

# Kirjastot

- [Tiedonsiirtorajapinnat](#)
- [Actiweb cpu-linkit](#)
- [Hälytysviestit howto](#)
- [Modbus-server](#)
- [smsdaemon-status-tiedostot](#)
- [Auto-login](#)
- [Webservice-OpenWeatherMap](#)
- [IO-profiilit](#)
- [Webservices](#)
- [SQLite tietokanta](#)
- [Webservice-nordpool](#)
- [HVAC-kirjasto](#)
- [Piste-skeemat](#)

# Tiedonsiirtorajapinnat

Tässä ohjeessa on esitelty protokollat, joita tuetaan suoraan pisteiden dataSource ja dataTarget -kentissä.

Yleistä:

dataSource ja dataTarget kenttiin annetaan URL muotoinen kuvaus siitä, millä tavalla pisteen arvo halutaan lukea jostakin ulkoisesta tietolähteestä, tai millä tavoin pisteen arvo halutaan kirjoittaa johonkin ulkoiseen kohteeseen. Edellä mainittu ulkoinen kohde voi olla jokin kenttäväylä, web-sivusto tai tiedosto.

Suurin osa dataSource ja dataTarget protokollista tukee seuraavia yleisiä URL parametreja.

## YLEISIÄ DATASOURCE JA DATATARGET URL PARAMETREJA:

?scale=0.1     skaalaa luetun raw arvon annetulla kertoimella.

?offset=     Tekee skaalattuun arvoon suuntaissiirtoa

?invert=1     Tällä parametrilla käännetään tavallisimmin DI tai DO pisteen polariteetti, mutta sillä voidaan tehdä myös muuta.

invert parametri toimii niin, että väylältä luettu arvo vähennetään

invert parametrin arvosta.

Digitaali pisteen polariteetin saa käännettyä antamalla "invert=1"

?bitmask=     Tällä parametrilla voi antaa maskin, jonka avulla valitaan vain halutut bitit raw arvosta. Käytännössä suorittaa binäärisen AND operaation RAW datan ja tämän maskin välillä.

?datatype=     Määrittää missä muodossa luettu RAW arvo tulee tulkita.  
Mahdollisia tyyppejä: int16, uint16, int32, uint32, float32

?x1=0&y1=0..x32=100&y32=100    Inline muunnostaulukko joka voidaan antaa

dataSource tai dataTarget URL:ssa. Käyrän taittopisteet annetaan koordinaatteina URL parametreissa, jotka ovat nimeltään muotoa "x1", "y1", x2 ja "y2". Pisteitä voi olla enimmillään 32 kpl, ja ne on nimettävä juoskevasti 1:stä lähtien. Raakadatan arvo on annettu X akselissa, ja vastaava skaalattu arvo y akselilla. Ero on tärkeätä huomata dataTargetissa, jolloin taulukkoa käytetään tavallaan takaperin - eli pisteen pv sijoitetaan x akselille, ja väylään lähetettävä data luetaan y akselilta. Tällöin samaa taulukkoa voidaan käyttää sekä sourcessa että targetissa.

?rejecthi    Raja-arvo, jota suuremmat (skaalatut) arvot hylätään, eli pisteen pv ei muutu.

?rejectlo    Raja-arvo jota pienemmät (skaalatut) arvot hylätään, eli pisteen pv ei muutu.

?hilimit    Ylin arvo jonka skaalattu arvo voi saada - eli rajoittaa pv:n maksimia. Arvoa ei hylätä, mutta se rajoitetaan.

?lolimit    Alin arvo jonka skaalattu arvo voi saada - eli rajoittaa pv:n minimiä. Arvoa ei hylätä, mutta se rajoitetaan.

?damping    Luo pisteen pv arvoon vaimennusta, eli käytönnössä tässä parametrissa voi antaa [ ] vaimennuksen aikavakion (lambda), aikayksikkönä funktiokutsut - jos kutsutaan kerran [ ] sekunnissa, aikavakion yksikkö on sekunti.

?precision    Pisteen tarkkuus. Arvo 0.001 tarkoittaa että piste näytetään 3 desimaalin tarkkuudella, [ ] ja tarkkuun 10 että arvo pyöristetään lähimpään kymmeneen.

?curve    [ ] Pisteen skaalaukseen voi käyttää tällä parametrilla hvac kirjaston hvacCurve tyyppisiä pisteitä. [ ] Käyttö tapahtuu niin, että halutun hvacCurve pisteen nimi annetaan URL:ssä ?curve=hvacCurvePisteenNimi [ ] HUOM! Toimii VAIN dataSource URL:issa (ei dataTarget). [ ] HUOM, Muutos! Tämä toimii myös luomalla pisteeseen "curve" niminen kenttä, ja antamalla sen arvoksi halu [ ] skaalaus taulukko.

?select    Jos RAW data on "table" muotoinen, tällä parametrilla voi valita taulukosta haluamansa solun, antamalla select parametrilla

oikean muotoisen polun haluttuun arvoon.

Polussa voi käyttää . ja [] syntaksia.

Esim. RAW data on {status="OK", data=  
    {pv={10, 20, 30, 40, 50}, description="My data"}  
}

voi pv kentän kolmannen rivin osoittaa select parametrilla:  
    ?select=data.pv[3]

HUOM! Vain toimii vain dataSource kentissä

#### PROTOKOLLAT:

slc://     Lukee ja kirjoittaa tietoa toisiin SLC pohjaisiin laitteisiin.  
Mikäli IP osoitteeksi annetaan "local" (tai 127.0.0.1) data luetaan  
tai kirjoitetaan paikallisesti laitteen omassa pistetietokannassa.

Verkon ylitse lukemiseen käytetään TCP porttia 30001

Muoto:

slc://local/dbPoint.field

Esimerkki:

slc://local/sys/settings/bacnetIP.instance

slc://192.168.0.130/ioPoints/TK01/TE10.pv

mbus://     Lukee dataa mbus väylästä. Tukee tällä hetkellä vain mbus-serial  
väylää.

Huomaa että URL alkaa %-koodatulla mbus -portti tietokantapisteellä  
joka osoittaa siis sen portin jota väyläkommunikaatioon käytetään.

[[[[mbus protokolla tulee URL parametreja:

[[[[interval[[[[MBus laitteen luentaväli sekunteina. Eli

interval=3600 tarkoittaa, että mittari luetaan vain tunnin välein

Muoto:

mbus://[mbusPortPointId]/[slaveAddress]/datarecords/[dataIndex]/value

Esimerkki (dataSource):

mbus://ioPorts%2FmbusSerial%2FP1/12016562/datarecords/5/value

wmb:// Lukee dataa, joka on vastaanotettu wireless M-Bus protokollalla.

VAIN LUKEMINEN TUETTU

Muoto:

wmb://[wmbusReceiveFile]/[dataKey]

Esimerkki (dataSource):

wmb://MC603\_heatMeter/temp\_in

modbusrtu:// Lukee ja kirjoittaa dataa modbus RTU väylään.

Tukee ylimääräisiä URL parametreja:

order sanajärjestys kun käytetään 32 bittisiä rekistereitä

arvot: "MSW" (enemmän merkitsevä sana ensin),

"LSW" (vähemmän merkistevä sana ensiksi).

Oletus: "MSW"

multistate Tällä parametrilla voidaan tulkita peräkkäiset rekisterit

arvoiksi esim. tasoa kuvaavaksi arvoksi. Eli RAW arvoksi

tulkitaankin rekisterin sisällön sijasta tieto siitä, kuinka

mones rekisteri ketjussa on arvoltaan suurempi kuin nolla.

onvalue Kun kirjoitetaan multistate rekistereihin,

kirjoitetaan tämä arvo niin monenteen rekisteriin, kuin

(skaalattu) PV määrittää.

VAIN dataTarget!

**forcewrite** Normaalisti modbus rekisterit kirjoitetaan vain silloin, kun PV arvo muuttu. Tällä voidaan kirjoitus pakottaa tapahtuvaksi jokaisella luku/kirjoitus kierroksella.

VAIN dataTarget.

**leap** in multistate sources, with this paramter, it is possible to define non-consecutive registers to behave like multistate input.  
NOTE! Use with caution! This can cause huge amount of bus traffic (meanning latecy will creep up).

**forcesinglewrite** Käyttää holding rekisterin kirjoittamiseen funktionkoodia 0x06 (write single holding) eikä normaalia 0x10, eli koodia 16 (Write multiple holding)

Muoto:

modbusrtu://[modbusPortPointId]/[slaveAddress]/regType/[regAddr]

Esimerkki (dataSource):

modbusrtu://ioPorts%2FmodbusRTU%2FP3/50/input/1?scale=0.1&datatype=int16

modbustcp:// Lukee ja kirjoittaa dataa modbusTCP -protokollalla.

Tukee ylimääräisiä URL parametreja:

**order** sanajärjestys kun käytetään 32 bittisiä rekistereitä  
arvot: "MSW" (enemmän merkitsevä sana ensin),  
"LSW" (vähemmän merkitsevä sana ensiksi).  
Oletus: "MSW"

**forcewrite** Normaalisti modbus rekisterit kirjoitetaan vain silloin, kun PV arvo muuttu. Tällä voidaan kirjoitus pakottaa tapahtuvaksi jokaisella luku/kirjoitus kierroksella.

VAIN dataTarget.

Muoto:

modbustcp://[modbusPortPointId]/[deviceId]/regType/[regAddr]

Esimerkki (dataSource):

modbustcp://ioPorts%2FmodbusTCP%2FM2/1/holding/1?scale=0.1&datatype=int32

bacnetip:// lukee ja/tai kirjoittaa dataa bacnet IP väylään.

Tuetut objekti muodot (objectTypeStr):

analog-input, analog-output, analog-value,

binary-input, binary-output, binary-value,

multistate-input, multistate-output, multistate-value

property voi olla joko propertyn numeroitu numero. bacnetDefines tiedostossa määritellään joillekin ominaisuuksille alias, joita ovat mm.

PV = 85

NAME = 77

DESCRIPTION = 28

OOS = 81

RELIABILITY = 103

STATE\_TEXTS = 110

ACTIVE\_TEXT = 4

INACTIVE\_TEXT = 46

Muoto:

bacnetip://[deviceId]/[objectTypeStr]/[objectNo]/[property]

Esimerkiksi:

bacnetip://421001/analog-input/2/PV

bacnetip://421001/analog-output/2/77

file:// Lukee tai kirjoittaa dataa paikallisiin tiedostoihin.

Muoto:

file://localhost/full/file/path

Huomaa! localhost tarvitaan aina semantiikan vuoksi, vaikka se jätetäänkin huomiotta - eli nykyiset versiot eivät huomioi host-nimen kirjoitusasusta. Tulevaisuudessa on mahdollista että file -protokolla laajenee tukemaan myös remote -hosteja, jolloin taaksepäin yhteensopivuuden vuoksi olisi hyvä että localhost on kirjoitettu pisteisiin jo valmiiksi oikein.

Esimerkki (dataSource tai dataTarget):

file://localhost/tmp/munloPiste.txt

file://localhost/tmp/munMittausData.txt?scale=10

http://

https:// Toimii tällä dataSource kentässä, ja vain webService objektissa

Hakee dataa HTTP tai HTTPS protokollan ylitse. Voi lukea esimerkiksi kokonaisen web sivuston, tai jonkin REST pohjaisen rajapinnan tarjoaman tiedon.

Muoto:

http://is.fi

Tarkempi kuvaus webservice dokumentissa!

## DATAN LUKEMINEN JÄRJESTELMÄSTÄ - SLC JÄRJESTELMÄN RAJAPINNAT

Järjestelmä toteuttaa joitakin rajapintoja joiden kautta tietokannan pisteitä pääsee lukemaan muista järjestelmistä.

RAJAPINTA: BACnet IP

Kaikki AI, AO, AV, BI, BO, BV, MSI ja MSV tyyppiset pisteet tietokannassa tulevat automaattisesti näkyviin BACnet/IP väylään, mikäli "BACnetIP" prosessi on käynnissä.



RAJAPINTA: WebService/REST

Data pisteitä on mahdollista lukea järjestelmästä myös REST tyyppisen rajapinnan ylitse. Kuvattu esillisessä dokumentissa.

RAJAPINTA: SLC konsoli

Dataa voi kysellä myös suoraan SLC palvelimesta. Tämä menetelmä on itseasiassa huomattavan tehokas, sillä sen kautta saa yhteyden suoraan SLC engineen Lua rajapintaan, ja tätä kautta on mahdollista vaikkapa telnet ohjelmalla, tai shell skriptistä netcat komennolla lähettää lua kielisiä komentojonoja SLC järjestelmälle. Komennot tulee päättää \0 eli eli niin "null" merkkiin. Tämä tarkoittaa yksinkertaisesti tavua nolla. Telnet ohjelmassa se saadaan usein tehtyä ^@ merkillä; Suomenkieliseltä näppäimistöltä [ctrl]-[alrGr]-[2].

Bash komentorivillä voi hyödyntää netcat komentoa, ja testata sitä esimerkiksi näin:

```
>> printf "return 'Hello SLC!'\0" | nc 192.168.0.128 30001
```

huomaa, että printf komennon merkkijonossa annetaan null merkki \0 muodossa.

Hyödyllisempi esimerkki:

```
>> printf "return Data.getReal ('ioPoints/TK01/TE10_AI.pv')\0" | nc 192.168.0.128 30001
```

Jolloin komento palauttaa laitteen 192.168.0.130 tietokannasta pisteen ioPoints/TK01/TE10\_AI.pv -kentän pv arvon, ja tulostaa sen näytölle

Saman voi tehdä myös telnet ohjelmassa, ottamalla yhteyden esimerkin tapauksessa osoitteeseen 192.168.0.128, porttiin 30001, ja kirjoittamalla:

```
return Data.getReal ('ioPoints/TK01/TE10_AI.pv')^@
```

ja painamalla enter. Lopussa oleva merkki ^@ on edellä jo mainittu "null" merkki.

Vastaukset joita SLC konsoli antaa, ovat aina JSON muodossa, joten voit kysellä rajapinnan ylitse ,yös taulukoita tai muita monimutkaisempia

data-rakenteita.

RAJAPINTA: fdxServer

Datapisteitä voidaan lukea tietokannasta myös fdx-rajapinnan ylitse.

Tämän rajapinnan kautta pääsee käsiksi pisteisiin, jotka on luotu  
/fdx/.. hakemistoon tietokannassa.

Esimerkiksi pisteen /fdx/TK01\_TE10\_M oloarvo on mahdollista kysellä fdx  
rajapinnan ylitse nimellä TK01\_TE10\_M.

# Actiweb cpu-linkit

Actiweb laitteiden välisen linkin luominen

□-----

Kun käyttöliittymään halutaan linkki jolla voi siirtyä laitteiden välillä, täytyy laitteet tutustuttaa toisiinsa.

Tämä tapa siirtää käyttäjäistunto käyttää SSH tunnelia, ja se on verkkoturvallinen.

Kun Actiweb ohjelmisto käynnistyy ensimmäistä kertaa, se luo SSH-avainparin.

Jotta käyttäjäistunto voidaan siirtää Actiweb laitteiden välillä, on kohdelaitteeseen syötettävä lähdelaitteen julkinen SSH avain.

Kun haluat luoda linkin laitteen A käyttöliittymään, josta haluat siirtyä samoilla kirjautumistiedoilla laitteen B käyttöliittymään, suorita alla olevien ohjeiden mukaiset toimenpiteet.

Laitteiden linkkaaminen:

1. Mene laitteen A käyttöliittymäsivulle "System->settings".
2. Paina "Security and SSH keys" kehyksessä paniketta "Show public RSA key"
3. Avautuvassa dialogissa, valitse ja kopioi kaikki koko julkinen avain (esim. ctrl-A ja ctrl-C)  
□  
3.1.□Huom! Poista avaimen lopusta host nimeen ja käyttäjänimeen viittaava osa, esim ..www-data@actiweb
4. Mene kohdelaitteessa (laite B) pistetietokantaan (System -> database)
5. Avaa tietokantapiste ..  
□□"sys/settings/authorizedSshKeys"  
□. ja napsauta sitten auki sen taulukko muotinen sshKeys -kenttä.
6. Jokainen rivi kuvaa yhtä laitetta, jonka grafiikkasivulla voi olla □linkki tähän laitteeseen. Luo taulukkoon uusi rivi, ja anna sille □kuvaava nimi, ja liitä sitten lähdelaitteesta (laite A) kopioimasi □julkinen SSH avain uuden taulukkorivin kenttään "key".
7. Aseta laitteen ip-osoite oikeaksi, jotta ohjelmisto löytää sen □käynnistyessään, ja saa ladattua oikean SSH sormenjäljen.
8. Paina OK, ja tallenna tietokanta, ja käynnistä ohjelmisto uudelleen.

HUOM!

SSH asetukset päivitetään ja listassa olevien laitteiden SSH sormenjäljet päivitetään verkon ylitse ohjelmiston käynnistyksessä.

SSH yhteyden, fingerprinttien ja avainten toimivuuden voi testata komennolla:

intel CPU:

```
sudo -u www-data ssh actiweb@[targetHostIP]
```

HUOM! Tiedosto-oikeudet

```
chmod 600 $HOME/.ssh/id_rsa
```

```
chmod 700 $HOME/.ssh
```

Nämä komennot pitää ajaa sekä actiweb että www-data käyttäjänä

Laitteiden välisen linkin luominen grafiikkasivulle

Tee grafiikkasivulle elementti (esim. label), jota haluat käyttää linkkinä

lisää elemntille Javascript koodi..

```
this.onclick = function (e) {  
    window.location = "handover.php?host=192.168.0.131";  
}
```

.. tai jos haluat siirtyä linkistä suoraan esimerkiksi sivulle "iv\_yhteenvedo.xml"..

```
this.onclick = function (e) {  
    window.location = "handover.php?host=192.168.0.131&page=iv_yhteenvedo";  
}
```

HUOM!

Ylläolevassa javascript lähdekoodissa kohdelaite on osoitteessa 192.168.0.131, joten se pitää korvata oikealla IP osoitteella.

# Hälytysviestit howto

## HÄLYTYSVIESTIT

Actiweb osaa lähettää mm. hälytyksistä viestin sähköpostilla, joka voidaan kääntää laitteen sisäisesti myös esimerkiksi tekstiviestiksi.

Jos viesti halutaan lähettää sähköpostilla, täytyy tehdä seuraavat asetukset:

☐\* Luo halutut ja tarvittavat hälytysryhmät

☐

☐\* Jaa hälytyspisteet käyttämään haluttuja hälytysryhmiä

☐

☐\* Asettele hälytysryhmään vastaanottajat

☐

System -> email settings sivulla:

☐

☐\* Lähettäjän osoite

☐

☐\* Aseta email hälytykset päälle (enabled)

☐\* Aseta lähtevän postin palvelin, esimerkiksi: smtp.gmail.com

☐

☐\* Jos palvelin käyttää kirjautumista, aseta käyttäjänimi ja salasana,

☐ sekä käännä "Login required" vipu päälle.

☐

☐\* Jos palvelin käyttää salausta, aseta palvelimen portti

☐☐TSL 587 (googlen palvelin toimii tällä)

☐☐SSL 465

☐HUOM! Kun käytetään googlen tiliä, n.s. turvattomat laitteet pitää joskus sallia.

Häiriöitä viestien lähettämisessä voi tutkia "Sytem > Log files > E-mail log" sivulla.

Hälytysryhmät:

Parametreilla enableToAlarmEvent, enableToNormalEvent ja enableAckEvent määritellään, luodaanko ryhmään kuuluvissa hälytyksissä tapahtumia kun hälytyspiste menee hälytykselle, palaa normaalitilaan tai kun se kuitataan.

pv tarkoittaa hälytysryhmän tilaa. Tämä tila määrittää, mitä vastaanottajalistaa ryhmä käyttää hälytysviestejä lähettäessä.

receiverList on lista vastaanottajista. Tämä lista koostuu itse asiassa kahdesta tasosta, eli tähän listaan voidaan mahdollistaa useampia vastaanottajalistoja, joiden välillä vaihdetaan ryhmän pv arvon mukaisesti.

Hälytysviestien pohjissa voidaan käyttää "tägejä" eli merkkijonoja jotka korvautuvat lähetetyssä viestissä hälytyskohtaisilla tiedoilla.

Nämä tägit muistuttavat html tägejä, ja niidem "nimi" on sama kuin hälytyspisteen kentällä, eli jos haluat näyttää viestissä hälytyksen kellonajan, lisää viestipohjaan tägi

□<ALARMDATE>

huomaa että tietokantapisteen kenttä on kirjoitettu tägissä aina SUURILLA kirjaimilla.

Mikäli haluat lisätä viestiin jonkin kentän hälytysryhmästä, voit periaate on sama, mutta tägiin tulee lisätä eteen "G\_", eli hälytysryhmän kuvaustekstin saa hälytysviestiin kirjoittamalla tägin

□<G\_DESCRIPTION>

□

kun itse hälytyspisteen kuvausteksti on <DESCRIPTION>.

On olemassa kaksi tägiä jotka eivät suoranaisesti ole tietokantapisteen kenttiä:

□<ID>□□hälytyspisteen nimi

□

□<TRANSITION>□Tila johon hälytyspiste siirtyi

□

Hälytysviestin protokolla:

Protokolla tai laite jonka kautta hälytysviesti lähetetään,

määrittäytyä hälytysryhmässä olevan vastaanottajan osoitteen perusteella.

Tarkemmin sanottuna, kun viestin vastaanottajaa määritetään hälytysryhmässä, on osoite (recipientList -> recipients -> address) oikeasti URL.

sähköpostiosoitteen pitkä muoto:

☐email://vastaanottaja@hostname.fi

ja SMS viestin puhelinnumeron pitkämuoto:

☐sms://+358231234123

☐

..mutta, jos osoite kenttä sisältää vain '+' ja numeroita, se tulkitaan  
puhelinnumeroksi, tai jos se muistuttaa sähköpostiosoitetta  
(sisältää . ja @ merkit) mutta URL skeemaan ei ole määritelty,  
se tulkitaan sähköpostiosoitteeksi.

Muita viestinlähetyksimuotoja:

☐rut://[puh numero]☐☐Teltonikan HTTP/GET tekstiviesti toiminto

☐☐☐☐☐☐muista tehdä asetukset /sys/settings/smsOverHttp

☐☐☐☐☐☐pisteeseen.

Operaattori saattaa vaatia puhelinnumeron annettavaksi maakoodin kanssa, esim. +358.

Tällöin asian tekee mielenkiintoiseksi se, että sekä viesti, että  
puhelinnumero lähetetään URL enkoodauksen läpi. Tämä tarkoittaa sitä,  
että '+' merkki tulkitaan välilyönniksi.

Lisäksi, ennen lähetystä puhelinnumerolle tehdään dekoodaus,  
joten + merkki täytyy dekodata "tuplasti" eli % merkki vielä erikseen:

Näin puhelinnumerosta +358501231234

tulee %252B358501231234 (eli + merkin tilalla %252B)

ja koko vastaanottajasta rut://%252B358503875050

☐☐☐☐☐☐Huomaa! Tekstiviestin lähetys GET rajapinnan läpi EI TOIMI

☐☐☐☐☐☐teltonikan ohjelmistoversiossa RUT9XX\_R\_00.06.05.3

☐☐☐☐☐☐

☐relay://[ipOsoite]☐☐atkohälytysten välittäminen CPU yksiköiden välillä.

☐☐☐☐☐☐Ketjun seuraava laite käsittelee viestin

☐☐☐☐☐☐käyttäen samannimisen hälytysryhmän asetuksia kuin

☐☐☐☐☐☐viestin alunperin lähettänyt CPU.

□□□□□□

□□□□□□ URL parametrilla ?group=[ryhmanNimi] voi valita

□□□□□□ mitä hälytysryhmää vastaanottava CPU käyttää

□□□□□□ hälytyksen käsittelyyn.



# Modbus-server

## Modbus server tila

Laitteen voi asettaa toimimaan modbus väylässä myös slave laitteena.

Tämä tuki saadaan aktivoitua seuraavasti:

1. Aseta haluttu modbus portti "slave" tilaan esimerkiksi vaihtamalla kenttä "mode" tietokantapisteessä "ioPorts/modbusRTU/P1".

2.

2. Aseta halutut slave -tilan asetukset pisteeseen

"sys/settings/modbusSlave"..

2.1

Haluttu väyläosoite kenttään "address"

2.2

Käytettävä kommunikaatioportti kenttään "port"

2.3

Valitse protokolla versio (RTU tai TCP)

2.4

Aseta slave-tila päälle vaihtamalla enabled kenttä arvoon "true"

2.5

Käynnistä task uudestaan (restart painike)

Modbus laitteesta luettavat rekisterit voidaan määritellä lisäämällä haluttuihin tietokantapisteisiin kenttä "modbusMapping", ja asettamalla siihen sopiva arvo.

modbusMapping kentäs arvo muodostuu seuraavasti:

[rekisteriTyyppi]/[rekisteriNumero]

esimeriksi:

1

Laitteen tietokantapiste "ioPoints/HOLDING\_0"

halutaan näkymään modbus väylällä holding rekisterinä 0

2

Pisteeseen luodaan kenttä "modbusMapping" ja sen arvoksi

asetetaan "holding/0"

3

Ohjelmisto käynnistetään uudestaan.

4

Nyt tietokantapistettä voi lukea ja kirjoittaa

modbus väylän kautta.

5

Muiden kuin pv kenttien näyttäminen modbusväylässä

Modbus slave tilassa väylälle kerrotana oletusarvoisesti pisteen .pv kenttä.

Jos väylälle halutaan kertoa jokin muu pisteen kenttä, voidaan se määrittää URL parametrilla "field".

Esimerkki:

Hälytyspisteen .av kenttä voidaan antaa modbus väylälle diskreetti-rekisterissä 0. Pisteen modbusMapping kenttään asetetaan silloin määritys:

```
□discrete/0?field=av
```

Kun tietokanta tallennetaan, ja ohjelmisto käynnistetään uudelleen, voidaan pisteen av kenttä lukea discrete 0 rekisteristä.

# smsdaemon-status-tiedostot

-----  
[ ] SMSDAEMON tile tiedostot  
-----

Jos SMS viestien lähettäminen on aktivoitu,  
palvelu täyttää laitteen levyn pienillä status tiedostoilla  
/var/log/smstools hakemistossa.

Ongelman voi todeta antamalla komennon  
ls /var/log/smstools/smsd\_stats/\* | wc -l

joka antaa tiedostojen määrän hakemistossa johon ylimääräisiä tiedostoja luodaan.

Jos tiedostojen lukumäärä on suuri (satoja tai yli), laitteessa on ongelma

Usein tiedostoja on jo syntynyt niin paljon, että ne täytyy poistaa käsin ennen korjauksen asentamista.

Status tiedostojen poistaminen käsin:  
sudo find /var/log/smstools/smsd\_stats/ -delete

Korjaus:  
Asenna uusi versio tiedostosta

/opt/slc/lib/libMSGGateway.lua

# Auto-login

## ☐AUTO-LOGIN toiminto

Ulkoinen laite voidaan asettaa kirjautumaan sisään halutulla käyttäjänimellä. Auto-login käyttää normaaleja käyttöliittymän käyttäjätilejä, joten automaattisesti sisäänkirjautuvan käyttäjän käyttöoikeuksia voidaan hallita kuten muidenkin käyttäjien.

IP osoitteet joista auto-login on mahdollista tehdä, täytyy listata asetustiedostoon /opt/slc/etc/autologin

Asetustiedoston formaatti on yksinkertainen, jokaisella on yksi IP osoite tai IP alue, josta auto-login toimii. Kommentti rivejä voi antaa kun aloittaa rivin # -merkillä.

esimerkki:

```
#IPaddr  
127.0.0.1
```

Merkkejä \* ja ? voi käyttää wild-cardena kun syöttä IP alueita.

## AUTOLOGIN KÄYTTÄJÄT:

Jotta käyttäjä voi kirjautua sisään automaattisesti ilman salasanaa, täytyy hänen kuulua "autologin" käyttäjäryhmään. Tämän muutoksen voi halutuille käyttäjätileille tehdä "system -> user management" sivulta.

Tämän jälkeen käyttöpaneelin (tai muun laitteen) saa kirjautumaan automaattisesti sisään käyttöliittymään, kun laittaa selaimen osoitteeseen URL parametrin

☐?autologin=[käyttäjänimi]

..jolloin koko web osoite voisi olla esimerkiksi..

☐http://192.168.0.128/index.php?autologin=user

Jolloin linkin kautta järjestelmään tullessa ei sisäänkirjautumisruutua ilmesty laisinkaan, vaan selain siirtyy suoraan järjestelmän etusivulle.

## HUOMAA:

Kun käyttäjä kuuluu autologin ryhmään, voi "../etc/autologin" tiedostossa mainituista IP osoitteista kirjautua sisään ILMAN SALASANAA kaikilla niillä käyttäjänimillä jotka kuuluvat autologin ryhmään.



# Webservice- OpenWeatherMap

Sääpalvelu openWeatherMap:

Palvelusta saa luettua nykyisen säätilan, tai ennusteita ympäri maailman.

Dokumentaatio:

<https://openweathermap.org/current>

<https://openweathermap.org/forecast5>

## NYKYINEN SÄÄTILA

URL:

<http://api.openweathermap.org/data/2.5/weather>

Parametrit:

?q={city name}

?q={city name},{country code}

?id={cityIdCode}

?lat=35&lon=139

?mode={dataMode} dataMode = "json" | "xml"

Vastaus:

```
"main":{  
  "temp":306.15,  
  "pressure":1013,  
  "humidity":44,  
  "temp_min":306,  
  "temp_max":306  
}
```

## ENNUSTE 5 VRK

Antaa ennusteen haluttuun paikkaan 3 tunnin välein, max 5 vuorokaudeksi.

Vastauksessa jokainen rivi edustaa 3 tunnin jaksoa.

URL:

<http://api.openweathermap.org/data/2.5/forecast>

Parametrit:

```
?q={city name}
?q={city name},{country code}
?id={cityIdCode}
?lat=35&lon=139
?mode={json|xml}
?cnt={nRows}
?units={metric|imperial}
```

Esimerkiksi:

```
http://api.openweathermap.org/data/2.5/forecast?q=tampere,fi&units=metric
```

Vastaus (rajoitettu 4 vastaus riviin):

```
{
  "cod": "200",
  "message": 0.0032,
  "cnt": 4,
  "list": [
    {
      "dt": 1487246400,
      "main": {
        "temp": 286.67,
        "temp_min": 281.556,
        "temp_max": 286.67,
        "pressure": 972.73,
        "sea_level": 1046.46,
        "grnd_level": 972.73,
        "humidity": 75,
        "temp_kf": 5.11
      },
      "weather": [
        {
          "id": 800,
          "main": "Clear",
          "description": "clear sky",
          "icon": "01d"
        }
      ],
      "clouds": {
        "all": 0
      },
      "wind": {
        "speed": 1.81,
        "deg": 247.501
      },
      "sys": {
        "pod": "d"
      },
      "dt_txt": "2017-02-16 12:00:00"
    },
    {
      "dt": 1487257200,
      "main": {
```

```
"temp": 285.66,
"temp_min": 281.821,
"temp_max": 285.66,
"pressure": 970.91,
"sea_level": 1044.32,
"grnd_level": 970.91,
"humidity": 70,
"temp_kf": 3.84
},
"weather": [
  {
    "id": 800,
    "main": "Clear",
    "description": "clear sky",
    "icon": "01d"
  }
],
"clouds": {
  "all": 0
},
"wind": {
  "speed": 1.59,
  "deg": 290.501
},
"sys": {
  "pod": "d"
},
"dt_txt": "2017-02-16 15:00:00"
},
{
  "dt": 1487268000,
  "main": {
    "temp": 277.05,
    "temp_min": 274.498,
    "temp_max": 277.05,
    "pressure": 970.44,
    "sea_level": 1044.7,
    "grnd_level": 970.44,
    "humidity": 90,
    "temp_kf": 2.56
  },
  "weather": [
    {
      "id": 800,
      "main": "Clear",
      "description": "clear sky",
      "icon": "01n"
    }
  ],
  "clouds": {
    "all": 0
  },
  "wind": {
```



```
    "speed": 1.41,  
    "deg": 263.5  
  },  
  "sys": {  
    "pod": "n"  
  },  
  "dt_txt": "2017-02-16 18:00:00"  
},  
{  
  "dt": 1487624400,  
  "main": {  
    "temp": 272.424,  
    "temp_min": 272.424,  
    "temp_max": 272.424,  
    "pressure": 968.38,  
    "sea_level": 1043.17,  
    "grnd_level": 968.38,  
    "humidity": 85,  
    "temp_kf": 0  
  },  
  "weather": [  
    {  
      "id": 801,  
      "main": "Clouds",  
      "description": "few clouds",  
      "icon": "02n"  
    }  
  ],  
  "clouds": {  
    "all": 20  
  },  
  "wind": {  
    "speed": 3.57,  
    "deg": 255.503  
  },  
  "rain": {},  
  "snow": {},  
  "sys": {  
    "pod": "n"  
  },  
  "dt_txt": "2017-02-20 21:00:00"  
}  
],  
"city": {  
  "id": 6940463,  
  "name": "Altstadt",  
  "coord": {  
    "lat": 48.137,  
    "lon": 11.5752  
  },  
  "country": "none"  
}  
}
```



# IO-profiilit

\*\*\*\*\*

IO-profiles kirjasto

\*\*\*\*\*

Väyläliitännäisille laitteille on mahdollista luoda valmiita kuvaustiedostoja, joissa määritellään, mitä asioita mistäkin rekisteristä tai objektista laitteen väylärajapinnassa löytyy.

Kuvaus tehdään XML muodossa, ja yksi tiedosto voi sisältää yhden laitetyypin profiilin.

Kun profiili on luotu, voidaan sitä hyödyntää kahdella eri tavalla:

Ensimmäinen tapa on lisätä laitteita auto-create listaan, jolloin laitetta varten luodaan kaikki väyläliitynnässä (ja profiili tiedostossa) määritellyt pisteet valmiiksi väyläasetuksineen.

Toinen tapa on viitata jossakin tietokantapisteessä lyhenteellä laitteen "kanaviin" tai objekteihin. Se yksinkertaistaa huomattavasti dataSource ja dataTarget kenttien luomista - nämä oikotiet tai lyhenteet ovat muodoltaan huomattavasti helpommin ihmisten luettavissa ja kirjoitettavissa.

PROFIILI TIEDOSTO

-----

Yksinkertainen esimerkki profiilitiedoston XML-kuvauksesta:

Laitteessa on yksi DI kanava, jota kutsutaan nimellä DI1

```
<ioprofile>
  <name>example</name>
  <description>This is an example device</description>
  <protocol>modbusrtu</protocol>
  <iodata>
    <DI1 dir="in" autocreate="true">
      <ioparams/>
      <pointschema>BI</pointschema>
      <url>discrete/0</url>
      <initdata/>
    </DI1>
  </iodata>
</ioprofile>
```

XML-puun muoto:

□Kuvauksen juurielementti on aina <ioprofile>

□Sen alle tulevat tagit

□<name>□□Tämä on profiilin tekninen tunnus, ja tällä  
□□□nimellä siihen viitataan ohjelmassa.

□<description> □Vapaamuotoinen kuvaus (esim. dokumentaatiota varten)

□<protocol>□Protokolla jota käytetään. Tätä tekstiä käytetään protokolla  
□osana kun profiilin pisteille muodostetaan dataSource ja  
□dataTarget kenttiä.

□<iodata> □Tämä tagi sisältää laitteen objektit, kanavat - tai  
□tavallaan tietokantapisteeet. Tämän tagin alle lisätään  
□yksi tagi per laitteen io-datapiste - eli modbus rajapinnan tapauksessa  
□yksi tagi olisi tavallisesti modbus rekisteri.

□IO data tagin nimeä käytetään myös io-datapisteen tai kanavan nimenä, jolla siihen viitataan mikäli  
□kanavalle ei ole määritelty alias tagia.

□jokaiselle kanavalle voidaan määritellä attribuutit

□dir□arvo joko "in" jolloin kanavasta tehdään source,  
□□□"out" jolloin kanavasta tehdään target, tai  
□□□"both" tai "mixed" jolloin kanavalla on molemmat (oletus).

□autocreate □määrittää luodaanko datapisteelle  
□□□tietokantapiste automaattisesti vai ei.

□□□arvo joko "true" tai "false", ja false on oletus arvo.

□Sen alle voidaan laittaa seuraavat tagit kuvaamaan io-datapistettä:

□  
□<alias>□Nimi jolla kanavaan viitataan, mikäli haluttu nimi EI ole sallittu XML elementin nimi.  
□□□Huomaa!  
□□□- Tämä nimi korvaa quick URL:eissa elementin nimen.  
□□□- Vain yksi alias sallittu. t.s. viimeinen alias elementti XML dokumenttia lukiessa jää voimaan.

□  
□<ioparams>□tämän alle tulevat dataSource/Target kentän URL osan  
□□□skaalaus parametrit muodossa

□□□<parametri>arvo</parametri>

□<pointschema>□kertoo mitä scheemaa käytetään tietokantapistettä  
□□□luotaessa autocreate ohjelmassa.

□<url>□dataSource ja Target kentän URL:n loppuosa, joka tulee  
□□□väyläosoitteen jälkeen. Kertoo usein rekisterityypin tai

rekisterin/tietueen numeron. Modbus:n tapauksessa  
muoto on "rekisteriTyyppi/rekisterinNumero"

<initdata> liittyy myös automaattisesti pisteiden automaattiseen  
luontiin; tämän tagin alla määritellään  
luodun pisteen kenttien alkuarvot (jos niiden halutaan  
olevan muuta kuin oletukset).

<terminals> Tässä kentässä voidaan kertoa dokumentaatiota varten,  
mitkä liittimet liittyvät tähän datapisteeseen.

## LYHENNETYT SOURCE JA TARGET KENTÄT

-----

Kun laitteesta ja sen väylärajapinnasta on olemassa profiilitiedosto,  
voidaan sen yhteydessä käyttää lyhennettyä "io://" alkuista dataSource ja  
dataTarget URL muotoa.

Käytämme yllä olevaa esimerkkiä modbus väylän kautta liitettävästä  
laitteesta, jossa on 1 kpl DI kanavia.

Pisteen dataSource kenttään voidaan kirjoittaa seuraava URL jolla  
kanavan tila voidaan lukea pisteen raw ja pv kenttiin:

io://modbus1/example/10/DI1

Jossa..

modbus1 tarkoittaa sellaista modbus porttia, jonka alias kentän  
arvoksi on määritetty "modbus1" (tämä on järjestelmän  
oletus modbus portti).

example on käytettävän ioprofiilin nimi

10 on väyläosoite - tässä tapauksessa laitteen modbus-osoite.

DI1 tämä on viittaus profiilin <iodata> osiosta löytyvään  
tagiin, josta halutun kanava tai io-datapisteen  
kuvaus löytyy.

Näitä lyhennettyjä URL:eja voi käyttää kaikkien sellaisten laitteiden  
yhteydessä, joista on saatavilla io-profiili.

\*\* FIN \*\*

# Webservices

Tuki tiedon lukemiseen WEB Service rajapintojen ylitse

Tuki lisätty 6.2.2018:

Pääsääntöisesti RESTful datan lukeminen web service rajapinnan kautta ei tarvitse laisinkaan ohjelmointia, mikäli data tarjotaan XML tai JSON muodossa. Kahden edellä mainitun datamuodon lisäksi web service -toteutus osaa hakea tiedon n.s. RAW muodossa, joka tarkoittaa sitä että web palvelun tarjoamaa dataa EI pyritä parsimaan millään tavalla. Sen sijaan siihen voidaan soveltaa Lua:n merkkijonoille tarkoitettuja pattern-matching funktiota, joiden avulla pyritään löytämään tärkeät osuudet datasta.

Toiminta:

Web service rajapinta hyödyntää cURL kirjastoa ja komentorivityökalua.

Käyttö:

Web service rajapintaa käytetään luomalla 'webService' -tyyppinen piste tietokantaan. Kyseisiä pisteitä pollataan noin 5 sekunnin välein, ja mikäli pisteen kohdalla edellisestä kyselystä kauemmin kuin pisteen updateInterval -kentässä on määritelty, suoritetaan uusi kysely.

Pisteeseen täytyy syöttää kaksi arvoa, dataSource joka on web palvelun URL osoite, ja selects -taulukko, joka kertoo mitä arvoja datasta halutaan kerätä talteen. Ohjelma kerää web palvelun palauttamasta datasta selects -taulukon perusteella arvoja, ja kirjoittaa ne pisteen ".data" -taulukkoon.

Esimerkki 1:

Luodaan webService tyyppinen piste "openWeatherMap"

web services URL osoite on:

<http://api.openweathermap.org/data/2.5/weather>

Lisäksi kyselyyn pitää asettaa parametrit appid (ikään kuin salasana), q (sijainti), sekä units (yksikkö)

query parametrit:

appid=a6d3f6617ad0c5ad24cc00d673ddceec&q=Pirkkala,fi&units=metric

Niinpä pisteen "openWeatherMap.dataSource" arvoksi tulee:

<http://api.openweathermap.org/data/2.5/weather?appid=a6d3f6617ad0c5ad24cc00d673ddceec&q=Pirkkala,fi&un>

Vastauksena saadaan JSON data:

```
{
  "coord": {
    "lon":23.65,
```

```

    "lat":61.47 },
    "weather":[
      {
        "id":600,
        "main":"Snow",
        "description":"light snow",
        "icon":"13d"
      }
    ],
    "base":"stations",
    "main": {
      "temp":-13,
      "pressure":1009,
      "humidity":92,
      "temp_min":-13,
      "temp_max":-13
    },
    "visibility":10000,
    "wind":{
      "speed":0.5,
      "deg":40
    },
    "clouds":{
      "all":75
    },
    "dt":1517988000,
    "sys":{
      "type":1,
      "id":5045,
      "message":0.0045,
      "country":"FI",
      "sunrise":1517985075,
      "sunset":1518014928
    },
    "id":641491,
    "name":"Pirkkala",
    "cod":200
  }
}

```

Jos meillä on tarkoitus lukea luetusta datasta esimerkiksi:  
 ulkolämpötila, ulkokosteus, auringon nousuaika, auringon laskuaika ja pilvisuus

Luodaan "openWeatherMap.selects" taulukkoon seuraavat rivit (selite suluissa,  
 sitä ei lisätä oikeasti pisteeseen):

```

#1 main.temp      (ulkolämpötila)
#2 main.humidity  (kosteus tieto)
#3 sys.sunrise    (auringon nousuaika)
#4 sys.sunset     (auringon laskuaika)
#5 clouds.all     (pilvisuus)

```

Syntaksi selects taulukossa on sama kuin Lua -kielessä taulukon soluihin viittaaminen!  
 Jos luetussa JSON datassa on taulukoita, soluihin viitataan numerolla [] sulkujen sisällä, ja  
 indeksointi alkaa numerosta yksi (1), ei nolasta.

Selects taulukon viittaamat data solut löytyvät "openWeatherMap.data" -taulukosta samoilla indekseillä. Eli koska main.temp on ensimmäisellä rivillä .selects taulukossa, löytyy ulkolämpötila .data taulukon ensimmäiseltä rivitä, ja toisella rivillä on suhteellinen kosteus.

-

Esimerkki 2:

Parsitaan HTML sivulta Chuck Norris "faktoja".

Luo piste "ws/norrisFacts"

Aseta web services URL osoitteeksi:

<http://bleacherreport.com/articles/43450-humor-top-100-chuck-norris-facts>

Encoding parametriksi pisteeseen annetaan "RAW", joka tarkoittaa että vastauksena tulee suoraan HTML/teksti-tyyppistä dataa, eikä sille tehdä dekoddausta.

Selects taulukko tarvitsee vain yhden rivin, jonka arvoksi tulee "<li>[%w%s%p]-</li>".

Tämän jälkeen pisteen data taulukosta pitäisi löytyä noin 108 riviä. Niitä tutkimalla voidaan havaita, että 3 ensimmäistä, ja 13 viimeistä eivät ole oikeata dataa. Ne tulee jättää huomiotta kun dataa haetaan pisteestä

Luomme tässä esimerkissä vielä pisteen, joka noukkii satunnaisen faktan ".data" riveiltä.

Luo piste "norris/fact", joka on tyyppiä "none".

lisää siihen kenttä "script", ja anna sille arvoksi seuraava pieni skriptin pätkä:

```
if ( (os.time() % 60) > 0) then return; end; local raw = Data.get ('ws/norrisFacts.data'); local n = (#raw or 0); local
```

Nyt pisteen pv -kenttään haetaan satunnainen Chuck Norris fakta, jonka voi vaikkapa näyttää käyttöliittymässä.



# SQLite tietokanta

## SQLite 3

SLC engine käyttää SQLite tietokantakirjastoa moniin eri toimenpiteisiin, ja sitä voi hyödyntää vapaasti sovellusohjelmissa API:n kautta myös omiin tarpeisiinsa.

Joissakin versioissa SQLite -tietokanta oli asetettu n.s. async moodiin, kirjoitus operaatiot saattoivat limittyä, tiedostojärjestelmän operaatioiden kanssa, jolloin tiedostojärjestelmässä oleva versio tietokannasta ei välttämättä ole aina sisäisesti täysin yhdenmukaisessa tilassa. Tällöin seuraava kirjoitusoperaatio voi alkaa jo siinä vaiheessa, kun edellisen muutoksen data on vielä kirjoituspuskurissa, odottamassa levyllekirjoittamista.

Tämä muodostuu ongelmaksi siinä vaiheessa, kun laitteelta katkaistaan syöttöjännite ilman "shutdown" komentoa, jolloin tiedostojärjestelmän kirjoituspuskureita ei ehdiä vielä levyille saakka.

Tällaisen epäkoherentissa tilassa olevan tietokantatiedoston voi osittain pelastaa tekemällä tietokannasta n.s. dumpin, esimerkiksi shell-komennolla:

```
sqlite3 mydata.db ".dump" | sqlite3 new.db
```

Trendlog tietokannan koon arviointi:

Demon pistetietokannassa on 423kpl analog-value pisteitä, ja jokaisen .pv arvosta 5000 näytettä.

Tietokannan koko on 122 612 736 tavua.

Näin ollen yhden historianäytteen koko on keskimäärin 58 tavua, tai yhden 1000 näytettä vastaa aina noin 58 kilotavua.

Näytteenottotaajuus:

ARM Cortex-A9 (1 GHz) pohjaisella TI Sitara prosessorilla 450 analogisen -pisteen pv-kentän arvojen tallentaminen kestää noin 3 - 4 sekuntia, joten tuolla näytetaajuudella on mahdollista päästä suhteellisen luotettavasti 5 sekunnin näytteistysväliin.

# Webservice-nordpool

Pohjoismaisen pörssisähkön hintojen lukeminen:

Rajapinnan URL:

[https://www.nordpoolgroup.com/api/marketdata/page/\[%i\]](https://www.nordpoolgroup.com/api/marketdata/page/[%i])

[%i] == luentatapa

HOURLY = 10, DAILY = 11, WEEKLY = 12, MONTHLY = 13, YEARLY = 14

Data muoto: JSON

MUUTA: ainakin seuraavat GET parametrit toimivat

'currency' valuutta koodit, esim: EUR, SEK

'endDate' Halutun datan loppu tai alku aika, muoto 'DD-MM-YYYY'

Esimerkki toteutuksia:

DataSource:

<https://www.nordpoolgroup.com/api/marketdata/page/11?currency=EUR>

select:

data.Rows[2]Columns[6].Value

data.Rows[1]Columns[6].Value

Python-kielinen toteutus malliksi:

<https://github.com/kipe/nordpool/tree/master/nordpool>

# HVAC-kirjasto

## HVAC kirjasto

Sisältää monia perusfunktioita ja pistetyyppejä LVIO automaatiota ajatellen.

Pistetyypit ja funktiot:

### \* hvacPIDController

Säädin piste. Voi sisältää vapaan määrän säätöportaita. Virittämiseen on porraskohtainen suhdealue, ja säädinkohtainen integrointi- ja derivointiaika.

Jotta säädin toimisi, se tarvitsee toimiakseen kutsun hvac\_runPID() funktiolle.

Funktio: hvac\_runPID (idPID [, input, en])

Funktiokutsu tarvitsee pakollisesti

idPID (string) säädin pisteen nimi tietokannassa (tyyppi vacPIDController)

input (valinnainen, real) säätimen feedback, eli mittauksen arvo real lukuna

en ( valinnainen, int) säätimen tila, eli vaikkapa pumpun käyntitila

Palauttaa true jos kutsu onnistui, muuten false

Esimerkki (suoraan vacPIDController pisteen script kentässä):

```
hvac_runPID (id)
```

### \* hvacCurve

Yleiskäyttöinen muunnoskäyrä, jota voi käyttää esimerkiksi lämpötilan säätöön.

Voi sisältää täysin vapaan määrän pisteitä. Muunnos käyrästöllä tehdään hvac\_curve() funktiolla.

Funktio: hvac\_curve (idCurve, dataIn)

idCurve (string) pakollinen. Käyrän tietokantapisteen nimi.

dataIn (real) pakollinen. Käyrälle vietävä (muunnettava) lukuarvo.

Funktio palauttaa käyrältä muunnetun arvon.

#### \* hvacSchedule

Viikko pohjainen aikaohjelma piste.

Jokainen päivä voi sisältää vapaan määrän tapahtumia. Käytetään yhdessä hvac\_scheduler() funktion kanssa.

Funktio: hvac\_scheduler (idSch [, tEpoch] )

idSch (string) Aikaohjelmapisteen nimi tietokannassa

tEpoch (int, valinnainen) jos halutaan, että ajastusta ei tehdä oikean kellonajan mukaan, tällä voidaan viedä haluttu aika funktiolle unix epookkina.

#### \* hvacTimer

Piste kuvailee yleiskäyttöisen viiveen. Käytetään yhdessä hvac\_timer() funktion kanssa.

Toimii siten, että seuraa pisteen input kenttää (tai kutsussa rValue parametria) ja mikäli siinä havaitaan nouseva tai laskeva reuna (eli sen arvo muuttuu), asettaa funktio pisteen output kentän samaan arvoon kuin input kenttä annettujen viiveiden kuluttua.

Eli jos input muuttuu arvosta 0 arvoon 2, muuttaa funktio pisteen output kentän arvoon kaksi onDelay (nouseva reuna) millisekunnin kuluttua. Jos viiveen aikana input palaa takaisin arvoon 0, resetoituu viiveen laskenta alkuun.

Samaa logiikkaa käytetään myös, kun input kentän arvo pienenee, eli siinä havaitaan laskeva reuna.

hvacTimer pisteellä oleva stepSize parametri määrää, kuinka paljon output voi yhden viivekierroksen jälkeen muuttua. Eli jos ..

stepSize = 1 ja output = 0

.. ja input muuttuu 0 -> 3, ja onDelay = 2000

Oletuksella että hvactimer() funktiota kutsutaan ainakin kerran kahdessa sekunnissa, muuttuu pisteen output kahden sekunnin päästä arvoon 1, ja jälleen kahden sekunnin päästä arvoon 2. Tällä ajastimella on siis mahdollista tehdä myös ramppoja (vaikkapa analogi pisteisiin), asettamalla stepSize arvoksi sopiva luku.

Funktio: hvac\_timer (idtimer [, rValue, rOnDelay, rOffDelay] )

idtimer (string) ajastin pisteen nimi tietokannassa.

rValue (real, valinnainen) Haluttu input arvo ajastimelle. Jos tätä ei anneta kutsussa, funktio käyttää tietokantapisteen input kentässä olevaa arvoa.

rOnDelay (real, valinnainen) Haluttu päällemeno viive millisekunteina. Jos tätä ei anneta, funktio käyttää pisteessä onDelay kentässä olevaa arvoa.

rOffDelay (real, valinnainen) Haluttu poismeno viive millisekunteina. Jos tätä ei anneta, funktio käyttää pisteessä offDelay kentässä olevaa arvoa.

\* Funktio: hvac\_limit (pointId [, hiLimit, loLimit] )

Tämä kutsu rajoittaa pisteen pv kentän haluttujen rajojen väliin. Tämä funktio ei liity mihinkään tiettyyn pistetyyppiin, vaan sitä voi käyttää kaikkiin sellaisiin pisteisiin, joilla on pv kenttä.

pointId (string, pakollinen) Rajoitettavan pisteen nimi

hiLimit (real, valinnainen) pisteen pv kentä numeraalisen arvon yläraja.

loLimit (real, valinnainen) pisteen pv kentä numeraalisen arvon alaraja.

Mikäli hiLimit tai loLimit parametreja ei anneta kutsussa, funktio koettaa löytää annetulta pisteeltä highLimit tai lowLimit kenttiä, ja käyttää niissä olevia arvoja rajoituksen tekemiseen.

HUOM! Nykyisessä slcenginen versiossa tehdään pisteen pv kentän rajoitus automaattisesti, jos pisteelle on luotu highLimit ja/tai lowLimit kentät. Tätä funktiota tarvitaan siis vain erikoistapauksissa.

# Piste-skeemat

## YLEISTÄ:

Tietokantapiste on tietyltä kannalta ajateltuna Lua taulukko - lua tyyppinä "table". Koska lua-tilukko ei sinänsä aseta mitään rajoituksia sille, millaista tietoa siihen voidaan tallentaa, SLC enginen pistetietokanta toteuttaa pisteille skeema tarkastelun. Aina kun pisteen johonkin kenttään kirjoitetaan arvo, kyseistä arvoa verrataan pisteen skeemaan, ja mikäli kirjoitettava arvo ei vastaa skeeman määritystä - ja mikäli sitä ei voida konvertoida oikeaan muotoon, kirjoitus epäonnistuu.

Tietokantapisteiden skeemat seuraavat pääpiirteittäin "JSON schemas" määritystä, mutta hieman yksinkertaistettuna ja toisaalta, sisältäen taas joitakin laajennoksia. Pistetietokanta asettaa jo lähtökohtaisesti kuitenkin joitakin lisärajoituksia; Koska skeemalla kuvataan pistetietokantaobjektia, tulee ylimmän tason tyyppin olla sen vuoksi aina "object" - skeema jossa näin ei ole, hylätään.

## SKEEMAN, ELI PISTETYYPIN LUOMINEN:

Uusi pistetyyppi - eli skeema, luodaan Data -kirjaston kutsulla createSchema.

```
Data.createSchema ( strSchemaName, schemaModel )
```

### Argumentit:

strSchemaName Merkkijono joka tulee uuden skeeman nimeksi. Voi sisältää merkkejä a-z, A-Z ja \_

### Paluuarvo:

Kutsu palauttaa arvon true jos onnistui, tai nil jos epäonnistui.

Jos haluaisi kuvata vaikkapa muuttujan, jonka täytyy sisältää kokonaisluku, väliltä 0 .. 1, olisi skeema esimerkiksi seuraavanlainen:

```
{type="int", default=0, minimum=0, maximum=1}
```

Mutta kuten ylempänä jo sanottiin, täytyy tietokantapisteen olla aina ylimmällä tasolla object tyyppinen - se on vaatimus, jonka ohjelma tarkistaa. object -tyyppi tarkoittaa taulukkoa, joka sisältää rivejä, joiden nimet ovat merkkijonoja. Kun asiaa miettii, juuri sellaisiahan datapisteet ovat; taulukko, joka sisältää rivejä, joilla on nimiä kuten "pv", tai "description".

Näitä tyypejä "object", "array", "int", "real", "string", "boolean" kutsutaan datatyypeiksi. Jokaisella datatyyppillä on lista lisävaatimuksia, joita voidaan määrätä kyseisen tyyppiselle tiedolle. Esimerkiksi object tyyppiselle datalle voidaan esittää vaatimuksia siitä, millaisia rivejä sen on pakko sisältää.

Array tyyppiselle, eli taulukolle (tai joskus kutsutaan listaksi) voidaan määrätä minimi tai maksimi määrä rivejä, ja jokaiselle riville jokin pakollinen muotovaatimus (esimerkiksi että rivit ovat kokonaislukuja).

Lua -kielen kannalta skeema on vain tietyllä tavalla muotoiltu taulukko. Kun kyseinen taulukko annetaan parametrina `Data.createSchema ()` -funktiolle, kyseinen funktio laittaa tuon annetus taulukon muistiin, ja järjestelmä vertaa niitä datapisteitä siihen, joiden on ilmoitettu olevan tuota tyyppiä.

Esimerkki: BI-piste:

```
Data.createSchema ("BI", {
  type="object",
  required={"class", "description", "pv", "priority", "inactiveText", "activeText", "faultStatus",
    "outOfService", "reliability", "description", "commStatus", "commTxt", "dataSource"},
  additionalProperties=true,
  properties={
    class={type="text", default="BI", fixed="BI"},
    description={type="text", default="BI object"},
    pv={type="int", default=0, minimum=0, maximum=1},
    priority={type="int", default=16, minimum=1, maximum=16},
    inactiveText={type="text", default="Off"},
    activeText={type="text", default="On"},
    faultStatus={type="integer", default=0, minimum=0},
    outOfService={type="integer", default=0, minimum=0, maximum=1},
    reliability={type="integer", default=0, minimum=0},
    commStatus={type="integer", default=0},
    commTxt={type="text", default=""},
    dataSource={type="text", default=""}}
  }
})
```

#### DATATYYPIT JA NIIDEN OMINAISUUDET:

**boolean** totuusarvo joka voi saada arvon true tai false. Yksinkertaisin skeemojen tukema tietotyyppi.

"fixed" tällä määreellä voidaan pakottaa muuttujan arvo halutuksi.

"default" oletusarvo kun muuttuja luodaan.

Aliakset: bool

**int** Kokonaisluku. ARM arkkitehtuurilla tämä datatyyppi muutetaan 32 bittiseksi kokonaisluvuksi.

"fixed" tällä määreellä voidaan pakottaa muuttujan arvo halutuksi.

"default" oletusarvo kun muuttuja luodaan.

"minimum" pienin numeraalinen arvo jonka muuttuja voi saada

"maximum" suurin numeraalinen arvo jonka muuttuja voi saada



"enumerated" lista arvoista joihin muuttuja voidaan asettaa.

Aliakset: integer

number Reaaliluku, jota kuvataan 64 bittisellä liukuluvulla. Lua kielen natiivi numerotyyppi.

"fixed" tällä määreellä voidaan pakottaa muuttujan arvo halutuksi.

"default" oletusarvo kun muuttuja luodaan.

"minimum" pienin numeraalinen arvo jonka muuttuja voi saada

"maximum" suurin numeraalinen arvo jonka muuttuja voi saada

"enumerated" lista arvoista joihin muuttuja voidaan asettaa.

Aliakset: real, float

array Lista tyyppinen juoksevasti kokonaisluvuilla indeksoitu taulukko, jonka rivit on nimetty juoksevasti kokonaisluvuilla.

HUOM! Vaikka lua -kieli ei varsinaisesti erottelekaan array ja object tyyppisiä taulukoita, on myös lua kielessä nopeampaa iteroita taulukon solut lävitse `ipair()` kuin `pair()` kutsulla, eli listat ovat nopeampia käsitellä kuin hash taulukot.

"itemNamees" Tällä listalla voidaan antaa taulukon riveille näyttönimet.

"minItems" minimi-määrä rivejä taulukossa.

"maxItems" maksimi-määrä rivejä taulukossa.

"items" on kutakin riviä kuvaava skeema - jokainen taulukon rivi joutuu seuraamaan tätä mallia!

object Taulukko, jonka rivit on indeksoitu merkkijonoilla, eli hash taulukko.

Huom! Tämä datatyyppi ei ole täysin natiivi lua -kielen kannalta, sillä lua -taulukko voisi sisältää sekä numeraalisesti indeksoituja rivejä, että merkkijonoilla indeksoituja. Eli se ei tee eroa object ja array tyyppien välillä. Tämä käytäntö periytyy Javascript kielestä, ja JSON notaatiosta.

"required" kenttä on lista merkkijonoja, joka määrää mitä rivejä objektin on pakko sisältää

"additionalProperties" on true/false arvo, joka ilmoittaa voiko objekti sisältää muitakin kuin 'properties' taulukossa kuvattuja rivejä.

"properties" on hash-taulukko jossa on annettu taulukon jokaisen rivin oma skeema, joka siis kuvaa kyseiseen riviin sovellettavaa

skeemaa.