Use of Node Profiles on the EpiSensor Gateway

Document Ref: EPI-208-00

Introduction

This document describes the use of Node Profiles on the EpiSensor Gateway. A Node Profile is a description of a node, its properties and sensors, in a JSON file format.

Background and Motivation

The motivation for Node Profile was two-fold; When a node joins the Zigbee network, the epi-gateway software sets about querying details of all properties and all sensors of the node. Depending on the node type, this can take some time and the effect is worsened for a network with many complex nodes joining the network at the same time. The Node Profile feature was designed to simplify the process of obtaining node property and sensor details on the gateway.

It is a common scenario to make the same configuration of the same node types connected to one Gateway or across multiple Gateways. The Node Profile feature was also designed to partly automate this process.

Description of a Profile File

Profile files are presented in JSON format and are validated against a Node Profile Schema File (see Appendix A). There are three sections in a profile file. The first section describes the profile file and the nodes that the profile file is applicable to. The JSON keys are described in the following table:

Key Name	Description	Example

`product_code`	Value should be a string describing the product codes that this profile file applies to. Wildcards are not supported but the profile may be applied to any node that starts with the product_code key value. So for example a profile with product_code key value “TES” may be applied to all nodes with product code starting with “TES”, for example “TES-21”, “TES-11”, “TES-22”.
	This key is mandatory.
		`“TES-21”`
		`“TES”`
		`“ZEM-6”`

`profile_name`	Value should be a string which identifies this profile.	`“Default_TES_Profile”`

	This key is mandatory.	
`profile_version`	Value should be a string which identifies the version of this profile. This key is mandatory.	`"1.0"`
`node_firmware_version_min`	Value should be a string which identifies the minimum node firmware version that this profile applies to. The profile is only applicable to nodes with a firmware version greater than or equal to this value. This key is mandatory.	`"3.00"`
`node_firmware_version_max`	Value should be a string which identifies the maximum node firmware version that this profile applies to. The profile is only applicable to nodes with a firmware version less than or equal to this value. This key is mandatory.	`"9.99"`
`profile_description`	This key is optional. It may be used to provide a user-friendly description of the profile file, as a string value, up to a maximum of 128 characters.	
`is_default_profile`	This key value should be of type boolean and indicates whether or not this is a default node profile. The role of the default profile will be discussed in more detail later in this document. This key is optional. If the key is not present, the profile will be assumed to be a non default profile.	`true` or `false`

The next section of the profile file is an array which describes the node level properties. These are listed in a `node_properties` JSON array which can be empty if there are no node level properties to be configured. Each entry in the array is a JSON object describing the property. The property level JSON keys are described in the following table:

Key Name	Description	Example
`property_name`	This key value is a string which is the name of the property. This value is not used by the gateway software to identify the node level property but serves the purpose of making the profile file more human readable. This key is mandatory.	`“switch_type”`
`property_id`	This key value is an integer which is the property ID. The gateway software will use this key to uniquely identify the node level property. This key is mandatory.	`6260`
`property_value`	This key value is a string which is the value of the property. Even if the property value is an integer it is represented in the profile file as a string. The gateway software will convert as necessary. This key is mandatory.	`“0”`
`read_only`	This key indicates if the property is read-only or read-write. Read only properties cannot be modified on the node. This key is mandatory.	`true` or `false`

The third section of the profile file is an array which describes the sensor and the sensor level properties. These are listed in a `sensors` JSON array which can be empty if there are no sensors to be configured.
Each entry in the array is a JSON object describing the sensor. The sensor level JSON keys are described in the following table:

Key Name	Description	Example
`sensor_id`	This key value is an integer which is the sensor ID. The gateway software will use this key to uniquely identify the sensor. This key is mandatory.	`4096`
`export_enabled`	This key value is a boolean type. If set to true,	`true` or `false`

	data from this sensor will be exported from the gateway. If set to false, data from this sensor will not be exported from the gateway. This key is mandatory.	
`export_id`	This key value is a string which is the export identifier to be used for this property, up to a maximum of 128 characters. Care should be taken not to assign the same export_id to more than one sensor on the same gateway. This is not checked by the gateway software when the profile is uploaded to the gateway or applied to the node. This key is optional. There is limited scope so in most cases this key will be null.	
`sensor_name`	This key value is a string which is the name of the sensor. This key is optional.	`“Battery Level”`
`units`	This key value is a string which are the units of measure of the data reported by the sensor. This key is optional. If this key is present, it will override the default units of measure for this sensor on the gateway.	`“Volts”`
`abbreviated_units`	This key value is a string which are the abbreviated units of measure of the data reported by the sensor. This key is optional. If this key is present, it will override the default abbreviated units of measure for this sensor on the gateway.	`“V”`
`divider`	This key value is an integer which is the divider to be used on the data reported by the sensor. This key is optional. If this key is present, it will override the default divider for this sensor on the gateway.	`1000`
`sensor_properties`	This is a JSON array which can be empty if there are no sensor level properties to be configured. Each element in the JSON array describes a property using the same keys as for node level	

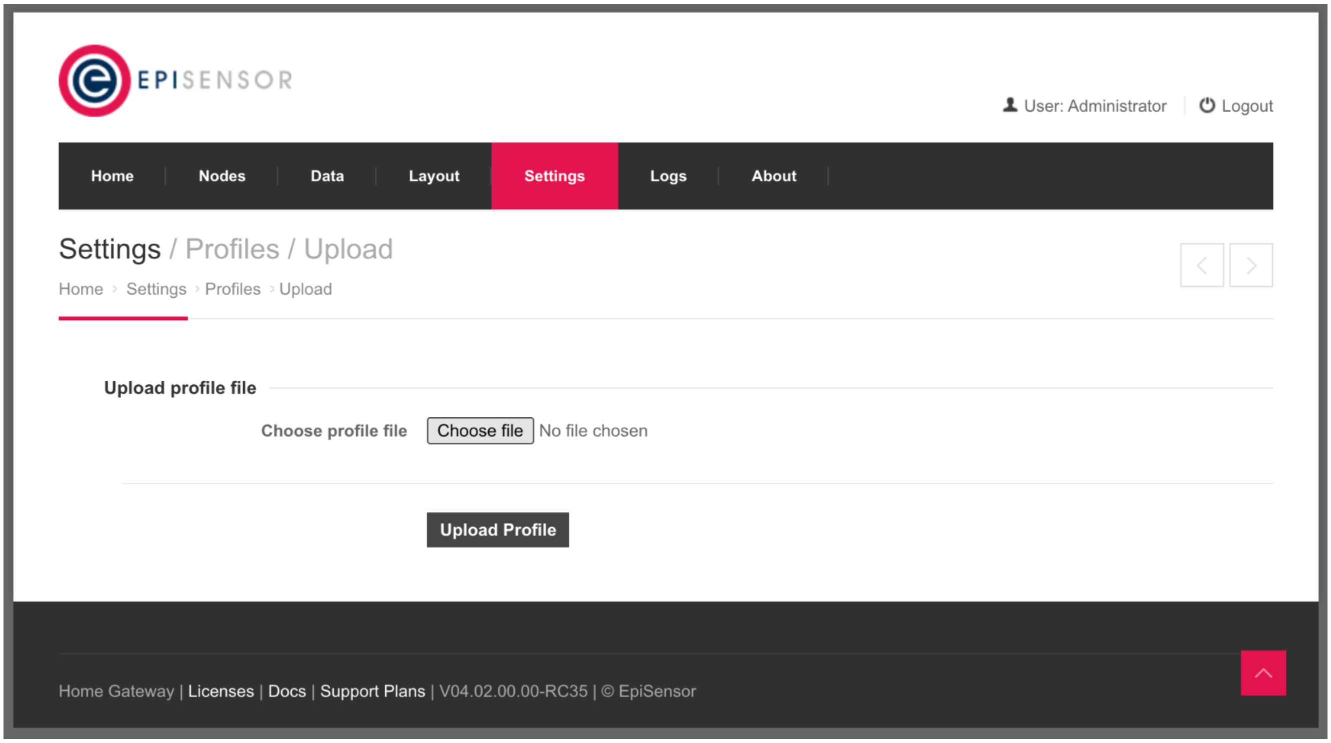
	properties (`property_name`, `property_id`, `property_value` and `read_only`). This key is mandatory but the array can be empty.	

Uploading Profiles to the EpiSensor Gateway

Profiles (via a profile file) must be uploaded to the gateway before they can be applied to nodes on that gateway. Profiles may be uploaded using the EpiSensor Gateway API or the web interface front end. Uploaded profiles will be immediately validated against the profile schema (see Appendix A). If validation fails, the profile will not be available on the gateway and an error code will be returned on the API or web interface.

Web Interface

From the EpiSensor Gateway web interface front end, the Settings->Profiles->Upload page allows for profile files to be uploaded from your local computer as shown in the following screenshot:



API

Profiles may be uploaded using the API using a PUT request to the settings/profiles endpoint. The profile file is specified as form-data in the request body.

An example in CURL is shown here (where the API password is `Admin/A1`):

```
curl --user Admin:A1 --location -g --request PUT '{{GATEWAY_URL}}/api/settings/profiles'
--form 'node_profile_file=@"/Users/jennymurphy/Downloads/TES-21_0012E95615-202105141111.json"'
```

Profile Management

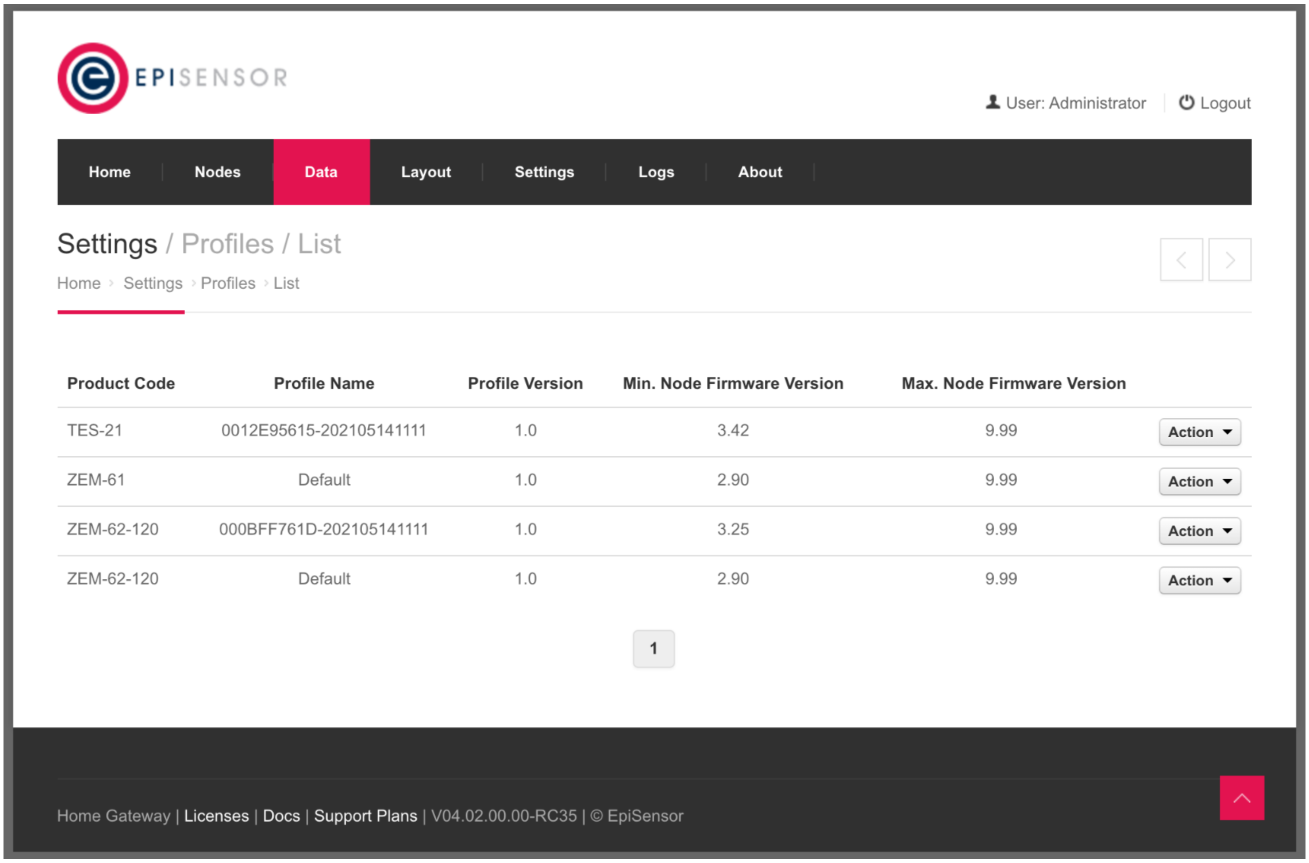
The scope of profile management on the EpiSensor Gateway includes listing, downloading and deleting files. These actions may be completed using the EpiSensor Gateway API or the web interface.

Web Interface

The profile management functionality is available from the Setting > Profiles > List page.

Listing

All valid profiles available on the gateway are listed on the Setting > Profiles > List as shown in the following screenshot. The product code, profile name, profile version and node firmware range are shown per profile. The profiles are listed alphabetically according to product code.



Settings / Profiles / List

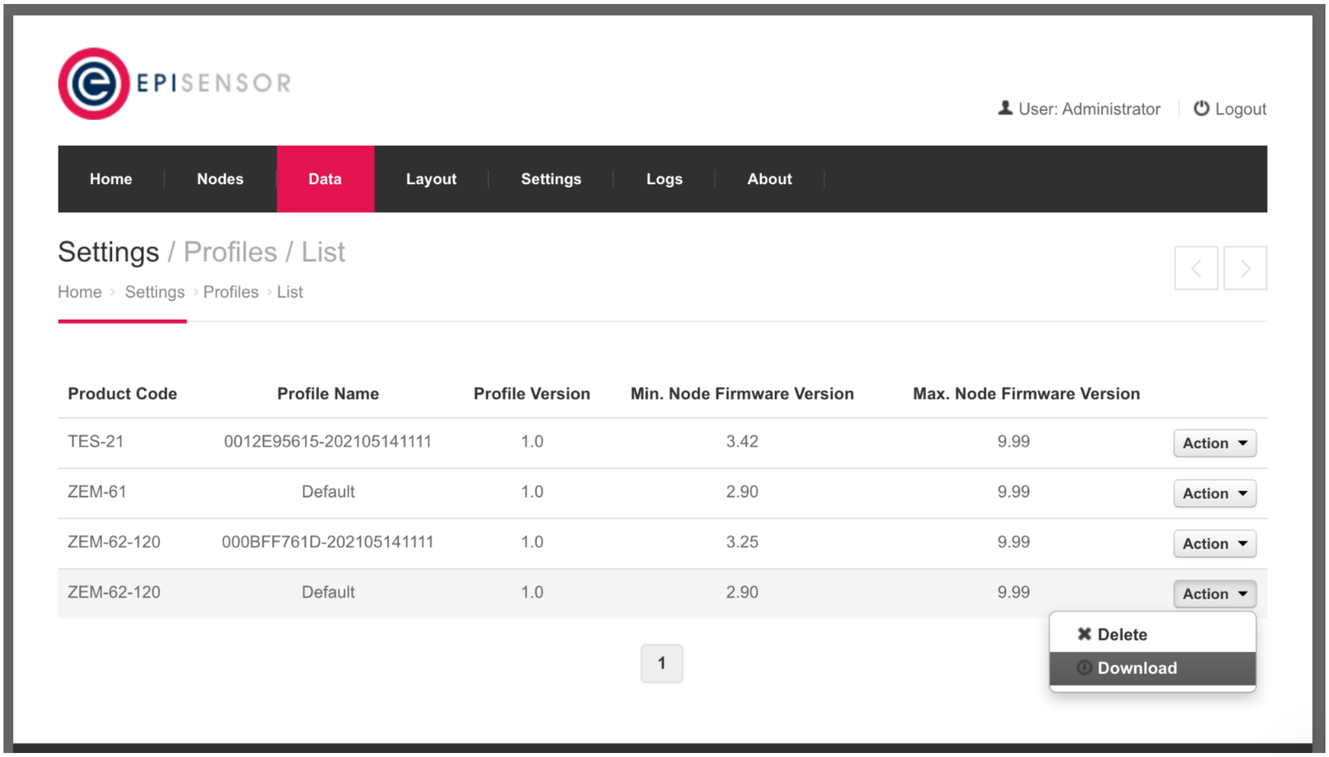
Home › Settings › Profiles › List

Product Code	Profile Name	Profile Version	Min. Node Firmware Version	Max. Node Firmware Version	
TES-21	0012E95615-202105141111	1.0	3.42	9.99	Action ▾
ZEM-61	Default	1.0	2.90	9.99	Action ▾
ZEM-62-120	000BFF761D-202105141111	1.0	3.25	9.99	Action ▾
ZEM-62-120	Default	1.0	2.90	9.99	Action ▾

Home Gateway | Licenses | Docs | Support Plans | V04.02.00.00-RC35 | © EpiSensor

Downloading

A listed profile may be downloaded to your local computer using the Action drop down list on the right hand side of the profile row as shown in the following screenshot.

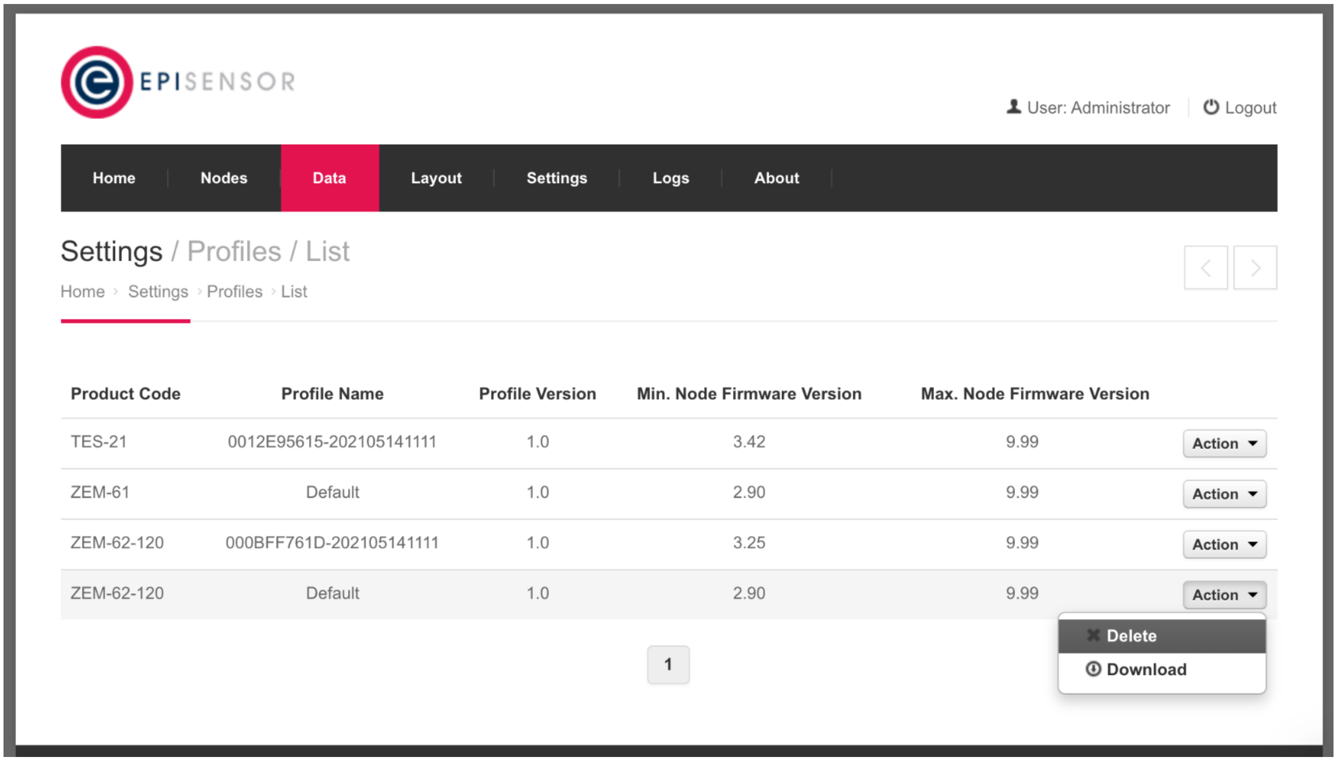


Settings / Profiles / List

Product Code	Profile Name	Profile Version	Min. Node Firmware Version	Max. Node Firmware Version	
TES-21	0012E95615-202105141111	1.0	3.42	9.99	Action
ZEM-61	Default	1.0	2.90	9.99	Action
ZEM-62-120	000BFF761D-202105141111	1.0	3.25	9.99	Action
ZEM-62-120	Default	1.0	2.90	9.99	Action

Deleting

A listed profile may be deleted from the gateway using the Action drop down list on the right hand side of the profile row as shown in the following screenshot.



Product Code	Profile Name	Profile Version	Min. Node Firmware Version	Max. Node Firmware Version	
TES-21	0012E95615-202105141111	1.0	3.42	9.99	Action ▾
ZEM-61	Default	1.0	2.90	9.99	Action ▾
ZEM-62-120	000BFF761D-202105141111	1.0	3.25	9.99	Action ▾
ZEM-62-120	Default	1.0	2.90	9.99	Action ▾

API

The profile management functionality is available from the /api/settings/profiles endpoint.

Listing

All valid profiles available on the gateway are listed in response to a GET request to the /api/settings/profiles endpoint.

An example in CURL is shown here (where the API password is `Admin/A1`):

```
curl --user Admin:A1  --location -g --request GET '{{GATEWAY_URL}}/api/settings/profiles'
```

For each profile the following information is shown:

```
{
        "is_default_profile": false,
        "node_firmware_version_max": "9.99",
        "node_firmware_version_min": "3.60",
        "product_code": "ZDR-16-r1k",
        "profile_description": "",
        "profile_name": "Default",
        "profile_version": "2.0",
        "uuid": "b896e4c4-873a-40b8-b873-14f543683b5e"
    },
```

The uuid field is important because it can be used to further specify a profile on the API.

Downloading

A profile on the gateway may be downloaded using the uuid specified for the profile in the profiles listing above. The profile is returned in response to a GET request to the /api/settings/profiles/{{uuid}} endpoint. The profile is returned in JSON format.

An example in CURL is shown here (where the API password is `Admin/A1`):

```
curl --user Admin:A1  --location -g --request GET
'{{GATEWAY_URL}}/api/settings/profiles/b896e4c4-873a-40b8-b873-14f543683b5e'
```

Deleting

A profile on the gateway may be deleted using the uuid specified for the profile in the profiles listing above. The profile is deleted in response to a DELETE request to the /api/settings/profiles/{{uuid}} endpoint.

An example in CURL is shown here (where the API password is `Admin/A1`):

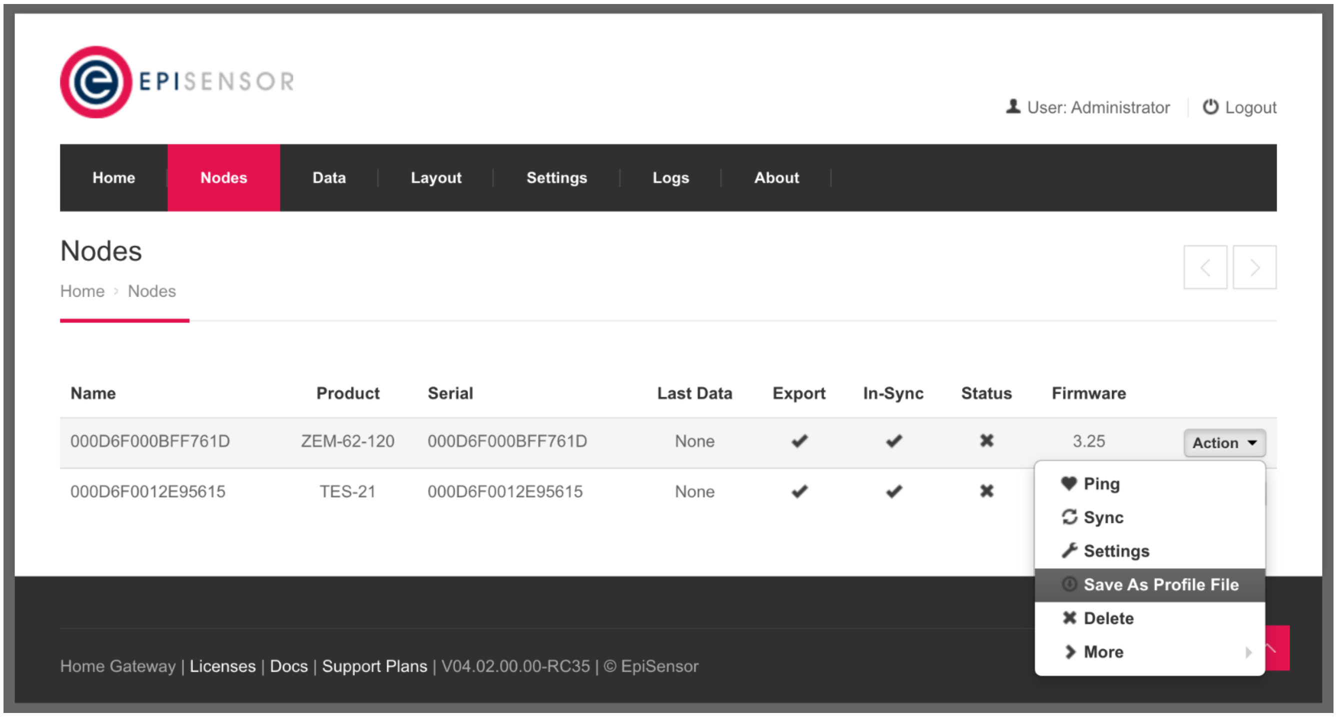
```
curl --user Admin:A1  --location -g --request DELETE
'{{GATEWAY_URL}}/api/settings/profiles/b896e4c4-873a-40b8-b873-14f543683b5e'
```

Saving Node Configuration as a Profile

A facility exists which allows the user to save the configuration of an existing node on the network as a profile file. The configuration includes the node property values, sensors and sensor property values. This can be convenient to replicate the configuration of a node to other nodes on the same network or on other networks. When a node configuration is saved as a profile file, it is provided to the user. This does not mean it exists as a profile on this gateway. To make the saved profile available on the gateway it must be uploaded to the gateway in a separate step. The profile name, description and the downloaded file name is auto-generated based on the node serial, the product code and a timestamp.

Web Interface

The configuration of an existing node may be saved as a profile file using the Action drop down menu in the Nodes List page as shown in the following screenshot. The profile file will be validated after it is generated and then downloaded to your local computer as a JSON file.



Nodes

Home › Nodes

Name	Product	Serial	Last Data	Export	In-Sync	Status	Firmware	
000D6F000BFF761D	ZEM-62-120	000D6F000BFF761D	None	✓	✓	✖	3.25	Action ▾
000D6F0012E95615	TES-21	000D6F0012E95615	None	✓	✓	✖		

Home Gateway | Licenses | Docs | Support Plans | V04.02.00.00-RC35 | © EpiSensor

API

The configuration of an existing node will be returned in response to a GET request to the /api/nodes/{{node_serial}}/profile endpoint. The profile will be in JSON format in the response body. As before, this profile is validated before it is returned. An example in CURL is shown here (where the API password is `Admin/A1`):

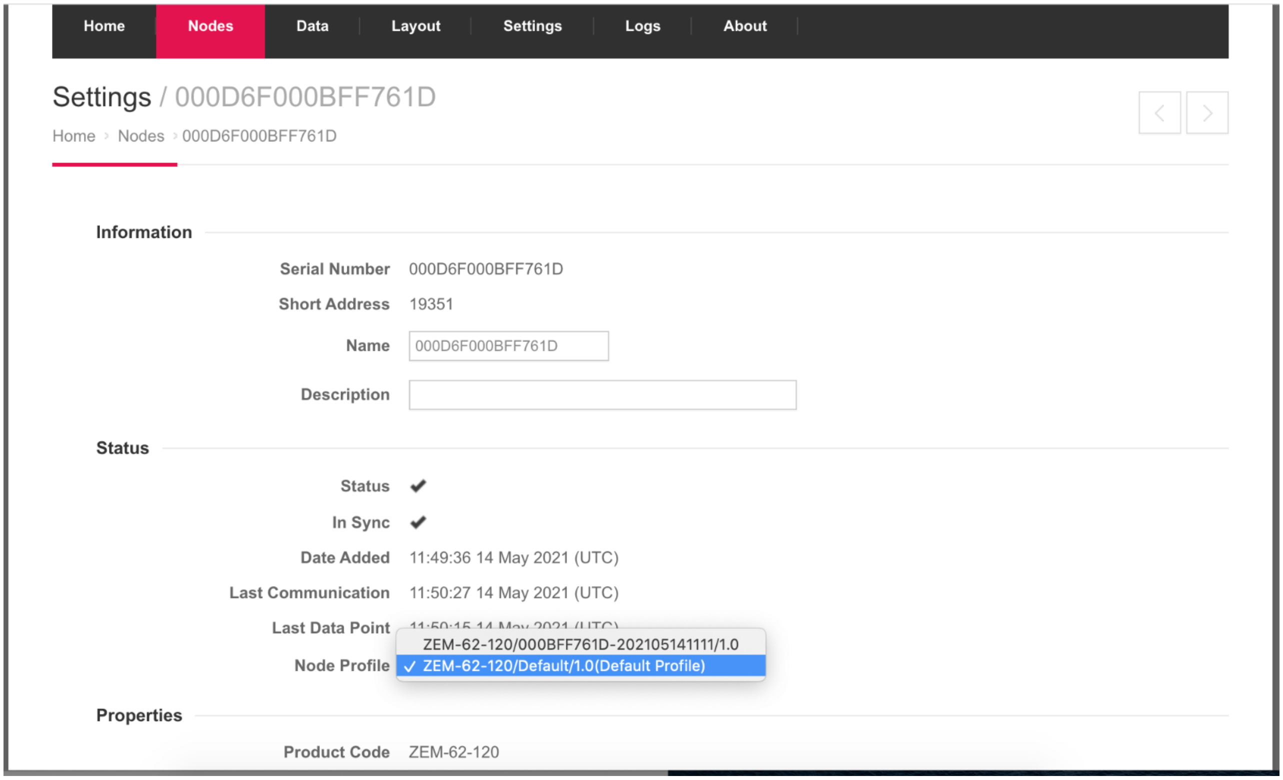
```
curl --user Admin:A1  --location -g --request GET
'{{GATEWAY_URL}}/api/nodes/{{NODE_SERIAL}}/profile'
```

Applying a profile to a node

A profile which has been uploaded to a gateway may be applied to a node on the gateway. Only profiles applicable to a node may be applied to a node. A profile is applicable to a node if the node code starts with or matches the `product_code` in the profile file AND the node firmware version is within the range specified by the `node_firmware_version_min` and `node_firmware_version_max` in the profile file.

Web Interface

The profiles applicable to a node are listed in the Node Profile drop down list on the Node Setting page. The profile which is currently applied to the node is marked with a tick. If no profile is applied to the node, ‘None’ will be shown in the drop down list. If there are no other (or none at all) profiles applicable to the node there will be no drop down list. After selecting the new profile, clicking ‘Save Changes’ will apply the profile to the node. The gateway software will apply the node property, sensors and sensor property settings to the node. This will include over-the-air commands to update the actual node and sensor property value.



Settings / 000D6F000BFF761D

Home › Nodes › 000D6F000BFF761D

Information

Serial Number 000D6F000BFF761D

Short Address 19351

Name 000D6F000BFF761D

Description

Status

Status ✓

In Sync ✓

Date Added 11:49:36 14 May 2021 (UTC)

Last Communication 11:50:27 14 May 2021 (UTC)

Last Data Point 11:50:15 14 May 2021 (UTC)

Node Profile
ZEM-62-120/000BFF761D-202105141111/1.0
✓ ZEM-62-120/Default/1.0(Default Profile)

Properties

Product Code ZEM-62-120

API

Similarly, using the EpiSensor Gateway API, a list of applicable profiles for a node is returned as part of the response to a GET request to the nodes endpoint for a specific node (or a GET request for a detailed view of all nodes). An example API response for a ZEM-62 node is shown below:

```
{
   "available_profiles": [
       {
           "is_default_profile": false,
           "node_firmware_version_max": "9.99",
           "node_firmware_version_min": "3.25",
           "product_code": "ZEM-62-120",
           "profile_description": "This profile was auto-generated from the settings of node
000D6F000BFF761D at date-time 2021-05-28 10:16:45.",
           "profile_name": "000BFF761D-202105281016",
           "profile_version": "1.0",
           "uuid": "cd123c54-8eb2-497d-9235-4bf23ee667ad"
       },
       {
           "is_default_profile": false,
           "node_firmware_version_max": "9.99",
           "node_firmware_version_min": "3.25",
           "product_code": "ZEM-62-120",
           "profile_description": "This profile was auto-generated from the settings of node
000D6F000BFF761D at date-time 2021-05-14 11:11:56.",
           "profile_name": "000BFF761D-202105141111",
           "profile_version": "1.0",
           "uuid": "0d8e4695-f099-497d-b5f1-d6efc3fa6107"
       }
   ],
   "child_list": [],
   "date_added": "2021-05-28T09:52:10",
   "description": "",
   "firmware_version": "3.25",
   "in_sync": true,
   "last_communication_date": "2021-05-28T10:33:06",
   "last_data_date": "2021-05-28T10:30:00",
   "name": "000D6F000BFF761D",
   "neighbour_list": [....],
   "power_amp": false,
   "product_code": "ZEM-62-120",
   "radio_power": 8,
   "routing_node": {....},
   "sensor_list": [....],
   "sensor_types": [],
   "serial_number": "000D6F000BFF761D",
   "status": true,
   "zem6x_properties": {
       "configuration": "NINE_S"
   }
}
```

In the above example, there are two available profiles for this node.

A profile may be applied to the node by sending a POST request to the node endpoint. The profile to apply may be specified using the uuid field or the key triplet of product_code, profile_name and profile_version. Examples in CURL is shown here (where the API password is `Admin/A1`):

```
curl --location -g --request POST '{{GATEWAY_URL}}/api/nodes/{{NODE_SERIAL}}' \
--data-raw '{
    "profile": {
        "product_code": "ZEM-62-120",
        "profile_name": "000BFF761D-202105281016",
        "profile_version": "1.0"
    }
}'
```

```
curl --location -g --request POST '{{GATEWAY_URL}}/api/nodes/{{NODE_SERIAL}}' \
--data-raw '{
    "profile": {
        "uuid": "af936e50-99fe-41f3-89bd-1be7c06b3a41"
    }
}'
```

After the profile has been applied, a subsequent GET request for that node will show the profile applied and the applicable profiles. An example is given below.

```
{
   "applied_profile": {
       "is_default_profile": false,
       "node_firmware_version_max": "9.99",
       "node_firmware_version_min": "3.25",
       "product_code": "ZEM-62-120",
       "profile_description": "This profile was auto-generated from the settings of node
000D6F000BFF761D at date-time 2021-05-14 11:11:56.",
       "profile_name": "000BFF761D-202105141111",
       "profile_version": "1.0",
       "uuid": "0d8e4695-f099-497d-b5f1-d6efc3fa6107"
   },
   "available_profiles": [
       {
           "is_default_profile": false,
           "node_firmware_version_max": "9.99",
           "node_firmware_version_min": "3.25",
           "product_code": "ZEM-62-120",
           "profile_description": "This profile was auto-generated from the settings of node
000D6F000BFF761D at date-time 2021-05-28 10:16:45.",
           "profile_name": "000BFF761D-202105281016",
           "profile_version": "1.0",
           "uuid": "cd123c54-8eb2-497d-9235-4bf23ee667ad"
       },
       {
           "is_default_profile": false,
           "node_firmware_version_max": "9.99",
           "node_firmware_version_min": "3.25",
           "product_code": "ZEM-62-120",
           "profile_description": "This profile was auto-generated from the settings of node
000D6F000BFF761D at date-time 2021-05-14 11:11:56.",
           "profile_name": "000BFF761D-202105141111",
           "profile_version": "1.0",
           "uuid": "0d8e4695-f099-497d-b5f1-d6efc3fa6107"
       }
   ],
   "child_list": [],
   "date_added": "2021-05-28T09:52:10",
   "description": "",
   "firmware_version": "3.25",
   "in_sync": true,
   "last_communication_date": "2021-05-28T10:49:47",
   "last_data_date": "2021-05-28T10:45:00",
   "name": "000D6F000BFF761D",
   "neighbour_list": [....],
   "power_amp": false,
   "product_code": "ZEM-62-120",
   "routing_node": {....},
   "sensor_list": [....],
   "sensor_types": [],
   "serial_number": "000D6F000BFF761D",
   "status": true,
   "zem6x_properties": {
       "configuration": "NINE_S"
   }
}
```

After the profile has been applied to the node, if there are any subsequent changes to the node settings (using the API or the web interface), the profile will no longer be strictly applied, and so the node settings page (or API response) will indicate that no profile is applied.

Default Profiles

A default profile is one which has the `is_default_profile` flag set to true in the profile file body. This means that when a node joins the gateway's network, and the profile is applicable to the node, it will automatically be applied to the node. A profile is applicable to a node if the node code starts with or matches the `product_code` in the profile file AND the node firmware version is within the range specified by the `node_firmware_version_min` and `node_firmware_version_max` in the profile file.

When a default profile is applied to a newly joined node, it means that the gateway will not query the node properties, sensors and sensor properties in the usual way. Instead it will use the information in the profile file to retrieve that information. As a result no over-the-air commands are issued and the node and sensor settings reported by the gateway will be according to the default profile.

In most cases the default profile should match the factory reset settings on the node.

More than one default profile

In the case that there is more than one applicable default profile for a node, the one that is applied will be the one that matches the node product code most closely. So for example if a TES-21 node arrives at the gateway and there are default profiles with `product_code` key "`TES`" and "`TES-2`", then the latter profile will be automatically applied to the new node.

Removing a profile from a node

The removal of a non-default profile from a node may be achieved by deleting the node from the gateway. A node may be deleted from the gateway using the web interface front end (Settings->Nodes page, node drop down list) or using the API (`{GATEWAY_URL}/api/nodes/{NODE_SERIAL}/command/deleteNode`).
The node will report to the gateway again and be added as a new node with no profile applied (unless there is an applicable default profile for that node).

It is important to note that when you delete the node which had a non default profile applied, it will reappear on the gateway but with the same settings as from the previously applied profile. If there is a default profile for the node, the default node profile will be applied to the node when it reappears at the gateway. However, there will be a mismatch between the reported node settings and the actual node settings in that scenario. A solution to this would be to factory reset the node and enable allow-join on the Gateway’s network to facilitate the node rejoining.

The removal of a default profile from a node may be achieved by deleting all applicable default profiles from the Gateway and then deleting the node from the Gateway.

The node will report to the Gateway again and be added as a new node with no profile applied).

Appendix A : Node Profile Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",

  "definitions": {
    "property": {
      "type": "object",
      "properties": {
        "property_name": {
          "type": "string",
          "maxLength": 128
        },
        "property_id": {
          "type": "integer" },
        "property_value": {
          "type": "string",
          "maxLength": 128
        },
        "property_value_type": {
          "type": "string",
          "maxLength": 8
        },
        "property_value_length": {
          "type": "integer" },
        "read_only": {
          "type": "boolean" }
      },
      "additionalProperties": true,
      "required": [
        "property_name",
        "property_id",
        "property_value",
        "read_only"]
    },
    "sensor": {
      "type": "object",
      "properties": {
        "sensor_id": {
          "type": "integer" },
        "export_enabled": {
          "type": "boolean" },
        "export_id": {
           "type": "string",
           "maxLength": 128
        },
        "sensor_name": {
           "type": "string",
           "maxLength": 256
        },
        "units": {
          "type": "string",
          "maxLength": 16
        },
        "abbreviated_units": {
          "type": "string",
          "maxLength": 16
        },
        "divider": {
          "type": "integer"
        },
        "sensor_properties": {
          "type": "array",
          "minItems": 0,
          "items":{"$ref": "#/definitions/property"}
        }
      },
      "additionalProperties": true,
      "required": [
        "sensor_id",
        "export_enabled",
        "sensor_properties"]
    }
  },

  "type": "object",
  "properties": {
    "product_code": {
      "type": "string",
      "maxLength": 16
    },
    "profile_name": {
      "type": "string",
      "maxLength": 64
    },
    "profile_version": {
      "type": "string",
      "maxLength": 8
    },
    "node_firmware_version_min": {
      "type": "string",
      "maxLength": 8
    },
    "node_firmware_version_max": {
      "type": "string",
      "maxLength": 8
    },
    "profile_description": {
      "type": "string",
      "maxLength": 128
    },
    "is_default_profile": {
      "type": "boolean"
    },
    "node_properties": {
      "type": "array",
      "minItems": 0,
      "items": {"$ref": "#/definitions/property"}
    },
    "sensors": {
      "type": "array",
      "minItems": 0,
      "items": {"$ref": "#/definitions/sensor"}
    }
  },
  "additionalProperties": true,
  "required": [
    "product_code",
    "profile_name",
    "profile_version",
    "node_firmware_version_min",
    "node_firmware_version_max",
    "node_properties",
    "sensors"
  ]
}
```

Appendix B: Sample Profile file

```
{
  "product_code": "TES-21",
  "profile_name": "TES_Profile_A",
  "profile_version": "1.0",
  "node_firmware_version_min": "3.00",
  "node_firmware_version_max": "9.99",
  "profile_description": "This profile may be used to configure your TES node.",
  "is_default_profile": false,
  "node_properties": [
    {
      "property_name": "command_list",
      "property_id": 6008,
      "property_value": "000E,000F,0010,000D",
      "read_only": true
    }
  ],
  "sensors": [
    {
      "sensor_id": 4096,
      "sensor_name": "Battery Level",
      "export_enabled": false,
      "units": "Volts",
      "abbreviated_units": "V",
      "divider": 1000,
      "sensor_properties": [
        {
          "property_name": "sensor_permissions",
          "property_id": 6211,
          "property_value": "0",
          "read_only": true
        },
        {
          "property_name": "reporting_mode",
          "property_id": 6202,
          "property_value": "5",
          "read_only": false
        },
        {
          "property_name": "reporting_interval",
          "property_id": 6203,
          "property_value": "15",
          "read_only": false
        },
        {
          "property_name": "logging_mode",
          "property_id": 6204,
          "property_value": "1",
          "read_only": false
        },
        {
          "property_name": "reporting_delta",
          "property_id": 6205,
          "property_value": "0.0",
          "read_only": false
        }
      ]
    },
    {
      "sensor_id": 350,
      "sensor_name": "Temperature T1",
      "export_enabled": true,
      "units": "C",
      "abbreviated_units": "C",
      "divider": 1,
      "sensor_properties": [
        {
          "property_name": "sensor_permissions",
          "property_id": 6211,
          "property_value": "0",
          "read_only": true
        },
        {
          "property_name": "reporting_mode",
          "property_id": 6202,
          "property_value": "5",
          "read_only": false
        },
        {
          "property_name": "reporting_interval",
          "property_id": 6203,
          "property_value": "15",
          "read_only": false
        },
        {
          "property_name": "logging_mode",
          "property_id": 6204,
          "property_value": "1",
          "read_only": false
        },
        {
          "property_name": "reporting_delta",
          "property_id": 6205,
          "property_value": "0.0",
          "read_only": false
        }
      ]
    }
  ]
}
```