ETAtouch RESTful Webservices

Version 1.1

October 31, 2013

Contents

1 Introduction

The ETAtouch device offers several web services to get access to the internal CAN subsystem. The webservices are implemented using the REST architectural style[1]. This allows the user to retrieve informations in a network-compatible, platform independent and programming language independent way. You can access the webservices of your ETAtouch device by sending your requests to port 8080 using the HTTP protocol:

http://ETAtouch-IP:8080/Resource

In order to use the ETAtouch webservices you must have ensured the following preconditions (take care of the sequence):

1. You must have installed system software version 1.20.0 or higher on your ETAtouch device.

2. You must have registered your ETAtouch device at `http://www.meineta.at`.

3. You must have applied for LAN access at `http://www.meineta.at` for your ETAtouch device.

4. You must have activated LAN access on your ETAtouch device in the system settings.

Figure 1 shows a use case diagram of the ETAtouch RESTful webservices. Figure 2 shows a typical sequence of requesting the menu tree which can be seen in the tree view on your ETAtouch device.

---

[1]`http://en.wikipedia.org/wiki/Representational_State_Transfer`

ETAtouch

Read menu tree

«extends» Read variable

Variable operations

«extends» Set variable

Read active errors

Create
variable set

«extends»

«extends» Read
variable set

Variable set
operations

«extends» Delete
variable set

«extends» Add variable

«extends»

Remove variable

User

Figure 1: Use cases of ETAtouch RESTful webservices.

```
sd ETAtouch REST

        c:HttpClient                                    s:ETAtouch

            GET http://{IP}:8080/user/menu HTTP/1.1

            xmlResponse: XmlResponse

        HTTP/1.1 200 OK
        Content-Encoding: utf-8
        Content-Type: application/xml
        <?xml version="1.0" encoding="utf-8"?>
        <eta version="1.0" xmlns="http://www.eta.co.at/rest/v1">
         <menu>
          <fub uri="/112/10021" name="Kessel">
            <object uri="/112/10021/0/0/12010" name="Counters">
             <object uri="/112/10021/0/0/12153" name="Full load hours"/>
            ...
          </fub>
         </menu>
        </eta>

            new                    :XmlParser

            parse(xmlResponse)
```
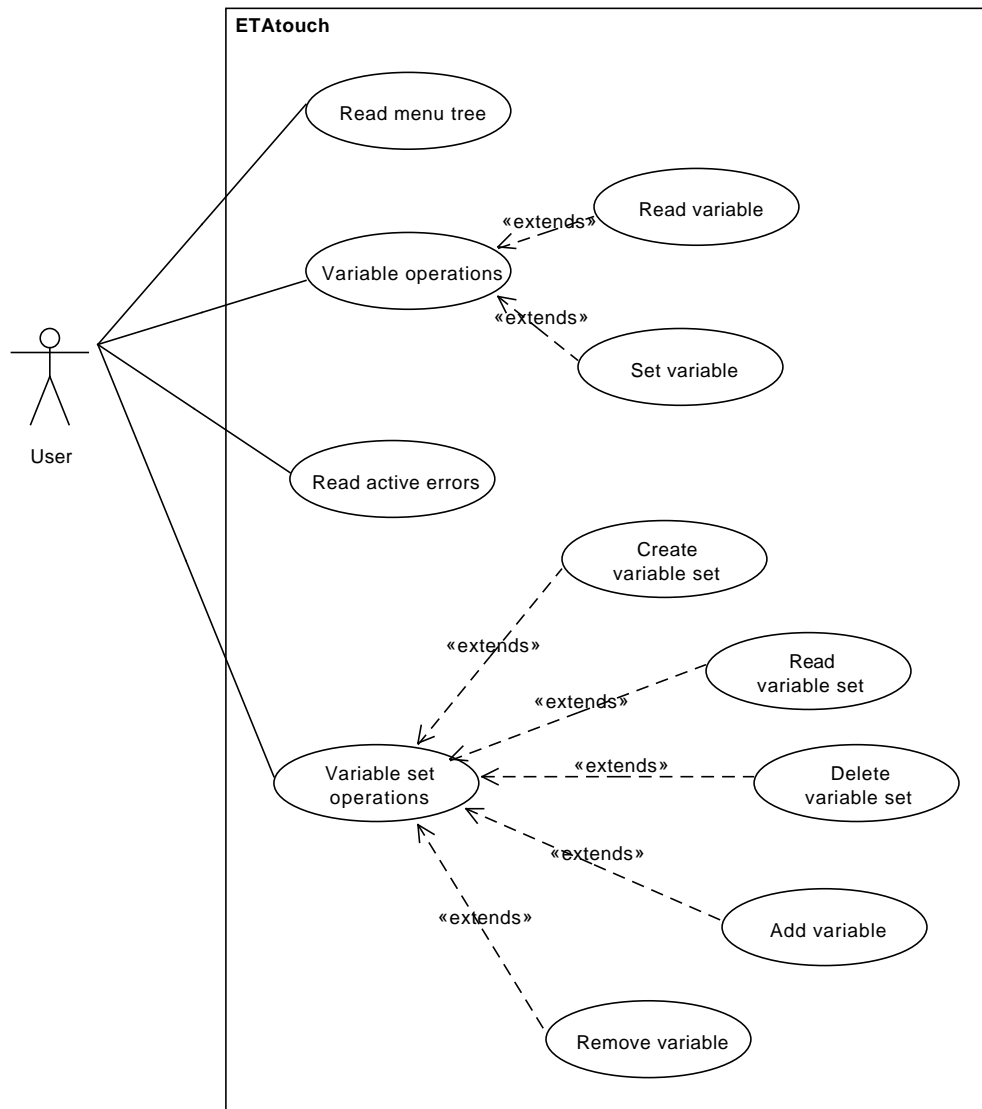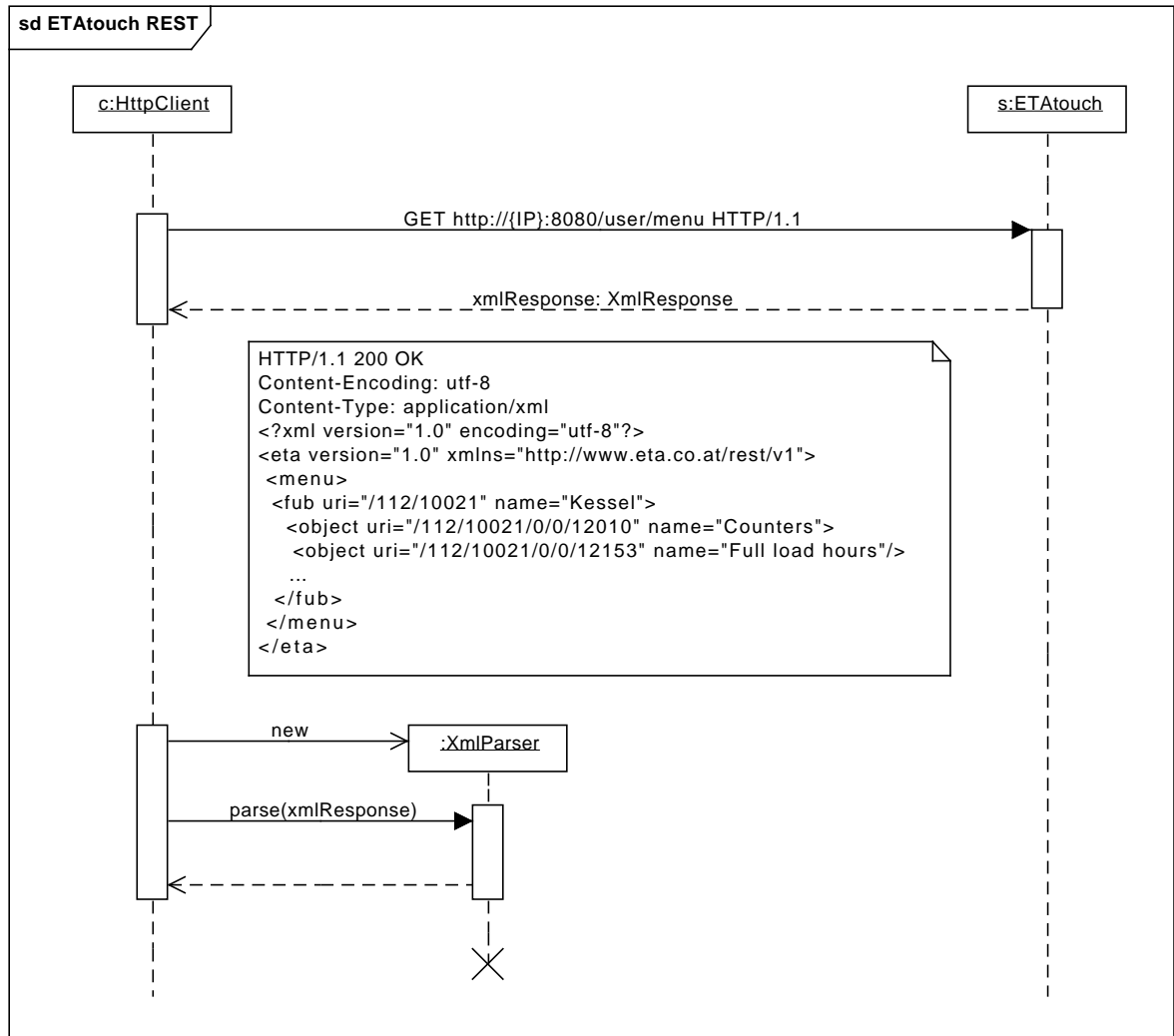
Figure 2: Typical sequence of getting the menu tree. After requesting the menu resource, the web service responses with the menu tree in XML format which can be processed by the client.

2 The resource /user/api

This resource returns the version of the API.

2.1 HTTP GET

Using HTTP GET you can read the current API version.

Example 1   Read the API version number.

─────────────────── *HTTP Request* ───────────────────
```
GET /user/api HTTP/1.1
```

─────────────────── *HTTP Response* ───────────────────
```
HTTP/1.1 200 OK
Content-Encoding: utf-8
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8"?>
<eta version="1.0" xmlns="http://www.eta.co.at/rest/v1">
  <api version="1.1" />
</eta>
```

2.2 HTTP POST

This method is not supported by this ressource.

2.3 HTTP PUT

This method is not supported by this ressource.

2.4 HTTP DELETE

This method is not supported by this ressource.

3 The resource /user/menu

This resource identifies the menu tree which you can see on the touch display in the tree view.

3.1 HTTP GET

Using HTTP GET you can read the menu tree as you can see it on the touch screen. Grouped by the various functional blocks you can find the particular objects in a hierarchical order.

Example 1 Retrieve the menu tree.

HTTP Request

```
GET /user/menu HTTP/1.1
```

HTTP Response

```
HTTP/1.1 200 OK
Content-Encoding: utf-8
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8"?>
<eta version="1.0" xmlns="http://www.eta.co.at/rest/v1">
  <menu>
    <fub uri="/112/10021" name="Kessel">
      <object uri="/112/10021/0/0/12010" name="Counters">
        <object uri="/112/10021/0/0/12153" name="Full load hours"/>
        ...
      </object>
      <object uri="/112/10021/0/0/12182" name="Miscellaneous">
        <object uri="/112/10021/0/0/12080" name="I/O key"/>
        <object uri="/112/10021/0/0/12112" name="Ash removal key"/>
        <object uri="/112/10021/0/0/12115" name="Emission Measurement"/>
        <object uri="/112/10021/0/0/12152" name="Feeding time"/>
      </object>
    </fub>
  </menu>
</eta>
```

3.2 HTTP POST

This method is not supported by this ressource.

3.3 HTTP PUT

This method is not supported by this ressource.

3.4 HTTP DELETE

This method is not supported by this ressource.

4 The resource /user/var

This resource identifies a variable in the CAN system.

7

4.1 HTTP GET

Using HTTP GET you can read a variable's value. Each variable is defined by a unique address in the CAN system. This address can be determined by evaluating the menu tree. Following you can see a snippet from the menu tree.

```
1  <object uri="/112/10021/0/0/12182" name="Miscellaneous">
2    ...
3    <object uri="/112/10021/0/0/12112" name="Ash_removal_key"/>
4    ...
5  </object>
```

Based on the XML snippet above, you can find the address 112/10021/0/0/12112 which identifies the ash removal key.

Example 1   Read the state of the ash removal key on CAN node 112 and Fub-ID 10021.

```
─────────────── HTTP Request ───────────────
GET /user/var/112/10021/0/0/12112 HTTP/1.1
```

```
─────────────── HTTP Response ───────────────
HTTP/1.1 200 OK
Content-Encoding: utf-8
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8"?>
<eta version="1.0" xmlns="http://www.eta.co.at/rest/v1">
  <value uri="/user/var/112/10021/0/0/12112" strValue="Off" unit=""
    decPlaces="0"  scaleFactor="1" advTextOffset="1802">1802</value>
</eta>
```

As you can see, the resulting XML document contains an element named `<value>`. The content of this element is the variable's raw value. Please refer to table 1 for a description of the attributes.

4.2 HTTP POST

Using HTTP POST you can set a variable to a certain value. Note, that you have to provide the the variable's raw value (without scaling)! If a variable with time interval datatype should be set, *value*, *begin* and *end* parameters must be provided. The begin and end time are integral multiples of 15 minutes since midnight. This implies a valid range of $[0, 96]$ for *begin* and *end*.

Note: The feature of setting the time slots requires an API version of 1.1 or higher (available since system software version 1.25.0/2.25.0 or higher).

Attribute	Description
uri	This is the requested URI.
strValue	The variable's value as formatted string.
unit	The variable's unit as string.
decPlaces	The number of decimal places.
scaleFactor	The scale factor for processing the raw value.
advTextOffset	The offset of text variables. If you read a text variable (e.g. the ash removal key) you can subtract this value from the variable's raw value in order to get a value from $[0, max]$ for e.g. boolean evaluations.

Table 1: Description of XML attributes of element `<value>`.

Example 1 Set the deash key to *On*.

HTTP Request

```
POST /user/var/112/10021/0/0/12112 HTTP/1.1
Content-Type: application/x-www-form-urlencoded

value=1803
```

HTTP Response

```
HTTP/1.1 200 OK
Content-Encoding: utf-8
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8"?>
<eta version="1.0" xmlns="http://www.eta.co.at/rest/v1">
  <success uri="/user/var/112/10021/0/0/12112"/>
</eta>
```

Example 2 Set monday's time slot 1 of the hot water tank's charging times to 00:00-12:00, 40 degrees:

HTTP Request

```
POST /user/var/112/10111/12130/0/1082 HTTP/1.1
Content-Type: application/x-www-form-urlencoded

value=400&begin=0&end=48
```

HTTP Response

```
HTTP/1.1 200 OK
Content-Encoding: utf-8
```

```
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8"?>
<eta version="1.0" xmlns="http://www.eta.co.at/rest/v1">
  <success uri="/user/var/112/10111/12130/0/1082"/>
</eta>
```

4.3 HTTP PUT

This method is not supported by this ressource.

4.4 HTTP DELETE

This method is not supported by this ressource.

5 The resource /user/vars

This resource identifies a set of variables. It allows you to define your own named variable set which acts as a container for different single variables. Having once created a variable set you can add or remove arbitrary variables from the CAN system to this set. The name of a variable set must match the following regular expression:

$$[a - zA - Z0 - 9]^{+}$$

The variable sets you have created are not persisted, so after a reboot the variable sets are lost.

5.1 HTTP GET

Using HTTP GET you can request two resources. (see table 2).

Resource	Description
/user/vars	Get a list of all defined variable sets.
/user/vars/{Varset}	Read all variables from the specified variable set *Varset*.

Table 2: Description of resources under /user/vars.

Example 1   Get a list of defined variable sets.

i

HTTP Request
```
GET /user/vars HTTP/1.1
```

```
───────────── HTTP Response ─────────────
HTTP/1.1 200 OK
Content-Encoding: utf-8
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8"?>
<eta version="1.0" xmlns="http://www.eta.co.at/rest/v1">
  <vars uri="/user/vars/myset1"/>
</eta>
```

Example 2 Read variable set *myset1*.

```
───────────── HTTP Request ─────────────
GET /user/vars/myset1 HTTP/1.1
```

```
───────────── HTTP Response ─────────────
HTTP/1.1 200 OK
Content-Encoding: utf-8
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8"?>
<eta version="1.0" xmlns="http://www.eta.co.at/rest/v1">
  <vars uri="/user/vars/myset1">
    <variable uri="112/10021/0/0/12112" strValue="Off" unit=""
      decPlaces="0" scaleFactor="1" advTextOffset="1802">1802</variable>
  </vars>
</eta>
```

5.2 HTTP POST

This method is not supported by this ressource.

5.3 HTTP PUT

HTTP PUT is used to create resources on the server. The semantics of HTTP PUT depends on your requested URI (see table 3).

Example 1 Create a new variable set named *myset1*.

```
───────────── HTTP Request ─────────────
PUT /user/vars/myset1 HTTP/1.1
```

```
───────────── HTTP Response ─────────────
HTTP/1.1 201 Created
Content-Encoding: utf-8
```

Resource	Description
/user/vars/{Varset}	Create a new variable set named *Varset*.
/user/vars/{Varset}/{Node-ID}/{Fub-ID}/{Fkt-ID}/{Io-ID}/{Var-ID}	Add the variable specified by the address *Node-ID/Fub-ID/Fkt-ID/Io-ID/Var-ID* to the variable set named *Varset*.

Table 3: Description of resources under `/user/vars`.

```
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8"?>
<eta version="1.0" xmlns="http://www.eta.co.at/rest/v1">
  <success uri="/user/vars/myset1"/>
</eta>
```

Example 2   Add the variable 112/10021/0/0/12112 to the variable set *myset1*.

— *HTTP Request* —
```
PUT /user/vars/myset1/112/10021/0/0/12112 HTTP/1.1
```

— *HTTP Response* —
```
HTTP/1.1 201 Created
Content-Encoding: utf-8
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8"?>
<eta version="1.0" xmlns="http://www.eta.co.at/rest/v1">
  <success uri="/user/vars/myset1/112/10021/0/0/12112"/>
</eta>
```

5.4 HTTP DELETE

HTTP DELETE is used to delete resources on the server. The semantics of HTTP DELETE depends on your requested URI (see table 4). If you do not need a variable set any more your client software should ensure, that all created variable sets are deleted in order to avoid resource conflicts.

Example 1   Remove the variable 112/10021/0/0/12112 from the variable set *myset1*.

— *HTTP Request* —
```
DELETE /user/vars/myset1/112/10021/0/0/12112 HTTP/1.1
```

Resource	Description
/user/vars/{Varset}	Delete the variable set named *Varset*.
/user/vars/{Varset}/{Node-ID}/{Fub-ID}/{Fkt-ID}/{Io-ID}/{Var-ID}	Remove the variable specified by the address *Node-ID/Fub-ID/Fkt-ID/Io-ID/Var-ID* from the variable set named *Varset*.

Table 4: Description of resources under `/user/vars`.

```
————————— HTTP Response —————————
HTTP/1.1 200 OK
Content-Encoding: utf-8
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8"?>
<eta version="1.0" xmlns="http://www.eta.co.at/rest/v1">
  <success uri="/user/vars/myset1/112/10021/0/0/12112"/>
</eta>
```

Example 2 Delete the variable set *myset1*.

i

```
————————— HTTP Request —————————
DELETE /user/vars/myset1 HTTP/1.1
```

```
————————— HTTP Response —————————
HTTP/1.1 200 OK
Content-Encoding: utf-8
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8"?>
<eta version="1.0" xmlns="http://www.eta.co.at/rest/v1">
  <success uri="/user/vars/myset1"/>
</eta>
```

6 The resource /user/errors

This resource identifies the active errors in the CAN system.

6.1 HTTP GET

Using HTTP GET you can read the active errors. A description of the resources can be found in table 5.

Resource	Description
/user/errors	Retrieve all active errors from the CAN system.
/user/errors/{Node-ID}	Retrieve all active errors from the CAN node with node number *Node-ID*.
/user/errors/{Node-ID}/{Fub-ID}	Retrieve all active errors from functional block *Fub-ID* on the CAN node with node number *Node-ID*.

Table 5: Description of resources under /user/errors.

Example 1 Retrieve all active errors from the system.

HTTP Request

```
GET /user/errors HTTP/1.1
```

HTTP Response

```
HTTP/1.1 200 OK
Content-Encoding: utf-8
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8"?>
<eta version="1.0" xmlns="http://www.eta.co.at/rest/v1">
  <errors uri="/user/errors">
    <fub uri="/112/10021" name="Kessel">
      <error msg="Flue gas sensor Interrupted" priority="Error"
        time="2011-06-29 12:47:50">Sensor or Cable broken or badly connected
      </error>
      <error msg="Water pressure too low 0,00 bar" priority="Error"
        time="2011-06-29 12:48:12">Top up heating water! If this warning
          occurs more than once a year, please contact plumber.
      </error>
    </fub>
    <fub uri="/112/10101" name="HK1"/>
  </errors>
</eta>
```

Example 2 Retrieve all active errors from CAN node 112.

HTTP Request

```
GET /user/errors/112 HTTP/1.1
```

```
                              ── HTTP Response ──
<?xml version="1.0" encoding="utf-8"?>
<eta version="1.0" xmlns="http://www.eta.co.at/rest/v1">
  <errors uri="/user/errors/112">
    <fub uri="/112/10021" name="Kessel">
      <error msg="Flue gas sensor Interrupted" priority="Error"
        time="2011-06-29 12:47:50">Sensor or Cable broken or badly connected
      </error>
      <error msg="Water pressure too low 0,00 bar" priority="Error"
        time="2011-06-29 12:48:12">Top up heating water! If this warning
          occurs more than once a year, please contact plumber.
      </error>
    </fub>
    <fub uri="/112/10101" name="HK1"/>
  </errors>
</eta>
```

Example 3 Retrieve all active errors from functional block 10021 on CAN node 112.

```
                              ── HTTP Request ──
GET /user/errors/112/10021 HTTP/1.1
```

```
                              ── HTTP Response ──
HTTP/1.1 200 OK
Content-Encoding: utf-8
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8"?>
<eta version="1.0" xmlns="http://www.eta.co.at/rest/v1">
  <errors uri="/user/errors">
    <fub uri="/112/10021" name="Kessel">
      <error msg="Flue gas sensor Interrupted" priority="Error"
        time="2011-06-29 12:47:50">Sensor or Cable broken or badly connected
      </error>
      <error msg="Water pressure too low 0,00 bar" priority="Error"
        time="2011-06-29 12:48:12">Top up heating water! If this warning
          occurs more than once a year, please contact plumber.
      </error>
    </fub>
  </errors>
</eta>
```

6.2 HTTP POST

This method is not supported by this ressource.

6.3 HTTP PUT

This method is not supported by this ressource.

6.4 HTTP DELETE

This method is not supported by this ressource.