VSCP Daemon Decision Matrix

The decision matrix of the VSCP daemon works much the same as a decision matrix on a low end node with the difference that it is much more capable. Events coming in through any of the interfaces of the VSCP daemon is all feed through the internal decision matrix, DM. There are also some events that are generated internally by the VSCP daemon (CLASS2.VSCPD) that is feed through the matrix.

Action parameter can be of any length. There is also no limit in the number of DM rows available even if there naturally is a practical system limit.

On late releases of the VSCP daemon the DM is stored in a database on disk which is read into memory at start up. On older systems and as an option the matrix can also be loaded form an XML file. While the database is configured through the web interface the XML file can be configured using a simple text editor.

Tags

row

The **row** tag specify one DM entry. The **enabled** attribute, which can be “true” or “false” to enable or disable a row. The **groupid** attribute can be used to group several DM-rows together with a user specified string value.

mask

The **mask** tag specify the mask for which part of an event that is of interest. Attributes are **priority**, **class**, **type** and **guid**. Bits of interest should be set to a one. So to compare the class set in filter set the corresponding mask to 0xffff or 65535 and so on. GUID should be on the form FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF, that is all numbers are in hexadecimal without the usual initial “0x”.

Simplest way of looking on the mask is to set it to zero if the tag should not be compared or to the max value if not. Max values are

Tag	Max value
Priority	0x0f/15
Class	0xFFFF/65535
Type	0xFFFF/65535
GUID byte	FF (no initial “ox”)

To trigger on a specific event. Set the mask for the class and the type both to 0xffff and the corresponding class/type of the filter to the event of interest. So in the sample below events of any priority and from any GUID will trigger if the class/type is the same as set in the filter.

```
<mask
```

```
    priority="0"
    class="0xFFFF"
    type="0xFFFF"
    guid="00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00">
</mask>
```

The sample below will not test priority,class, and the upper fifteen bytes of the GUID. But will test the LSB. So events of any priority and with any class/type but with the LSB if the GUID set to the value set in the corresponding GUID position of teh filter will trigger the DM action.

```
<mask
    priority="0"
    class="0"
    type="0"
    guid="00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:FF">
</mask>
```

filter

The **filter** tag specify the filter for which part of an event that is of interest. Attributes are **priority**, **class**, **type** and **guid**. Set the attribute to the value of interest and enable a test of it by setting the corresponding mask to all ones. For example to test for events with class=10 set the class attribute of the filter to 10 and the class attribute of the mask to 0xffff/65535. GUID should be on the form FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF, that is all numbers are in hexadecimal without the usual initial “0x”.

The simple way to look at the mask/filter combination is to set a mask to zero if the corresponding filter part should not be compared and is a don't care. You can create more complex scenarios of course as explained below.

The filter is combined with the mask. First the incoming event is masked this means all bits where the mask is zero is also set to zero in the event. The result is compared to the filter, if they are the same we have a match and the decision matrix row is triggered.

If you set the mask for a row to all ones and the filter equal to the event you are interested in, then that row will be triggered when that event is received by the daemon. In this case you can have the mask for priority, GUID set to zero so that events with all priorities and from all units (with different GUID's) will trigger the row. If you only are interested in events from a specific node set the GUID part of the mask to ones (FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF) and the GUID part for the filter to the node's GUID.

Another way one can use to match a single event is to set both filter and mask to the same value. This will have the same effect as the above method. Just remember that the mask should have a zero in a bit position for a don't care and a one for a significant bit and the filter should have the corresponding bit set to the value that is wanted and all other bits set to zero.

The truth table for all this looks like this:

filter ^ event-bit	mask	result
0	0	1

filter ^ event-bit	mask	result
0	1	1
1	0	1
1	1	0

control

Deprecated from version 1.12.11.0

This is the 32-bit DM control word.

Bit	Description
31	Reserved.
30	This bit can be used to disable all decision matrix entries below the row that has this bit set.
5	Specifies that the index should be checked.
4	Specifies that the zone should be checked.
3	Specifies that the sub-zone should be checked.

index

Index is in the first data byte for events that support it. If the **bEnable** attribute is set to *true* a check will be done to check if the value set here is equal to the first byte of the incoming events data.

measurementindex

Measurement events (for example CLASS1.MEASUREMENT events) usually have an index that identify a specific sensor (0-7 or 0-255). If the **bEnable** attribute is set to *true* a check will be done to check if the measurement index of the incoming event have a value equal to the one set here.

zone

If the **bEnable** attribute is set to *true* a check will be done to check if the value set here is equal to the second byte of the incoming events data.

subzone

If the **bEnable** attribute is set to *true* a check will be done to check if the value set here is equal to the third byte of the incoming events data.

allowed_from

This is the ISO date time from which the DM action can happen. The format is a standard ISO date/time on the form YYYY-MM-DD HH:MM:SS or YYYY-MM-DDTHH:MM:SS.

allowed_to

This is the ISO date time up to which the DM action can happen. The format is a standard ISO date/time on the form YYYY-MM-DD HH:MM:SS or YYYY-MM-DDTHH:MM:SS.

allowed_weekdays

This is a string with eight entries each representing a day of the week. If a DM action is allowed to occur every day of a week the string should be given as “mtwtfss”. Use a dash for a day which the action is not allowed to occur. For example if the DM action can occur on all days except Wednesdays use the string “mt-tfss”.

allowed_time

This is a string that tells at which time(s) a DM event is allowed to occur at. The string has the form

```
YYYY-MM-DD HH:MM:SS
```

You can set any of the items to a * meaning any time so

- -*-* *:*:*

is any time. You can also use a slash to give a list of times. So

- -*-* *:0/5/10:0

will trigger the DM action every hour, five minutes over the hour and ten minutes over the hour, every day, every month, every year. On the other hand

- -1/6-* *:0/5/10:0

will just do the same in January and in June.

measurement

Added in version 1.12.11.0

This makes it possible to compare a measurement value for measurement events (for example CLASS1.MEASUREMENT events) with a pre-set value defined here.

Format is

```
<measurement enable="true|false" compare="noop|eq|neq|gt|gteq|lt|lteq"
unit="0" value="78.2" />
```

where the comparison value is represented by “78.2” here and unit is set to the default value zero.

The attribute **compare** defines which comparison that should be performed with the following possible values

Comparison	Alternative	Internal code	Description
noop	empty	0	No comparison. **Default**
eq	**==**	1	Check for equal.
neq	**!=**	2	Check for NOT equal.
gt	**>**	3	Check for greater than.
gteq	**>=**	4	Check for greater than or equal.
lt	**<**	5	Check for less than.
lteq	**⇐**	6	Check for less than or equal.

The attribute **unit** is a numerical code that specifies which unit the incoming value should have (0-3 for Level I/0-255 for Level II).

Sensor index for the sensor is set in the measurementindex tag and is (0-7 for Level I/0-255 for Level II).

action

This is a 32-bit action code that specifies which action should be executed if the row is triggered. Actions and their codes are described here.

param

The parameter is a text string that makes it possible to control how the action is performed. Before the action is performed the string is checked for escapes and this makes it possible to pass run time information in an easy way. Escapes are defined here.

comment

Make your own comments of what the row is there for here.

The decision matrix XML file

Note! *The XML file format is deprecated and replaced by a database which is edited in the web interface. It is however still possible to read DM entries from the XML file.*

When the daemon is started up the internal decision matrix is loaded from the dm.xml in the folder set as configuration path.

The dm.xml file can be edited but easier is to use the web interface of the VSCP daemon to updated and add entries. The location is

```
http://localhost:8080/vscp/dm
```

if VSCP daemon is on a local host.

The format for the dm.xml file is as follows

```
<?xml version = "1.0" encoding = "UTF-8" ?>

<!-- 2017-01-05                                                  -->
<!-- This files holds the decision matrix for the daemon         -->

<dm>
    <!-- Enabled row, belongs to group "test" -->
    <row enabled="true" groupid="test">

        <!-- Mask to trigger row -->
        <mask
                priority="1"
                class="0xFFFF"
                type="55"
                guid="0F:0E:0D:0C:0B:0A:09:08:07:06:05:04:03:02:01:00">
        </mask>

        <!-- Filter to trigger row -->
        <filter
                priority="7"
                class="1000"
                type="0xAA"
                guid="FF:EE:DD:CC:BB:AA:99:88:77:66:55:44:33:22:11:00">
        </filter>

        <!-- Date and time from which action should trigger -->
        <allowed_from>1970-01-01 00:00:00</allowed_from>

        <!-- Date and time up to which action should trigger -->
        <allowed_to>2099-12-31 23:59:59</allowed_to>

        <!-- List of weekdays which action should trigger -->
        <allowed_weekdays>mtwtfss</allowed_weekdays>

        <!-- A specific time (or pattern) when the action should trigger -->
        <allowed_time>*-*-* *:0/5/10:0</allowed_time>
                              <!-- Every zero,five and ten minutes-->
        <!-- Optional - Set Index to check -->
        <index bEnable="true|false">0</index>

        <!-- Optional - Set Zone to check -->
        <zone bEnable="true|false">0</zone>

        <!-- Optional - Set Subzone to check -->
        <subzone bEnable="true|false">0</subzone>

        <!-- Control code (deprecated) -->
```

```
        <control>0x80000000</control>

        <!-- measurement index -->
        <measurementindex bEnable="true|false">0</subzone>

        <!-- measurement -->
        <measurement bEnable="true|false" compare="eq|neq|gt|gteq|lt|lteq"
unit="0">78.2</measurement>

        <!-- Action code -->
        <action>0x10</action>

        <!-- Action parameter -->
        <param>1,t44;canal;12;25;abc;90</param>

        <!-- Comment for decion matrix row -->
        <comment>This is a dumb comment</comment>

    </row>

</dm>
```

Timing parameter data format

- String with days the action can occur in as mtwtfss use '-' for day when action should not be performed.
- Start Data/time for when the action can occur in ISO format as YYYY-MM-DD HH:MM:SS Use a wild-card character '*' if any of them is don't care.
- End Data/time for when the action can occur in ISO format as YYYY-MM-DD HH:MM:SS Use a wild-card character '*' if any of them is don't care.
- Action date/time when action should occur in ISO format as YYYY:MM:DD HH:MM:SS

The start and the end Date/time fields set the date/time range when the action is allowed to occur.

The action date set the date/time when the action should occur. This can be a one shot in which case the full date/time is filled in as in 2009-11-02 14:30:00 which will perform action at this time once if the start/end dates allow for that. To get repeats it is possible to use wild-cards. For example *-*-* 14:30:* (can also be written as * 14:30:*) will perform the operation every seconds from 14:30 to (but not including) 14:31 in the time range set by the start/end date/time range.

For repeating operation that is just dependent on time filter on the SECOND event so that this is the only event that trigger the row.

It is also possible to specify more then one of each element in the date/time by separating them with a slash. For example * 14:0/10/20/30/40/50:00 will do the action every ten minutes between two a clock and three a clock in the afternoon.

Scheduler

Each event that is received by the daemon is feed through the decision matrix. When the event is received it is placed on the DM input queue after passing an initial set of filter that removes events that are of no interest to the specific implementation.

The first event in the DM queue is then run through the matrix and each DM row is compared and if there is a match the action for that row is executed.

The daemon itself place some events on the DM queue. See Internal DM events below. For example the internal LOOP event is run through the matrix between every external event feed to the queue. The LOOP event will also be seen in the matrix when no other events are present. In this case a configuration value is used to set the time between two LOOP events (sleep time) if no other events arrive.

Variable substitution for parameters (escapes)

Action parameters are strings that are passed to actions and in that way can be used to configure the action to solve different problems. The action string can contain escape values which are replaced with real values before the action is performed. The following escapes are currently defined.

Escape sequence	Description
%%	The character '%'
%;	The character ';'. Semicolon is normally used to separates arguments of an action.
%cr	A carriage return.
%lf	A line feed.
%crlf	A carriage return + a line feed.
%tab	A tab.
%bell	A bell.
%amp	Insert an ampersand character
%amp-html	Insert HTML representation of ampersand (&).
%lt	Insert a less than character
%lt-html	Insert HTML representation of less than (<).
%gt	Insert a greater than character
%gt-html	Insert HTML representation of greater than (>).
%variable:[*variable_name***]**	Value of a variable. BASE64 data **is not** decoded.
%vardecode:[*variable_name***]**	Value of a variable. BASE64 data **is** decoded.
%file:[*path***]**	Content of a named file.
%event	A full event in the standard text form.
%event.class	The class for the event as a number.
%event.class.str	The class for the event as a a descriptive string.
%event.type	The type for the event as a number.
%event.type.str	The type for the event as a a descriptive string.
%event.head	head of the event.
%event.priority	The priority for the event (from head).

Escape sequence	Description
%event.sizedata	The number of databytes the event have.
%event.data	All data-bytes as a comma separated list. If no data 'empty' is set.
%event.data[n]	Data byte n. If no data 'empty' is set.
%event.obid	obid of event.
%event.hardcoded	Hardcoded bit f head as 0/1.
%event.guid	GUID of the event.
%event.nickname	nickname for the GUID of the event (LSB of GUID).
%event.timestamp	Timestamp of the event.
%event.index	This is the same as %event.data[0].
%event.zone	This is the same as %event.data[1].
%event.subzone	This is the same as %event.data[2].
%isodate	Date in ISO format YY-MM-DD
%isotime	Time in ISO form HH:MM:SS.
%isoboth	Date + Time in ISO form YY-MM-DDTHH:MM:SS.
%isobothms	Date + Time + milliseconds in ISO form YY-MM-DDTHH:MM:SS.nnn.
%unixtime	Unix time as a 32-bit unsigned number.
%mstime	Current time in milliseconds.
%hour	Current hour.
%minute	Current minute.
%second	Current second.
%week0	Current week number (1-52(53)). Week starts with Sunday.
%week1	Current week number (1-52(53)). Week starts with Monday.
%weektxt	Get week name in textual short form.
%weektxtfull	Get week name in textual long form.
%month	Current month number (1-12).
%monthtxt	Get current month in textual short form.
%monthtxtfull	Get current month in textual long form.
%year	Current year.
%quarter	Current quarter(1-4).
%path.config	Return the directory containing the system config files. Unix: /etc Windows: C:\Documents and Settings\All Users\Application Data Mac: /Library/Preferences
%path.datadir	Return the location of the applications global, i.e. not user-specific, data files. Unix: prefix/share/appname Windows: the directory where the executable file is located Mac: appname.app/Contents/SharedSupport bundle subdirectory
%path.documentsdir	Return the directory containing the current user's (the account the daemon/server is run as) documents. Unix: ~ (the home directory) Windows: C:\Documents and Settings\username\Documents Mac: ~/Documents
%path.executable	Return the directory and the filename for the current executable. Unix: /usr/local/bin/exename Windows: C:\Programs\AppFolder\exename.exe Mac: /Programs/exename
%path.localdatadir	Return the location for application data files which are host-specific and can't, or shouldn't, be shared with the other machines.
%path.pluginsdir	Return the directory where the loadable modules (plugins) live. Unix: prefix/lib/appname Windows: the directory of the executable file Mac: appname.app/Contents/PlugIns bundle subdirectory

Escape sequence	Description
%path.resourcedir	Return the directory where the application resource files are located. The resources are the auxiliary data files needed for the application to run and include, for example, image and sound files it might use. Unix: prefix/share/appname Windows: the directory where the executable file is located Mac: appname.app/Contents/Resources bundle subdirectory
%path.tempdir	Return the directory for storing temporary files.
%path_userconfigdir	Return the directory for the user config files. Unix: ~ (the home directory) Windows: C:\Documents and Settings\username\Application Data Mac: ~/Library/Preferences
%path.userdatadir	Return the directory for the user-dependent application data files: Unix: ~/.appname Windows: C:\Documents and Settings\username\Application Data\appname Mac: ~/Library/Application Support/appname
%path.localdatadir	Return the directory for user data files which shouldn't be shared with the other machines.
%toliveafter1	Inserts '*Carpe diem quam minimum credula postero.*'
%toliveafter2	Inserts '*Be Hungry - Stay Foolish.*'
%toliveafter3	Inserts '*Stay Foolish - Be Hungry.*'
%measurement.index	Index from measurement data coding (0-7).
%measurement.unit	Unit from measurement data coding (0-3).
%measurement.coding	Index from measurement data coding (0-7).
%measurement.float	Convert event data to a floating point value. The class must be one of the measurement classes VSCP_CLASS1_MEASUREMENT, VSCP_CLASS2_LEVEL1_MEASUREMENT, VSCP_CLASS1_MEASUREZONE, VSCP_CLASS1_SETVALUEZONE, VSCP_CLASS1_MEASUREMENT64 or VSCP_CLASS2_MEASUREMENT_STR
%measurement.string	Convert event data to a string. The class must be one of the measurement classes VSCP_CLASS1_MEASUREMENT, VSCP_CLASS2_LEVEL1_MEASUREMENT, VSCP_CLASS1_MEASUREZONE, VSCP_CLASS1_SETVALUEZONE, VSCP_CLASS1_MEASUREMENT64 or VSCP_CLASS2_MEASUREMENT_STR
%measurement.convert.data	This converts a level I measurement from any of the measurement classes into string data in the form needed by the action send event. You can use it to always send a VSCP_CLASS2_MEASUREMENT event, that is easy to interpret, when a measurement event of any kind is received. Just add index,zone,subzone,%measurement.unit,%measurement.convert.data instead of the data for the event. The class should always be VSCP_CLASS2_MEASUREMENT_STR and the type %event.type
%eventdata.realtext	Tries to convert the event data into a string in a human readable format.

Can perfectly be used as arguments for triggered external program execution.

Actions

Actions can have long argument lists. If you are an end user don't be scared by this. The setup of DM

rows is done with program support and you fill in the values in a nice GUI.

The semicolon character ';' is used as a separator for arguments and if needed in an argument ”%;” should be used. Also % is a special character used for escapes and should be written as %%.

NOOP

```
VSCP_DAEMON_ACTION_CODE_NOOP    0x00/0
```

No operation.

Execute external program

```
VSCP_DAEMON_ACTION_CODE_EXECUTE    0x10/16
```

Run an external program. The daemon must have execute access rights to be able to launch the program. It is important that this program returns as fast as possible. If lengthy operations is needed please fork another process in your external program and then return.

It is also important that you give the full path to the program and where used, it's extension. So instead of writing an executable as “mysql” your should write “C:\mysql-5.6.21-winx64\bin\mysql.exe”. The reason for this is that the VSCP daemon checks if the file is available before trying to execute it.

Parameters

- Full path to program + arguments to program. (possibly with with substitution elements %event.class %event.type %date etc).
- Optional variable that is set to true if the execution was successful.

Note that the semicolon symbol (';') is used internally and if it is part of a program argument %; should be written.

Example 1 Run a scripts when a CLASS1.MEASUREMENT, Type=6, Temperature event is received from a node with nodeid=1 and from sensor with index equal to 2 on that device.

```
<row enable="true" groupid="" >
    <mask  priority="0"  class="65535"  type="65535"
        GUID=" 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:FF" >
    </mask>
    <filter  priority="0"  class="10"  type="6"
        GUID=" 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:01" >
    </filter>
    <control>0x20</control>
    <action>0x10</action>
    <param>/srv/vscp/thingspeak.sh XXXXXXXXXXXXXXXX 3
%measurement.string</param>
    <comment>Send value for compressor sensor</comment>
```

```
    <allowed_from>0000-01-01 00:00:00</allowed_from>
    <allowed_to>9999-12-31 23:59:59</allowed_to>
    <allowed_weekdays>mtwtfss</allowed_weekdays>
    <allowed_time>*-*-* *:*:*</allowed_time>
    <index  bMeasurement="true"  > 2</index>
    <zone>0</zone>
    <subzone>0</subzone>
</row>
```

Example 2

Turn on lights when the sun goes down (calculated).

```
<row enable="true" groupid="Tellstick" >
    <mask  priority="0"  class="65535"  type="65535"
         GUID=" 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00" >
    </mask>
    <filter  priority="0"  class="20"  type="45"
         GUID=" 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00" >
    </filter>
    <control>0x0</control>
    <action>0x10</action>
    <param>/usr/local/bin/tdtool --on 1</param>
    <comment>Turn on window ligts when sun goes down</comment>
    <allowed_from>0000-01-01 00:00:00</allowed_from>
    <allowed_to>9999-12-31 23:59:59</allowed_to>
    <allowed_weekdays>mtwtfss</allowed_weekdays>
    <allowed_time>*-*-* *:*:*</allowed_time>
    <index  bMeasurement="false"  > 0</index>
    <zone>0</zone>
    <subzone>0</subzone>
</row>
```

Example 3 Log data to a MySQL database

```
<row enable="true" groupid="" >
    <mask  priority="0"  class="65535"  type="65535"
        GUID=" 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:FF" >
    </mask>
    <filter  priority="0"  class="10"  type="6"
        GUID=" 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:01" >
    </filter>
    <control>0x0</control>
    <action>0x10</action>
    <param>
     mysql -u'user' -p'password' -h'host' -e"INSERT
       INTO temperature(GUID,SensorIndex,Date,Value)
       VALUES ('%event.guid',%measurement.index,'%isodate %isotime',
       %measurement.float)";
    </param>
    <comment></comment>
```

```
    <allowed_from>0000-01-01 00:00:00</allowed_from>
    <allowed_to>9999-12-31 23:59:59</allowed_to>
    <allowed_weekdays>mtwtfss</allowed_weekdays>
    <allowed_time>*-*-* *:*:*</allowed_time>
    <index  bMeasurement="true"  > 1</index>
    <zone>0</zone>
    <subzone>0</subzone>
</row>
```

Example 4

Start a new instance of notepad every minute at a specific date and time interval.

```
<row enabled="true">

    <!-- Mask to trigger row - zero for a bit is don't care -->
    <mask priority="0"
        class="0xFFFF"
        type="0xFF"
        guid="00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00">
    </mask>

    <!-- Filter to trigger row -
          if mask have a one in a bit it is compared with filter -->
    <filter priority="7"
            class="0xFFFF"
            type="5"
            guid="00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00">
    </filter>

    <!-- Date and time from which action should trigger -->
    <allowed_from>2020-01-01 00:00:00</allowed_from>

    <!-- Date and time up to which action should trigger -->
    <allowed_to>2020-01-01 05:59:59</allowed_to>

    <!-- Date and time up to which action should trigger -->
    <allowed_weekdays>mtwtfss</allowed_weekdays>

    <!-- A specific time (or pattern) when the action should trigger -->
    <!-- Every ten seconds-->
    <allowed_time>*-*-* *:*:0</allowed_time>

    <!-- Control code - Note that the row property enabled is doubled
             as the highest bit here -->
        <control>0x80000000</control>  <!-- enable row -->

    <!-- Action code  -->
    <action>0x00000010</action>

    <!-- Action parameter -->
```

```
    <param>c:\windows\notepad.exe</param>

    <!-- Comment for decision matrix row -->
    <comment>Start a new instance of notepad every minute at a
             specific date and time interval.
    </comment>

</row>
```

HTTP GET, POST, (PUT)

```
VSCP_DAEMON_ACTION_CODE_GET_PUT_POST_URL    0x75/117
```

This action access a given URL given by the action parameter. Can be used to write data to a remote data source using web server scripts or similar.

Action parameters

```
    method;url[;data;headers;proxy]
```

- Method: 'GET' (default), 'POST' (or 'put', 'options', 'delete', 'patch')
- URL (e.g. http://www.host.com)
- Optional data. This data should be on the form *name=Lucy&neighbours=Fred+%26+Ethel* and it is ignored if access method is 'get'
- Optional extra header row(s). Each row must be ended with “\n”.
- Optional proxy on the form <hostname>:<port number> (just leave blank if no proxy should be used).

put/options/delete/patch is not available at the moment.

Note 1

Remember to us **%amp** instead of the ampersand character as it is invalid in XML file.

Note 2

Currently no login capabilities is available but you can use

```
"http://<user>:<password>@mysite.com/mypath"
```

but should remember that this method is deprecated in RFC-1396

Example 1

Use a HTTP POST to update a Thingspeak channel every minute. We use CLASS2.VSCPD Type=6, MINUTE to update field1 which get the current minute value written using the escape **%minute**. The event is executed all time, every day, from beginning of time to the end of time.

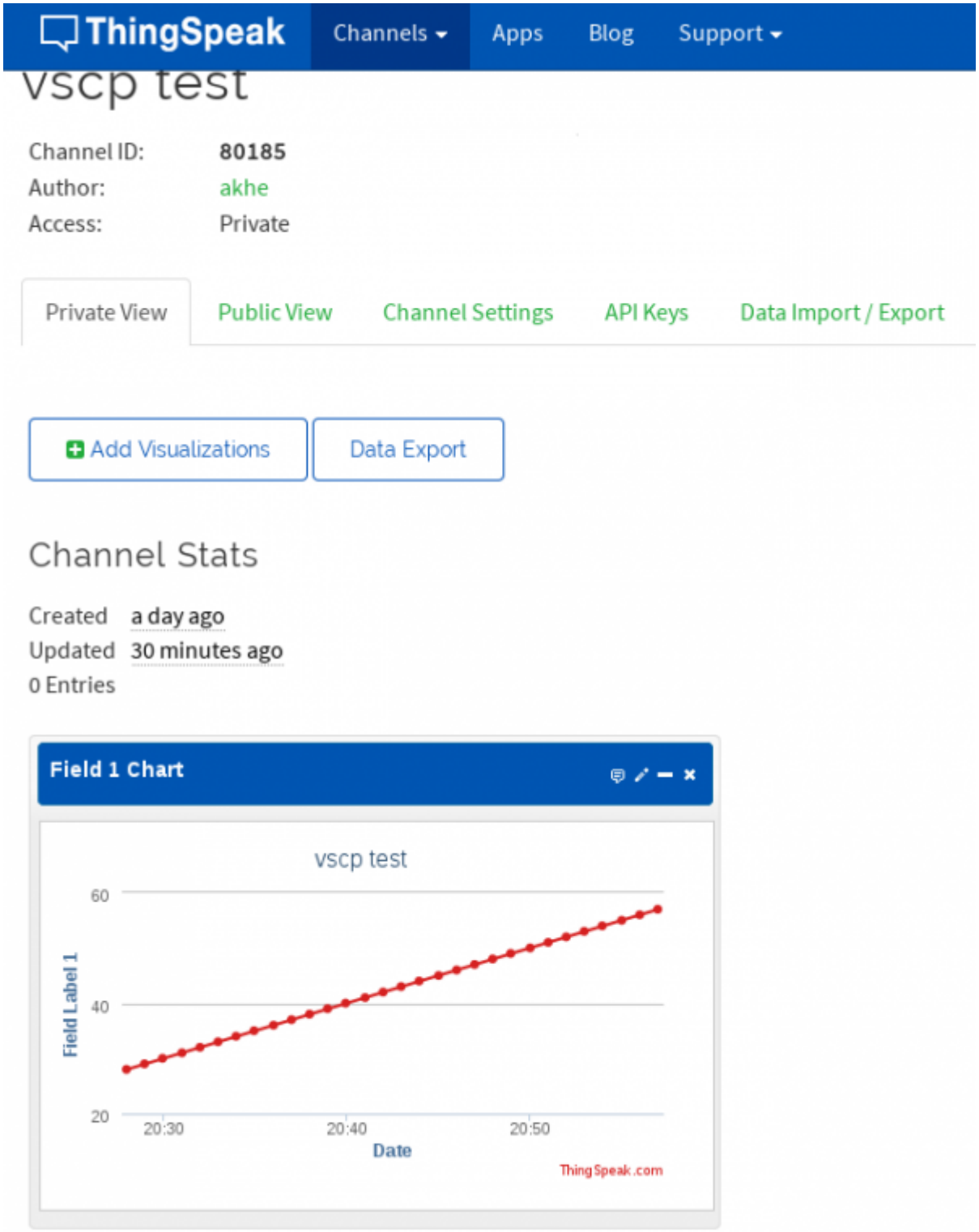
Replacing the current event with a measurement event and using %measurement.string instead of %minute for field1 is an easy way to get a dynamic diagram from measurement data.

The XML DM entry for this functionality looks like this. As always it is easier to use the web interface of the VSCP daemon to enter decision matrix rows.

“xxxxxxxxxxxxxxxx” in the action parameter is the API key that one get from TingSpeak.

```
<row enable="true" groupid="" >
    <mask priority="0" class="65535"  type="65535"
        GUID=" 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00" > </mask>
    <filter priority="0"  class="65535" type="6"
        GUID=" 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00" > </filter>
    <control>0x0</control>
    <action>0x75</action>
<param>POST;http://api.thingspeak.com/update;field1=%minute;X-THINGSPEAKAPIK
EY: xxxxxxxxxxxxxxxx\n</param>
    <comment></comment>
    <allowed_from>0000-01-01 00:00:00</allowed_from>
    <allowed_to>9999-12-31 23:59:59</allowed_to>
    <allowed_weekdays>mtwtfss</allowed_weekdays>
    <allowed_time>*-*-* *:*:*</allowed_time>
    <index  bMeasurement="false"  > 0</index>
    <zone>0</zone>
    <subzone>0</subzone>
  </row>
```

The result will be this



The format for updating several fields is

```
field1=value&field2=value&field3=value...
```

As the ampersand is an invalid character in an XML file this has to be written as

```
field1=value%ampfield2=value%ampfield3=value...
```

You can do the same using HTTP GET but the format is a bit different

```
<row enable="true" groupid="" >
    <mask  priority="0"  class="65535"  type="65535"
        GUID=" 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00" >
    </mask>
    <filter  priority="0"  class="65535"  type="6"
        GUID=" 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00" >
    </filter>
    <control>0x0</control>
    <action>0x75</action>
<param>GET;http://api.thingspeak.com/update?key=xxxxxxxxxxxxxxxx%ampfield1=%
minute</param>
    <comment></comment>
    <allowed_from>0000-01-01 00:00:00</allowed_from>
    <allowed_to>9999-12-31 23:59:59</allowed_to>
    <allowed_weekdays>mtwtfss</allowed_weekdays>
    <allowed_time>*-*-* *:*:*</allowed_time>
    <index  bMeasurement="false"  > 0</index>
    <zone>0</zone>
    <subzone>0</subzone>
  </row>
```

ThingSpeak API is described here

Send event to remote VSCP daemon

```
VSCP_DAEMON_ACTION_CODE_SEND_TO_REMOTE    0x43/67
```

This action can be used to send an event to a remote VSCP daemon.

Parameters

- address to server.
- port
- username
- password
- event to send

Store in variable

```
VSCP_DAEMON_ACTION_CODE_STORE_VARIABLE    0x50/80
```

Store data in a variable. The variable is named in the argument. If the variable is not available it is created.

This action is available in two forms. One that store the data supplied in the action parameter in a named variable and one form that store the event in the variable.

Form 1 - Store action data in variable

Parameters

- Variable name.
- Variable type (defaults to 1 = string).
- Persistence as true|false. (Defaults to false = non-persistent).
- value in the form that type of variable expect

Variable write formats are here.

Example parameter data

```
Store in boolean: my_boolean;2;true;true
Store in integer: my_int;3;true;24000
Store in string: my_string;1;true;"This is a string"
Store on event: my_event;7;false;0,20,1,2,3,4,5,6
```

Form 2 - Store event data in variable

Parameters

- Variable name.
- Variable type (defaults to 1 = string) (only used if the variable does not exist yet).
- Persistence as true|false (Defaults to false = non-persistent) (only used if the variable does not exist yet).

What data that is stored in the variable depends on the type of the variable. The variable is created if it does not exist.

Variable type	Type	Data stored
VSCP_DAEMON_VARIABLE_CODE_UNASSIGNED	Unspecified data	Nothing
VSCP_DAEMON_VARIABLE_CODE_STRING	String value	If the event is a measurement the value is stored in string form.
VSCP_DAEMON_VARIABLE_CODE_BOOLEAN	Boolean value	
VSCP_DAEMON_VARIABLE_CODE_INTEGER	Integer value	If the event is a measurement it will be stored (possibly truncated) in the integer (if possible).
VSCP_DAEMON_VARIABLE_CODE_LONG	Long value	If the event is a measurement it will be stored (possibly truncated) in the long (if possible).

Variable type	Type	Data stored
VSCP_DAEMON_VARIABLE_CODE_DOUBLE	Floating point value	If the event is a measurement it will be stored in the double.
VSCP_DAEMON_VARIABLE_CODE_VSCP_MEASUREMENT	VSCP data coding	If the event is a measurement it will be stored in the variable.
VSCP_DAEMON_VARIABLE_CODE_VSCP_EVENT	VSCP event	The complete event will be stored in the variable.
VSCP_DAEMON_VARIABLE_CODE_VSCP_EVENT_GUID	VSCP event GUID	The GUID of the event will be stored in the variable.
VSCP_DAEMON_VARIABLE_CODE_VSCP_EVENT_DATA	VSCP event data	The data of the event will be stored in the variable
VSCP_DAEMON_VARIABLE_CODE_VSCP_EVENT_CLASS	VSCP event class	The class of the event will be stored in the variable.
VSCP_DAEMON_VARIABLE_CODE_VSCP_EVENT_TYPE	VSCP event type	The type of the event will be stored in the variable.
VSCP_DAEMON_VARIABLE_CODE_VSCP_EVENT_TIMESTAMP	VSCP event timestamp	The timestamp of the event will be stored in the variable.
VSCP_DAEMON_VARIABLE_CODE_DATETIME	Date + Time in iso format	Date and time when the event was received will be stored in the variable.
VSCP_DAEMON_VARIABLE_CODE_DATE	Date in iso format	Date when the event was received will be stored in the variable.
VSCP_DAEMON_VARIABLE_CODE_TIME	Time in iso format	Time when the event was received will be stored in the variable.
VSCP_DAEMON_VARIABLE_CODE_BLOB	Base64 binary encoded data	Nothing is stored.
VSCP_DAEMON_VARIABLE_CODE_MIME	Mime type (mime-type;base64 encoded content)	Nothing is stored.

Variable type	Type	Data stored
VSCP_DAEMON_VARIABLE_CODE_HTML	HTML Page	Nothing is stored.
VSCP_DAEMON_VARIABLE_CODE_JAVASCRIPT	JavaScript code	Nothing is stored.
VSCP_DAEMON_VARIABLE_CODE_JSON	JSON data	Event stored on VSCP JSON form.
VSCP_DAEMON_VARIABLE_CODE_XML	XML data	Event stored on VSCP XML form.
VSCP_DAEMON_VARIABLE_CODE_SQL	SQL data	Nothing is stored.
VSCP_DAEMON_VARIABLE_CODE_LUA	LUA script	Nothing is stored.
VSCP_DAEMON_VARIABLE_CODE_LUA_RESULT	LUA executed script result string	Nothing is stored.
VSCP_DAEMON_VARIABLE_CODE_UX_TYPE1	Contains XML file for User interface	Nothing is stored.
VSCP_DAEMON_VARIABLE_CODE_DM_ROW	A DM row, comma separated	Nothing is stored.
VSCP_DAEMON_VARIABLE_CODE_DRIVER	Driver item, comma separated	Nothing is stored.
VSCP_DAEMON_VARIABLE_CODE_USER	User item, comma separated	Nothing is stored.
VSCP_DAEMON_VARIABLE_CODE_FILTER	Filter item, comma separated	Nothing is stored.

Add to a variable

```
VSCP_DAEMON_ACTION_CODE_ADD_VARIABLE    0x52/82
```

Add data to a variable. The variable is named in the argument.

Parameters

- variable
- Value to add to variable

Example

```
myvariable;1341
```

If the variable does not exist it is created.

Note that there is possible to create dynamically created variables by using the escapes available. As an example a variable energy_%monthtxtfull_%year In this case adding energy consumption for a month in a variable.

Subtract from a variable

```
VSCP_DAEMON_ACTION_CODE_SUBTRACT_VARIABLE    0x53/83
```

Subtract the data from a variable. The variable is named in the argument.

Parameters

- Variable
- Value to subtract from variable

Example

```
myvariable;10
```

Multiply with a variable

```
VSCP_DAEMON_ACTION_CODE_MULTIPLY_VARIABLE    0x54/84
```

Multiply the data with a variable. The variable is named in the argument.

Parameters

- Variable
- Value to multiply variable with

Example myvariable;3

Divide the data with a variable

```
VSCP_DAEMON_ACTION_CODE_DIVIDE_VARIABLE    0x55/85
```

Divide the data with a variable. The variable is named in the argument.

Parameters

- variable
- value to divide variable with

Example

```
myvariable;2
```

Check variable, set variable

```
VSCP_DAEMON_ACTION_CODE_CHECK_VARIABLE    0x58/88
```

Check if variable is greater etc. then a specified value and set some other variable to the logic

outcome of the compare. The variable is named in the argument. If the variable is not available it is created.

Parameters

- Value to check.
- unit (defaults to zero).
- Operation (see table below).
- Variable to check.
- Variable to set to logic outcome.

Compare operators

Operator	Operator alternative	Description
noop		No operation
lt	<	Less then
gt	>	Greater then
lteq	⇐	Less then or equal
gteq	>=	Greater then or equal
eq	==	Equal
neq	!=	Not equal

Example parameter data

```
500;0;gt;counter;flag
```

Check variable, set to true

```
VSCP_DAEMON_ACTION_CODE_CHECK_VARIABLE_TRUE    0x56/86
```

Check if variable is greater etc. then a specified value and set some other variable to true if it is. The variable is named in the argument. If the variable is not available it is created.

Parameters

- Value to check.
- unit (defaults to zero).
- Operation (see table below).
- Variable to check.
- Variable to set to true.

Compare operators

Operator	Operator alternative	Description
noop		No operation
lt	<	Less then
gt	>	Greater then

Operator	Operator alternative	Description
lteq	⇐	Less then or equal
gteq	>=	Greater then or equal
eq	==	Equal
neq	!=	Not equal

Example parameter data

```
500;0;gt;counter;flag
```

Will check if 500 is greater then counter and set the variable flag to true if it is.

Check variable, set to false

```
VSCP_DAEMON_ACTION_CODE_CHECK_VARIABLE_FALSE    0x57/87
```

Check if variable is greater etc. then a specified value and set some other variable to false if it is. The variable is named in the argument. If the variable is not available it is created.

Parameters

- Value to check.
- unit (defaults to zero).
- Operation.
- Variable to check.
- Variable to set to false.

Compare operators

Operator	Operator alternative	Description
noop		No operation
lt	<	Less then
gt	>	Greater then
lteq	⇐	Less then or equal
gteq	>=	Greater then or equal
eq	==	Equal
neq	!=	Not equal

Example parameter data

```
500;0;gt;counter;flag
```

Will check if 500 is greater then counter and set the variable flag to false if it is.

Check Measurement

```
VSCP_DAEMON_ACTION_CODE_CHECK_MEASUREMENT    0x59/89
```

This is only valid for the measurement event types. Both Level I and Level II events. The value is compared to the argument value and the flag variable is set to the logic outcome.

The unit and the sensor index must match for the comparison to be performed.

Parameters

- Unit.
- Sensor index.
- Value to check.
- Operation.
- Variable to set to logic outcome.

Compare operators

Operator	Description
<	Less then
>	Greater then
⇐	Less then or equal
⇒	Greater then or equal
==	Equal
!=	Not equal

Example parameter data

```
0;0;500;>;counter;flag
```

Will check if 500 is greater then counter and set the variable flag to true if it is.Unit is 0 and sensor index is 0.

Store minimum

```
VSCP_DAEMON_ACTION_CODE_STORE_MIN    0x71/113
```

This is only valid for the measurement event types. Both Level I and Level II events. The value is stored in the variable if it is less than the current value stored in the variable.

The unit, the sensor index, zone, and subzone must match for the check to be performed.

If the variable is numerical the new value is stored in it. If the variable is a measurement type variable the value is also stored in it and when it is created sensor-index, zone and subzone is stored in it to.

Parameters

- Variable name.

- Unit. (defaults to zero).
- Sensor index. (Only checked for events that have it defined, defaults to zero).
- Zone. (Only checked for events that have it defined, defaults to zero).
- Subzone.(Only checked for events that have it defined, defaults to zero).

Store maximum

```
VSCP_DAEMON_ACTION_CODE_STORE_MAX    0x72/114
```

This is only valid for the measurement event types. Both Level I and Level II events. The value is stored in the variable if it is higher than the current value stored in the variable.

The unit, the sensor index, zone, and subzone must match for the check to be performed.

If the variable is numerical the new value is stored in it. If the variable is a measurement type variable the value is also stored in it and when it is created sensor-index, zone and subzone is stored in it to.

Parameters

- Variable name.
- Unit. (defaults to zero).
- Sensor index. (Only checked for events that have it defined, defaults to zero).
- Zone. (Only checked for events that have it defined, defaults to zero).
- Subzone.(Only checked for events that have it defined, defaults to zero).

Send event

```
VSCP_DAEMON_ACTION_CODE_SEND_EVENT    0x40/64
```

Send event when another event is received.

Parameters

- Event to send. Should be on the form head,class,type,obid,time-stamp,GUID,data1,data2,data3....
- Optional variable that will be set to true. If the variable is used it should be separated from the event with a | so a full parameter list with a variable should take the following form 0,20,3,0,0,0:1:2:3:4:5:6:7:8:9:10:11:12:13:14:15,0,1,35|bSent

Example

This example send an event every ten seconds. It happens all weekdays during a specified time period from 1977-01-01 00:00:00 to 2099-12-31 23:59:59. The optional variable bSent is set to false when the event has been sent.

```
<row enabled="true">
```

```
    <!-- Mask to trigger row - zero for a bit is don't care -->
    <mask priority="0"
        class="0xFFFF"
        type="0xFF"
        guid="00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00">
    </mask>

    <!-- Filter to trigger row -
        if mask have a one in a bit it is compared with filter -->
    <filter priority="7"
        class="0xFFFF"
        type="5"
        guid="00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00">
    </filter>

    <!-- Date and time from which action should trigger -->
    <allowed_from>1977-01-01 00:00:00</allowed_from>

    <!-- Date and time up to which action should trigger -->
    <allowed_to>2099-12-31 23:59:59</allowed_to>

    <!-- Date and time up to which action should trigger -->
    <allowed_weekdays>mtwtfss</allowed_weekdays>

    <!-- A specific time (or pattern) when the action should trigger -->
    <!-- Every ten seconds-->
    <allowed_time>*-*-* *:*:0/10/20/30/40/50</allowed_time>

    <!-- Control code - Note that the row property enabled is doubled
            as the highest bit here -->
    <control>0x80000000</control>  <!-- enable row -->

    <!-- Action code  -->
    <action>0x00000010</action>

    <!-- Action parameter -->
<param>0,20,3,0,0,0:1:2:3:4:5:6:7:8:9:10:11:12:13:14:15,0,1,35|bSent</param>

    <!-- Comment for decion matrix row -->
    <comment>Send an event every ten seconds</comment>

</row>
```

Send event conditional

```
VSCP_DAEMON_ACTION_CODE_SEND_EVENT_CONDITIONAL    0x42/65
```

Send an event if a specified variable is true.

This action is specified to make it possible to define DM entries that wait for a response form a node

and resend a specified event a number of times until this response is received or a timeout occurred.

Parameters

- Control variable name. Event is sent if variable is true.
- Event to resend.

The variable is set to true by the action that initially sent the event that needs a reply. Typically this can be an event that trigged an ON event and expects a confirm. When the variable is true this action get triggered and if triggered by one of the internal time events it will start to send periodic events with a period equal to the internal time events period. The filter/mask should be set to trigger on some of the internal time events (loop, second, minute etc). This choice determines the interval between resend of events.

To get all this to work four DM rows are required.

1. A row that send the original event and set the flag to true.
2. A row where a timer is started that set the variable to false when it elapses.
3. A row with this action. The event is resent until a response is received which sets the flag to false. Check variable set to false can be used for this as well.
4. A row that set the flag to true when the correct response is received. The stop timer action is perfect for this. The trigger should be the event which is the expected response.

Send event(s) from file

```
VSCP_DAEMON_ACTION_CODE_SEND_EVENTS_FROM_FILE    0x42/66
```

This action sends event(s) from a named file.

Parameters

- Path to file.

The format for the XML file is

```
<events>
    <event>
        <head>0</head>
        <class>0</class>
        <guid>FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF:FF</guid>
        <type>0</type>
        <data>0,1,2,3,0x67</data>
    </event>
</events>

<event>
    ....
</event>
```

Write to file plain

```
VSCP_DAEMON_ACTION_CODE_WRITE_FILE    0x70/112
```

This action writes (appends) the substituted action argument string to a file. If the string should be written one on each row be sure to add %crlf for windows and %lf for Unix at the end of the action-string.

Parameters

- Path to file
- 1 for append. 0 for overwrite.

Start a timer

```
VSCP_DAEMON_ACTION_CODE_START_TIMER    0x60/96/
```

This action starts a timer identified by a numerical ID that will count down from a specified time set in seconds. If the timer is already defined it will be reused and if active it will be reinitialized.

Parameters

- Timer ID.
- Count down time in seconds.
- Variable that is set to setvalue when the timer elapses.
- setvalue (true or false).

Pause a timer

```
VSCP_DAEMON_ACTION_CODE_PAUSE_TIMER    0x61/97
```

This action stops an active timer identified by a numerical ID. If the timer does not exist or is inactive the action does nothing.

Parameter

- Timer ID.
- Variable that will be set to false.

Stop a timer

```
VSCP_DAEMON_ACTION_CODE_STOP_TIMER    0x62/98
```

This action stops an active timer identified by a numerical ID. If the timer does not exist or is inactive the action does nothing.

Parameters

- Timer ID.
- Variable that will be set to false.

Resume a timer

```
VSCP_DAEMON_ACTION_CODE_RESUME_TIMER    0x63/99
```

This action stops an active timer identified by a numerical ID. If the timer does not exist or is inactive the action does nothing.

Parameters

- Timer ID.
- Variable that will be set to false.

Write Table

```
VSCP_DAEMON_ACTION_CODE_WRITE_TABLE    0x80/128
```

Write table data to a named table. You can access the written table data through the web socket, TCP/IP or the REST interface. Previous version wrote data to a text file on disk now a full SQL table is used.

Parameters

Two formats are currently supported selected by the first part of the action parameter string.

Row type 0

```
0;tablename;datetime;value[;[BASE64:]sql-expression]
```

- **0** - Select the first type.
- **tablename** - Name of the table (system unique)
- **datetime** - Date/time on ISO format (YY-MM-DDTHH:MM:SS)
- **value** - A floating point value for the data point.
- *Optional* **[BASE64:]sql-expression** - Optional SQL expression. If left out the sqlinsert expression set in the table configuration is used. If set here a SQL expression updating the 'vscptable' is expected. The table always have this name but the database can have other user defined tables which can be updated here. If the SQL expression is preceded by **BASE64:** a conversion from base64 is done before the SQL expression is used. After the decoding from

base64 is done the VSCP decision matrix escapes are resolved which so the SQL expression can have VSCP decision matrix escapes in it even if it is encoded in base64.

Before the configured SQL insert expression for the table is used the VSCP decision matrix escapes are resolved. So

```
0;test1;%both;%measurement.float
```

will be translated to

```
0;test1;2017-02-17T23:47:19;-19.2
```

and then if the table stored sqlinsert expression is

```
INSERT INTO 'vscptable' (date,value,event) VALUES ('%%s','%%f','%event')%;
```

will have the escapes handled. That is

- s will be %s * f will be %f
- %event will be translated to the string form of the event.
- %; will be translated to ; **Important** if you have multi line SQL expressions.

Row type 1

```
1;tablename;[BASE64:]sql-expression]
```

- 0
- **tablename** Name of table (system unique)
- sql expression. Se description of SQL expression above.

The requirements for the table is that it should be named **vscptable**, it should have a text field named **time** and a numeric (integer) field named **value**. Apart from this the database can contain any number of additional fields or tables.

You can read more about tables here.

Clear Table

```
VSCP_DAEMON_ACTION_CODE_CLEAR_TABLE     0x81/129
```

Clear data for a named table. An optional SQL expression can be given but normally the table configured delete SQL expression is used. Typical use is when a table should collect data over a period of time and then restart that collection when the time have passes. Note that the MAX table type can be used for this to, at least if the the period of the measurements are constant.

Parameters

```
tablename[;[BASE64:]sql-delete-expression]
```

- tabelname - Name for an existing table.
- sql-delete-expression - This optional expression can be used to delete the table data. Normally the configured delete expression is used.

You can read more about tables here.

Run LUA Script

```
VSCP_DAEMON_ACTION_CODE_RUN_LUASCRIPT    0x100/256
```

Only available if LUA is compiled in.

Run a LUA script. The standard VSCP decision matrix escapes can be used to insert values so

* a parameter with %variable:variable_name will execute the value of the variable *variable_name*. * a parameter with %file:path will execute the content of the file at *path*.

Parameters

* LUA code to execute. Can contain VSCP decision matrix escapes.

Run JavaScript

```
VSCP_DAEMON_ACTION_CODE_RUN_JAVASCRIPT    0x200/512
```

Run JavaScript

The standard VSCP decision matrix escapes can be used to insert values-

The script can use internal VSCP functions to read and write variables and to send and receive events.

The JavaScript engine used is Duktape (ECMAScript 5.1), with some semantics updated from ES2015+ .

The JavaScript is executed on it's own thread. This means that it is possible to construct a script that is started when the VSCP daemon is started and will run until it is killed. All that is need to do this is to trigger on the internal CLASS2.VSCPD, Type=23 Starting up event.

Parameters

The JavaScript code to execute in real text or if preceded with **BASE64:** the JavaScript code encoded in BASE64. The parameter contents can contain standard VSCP decision matrix escapes which will be filled in after the BASE64 code is decoded and and before the JavaScript code is executed.

The BASE64 encoded form is the preferred form as XML does not allow some characters like ampersand (&) which must be coded as & to not break the XML parsing. This is true for other characters like < (less that), > (greater than), etc.

Even better JavaScript variables an be used. This is a VSCP variable of type JavaScript which can be edited in the administration interface in a coder friendly editor and thus make it easier to write the code. To insert the value of a variable two forms can be used.

%vardecode:[*variable_name***]** will be replaced by the value of the variable *variable_name* decoded from BASE64. This is the usual selection for a JavaScript variable as it's value always is encoded in BASE64.

```
%varddecode:[varTestScript]
```

%variable:[*variable_name***]** will be replaced by the value of the variable *variable_name* as it is stored so if the value is stored in BASE64 the coding will be preserved.

```
%variable:[varTestScript]
```

or

```
BASE64:%variable;[varTestScript]
```

which actually is the same as

```
%varddecode:[varTestScript]
```

There is also a third possibility

%file:[*path***]** will be replaced with the content of the file at *path*. This means that a parameter

```
%file:[path_to_script]
```

will execute the JavaScript stored in the file pointed to by path_to_script.

The code can use callbacks into the VSCP daemon for VSCP functionality. The callbacks are defined here.

example

Read the value of the variable *test1* and then log this value to the logs. Note that the VSCP DM escape %isodate will be replaced with the current date before the script is executed.

```
   var dd = vscp_readVariable(vscp_clientItem,"test1");
   vscp_log( "test1=" + dd.value + " Hello world! %isodate\n");
```

The BASE64 version of this parameter looks like this

```
BASE64:dmFyIGRkID0gdnNjcF9yZWFkVmFyaWFibGUodnNjcF9jbGllbnRJdGVtLCJ0ZXN0MSIpO
w0KICAgdnNjcF9sb2coICJ0ZXN0MT0iICsgZGQudmFsdWUgKyAiIEhlbGxvIHdvcmxkISAlaXNvZ
GF0ZVxuIik7
```

Internal Decision Matrix Events

```
CLASS2.VSCPD (65535)
```

CLASS2.VSCPD is reserved for internal events used by the decision matrix mechanism of the VSCP daemon. Events of this type is never be visible on a physical bus.

Events of this type can be used for timekeeping and many more things. As an example the CLASS2.VSCP, Type=7, Hour can be used to perform things on a specific hour of the day.

Copyright 2000-2017 Åke Hedman, Grodans Paradis AB/Paradise of the Frog

Grodans Paradis AB

From:
http://www.vscp.org/docs/vscpd/ - **the VSCP Server**

Permanent link:
http://www.vscp.org/docs/vscpd/doku.php?id=vscp_daemon_decision_matrix

Last update: **2017/09/03 21:34**