Virtual Power Solutions

iEnergy3 - API

Data Provisioning

March 2016

Index

1 Introduction

This section describes the Data Server application programming interface (API). This API is provided by the iEnergy3 for the other application or components to use. The API tries to conform to the RESTful principles of the HTTP protocol thus providing a machine friendly, robust and predictable interface to the system functionalities. Except otherwise noted, all data is to be transmitted using the JSON data format.

Before proceeding to the specification of the above mentioned API, there are four main concepts that are central to the Data Server platform and that should be clearly understood. These four concepts are:

- Local: A physical location that has the remote metering hardware installed. Locations usually correspond to a house or a building where the metering hardware is installed.
- Unit: The hardware that is located at the customer's location and that receives the data readings from devices. A unit usually communicates with the server using a TCP/IP connection.
- Device: Another piece of hardware that is located at the site being monitored. Devices can read several parameters (current, power, etc.) and send the data that has been read to its governing unit. There can be more than one device associated to unit. A device can be, for instance, a socket plug that is used to monitor the consumption of a specific appliance, or it can be a clamp that is used to monitor the current on a specific wire. A device usually communicates with its unit using some form of wired or wireless connection.
- Tag: A measurement end point that stores the data readings of a specific device. If a device measures current and power then it has two tags associated, one for current and another for power. A device may have one or more tags; in other words, it may measure one or more parameters.

All operations, except otherwise noted, must be invoked using HTTPS protocol and provide an HTTP Authorization header like "Authorization: ISA <session token>" where session token is received as a response when a session is created. See Sessions' methods documentation for more details about authentication.

To ease reading and add meaning, this document follows these simple writing conventions.

Variables are represented as <type: description of the value>. The possible data types are string, number, date, time and Boolean.

Resources can be represented in the following formats:

- Single resource – Resource
- List of resources - [Resource]
- Paged list of resources - [Resource]+

In result to each request performed to the API, a specific response code is returned in the reply to the requester. The following table lists the possible response codes and their meaning:

Response Status Code	Description
200 OK	Successful request. Response body contains the resource representation requested
201 Created	Resource was successfully created.
204 No Content	Resource was successfully updated and its representation didn't change compared to what was sent by the client. Or, resource was successfully deleted.
304 Not Modified	Your cached resource is still up-to-date.
400 Bad Request	The request contains invalid data, check error details.
401 Unauthorized	Insufficient credentials to access resource.
404 Not Found	Resource not found.
405 Method Not Allowed	The verb used in the request is incorrect.
500 Internal Server Error	General server side error. Please try again later.

1.1 General conventions

Operations that return lists of resources can return these as a list (represented as [Resource]) or as a paged list (represented as [Resource]+). In either case, the results can be filtered and ordered by fields defined on an operation basis.

Non-paged lists are returned as [] (arrays) of resources whereas paged lists are returned as {} (objects) with the following properties:

```
{
 "PAGE": <NUMBER: CURRENT PAGE NUMBER (ZERO BASED)>,
"PAGE_SIZE": <NUMBER: NUMBER OF MAXIMUM RECORDS PER PAGE>,
"TOTAL": <NUMBER: TOTAL NUMBER OF RECORDS>,
"LIST": [RESOURCE]
 }
```

Methods which return paged lists, accept the following URL parameters to customize the list returned:

```
?page=<page>&page_size=<page size>
```

All these parameters are optional. Defaults and allowed values are:

- page - default is 0;
- page_size - default is 50, allowed values are between 1 and 200;

Search operations allowing user-defined ordering specify valid order properties as:

Order: <PropertyName1>, <PropertyName2>

Software

For example, an operation documented as:

Order: Id, Type

Allows the caller to specify none, one or both properties as ordering. If none, the bold property is used as the default (ascending) order. If a '-' (minus) is prepended to a property, the ordering is descending. Order is specified as the 'order' query parameter with a list of values. All of the following are valid order parameters:

```
?order=[Id]
?order=[-Id]
?order=[Type]
?order=[Type,-Id]
```

Search operations allowing user-defined filters, specify valid filter properties as:

Filter: <PropertyName1>, <PropertyName2>

Filters are defined using a custom language which allows use logical and relational expression to build the desired filter. A simplified language grammar is:

Logical Expression		
Relational Expression	|	Relational Expression
	&	

Relational Expression		
Property	=	"string"
	<	number
	<=	@timestamp
	>	
	>=	
	!=	
Property	=%	"string"
	%=	
	%	
Property	in	["string",…,"string"]
		[number,…,number]

Symbol	Operation
&	Logical AND
|	Logical OR
=	Equals to
<	Less than
<=	Less or equal than
>	Greater than
>=	Greater or equal than
!=	Different from
=%	Starts with
%=	Ends with
%	Contains
In	Exists in list of values

The expected and returned format for the date data type, except where noticed, is the number of milliseconds between January 1, 1970, 00:00:00 UTC, and the desired date. For instance, 2012 January's first would be sent as 1325376000000.

The time data type is used for specifying a time stamp in a generic day. For this data type, the expected format, except where explicitly stated, is the number of milliseconds since the beginning of one day using the UTC standard. This means that the time stamp 20:00:00 of one day would be sent as 72000000. This also means that this data type only accepts values between 0 and 86400000, including these two values.

For numbers, the decimal separator used is '.' (dot).

1.2 API Entities

1.2.1 Actuation

This entity represents an actuation.

```
{

"ActuationLogId": <number: actuation log identifier>,

"Command": <string: command (0 for off, 1 for on and 3 for commuting the current
state to the other one)>,

"TagId": <number: tag identifier>

}
```

1.2.2 ActuatorState

This entity represents the actuation state of a tag

```
{

“TagId”: <number: tag identifier>,

“State”: <string: actuator state (0 for off, 1 for on or 3 for unknown, if the actuator did not reply for some reason)>

}
```

1.2.3 Consumption

This entiry represents a consumption report.

```
{ “UnitId”: <number: unit id>,

 “Granularity”: <string: type of granularity (instant, daily, hourly, monthly or yearly)>,

 “TagId”: <number: tag id>,

 “Date”: <date: begin date (using the unit time zone) of the time interval to which this consumption data corresponds>,

 “Read”: <number: value of the consumption in the default measurement unit of the respective tag type>,

 “ReadCurrency”: <number: value of Read converted to default currency of the supplier>,

 “ReadCarbon”: <number: value of Read converted to carbon dioxide values>,

 “Trees”: <number: number of trees necessary to compensate the emission of ReadCarbon>,

 “Cars”: <number: number of cars which emit ReadCarbon to the atmosphere>,

 “CurrencySymbol”: <string: symbol that represents the currency which is used in the value contained in the field ReadCurrency>

}
```

1.2.4 Device

This entity represents a device.

```
{

“Id”: <number: id>,

“Name”: <string: name>,

“DeviceTypeId”: <number: device type id>,
```

```
“TariffId”: <number: tariff id>,

“Tariff”: <Tariff: tariff>,

“ModuleType”: <[ ModuleType]: list of module types that are associated to the
device>,

“FixedCostId”: <number: id of the fixed cost associated to a tariff>,

“UnitId”: <number: unit id>,

“LocalId”: <number: local id>,

“Interval”: <number: interval between data acquisitions>,

“IsCommunicating”: <boolean: flag indicating if it is communicating>,

“Index”: <number: index of the device in the unit>,

“Address”: <string: Device ID that comes with the Device>,

“Deleted”: <boolean: flag indicating if the device is deleted or not>,

“State”: <Number: string: the device initial state. By default it is deactivated >,

}
```

1.2.5 Session

This entity represents a session in the API.

```
{

“Token”: <string: session token>,

“Timeout”: <number: session timeout in seconds>

}
```

1.2.6 Tag

This entity represents a tag.

```
{ “Id”: <number: id>,

 “UnitId”: <number: unit id>,

 “Name”: <string: name>,

 “Index”:<number: The address of the tag in device>,

 “LocationId”: <number: location id>,

 “TagTypeId”: <number: tag type id>,

 “Created”: <date: date when it was created>,
```

```
 “Active”: <boolean: flag indicating if the tag is active>,

 “Deleted”: <boolean: flag indicating if the tag is deleted>,

 “Communicate”: <boolean: flag indicating if the tag is communicating>,

 “Manual”: <boolean: flag indicating if the tag is manual>,

 “Visible”: <boolean: flag indicating if the tag is visible>,

 “Validate”: <boolean: flag indicating if the tag should be validated>,

 “InternalAddress”: <number: internal address>,

 “VirtualField”: <string: data value over which the virtual tag is calculated (value
of the tag reading or difference between two values)>,

 “DeviceId”: <number: id of the device>

}
```

1.2.7 Unit

This entity represents an unit

```
{

“Id”: <number: id>,

“Name”: <string: name>,

“State”: <number: unit state id>,

“MacAddress”: <string: mac address>,

“LocalId”: <number: local id>,

“ModuleType”: <[ ModuleType]: list of module types that are associated to the unit>,

“B64AccessKey”: <string: access key for direct communication with unit. Note:
currently this field has the same value for all units, but this will be changed in
the future>,

“TimeZoneId”: <string: time zone id for this unit>,

“HashedSerial”: <string: public unit identifier>,

“Interval”: <number: communication interval>,

“LastComm”: <date: date of last communication>,

“DomainId”: <number: id of the domain this unit belongs to>,

“TypeId”: <number: unit’s type id>,

“IsActive”: <boolean: flag indicating whether this unit is valid or not>,

“IsComm”: <boolean: flag indicating whether this unit is communicating>,

“Local”: <Local: local associated to the unit>,
```

```
"Deleted": <boolean: flag indicating if the unit is deleted or not>,

"Dev

"Devices": <[Device]: list of devices that are associated to the unit>,

MonthlyObjective": <number: decimal value that corresponds to the value, in the
currency associated to the user account, of the monthly objective for the unit>,

"LastConfigDate": <date: date of the last configuration change>

}
```

1.2.8 TagDatas

This entity represents a TagDatas element, used for insering data into the platform.

```
{

"TagDatas": <[TagData]: List of TagData elements>

}
```

1.2.9 TagData

This entity represents a TagData, used to carry data for a specific Tag ID.

```
{

"TagID": <number: the tag id>

"Values" : <[TagValue]: List of samples to insert into the indicated tagId>

}
```

1.2.10 TagValue

This entity represents a TagValue, used to carry a sample.

```
{

"Value": <number: the sample value>

"Date" : <number: the Unix timestamp of the sample>

}
```

1.3 API Operations

This section describes the most important API operations

1.3.1 Create a session on the API (Authenticate)

For creating a session on the API, the CreateSession method Logs a user in the API.

Request	*POST /sessions* *User* *Required fields:* *Login* *Password*
Responses	*201 Created* *Session* *Possible error codes:* *400 Bad Request* *null_or_empty.user.password* *invalid_length.user.password* *null_or_empty.user.login* *invalid_length.user.login* *invalid.social_provider* *null_or_empty.facebook.access_token* *null_or_empty.facebook.email* *null_or_empty.user.email* *401 Unauthorized* *no_permissions_to_access*

1.3.2 Find a Unit

The FindUnits method returns one list with the details of one set of units, optionally filtered and ordered by lists of its ids and local ids, and optionally showing the information of the local of each unit returned.

Request	*GET /units* *Include: Local,Device,FixedCost,Tariff,Interval* *Where: Id, LocalId, HashedSerial, IsComm* *Order:* ***Id****, LocalId, HashedSerial, IsComm*

	Required fields: *None* *Optional fields:* *Include* *Where* *Order*
Responses	*200 OK* *[Unit]+* *Possible error codes:* *400 Bad Request* *invalid.include_parameter* *invalid.where_parameter* *invalid.order_parameter* *404 Not Found* *not_found.unit* *not_found.local (not implemented yet)*

1.3.3 Find Tags

This method returns one list with the details of a set of tags, optionally filtered and ordered by lists of its ids, unit ids, tag type ids and device ids.

Request	*GET /tags* *Where: Id, UnitId, TagTypeId, DeviceId* *Order:* ***Id****, UnitId, TagTypeId, DeviceId* *Required fields:* *None* *Optional fields:* *Where*

	Order
Responses	*200 OK* *[Tag]+* *Possible error codes:* *400 Bad Request* *invalid.where_parameter* *invalid.order_parameter* *401 Unauthorized* *no_permissions_to_access.tag* *no_permissions_to_access.unit* *no_permissions_to_access.device* *404 Not Found* *not_found.unit* *not_found.tag_type* *not_found.device* *not_found.tag*

1.3.4 Get Consumptions

This method returns a set of consumptions filtered by granularity, begin date, end date and a list of tag identifiers. It is important to note that some of the tag types are instantaneous (for instance, temperature and humidity). In those cases the request to this method should include the optional field InstantsType, which will determine how the value of Read (one of the fields of the objects Consumption that are returned in this method) will be calculated. If the optional field InstantsType is passed with a null value, the method will treat the request as it was passed the value Avg in that field, for each tag that is noncumulative. If the tag is cumulative, the value passed in the field InstantsType is ignored.

Request	*GET /consumptions/<string: the type of aggregation in which we want to obtain the consumptions >?from=<string: begin date (in the unit time zone) of the time interval to which the calculation will be performed>&to=<string: end date (in the unit time zone) of the time interval to which the calculation will be performed>&tags=<string: list of comma separated tag identifiers, delimited by characters [and]>&instantsType=<string: the instants type>* *Granularity: Instant, Hourly, Daily, Monthly, Yearly*

	InstantsType: Avg, Max, Min, Stdev *Avg – Average* *Max – Maximum* *Min – Minimum* *Stdev – Standard deviation* *Required fields:* *Granularity* *From* *To* *Tags* *Optional fields:* *InstantsType*
Responses	*200 OK* *[Consumption]* *Possible error codes:* *400 Bad Request* *null_or_empty.granularity* *null_or_empty.from_date* *null_or_empty.to_date* *null_or_empty.tags* *invalid.consumption_granularity – invalid_value.granularity* *invalid.generic.dates – invalid_value.from_date ; invalid_value.to_date ; invalid_date_relation.from_date_vs_to_date* *invalid.instant_type – invalid_value.instant_type* *401 Unauthorized* *no_permissions_to_access.tag (not implemented yet)* *404 Not Found* *not_found.tag (not implemented yet)*

1.3.5 Actuate

This method creates a new actuation command and sends it, 'on the fly', to the respective unit.

Request	*POST /actuations* *Actuation* *Required fields:* *TagId* *Command* *Optional fields:* *None*
Responses	*201 Created* *Actuation* *Possible error codes:* *400 Bad Request* *non_actuation.tag* *invalid.command invalid_value.command* *401 Unauthorized* *no_permissions_to_use.tag– no_permissions_to_access.tag* *404 Not Found* *not_found.tag*

2 API Usage Scenario for Data Retrieval

In order to use the API, first an authentication has to be performed using the CreateSession method. This method needs a valid username and password and, if the authentication is valid, returns a token that can be used to identify the user in the following requests. An example of one possible pair request/response to this method is shown below.

Request

```
POST http://innov.isaenergy.pt:6600/api/1.4/sessions HTTP/1.1

Accept-Encoding: gzip,deflate

Content-Type: application/json

User-Agent: Jakarta Commons-HttpClient/3.1

Host: innov.isaenergy.pt:6600

Content-Length: 48

Body:

{

"Login":"my_username",

"Password":"my_password"

}
```

Response

```
HTTP/1.1 201 Created

Server: ASP.NET Development Server/10.0.0.0

Date: Fri, 29 Jul 2011 14:17:15 GMT

X-AspNet-Version: 4.0.30319

Content-Length: 117

Cache-Control: private

Content-Type: application/json; charset=utf-8

Connection: Close

{

"Data":null,

"RefreshToken":"RD5WUDp3X08ujvkUAhn+E92xaPeiVvyLH1N4lZVyFRzHT8+I2skqaQ51znDq4eru
adf09Qer9fmQxdFe7\/LRhO7GynFGZDw=",

"Timeout":7200,

"Token":"oQVokEr7ud6e2GuDQRnteQgD8Dq\/PUPsZ+AXqlsCTSujRLHYQq7D8mcI\/2n8k20L\/kDh
cRrCGt\/gthK7yxa10yVWp+xxPd0="

}
```

The authentication token is valid for the "timeout" value, which is in this case 900 seconds. After the timeout has expired, a new token must be generated using the same method.

After the authentication has been performed, a method to discover the user's units may be called: the FindUnits Method. An example request and response follows.

Request

```
GET http://innov.isaenergy.pt:6600/api/1.4/units?Where=Id%3D94%7CId%3D2&Order=-LocalId%2CId&Include=Local HTTP/1.1

Accept-Encoding: gzip,deflate

Accept: application/json

Authorization: ISA oQVokEr7ud6e2GuDQRnteQgD8Dq\/PUPsZ+AXqlsCTSujRLHYQq7D8mcI\/2n8k20L\/kDhcRrCGt\/gthK7yxa10yVWp+xxPd0=

User-Agent: Jakarta Commons-HttpClient/3.1

Host: innov.isaenergy.pt:6600

Body:
```

Response

```
HTTP/1.1 200 OK

Server: ASP.NET Development Server/10.0.0.0

Date: Fri, 29 Jul 2011 15:18:09 GMT

X-AspNet-Version: 4.0.30319

Content-Length: 508

Cache-Control: private

Content-Type: application/json; charset=utf-8

Connection: Close

{

"List": [

{

"B64AccessKey": "0001112223333444555",

"Id": 94,

"Local": {

"CountryCode": "PT",

"Id": 85,

"IsStorage": false,

"Name": "B0605-Trofa",

"UserId": 1,

"ZipCodeId": 190856
```

```
},
"LocalId": 85,
"MacAddress": "00-04-A3-22-81-D8",
"Name": "B0605-Trofa",
"State": 1
},
{
"B64AccessKey": "000111222333444555",
"Id": 2,
"Local": {
"CountryCode": "PT",
"Id": 4,
"IsStorage": false,
"Name": "iMeter ESI Testes",
"UserId": 1,
"ZipCodeId": 71756
},
"LocalId": 4,
"MacAddress": "00-04-A3-22-88-C7",
"Name": "iMeter Teste ESI",
"State": 1
}
],
"Page": 0,
"PageSize": 50,
"Total":2
}
```


This response shows the list of units in the database that match the request criteria and their respective information. The list is composed by 2 elements, the ones whose id is 94 or 2, using the fields “LocalId” (in decreasing order) and “Id” (in crescent order) for ordering. The page size used to show the results list is 50 and only the first page (page 0) of that list is shown (which is the only one, in this particular case). The information of the each unit’s local is also displayed.

The tags belonging to a unit can also be retrieved. For this task, the FindTags request should be issued. The following table illustrates an example of the retrieval of the tags from a Unit.

Request

```
GET http://innov.isaenergy.pt:6600/api/1.4/tags?
where=TagTypeId+in+%5B20001%2C21001%5D+%26+UnitId+%3D+2 HTTP/1.1

Accept-Encoding: gzip,deflate

Authorization: ISA
oQVokEr7ud6e2GuDQRnteQgD8Dq\/PUPsZ+AXqlsCTSujRLHYQq7D8mcI\/2n8k20L\/kDhcRrCGt\/g
thK7yxa10yVWp+xxPd0=

User-Agent: Jakarta Commons-HttpClient/3.1

Host: innov.isaenergy.pt:6600

Body:
```

Response

```
HTTP/1.1 200 OK

Server: ASP.NET Development Server/10.0.0.0

Date: Fri, 29 Jul 2011 15:18:09 GMT

X-AspNet-Version: 4.0.30319

Content-Length: 508

Cache-Control: private

Content-Type: application/json; charset=utf-8

Connection: Close

{

"List": [

{

"Active": true,

"Communicate": false,

"Created": 1311870060000,

"Deleted": false,

"DeviceId": 895,

"DomainId": 2,

"Id": 2684,

"InternalAddress": 0,

"LocationId": 151,

"Manual": false,

"Name": "Energia Activa",
```

```
"RemoteId": null,
"TagTypeId": 20001,
"UnitId": 2,
"Validate": false,
"VirtualField": null,
"Visible": true
},
{
"Active": true,
"Communicate": false,
"Created": 1311870060000,
"Deleted": false,
"DeviceId": 896,
"DomainId": 2,
"Id": 2687,
"InternalAddress": 0,
"LocationId": 151,
"Manual": false,
"Name": "Energia Activa",
"RemoteId": null,
"TagTypeId": 20001,
"UnitId": 2,
"Validate": false,
"VirtualField": null,
"Visible": true
}],
"Page": 0,
"PageSize": 50,
"Total": 2
}
```

This response shows the list of tags in the database that match the request criteria and their respective information. The list is composed by 2 elements, the ones whose tag type id is *20001* or *21001* and unit id is *2*, using the default ordering (by the field "Id", in crescent order). The page size used to show the results list is 50 and only the first page (page 0) of that list is shown (which is the only one, in this particular case).

The values from each tag may be also retrieved from the API. For this task the GetConsumptions method should be used. The following table illustrates a GetConsumptions request.

Request

Full path:
http://innov.isaenergy.pt:6600/api/1.4/consumptions/daily?to=1406105200000&from=1304982000000&tags=%5B2684%5D

HTTP method: *GET*

Authorization HTTP header: *"ISA oQVokEr7ud6e2GuDQRnteQgD8Dq\/PUPsZ+AXqlsCTSujRLHYQq7D8mcI\/2n8k20L\/kDhcRrCGt\/gthK7yxa10yVWp+xxPd0="*

Media type: *Application/json*

Body:

Response

Media type: *Application/json*

Body:


```
[
{
"Cars": 329.622,
"Date": 1311897600000,
"Granularity": "daily",
"Read": 109.874,
"ReadCarbon": 38.4559,
"ReadCurrency": 13.3052,
"TagId": 2684,
"Trees": 219.748,
"UnitId": 150
}
]
```