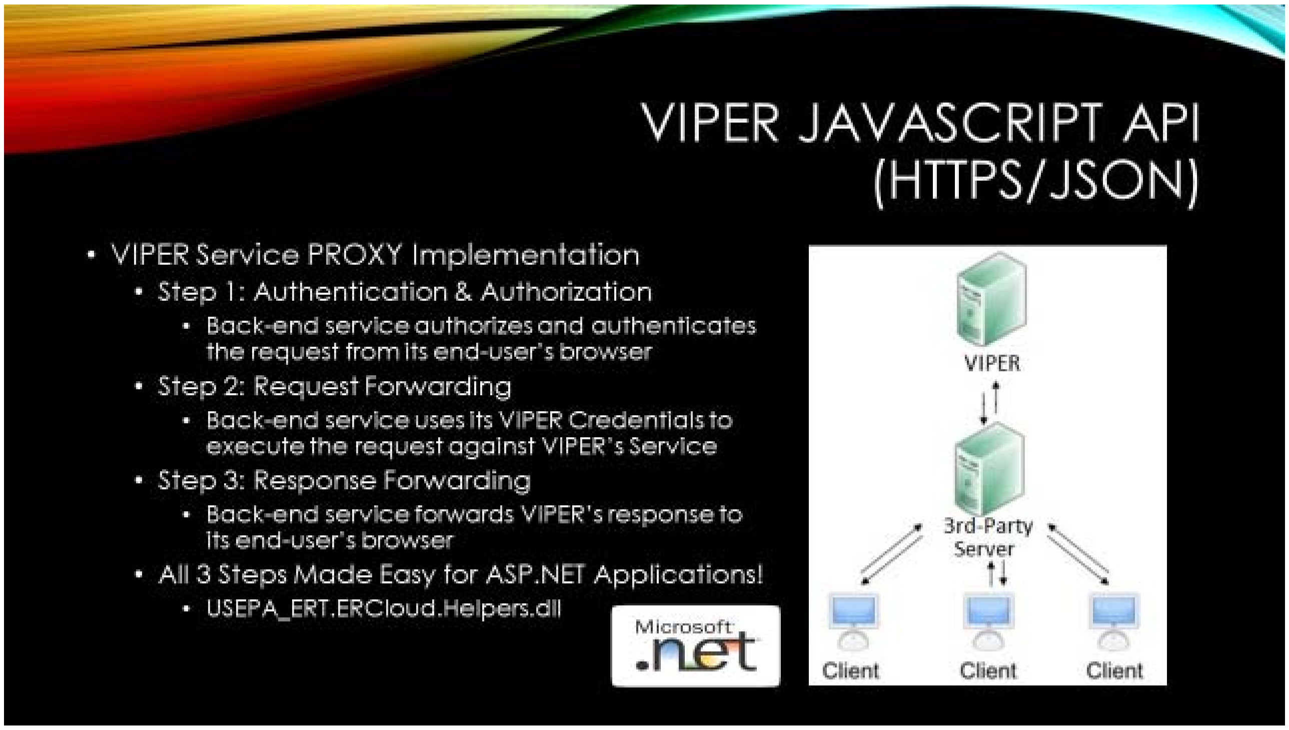


VIPER JAVASCRIPT API (HTTPS/JSON)

- VIPER Credentials MUST Be Protected!
  - Accessing VIPER Services **DIRECTLY**
    - **Back-end services** can access VIPER Services DIRECTLY
    - Credentials are never exposed to end-users
  - Accessing VIPER Services **BY PROXY**
    - **Browsers** MUST access VIPER Services BY PROXY
    - PROXY MUST be implemented as a back-end service of the Third-Party Application
      - Only the Third-Party Application is in a position to authenticate and authorize the originating request from its end-user's browser

SSL Encryption

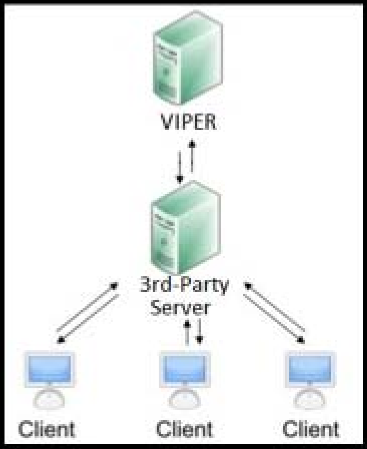


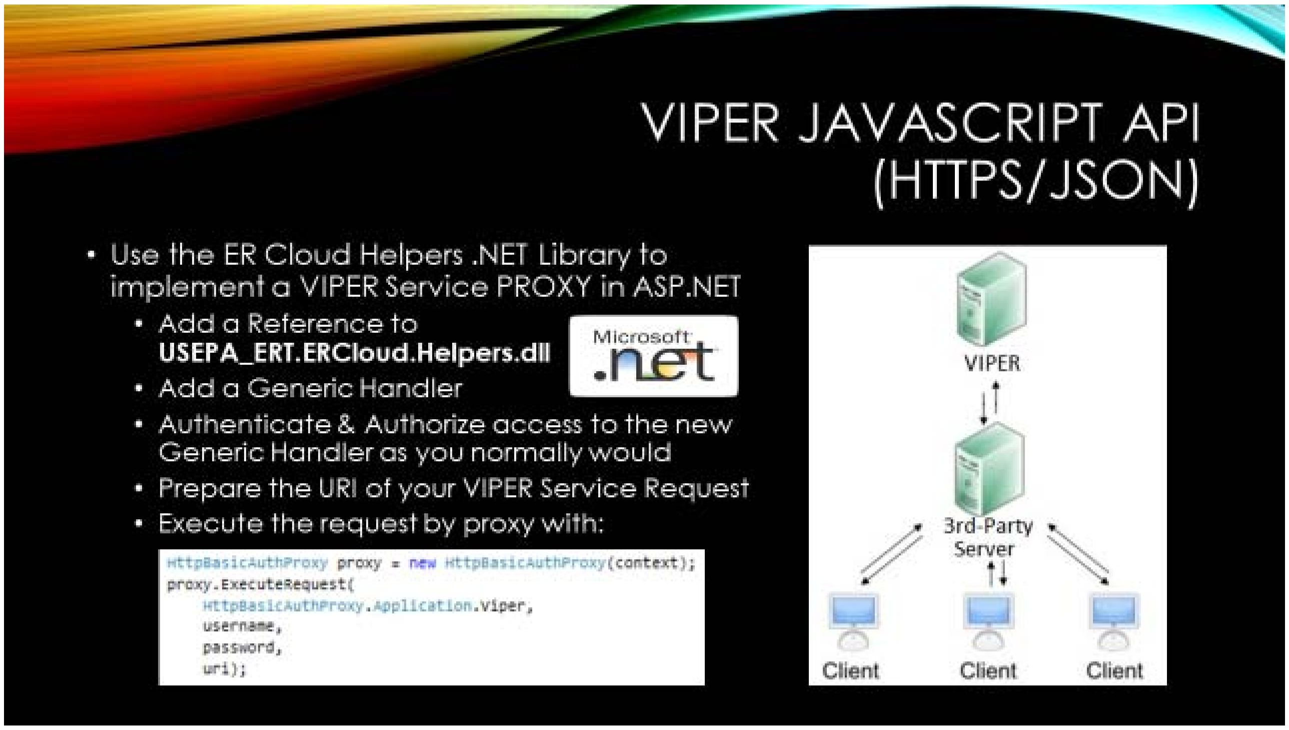


VIPER JAVASCRIPT API (HTTPS/JSON)

- VIPER Service PROXY Implementation
  - Step 1: Authentication & Authorization
    - Back-end service authorizes and authenticates the request from its end-user's browser
  - Step 2: Request Forwarding
    - Back-end service uses its VIPER Credentials to execute the request against VIPER's Service
  - Step 3: Response Forwarding
    - Back-end service forwards VIPER's response to its end-user's browser
  - All 3 Steps Made Easy for ASP.NET Applications!
    - USEPA_ERT.ERCloud.Helpers.dll

Microsoft .net



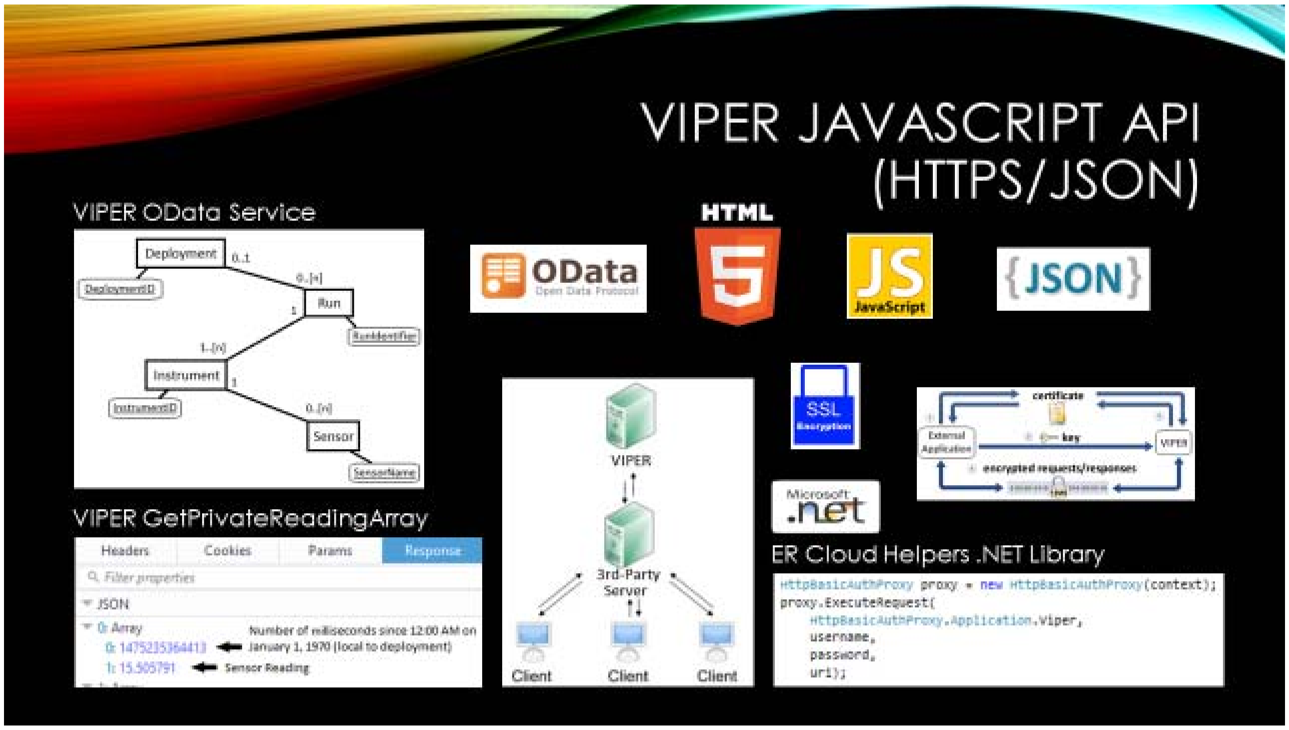


VIPER JAVASCRIPT API (HTTPS/JSON)

- Use the ER Cloud Helpers .NET Library to implement a VIPER Service PROXY in ASP.NET
  - Add a Reference to **USEPA_ERT.ERCloud.Helpers.dll**
  - Add a Generic Handler
  - Authenticate & Authorize access to the new Generic Handler as you normally would
  - Prepare the URI of your VIPER Service Request
  - Execute the request by proxy with:

```
HttpBasicAuthProxy proxy = new HttpBasicAuthProxy(context);
proxy.ExecuteRequest(
    HttpBasicAuthProxy.Application.Viper,
    username,
    password,
    uri);
```

Microsoft .net
VIPER
3rd-Party Server
Client
Client
Client



VIPER JAVASCRIPT API (HTTPS/JSON)

VIPER OData Service

Deployment
DeploymentID
0..1
0..[n]
Run
1
RunIdentifier
1..[n]
Instrument
1
InstrumentID
0..[n]
Sensor
SensorName
OData
Open Data Protocol
HTML 5
JS JavaScript
{JSON}
VIPER
3rd-Party Server
Client
Client
Client
SSL Encryption
certificate
External Application
key
VIPER
encrypted requests/responses
Microsoft .net

VIPER GetPrivateReadingArray

```
Headers    Cookies    Params    Response
Filter properties
▾ JSON
▾ 0: Array
    0: 1475235364413  ◀ Number of milliseconds since 12:00 AM on
                         January 1, 1970 (local to deployment)
    1: 15.505791      ◀ Sensor Reading
```

ER Cloud Helpers .NET Library

```
HttpBasicAuthProxy proxy = new HttpBasicAuthProxy(context);
proxy.ExecuteRequest(
    HttpBasicAuthProxy.Application.Viper,
    username,
    password,
    uri);
```