HSYCO Documentation

VER 3.3.0

Contents

1	About HSYCO	10
1.1	Files Organization	11
1.1.1	Access Control List	17
1.1.2	Log Files	19
1.2	User Interface	23
1.2.1	Project	25
1.2.1.1	Project Style	31
1.2.2	Page	33
1.2.2.1	Directives	37
1.2.3	UI Objects	40
1.2.3.1	Hephais (UI Object)	42
1.2.3.2	Text	43
1.2.3.3	Wyrestorm (UI Object)	45
1.2.3.4	Tecnoalarm (UI Object)	46
1.2.3.5	Paradox (UI Object)	47
1.2.3.6	NuvoMini (UI Object)	48
1.2.3.7	Nuvo (UI Object)	49
1.2.3.8	Fp2000small (UI Object)	50
1.2.3.9	Fp2000 (UI Object)	51
1.2.3.10	Bentel (UI Object)	52
1.2.3.11	ButtonIcon	53

Contents

1.2.3.12	**Button**	54
1.2.3.13	**3button**	55
1.2.3.14	**Submit**	56
1.2.3.15	**CSx75 (UI Object)**	58
1.2.3.16	**Inim (UI Object)**	59
1.2.3.17	**Video**	60
1.2.3.18	**Weather**	62
1.2.3.19	**UserList**	63
1.2.3.20	**UserImage**	65
1.2.3.21	**User**	68
1.2.3.22	**Timer**	71
1.2.3.23	**TimePanel**	72
1.2.3.24	**Time**	74
1.2.3.25	**TempMini**	77
1.2.3.26	**Temp**	78
1.2.3.27	**SubmitImage**	79
1.2.3.28	**SliderButton**	81
1.2.3.29	**Slider**	83
1.2.3.30	**SelectPanel**	86
1.2.3.31	**Select**	89
1.2.3.32	**Selector**	92

Contents

1.2.3.33	**Scheduler**	**93**
1.2.3.34	**RadioButton**	**96**
1.2.3.35	**Panel**	**98**
1.2.3.36	**Marquee**	**100**
1.2.3.37	**Link**	**102**
1.2.3.38	**KeypadPanel**	**105**
1.2.3.39	**Keypad**	**108**
1.2.3.40	**IR**	**111**
1.2.3.41	**ImageLink**	**113**
1.2.3.42	**Image**	**115**
1.2.3.43	**Hsycomonitor**	**117**
1.2.3.44	**VBar**	**118**
1.2.3.45	**Hbar**	**119**
1.2.3.46	**DMXRGB**	**120**
1.2.3.47	**DMX**	**121**
1.2.3.48	**Dimmer**	**122**
1.2.3.49	**DatePanel**	**123**
1.2.3.50	**Date**	**126**
1.2.3.51	**Datalogger**	**129**
1.2.3.52	**SMS (UI Object)**	**133**
1.2.3.53	**Guardall (UI Object)**	**134**

Contents

1.2.3.54	**Elmo (UI Object)**	135
1.2.3.55	**Galaxy (UI Object)**	136
1.2.3.56	**Input**	137
1.2.3.57	**Container**	140
1.2.3.58	**Checkbox**	141
1.2.3.59	**Chart**	143
1.2.3.60	**CameraPanel**	147
1.2.3.61	**CameraLink**	150
1.2.3.62	**Camera**	151
1.2.3.63	**ButtonImage**	153
1.2.3.64	**Background**	154
1.2.3.65	**Ats (UI Object)**	155
1.2.3.66	**Aton (UI Object)**	156
1.2.3.67	**Airzone (UI Object)**	157
1.2.3.68	**AdminLink**	158
1.2.4	**The User Interface Configuration Database**	159
1.2.5	**URL Keywords**	162
1.3	**Manager**	163
1.3.1	**Admin**	165
1.3.2	**Project Editor**	169
1.3.2.1	**JavaScript Command and Utility Functions API**	173

Contents

1.3.2.2	Audio and Public Announcement	202
1.3.3	File Manager	204
1.3.4	Log Viewer	206
1.3.5	Status Browser	207
1.3.6	System Monitor	209
1.3.7	Wiring Editor	210
1.3.7.1	Contatto	216
1.3.7.2	Domino	259
1.3.7.3	Dummy	292
1.3.7.4	HwgIO	295
1.3.7.5	KNX	297
1.3.7.6	Knxtopo.txt	309
1.3.7.7	MyHome	311
1.3.8	KNX Utility	329
1.3.9	BACnet Utility	336
2	Products	338
3	Support	339
4	FAQ	340
5	Installation Guide	342
6	Settings	351
6.1	Camera URLs	390

Contents

6.2	Encoder URLs	391
6.3	Enttec	392
7	Programming	394
7.1	Action Keywords	396
7.2	Draw API	410
7.3	Event Keywords	416
7.4	Events Programming	430
7.5	JavaScript Programming	440
7.6	Java Programming	445
7.7	Java Callback Methods API	447
7.8	Java Command and Utility Methods API	460
7.9	JavaScript Callback Functions API	491
7.10	Working with Forms	504
7.11	Working with User Objects	508
8	Advanced Programming	511
8.1	Custom Drivers Callback Methods API	512
8.2	Custom Drivers Command and Utility Methods API	515
8.3	Custom I/O Drivers	528
8.4	Client JavaScript Extension API	532
8.5	Plugins Applications	537
8.6	Using the Embedded SQL Database Engine	540

Contents

9	**Introduction to I/O Servers**	**543**
10	**I/O Servers**	**547**
10.1	**Airzone**	**550**
10.2	**Arduino**	**567**
10.3	**Arteco**	**575**
10.4	**Aton**	**578**
10.5	**Ats**	**589**
10.6	**AxisDecoder**	**599**
10.7	**BACnet**	**605**
10.8	**Bentel**	**614**
10.9	**CSx75**	**621**
10.10	**Daikin**	**635**
10.11	**ESC/VP**	**643**
10.12	**Elmo**	**647**
10.13	**Elsner**	**654**
10.14	**Fp2000**	**657**
10.15	**GSM**	**670**
10.16	**Galaxy**	**676**
10.17	**Giacomini**	**682**
10.18	**Guardall**	**690**
10.19	**HSYCO Remote**	**698**

Contents

10.20	**Hephais**	**700**
10.21	**Hidedgesolo**	**708**
10.22	**HwgMonitor**	**712**
10.23	**Inim**	**714**
10.24	**Mitsubishi**	**725**
10.25	**Modbus**	**734**
10.26	**Modbus Server**	**745**
10.27	**Notifier**	**752**
10.28	**Nuvo**	**762**
10.29	**Paradox**	**771**
10.30	**PowerOne**	**777**
10.31	**RayCONTROL**	**780**
10.32	**SSH**	**786**
10.33	**SamsungTV**	**791**
10.34	**Sip**	**794**
10.35	**System**	**799**
10.36	**Tecnoalarm**	**804**
10.37	**Tridonic**	**811**
10.38	**WXonline**	**820**
10.39	**Wyrestorm**	**829**
10.40	**Yamaha RX-V**	**833**

About HSYCO

Home Systems Consulting S.p.A. develops state-of-the-art solutions for home and building automation systems.

Combining a clever design and unparalleled integration of leading technologies, we upgrade buildings to a different level of comfort, security and energy efficiency.

The HSYCO web-based project editor

HSYCO is an innovative Java-based software framework for home and building automation applications.

It is highly scalable, portable, secure, reliable and customizable.

HSYCO combines an efficient HTML5 GUI, with web-based design tools, and an extensive Java and JavaScript server API for the automation and control logic.

HSYCO scales well from the needs of a small home to the ones of large buildings and to those of distributed, multi-layer, supervision architectures.

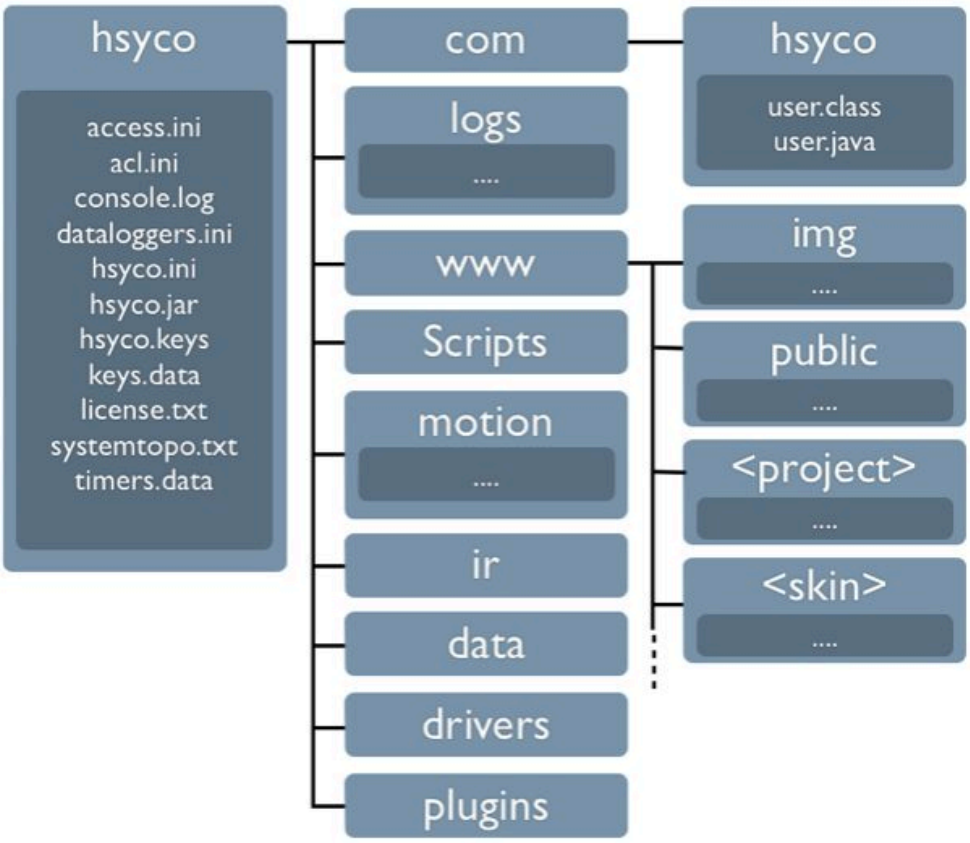
HSYCO has built-in redundancy, tasks isolation and parallel execution of user code and drivers, independent watch-dog processes for each connected device, and a high-availability model inspired by the flight control systems of the aviation industry.

HSYCO is an open environment for developers, providing a powerful application programming interface that is based on the standard and widely known Java and javaScript languages.

It can be deeply customized to respond to all specific requirements of the most complex installation, and at the same time it is easy to maintain over the typical long life cycle of any building infrastructure.

Files Organization

The general structure of HSYCO's main directory is shown in the following diagram.



hsyco/access.ini

This file contains the information corresponding to the users' PIN/PUK access codes. This file can be manually modified to create or delete a PIN code, though it is usually modified through the users management pages in the HSYCO Web interface. It is also updated by HSYCO after every access, showing the time and the IP address of the last access and the number of access errors made by every user.

> In order to guarantee the highest security, the access.ini file must be treated as a secret document (though the content of access.ini is partly encrypted, it is theoretically possible to retrieve the PIN and PUK codes with special decryption procedures.), avoiding copying or sending it via not-secure systems and restricting any access by unauthorized subjects.

---

hsyco/acl.ini

This file is used to define server-side access control lists for Web-based user commands. The optional acl.ini file defines the rules to allow or reject commands.

See the Access Control List section for additional information.

---

hsyco/console.log

console.log is a system log file. It contains information and error messages generated by the operating system or by the Java Virtual Machine.

For example, the following message:

```
Exception in thread "main" java.lang.NoClassDefFoundError: com/hsyco/OpenWebNetMonitor
at com.hsyco.hsyco.main(hsyco.java:4794)
Caused by: java.lang.ClassNotFoundException: com.hsyco.OpenWebNetMonitor
```

suggests that one of the HSYCO Java libraries is missing. This file is therefore useful in the testing stage or during the development of an additional Java component, while it does not provide any relevant information about the normal execution of the HSYCO software.

---

hsyco/hsyco.ini

This is the main configuration file, containing all HSYCO configuration parameters. It is fully described in the Configuration chapter. Any change to this file causes the automatic restart of HSYCO after a few seconds.

---

hsyco/hsyco.jar

This file contains all the components of the HSYCO software. In order to update HSYCO SERVER to a newer version, you replace this file with a new one, containing the whole code and the resources of the software in a compressed format. Any change to this file causes the automatic restart of HSYCO after a few seconds.

---

hsyco/hsyco.keys

This file contains the **SSL** certificate for the **HTTPS** web traffic cryptography. When HSYCO is started, if the file is not available, a new SSL certificate is automatically generated according to the name defined in hsyco.ini. Otherwise, HSYCO simply uses the certificate contained in this file, which could have been also generated by an official **Certification Authority (CA)**, like Verisign or Thawte. If an official certificate is not used, this file is created and managed by HSYCO without any manual intervention.

---

hsyco/keys.data

It is the database of the secure authentication keys between the Web Browser and HSYCO. This file is created and managed by HSYCO and must not be modified or erased.

---

hsyco/license.txt

This file contains the software license key needed to make HSYCO fully operational. When HSYCO is started for the first time, or if this file is removed before start-up, HSYCO creates a new file containing an encrypted unique code that identifies the computer on which the software is running. In order to unlock HSYCO, the file needs to contain the software license key, issued by Home Systems Consulting.

If the license key is not available in the file or if it is not correct, HSYCO disables all the functions of access to any field device, keeping only the HTTP and HTTPS Web Server running. It will be therefore possible to access the Web pages, but not to execute any action on or interface to the systems connected to HSYCO.

---

hsyco/systemtopo.txt

Contains the list of all the devices of the system and names of the **Wi-Fi** access points used for the location services.

This file format is described in details in the Configuration chapter.

systemtopo.txt can be modified at any time; any change is applied within a few seconds without restarting HSYCO. Web pages are automatically reloaded after any change to systemtopo.txt.

---

hsyco/timers.data

It is the database of the timers' settings, such as alarms, irrigation or light timers, managed by HSYCO. This file is created and managed by HSYCO and should not be modified or erased.

---

hsyco/com/hsyco

This directory contains the customizable Java files. In particular, the file **user.class** must always exist, even just in its original version if it has not been customized. The corresponding source file **user.java** is not necessary for the correct execution of the software.

---

hsyco/www

The www directory contains all the customizable files corresponding to the Web pages of the user interface. All the required standard files can be found in the **hsyco.jar** package and therefore are not included in www; however it is always possible to add new customized files in appropriate locations in www; these files will have priority over the standard files contained in hsyco.jar.

> Whenever HSYCO detects any change to the files in www and in its sub-directories, it automatically forces the refresh of the Web page on the connected browsers, so that all the changes are immediately visible.

The directories and the standard files will be described in the following paragraphs, although the directories and files structure can be customized as needed.

---

hsyco/www/<project>/index.hsm

The <project> directory, contains the definition of the user interface pages. Each <project> directory contains the **index.hsm** file, with the complete definition of the whole Web interface. This file format is described in the Personalization chapter.

It is not possible to use names such as img and the skins' names, which are reserved. It is also not possible to have more than one Web interface description file in the same directory.

---

hsyco/www/<project>/img

Contains the files of the images associated to the automation and lighting devices defined in the systemtopo.txt file, or any other image file used by the Web interface.

Files in this subdirectory have priority over files in the hsyco/www/img directory.

---

hsyco/www/<skin>

Customized skins can be defined and saved in www subdirectories. The files of the standard skins are included in the hsyco.jar package file. Creating a directory with the same name of a standard skin is possible, but should be done carefully, as it will override the standard skin files.

---

hsyco/www/img

Contains the files of the images associated to the automation and lighting devices defined in the systemtopo.txt file, or any other image file used by the Web interface. Files in the img/ sub-directory under the project's main directory have priority over files in this directory.

---

hsyco/www/public

The *HTTPServerPublicDirectory* parameter defined in System Settings file enables a basic Web server that serves files, without any parsing, contained in a specific directory, via HTTP and HTTPS, only to clients in the trusted range of IP addresses.

---

hsyco/motion

Contains the images of every single frame recorded from each camera. The motion directory contains a distinct subdirectory for each camera, named as the cameras ids defined in hsyco.ini. In these directories the files are organized in two additional subdirectory levels. HSYCO automatically manages these directories and the individual files.

> In order to avoid any malfunction in the reproduction and, potentially, in the recording of the events you should not partially delete the files. If needed, it is possible to delete all the files of a camera and then restart HSYCO so that the index of the images can be correctly updated.

HSYCO will automatically erase the files and their corresponding subdirectories after the time of images storage has expired, as set in hsyco.ini for each camera.

Moreover, the older frames are automatically erased before their times of expiration if the space available on disc drops below a safe margin.

---

hsyco/logs

Contains the log files generated by HSYCO. The information details contained in the log files depend on the log level settings defined in hsyco.ini. All the relevant events detected from the field, the commands sent, the users access information and errors can be traced.

When starting HSYCO the information corresponding to the execution of the different HSYCO modules will be logged. See the Log Files section for the details.

HSYCO writes a separate log file each day, in a different directory for each year, in the ***MMGG-message.log'*** *format. For example, the log file for the 12th April 2012 will be hsyco/logs/2012/0412-message.log*.

If anti-intrusion systems are integrated with HSYCO, the file **security.log** will be written too, in the same directories as the daily log files. This file contains the list of all the events concerning activation, deactivation and alarms on the anti-intrusion systems.

The log files can be removed anytime without causing any problem. HSYCO does not provide an automatic procedure to erase the log files.

---

hsyco/ir

This directory can be empty or might not exist at all. It contains the files with CCF encoded IR commands, to be used when sending IR commands with IRTrans devices.

These files should have the extension .ccfhex and the name corresponds to the IR database name.

Each file contains one or more lines per command:

```
command_name = hex_ccf_data
```

The CCF data for each command is in hexadecimal format, and could be broken in multiple lines.

---

hsyco/dataloggers.ini

This file contains the definitions of time slots for data loggers of type counter. Each line sets some rules for a time slot of a data logger, the format to be used is the following:

```
datalogger_name; slot_id; time_range; days_of_week; date; rate
```

Whenever a counter data logger listed in this file is updated, the rules are matched from top to bottom to establish the time slot the new value belongs to and thus which rate to apply.

The following table describes each field of a rule line:

Field	Format	Description
datalogger_name	string	specifies which data logger this slot is applied to
slot_id	positive integer	identifier for the slot. More lines can refer to the same slot identifier
time_range	hh:mm-hh:mm	specifies the time range in which the slot applies. An asterisk (*) in this field will always result in a match
days_of_week	list of week days	specifies the week days matched by this rule. The days are identified by a number from 1 (Monday) to 7 (Sunday). For instance, writing “135” into this field will result in a match on Mondays (1), Wednesdays (3), and Fridays (5). An asterisk in this field will always result in a match
date	yyyy/mm/dd or yyyy/mm/dd-yyyy/mm/dd	specifies the date or the date range in which the slot applies. In case of date ranges (yyyy/mm/dd-yyyy/mm/dd) the starting date or the ending date can be replaced by an asterisk indicating that this rule will apply respectively until or from the specified date. For instance the date range “*-2000/12/31” will match all the dates preceding and including December 31st 2000. If the exact date format is used (yyyy/mm/dd) then each field can be replaced by an asterisk which will match every year, month, or day. For instance “*/12/25” will match December 25th of every year. An asterisk in this field will always result in a match
rate	positive float number	specifies the rate (i.e. the multiplicative factor) to be applied for this time slot.

hsyco/data

HSYCO has an integrated SQL database engine (HSQLDB - HyperSQL DataBase) you can use in your applications via the standard JDBC API.

The database that HSYCO uses is called “hsyco” and its files are saved in the data/ directory under the base directory. We consider the data/ directory a reserved HSYCO path and discourage the use of it for storing user’s databases or in general for any customized purpose.

hsyco/Scripts

This directory can be empty or might not exist at all. It contains a copy of operating system specific scripts needed for the automatic execution of HSYCO when the server is started[Note 3].

In this directory it is also possible to write additional scripts called by the customized code in user.java.

Note 3

HSYCO SERVER is based on a Linux Ubuntu 12.04 LTS Server distribution. This version uses the Upstart system for the task and services management of the operating system. The configuration file is hsyco.conf. It can be found in the /etc/init directory and allows Upstart to correctly manage the execution of HSYCO.

hsyco/drivers

HSYCO offers a development framework to easily build your own I/O server, written in Java, so you can interface with external systems that are not part of the built-in library of supported systems.

Each driver has a dedicated sub-directory under the “drivers” directory in HSYCO’s root directory.

See the Custom I/O Drivers Development chapter for additional information.

hsyco/plugins

HSYCO allows you to bundle both the user interface design and the application code in one package that is easier to distribute and install.

Each plugin has a dedicated sub-directory under the “plugins” directory in HSYCO’s root directory.

See the Plugins Applications Development chapter for additional information.

Access Control List

HSYCO supports server-side access control lists for Web-based user commands.

The acl.ini file is a text file that defines the rules to allow or reject commands.

Changes to acl.ini are immediately effective.

Each line defines an access rule, with the following format:

```
action; user_id; location; type; command
```

Rules are evaluated starting from the first line and progressing down the file until a match is found on the user, the IP address of the Web browser, the command type and command.

On the rule that matches, the “allow” or “deny” action is taken, executing or rejecting the command.

You can user the “*” character in the user_id, service and command fields, to match any value or values starting or ending with a specific string.

The following table describes each field of a rule line.

Field	Format	Description
action	allow | deny	when this rule matches, the command is either executed or rejected
user_id	string [, string] * matches any user	one or more user names (comma separated)
location	local | remote | *	clients with an IP address that is part of the trusted range are “local”; clients outside the trusted range are “remote”
type	io | timer | camera | * * matches any service	command type
command	string * matches any string or substring	matching command string. “io” commands have the <datapoint>=<value> format for I/O Servers, dmx. <channel>=<value> for DMX, <IRTrans_id>=<command> for IRTrans commands, and <name>=<param> for user commands. “camera” type commands are formatted as <camera_name>=<function>.<action> for PTZ, or<camera_name>=<mail | download> when sending a frame via e-mail or downloading a recorded video. “timer” commands are formatted as <timer_name>=<action>

If no matching rule is found in the acl.ini file, the command is rejected. If the verbose log level is enabled, an error message is logged in the daily log file:

```
2012.01.04 16:25:10.223 + ACL ALERT: DENIED – USER: staff LOCATION: local TYPE: io COMMAND: user=test
```

Examples:

```
deny; *; *; *; dmx*
allow; *; *; *; *
```

Allows all commands from the Web interface, except for DMX control.

```
deny; *; *; camera; *zoom*wide
allow; *; *; camera; *
```

Disables all commands, only allowing camera control commands, but not the zoom-out command.

```
deny; guest; *; *; *
allow; *; *; *; *
```

User “guest” is not authorized to execute any command, all other users are allowed.

```
deny; *; remote; *; *
```

Prevents the execution of any command from browsers that are outside of the trusted range of IP addresses, as defined with the trustedNet configuration parameter.

Log Files

The log files generated by HSYCO are contained in sub-directories under hsyco/logs.

Message Log

HSYCO writes a separate log file each day, in a different directory for each year, in the MMGG-message.log format. For example, the log file for January 4th 2012 will be hsyco/logs/2012/0104-message.log.

The information details contained in the log files depend on the settings of the log options defined in hsyco.ini.

All the relevant events detected from the field, commands sent, users access information and possible errors can be traced.

Every log line starts with date and time (local time is always used according to the server settings), in the format:

```
AAAA.MM.GG HH:MM:SS:MMM
```

followed by the “+” character for errors, “-” for information messages and “*” for the remote logs generated by the JavaScript code of the Web interface.

Some examples of log messages that appear in typical situations are described below.

Starting HSYCO Without a License Key

```
2012.01.04 11:39:14.402 - HSYCO Ver. 3.0.0 Build 0086 (USER Ver. No User Code) started
2012.01.04 11:39:16.133 - Starting HTTP Server
2012.01.04 11:39:16.180 - Starting HTTPS SSL Server
2012.01.04 11:39:16.751 - HTTP server accepting connections on port: 80. Document root: www
2012.01.04 11:39:18.126 - HTTPS server accepting connections on port: 443. Document root: www
```

When the software license key is not available or is not valid, only the HTTP and HTTPS servers are activated.

HSYCO Normal Start-up Process

```
2012.01.04 11:39:14.402 - HSYCO Ver. 3.0.0 BETA Build 0086 (USER Ver. No User Code) started
2012.01.04 08:14:10.943 - Starting HTTP Server
2012.01.04 08:14:10.951 - Starting HTTPS SSL Server
2012.01.04 08:14:11.167 - HTTP server accepting connections on port: 80. Document root: www
2012.01.04 08:14:12.012 - Starting I/O Monitor [0]
2012.01.04 08:14:12.015 - Starting OpenWebNet Polling [0]
2012.01.04 08:14:12.019 - Starting Cameras Polling [0]
2012.01.04 08:14:12.021 - Starting Cameras Polling [1]
2012.01.04 08:14:12.022 - Cameras polling started [0]
2012.01.04 08:14:12.049 - Cameras polling started [1]
2012.01.04 08:14:12.078 - Starting User Time Callback Server
2012.01.04 08:14:12.110 - Starting Device Timer
2012.01.04 08:14:12.111 - User Time Callback Server started
2012.01.04 08:14:12.113 - Starting Timer Server
2012.01.04 08:14:12.115 - Device Timer started
2012.01.04 08:14:12.122 - Starting SysLog UDP Server
2012.01.04 08:14:12.124 - Timer Server started
2012.01.04 08:14:12.127 - Starting Music Players Status Polling
2012.01.04 08:14:12.134 - SysLog UDP server accepting packets on port: 514
2012.01.04 08:14:12.138 - Music Players Polling started
2012.01.04 08:14:12.143 - Starting IRTrans Monitor [0]
```

```
2012.01.04 08:14:12.146 - Starting SlimServer Monitor
2012.01.04 08:14:12.151 - IRTrans Monitor [0] connection established
2012.01.04 08:14:12.179 - SlimServer Monitor connection established
2012.01.04 08:14:12.200 - HTTPS server accepting connections on port: 443. Document root: www
2012.01.04 08:14:33.480 - ioMonitor - started [knx]
```

In normal conditions, all HSYCO subsystems are activated:

- HTTP Server
- HTTPS Server
- I/O Server (for each I/O sub-system)
- Cameras Polling (for each camera)
- User Time Callback Server
- Device Timer
- Timer Server
- Music Players Status Polling
- SysLog UDP Server
- IRTrans Monitor (for each IRTrans).

For each of these systems, you will see a log line at the beginning of the activation and one to confirm the successful activation.

I/O Servers Connection Errors

```
2012.01.04 18:22:45.989 + ioMonitor - Exception [knx] - connect timed out
2012.01.04 18:22:46.401 + ioMonitor - Handshake error (2) in OpenWebNet monitor thread [myhome] - SCS server
requires authentication
```

In this case, we have a time-out connection error on the KNX server and an authentication error on the OpenWebNet gateway.

IRTrans Connection Errors

```
2012.01.04 18:22:46.916 + IRTransMonitor - Couldn't get I/O for IRTrans [0] - Connection refused
```

This message signals a connection error to the first IRTrans – [0] IRTrans – defined in the IRTrans parameter in hsyco.ini.

Writing Errors of the Cameras Images

```
2012.01.04 20:00:00.000 + cameraRecSet - Camera recording skipped - Disk full
```

This message indicates that the disk storage space for saving camera images is insufficient to record new camera images.

```
2012.01.04 20:00:00.000 + cameraFlushAgedFrames -  Early delete [camera] frames - Disk full
```

This message indicates that the disk storage space for saving camera images is low, and HSYCO is deleting older images sooner than the configured expiration time.

Security Messages

All messages concerning security are identified by the heading:

```
SECURITY ALERT
```

PIN not found; only the first and the last digit of the wrong PIN are shown in the log:

```
2012.01.04 19:27:02.121 + SECURITY ALERT: CHECKAUTHREQ FAILED – REASON: PIN NOT FOUND – IP: 127.0.0.1 PIN:
1***5
```

Valid PIN, but wrong PUK; the PIN is not written in the log. HPIN is the encrypted PIN saved in access.ini:

```
2012.01.04 19:30:44.739 + SECURITY ALERT: CHECKAUTHREQ FAILED – REASON: PIN/PUK AUTH FAILED – IP: 127.0.0.1
HPIN: 556342...
```

Error in the URLKEY used to access the HSYCO Web interface:

```
2012.01.04 19:29:47.111 + SECURITY ALERT: CHECKPINPAGEREQ FAILED – REASON: KEY NOT FOUND – IP: 127.0.0.1
KEY: alsldkekds
```

The access.ini file has been modified and, as a consequence, reloaded by HSYCO:

```
2012.01.04 19:31:48.383 – SECURITY ALERT: access.ini file reloaded
```

Login and logout messages of the user identified by the default name:

```
2012.01.04 19:28:16.734 – SECURITY ALERT: USER LOGIN – IP: 127.0.0.1 USER: default
2012.01.04 19:29:08.159 – SECURITY ALERT: USER LOGOUT – IP: 127.0.0.1 USER: default
```

New user created through the users management Web page:

```
2012.01.04 09:48:41.275 – SECURITY ALERT: WEB ADMIN – USER [myuser]  NEW [ack]
```

Changes to an existing user from the users management Web page:

```
2012.01.04 09:50:40.000 – SECURITY ALERT: WEB ADMIN – USER [myuser] UPDATE [ack]
```

User deleted from the users management Web page:

```
2012.01.04 09:52:40.000 – SECURITY ALERT: WEB ADMIN – USER [myuser] DELETE [ack]
```

Commands Sent from the Web Interface

Examples of commands sent to I/O Servers data points, to DMX channels and user buttons:

```
2012.01.04 20:04:25.956 – WEB I/O COMMAND [domino.o133.20=flip]: [OK]
2012.01.04 21:44:18.493 – WEB I/O COMMAND [dummy.light.1=0]: [OK]
2012.01.04 21:44:31.252 – WEB I/O COMMAND [myhome.scene.94.5=flip]: [OK]
2012.01.04 21:45:11.412 – WEB USER COMMAND [var1]: 2 [OK]
2012.01.04 21:47:21.300 – WEB DMX COMMAND [100*190]: [OK]
```

Web Wi-Fi Clients Location services

When the association of a Wi-Fi device (the device should be running a Web Browser with an active HSYCO session) to an Access Point is detected, the log message is:

```
2012.01.04 20:14:45.228 – LOCATION: 192.168.0.209 LOCATED AT 2p
```

When the device is disconnected, or de-associated from an Access Point, the message is:

```
2012.01.04 20:16:17.752 – LOCATION: 192.168.0.180 POSITION UNKNOWN
```

Phone Calls Log

Call notifications managed by the telephone systems integrated to HSYCO are recorded in the log file.

```
2012.01.04 20:14:45.228 – PBX CALL [192.168.1.13]: FROM: 210 TO: 310 [OK]
```

The security.log File

If any anti-intrusion units is connected to HSYCO, the security.log file will be written in the same directories as the daily log files.

This file contains the list of all events of activation, deactivation and alarm detected on the anti-intrusion units.

```
2012.01.04 12:38:25.861 [0] OFF – ZONES 1
2012.01.04 12:46:36.392 [0] ON – ZONES 1234
2012.01.04 12:47:08.703 [0] ALARM – INTRUSION – ZONE 1
2012.01.04 12:47:25.055 [0] OFF – ZONES 1234
2012.01.04 13:14:23.629 [0] ON – ZONES 1
```

User Interface

HSYCO allows the interaction with the system through a custom user interface. We can identify three entities:

- **Project**: a single interface
- **Page**: a project can contain pages
- **Object**: a page can contain objects

A project is defined by a subdirectory inside www that contains an index.hsm file.

Whenever HSYCO detects any change to the files in www and in its subdirectories, it forces the reload of all the projects on the connected browsers.

For each sub-directory in www in which the file index.hsm exists, you can access the page with the following URL:

```
https://<name server>/<URLKey>/<subdirectory name>
```

For additional interface options, see **URL Keywords**.

To create or edit a project there are two methods: visually, through the Project Editor or by manually editing the project's source code, the index.hsm file.

Projects Syntax

Projects are described through a custom markup language that defines pages, objects and various directives that apply to the project (skin, language, size...).

The syntax for every object is described in the corresponding section.

index.hsm files can be viewed from the Project Editor (tools > source) and edited from the Editor in the File Manager.

When errors are detected, the page is not served to the Web browser. File parsing stops at the first error, so only the first line with errors is reported in the log file.

Each type of graphic object could require one or more parameters, separated by the “;” character. Empty lines or blank spaces are ignored.

> Changes to index.hsm cause an automatic page reload on the connected Web browsers.

You can insert comment lines, which are ignored. A comment line starts with the “#” character at the beginning of the line.

> Names in index.hsm are case sensitive, small and capital letters are significant; all objects defined in the standard skin have names with lower-case letters.
>
> Some objects support a variant called **identified version** to allow the **!id** dynamic modification of the text, visibility, color several other attributes with the uiSet() Java method or the UISET action.

The User Interface Configuration Database

The graphical representation of user interface objects that are directly associated with I/O servers data points is defined in a file named **systemtopo.txt**.

This file is usually created and updated automatically by the I/O servers, but can also be edited manually. It also contains location information for the WiFi location services.

The systemtopo.txt file is only used by the Project Editor, to list the user interface data points in the address field of device objects, and in the run-time user interface, for the graphic representation of each object.

The systemtopo.txt file is not required for any other core function. If your HSYCO server does not have a user interface, but only application logic implemented with the various programming APIs, systemtopo.txt would be useless and could be ignored or even removed.

See The User Interface Configuration Database section for detailed information on systemtopo.txt.

Project

A project defines a custom user interface, that allows the user to interact with the system. Each project has one or more pages that contain UI Objects.

On the index.hsm, the project's parameters are defined as directives (#<directive>)

Parameters

Skin

Defines the graphical appearance of the project.

Values

- **skin name**: the name of the skin. The standard skin provided with HSYCO is "blue"

Syntax

```
(#skin <skin name>)
```

Language

Defines the language used for all standard text messages of the Web interface.

Values

- **language id**: the language used for the project (it | en | fr | cn)

Syntax

```
(#language <language id>)
```

Header

Specifies the page header, the text that normally appears in bookmarks and as the title of the browser's window.

Values

- **title**: the project's title

Syntax

The header must be declared after all the directives and before the pages.

```
(header <title>)
```

Size

Sets the project size as width and height. If the orientation mode is enabled, a second set of width and height is required. The first set refers to landscape mode, the second to portrait mode.

Values

- **width**: the project's width in pixels
- **height**: the project's height in pixels

Syntax

```
(#size <width>x<height>)
```

Or to enable orientation mode:

```
(#size <width (landscape)>x<height (landscape)>x; <width (portrait)>x<height (portrait)>x)
```

Device Image

Enables or disables the camera or image pop-ups associated to devices in the project, as configured in the systemtopo.txt file.

Syntax

If omitted, the default value is "enabled".

```
(#deviceimage disable)
```

Kiosk

The kiosk mode removes the menu bar and borders, for digital signage or kiosks applications where you want to have a full-screen display of your pages. "lock" also disables the navigation functions.

Values

- **mode**: disabled, enabled or locked

Syntax

If omitted, the default value is "disabled". "enabled" mode:

```
(#kiosk nolock)
```

"locked" mode:

```
(#kiosk lock)
```

Scale

Sets a scaling factor to resize the whole page area from the original size.

Values

- **factor**: scale factor, as a decimal positive number. Numbers greater than 1.0 create a magnification effect, proportionally increasing the size of the page and its content; numbers smaller than 1.0 make the page and its content smaller than the original

Syntax

```
(#scale <factor>)
```

Locked

Disables any user command for this project. Users will be able to navigate between pages and see the current status, but all commands will be locked.

Syntax

If omitted, the project is unlocked.

```
(#locked <true>)
```

Background

Defines the project's background for all pages.

Values

- **image**: the image file name

Syntax

It must be defined after the header object, before the pages.

```
(background!all.background <image>)
```

> Image files are searched inside the ***www/<project's name>/img*** directory first, then inside the ***www/img*** directory.
> If the filename is set with a UISet, it's possible to load an image from the web by entering the URL, or a path relative to ***www/<skin's name>/pic*** by specifying ***/pic/*** followed by the filename

Camera List

Defines the display name of each camera. It is required when you are displaying cameras in the index.hsm file. If you need to prevent access to one or more cameras for a specific index.hsm page, you can set that camera with an empty name, “”, or the reserved words “null” or “empty”. These cameras will be skipped by the camera rolling function, or when manually going from one camera to the next.

Values

- **list**: the comma separated list of all cameras, in the same order in which the cameras' ids appear in the Cameras parameter in hsyco.ini. The names must be enclosed in double quotes.

Syntax

```
(#cameralist <list>)
```

E.g.

```
(#cameralist "garden","openspace")
```

Camera Overlay

Defines custom overlay image masks for each camera. It is optional, and if not specified HSYCO will use the standard overlays.

Values

- **list**: the list of the type of overlay for each camera defined in hsyco.ini, the elements are separated by commas:
  - "default": default overlay for the PTZ cameras
  - "": default overlay for the PTZ cameras
  - "null": no overlay
  - "filename.png": customized file, in the www/<project's name>/img or www/img directory.

Syntax

```
(#cameraoverlay <overlay list>)
```

E.g.

```
(#cameraoverlay "","null","myoverlay.png")
```

Camera Grid Headers

Defines the display name of each camera grid. It is required when you are including camera grids in the project.

Values

- **list**: the list of the cameras grids' names, listed in order starting from 1, as they appear in the CameraGrid.id parameters in hsyco.ini. The names must be enclosed in double quotes and separated by commas.

Syntax

```
(#cameragridlist <list>)
```

E.g.

```
(#cameragridlist "Garden","Living Room")
```

Style

Defines style attributes that modify the appearance of the current skin. It is used to customize a skin, changing for example the color of the background. For a complete description of possible values, see Project Style.

Values

- **style**: the semicolon separated list of attributes and values.

Syntax

```
(#style <style>)
```

E.g.

```
(#style body-background-color=red; pages-background-color=green; header-background-color=blue)
```

UISet

Defines a set of UISets that apply to the specified UI Objects. Sets are specified as *<object's id>.<attribute name>=<value>*. A UISet without the object's id is applied directly to the project object (E.g. *autoreload=false* is the same as *<project's name>.autoreload=false*)

Values

- **uiset list**: list of attributes and values.

Syntax

```
(#uiset <uiset list>)
```

E.g.

```
(#uiset autoreload=false; myinput.value=hello)
```

Accessibility

The accessible mode, which can be enabled through a UISet (see the **accessibility** attribute below), adds enhancements for voice over when the browser supports it. These allow voice over to recognize buttons and their states. A custom interface optimized for voice over can include the following UI objects:

- link
- text
- button
- user

UI Attributes

Project attributes

Name	Value	Description
kiosk	nolock	Kiosk mode, not locked
	lock	Kiosk mode, locked (user navigation is disabled)
	strict	Kiosk mode, total height excludes the height of the header/footer, so the kiosk layout can stay the same as a non-kiosk
	false	Kiosk mode disabled
lock	true	Lock the interface, so user navigation is disabled
	false	Unlock the interface, default
page	<page name>	Show specified page. It has to be different from the previous set value
pageback	<page name>	If the current page was set with a "page" attribute and its name is the same as <page name>, navigate to the previous page
accessibility	true	Enabled accessible mode
	false	Disable accessible mode

blink	false	Disable all blinking animations in the project
autoreload	true	Default, the browser will automatically reload the page when the project changes
	false	The browser won't automatically reload the page when the project changes

Project Style

The appearance of a project is defined mainly by the selected skin. It can be customized by specifying a project style or by adding a "custom.css" file to the project's directory.

Project Style Attribute

The project style attribute requires a list of semicolon-separated attribute ids and values. What follows is a list of all the possible ids and how they affect the skin's appearance.

body-background-color

Defines the color of the overall background (outside the project's page area). Requires a css color as value. E.g.

```
body-background-color=red
```

pages-background-color

Defines the color of the pages' background. Requires a css color as value. E.g.

```
pages-background-color=rgba(255,255,0)
```

pages-shadow

Defines the shadow of the project's pages area. Requires a css shadow as value. E.g.

```
pages-shadow=0 0 0 3px yellow
```

header-background-color

Defines the color of the page header's background. Requires a css color as value. E.g.

```
header-background-color=#0000FF
```

footer-background-color

Defines the color of the page footer's background. E.g.

```
footer-background-color=rgba(0,0,255,0.5);
```

popup-background-color

Defines the color of the popup's background. Requires a css color as value. E.g.

```
popup-background-color=purple
```

popup-shadow

Defines the shadow of the project's pages area. Requires a css shadow as value. E.g.

```
popup-shadow=0 0 0 3px yellow
```

> A **CSS color** can be specified as
> - hex value: *#<rr><gg><bb>* E.g. *#FF1010*
> - color name: red, blue, white...
> - as rgb: *rgb(<r>,<g>,<b>)* E.g. *rgb(255,30,30)*
> - as rgba: *rgba(<r>,<g>,<b>,<alpha>)* E.g. *rgba(255,30,30,0.5)*

custom.css

The *custom.css* file can be created inside the **www/<skin's name>** if it's to be applied to all projects with that skin, or inside the **www/<project's name>** if it should apply to a specific project.

Any css class specified in this file is applied on top of the skin's css. Here's an example of a *custom.css* file:

```
.body { /* body's background */
    background-color: yellow;
}

.pagesContainer { /* pages' background */
    background-color: red;
}

#pageheader, #pagefooter { /* header and footer's background */
    background-color: blue;
}
```

Page

Interfaces are organized in pages. There are three types of pages: **menu**, **page**, **popup**.

Each page is identified by an **id**. The id "menu" is reserved for **menu** pages. If the project supports the orientation mode, pages (of **page** and **menu** types) can have two versions with the same id and an orientation that can be portrait or landscape. When opening a page with a specific id, the page with the corresponding id and current orientation will be chosen. If it's missing, the page with same id but different or missing orientation value will be chosen instead.

Pages (of **page** and **menu** types) can also have custom gestures defined with the **gestures** attribute.

Performing a gesture when a popup is open will

Menu

Main page of the project. The home link on the interface's top bar leads to this page. There has to be one menu in each project. There can be a landscape and a portrait version, specified with an attribute.

Parameters

- **page type**: menu, page, or popup
- **orientation**: if the project is orientation enabled, specifies the orientation (landscape or portrait)
- **gestures**: list of gestures that can be performed on the page (see gestures)

Syntax

(menu) defines the beginning of the page. Orientation is specified as (menu#landscape) or (menu#portrait) when the project is orientation-enabled. (endofmenu) defines the end of the page.

```
(menu[#<orientation>])
...
(endofmenu)
```

E.g.

```
(menu)
...
(endofmenu)

(menu#portrait)
...
(endofmenu)
```

Page

A generic page, identified by a name. If the project is orientation-enabled, the orientation can be specified (if omitted, the default is landscape).

Parameters

- **page type**: menu, page, or popup
- **id**: identifies the page. It's used on link objects to link it

- **title**: title shown at the top of the page, if the skin includes it
- **orientation**: if the project is orientation enabled, specifies the orientation (landscape or portrait)
- **locations**: locations assigned to this page
- **protected**: can be null, pin or puk. Specifies if to access the page, a pin or pin and puk is requested
- **gestures**: list of gestures that can be performed on the page (see gestures)

Syntax

(page) defines the beginning of the page. Orientation is specified as (page#landscape) or (page#portrait) when the project is orientation-enabled. (endofpage) defines the end of the page.

```
(page[#<orientation>])
...
(endofpage)
```

E.g.

```
(page)
...
(endofpage)

(page#portrait)
...
(endofpage)
```

UI Attributes

Page attributes

Name	Value	Description
gestures	list of ids and values	List of gestures. E.g. *swipeup=page:menu; pinchin=user*

Popup

A popup page, identified by a name and a popup type: simple or modal. It can have a close button, if it's necessary or if specified by the *closebutton* attribute.

Simple Popup

A simple popup opens up next to the object that generated it (eg. a link or user object) and it's closed by the same object. If there's no space, the popup will be opened at the center of the page, with a close button to close it.

Modal Popup

A modal popup is an exclusive popup that will open in the center of the page, while the background is grayed out. To close it, click on the background or on the close button (if it's visible).

Parameters

- **page type**: menu, page, or popup
- **id**: identifies the page. It's used on link objects to link it
- **title**: title shown at the top of the page, if the skin includes it
- **size**: width and height of the popup

- **type**: popup type: simple or modal. The default is simple

Syntax

(popup) defines the beginning of the page, (endofpopup) defines the end of the page. A size directive is used to define the popup's size.

```
(popup <id>; <title>)
(#size <width>x<height>)
...
(endofpopup)
```

E.g.

```
(popup mypopup; My Title; modal)
(#size 300x250)
...
(endofpopup)
```

UI Attributes

Page attributes

Name	Value	Description
gestures	list of ids and values	List of gestures. E.g. *swipeup=page:menu; pinchin=user*

Popup attributes

Name	Value	Description
popuptype	simple	Simple Popup
	modal	Modal Popup
dirs	list of any directions (up,right,down or left)	Specify the priority by which the opening direction of the popup is chosen. E.g. if dirs is *up,down,right*, when opening the popup it will try to position it on top of the object that generated it. If there's not enough space (if the popup exceeds the page area), it will try opening it below the object. If there's still not enough space, it will try on the right. If all the directions fail, it will open it at the center of the page, showing a close button to close it.
locked	true	Locked: the close button is hidden and the popup can't be closed by a back gesture or by clicking on the background (if it's a modal popup).
	false	Default behaviour: the popup can be closed by a back gesture or by clicking on the background (if it's a modal popup)

closebutton	auto	The close button is visible only when needed (on modal popups and on simple popups that are shown at the center of the page)
	true	The close button is always visible
	false	The close button is never visible
width	<width>	Popup's width in pixels
height	<height>	Popup's height in pixels

Directives

General directives.

(#skin <skin name>)

This mandatory directive must be placed at the beginning of the index.hsm file, before the (header) object. The standard skin provided with HSYCO is "blue". Parameters:

- **skin name**: the name of the skin used for this Web interface

Eg.

```
(#skin blue)
```

(#language <language id>)

This directive must be placed before the (header) object, and defines the language used for all standard text messages of the Web interface. Parameters:

- **language id**: the language used for this Web interface (it | en | fr).

(#size <page size>)

This directive sets the page size of a project, and must be placed at the beginning of the index.hsm file, before the (header) object. For projects that support orientation, the syntax is:

```
(#size <landscape size>; <portrait size>)
```

(#size) can also be used to define the size of pop-up windows. In this case it should be placed just after the (popup) object, before any other object defined inside the pop-up. For example:

```
(#size 1200x800)
```

sets the size to 1200 pixels width and 800 pixels height.

```
(#size r12c18)
```

sets the size to 12 rows and 18 columns, based on the reference grid used for standard size buttons. Parameters:

- **page size**: the size in pixels or rows and columns.

(#scale <factor>)

Sets a scaling factor to resize the whole page area from the original size. It is supported on Safari iOS only. Parameters:

- **factor**: scale factor, as a decimal positive number. Numbers greater than 1.0 create a magnification effect, proportionally increasing the size of the page and its content; numbers smaller than 1.0 make the page and its content smaller than the original.

(#kiosk <mode>)

Use this directive to remove the menu bar and borders, for digital signage or kiosks applications where you want to have a full-screen display of your pages. (#kiosk lock) also disables the navigation functions. Parameters:

- **mode**: optional parameter. Set to “lock” to disable link keys and all

other navigation features.

(#locked <mode>)

Disables any user command for this project. Users will be able to navigate between pages and see the current status, but all commands will be locked. Parameters:

- **mode**: set to “true” to disable commands execution.

==(#deviceimage disable) Enables or disables the camera or image pop-ups associated to devices in the Web interface, as configured in the systemtopo.txt file.

(#location <list>)

Associates a page to one or more location ids, as defined with the LocationBases parameter in hsyco.ini. Thanks to the association between pages and locations, you can click the location name that is shown in the top- right area of the menu bar to jump to the pages associated to that area. This directive should be defined immediately after the (page) object, at the beginning of the definition of each page that you want to associate to locations. Parameters:

- **list**: semi colon separated list of location ids, ad defined with

LocationBases in hsyco.ini.

(#cameralist <list>)

This directive defines the display name of each camera. It is required when you are displaying cameras in the index.hsm file. If you need to prevent access to one or more cameras for a specific index.hsm page, you can set that camera with an empty name, “”, or the reserved words “null” or “empty”. These cameras will be skipped by the camera rolling function, or when manually going from one camera to the next. Parameters:

- **list**: the comma separated list of all cameras, in the same order in which the cameras’ ids appear in the Cameras parameter in hsyco.ini. The names must be enclosed in double quotes.

(#cameragridlist <list>)

This directive defines the display name of each camera grid. It is required when you are including camera grids in the index.hsm file. Parameters:

- **list**: the list of the cameras grids’ names, listed in order starting from 1, as they appear in the CameraGrid.id parameters in hsyco.ini. The names must be enclosed in double quotes and separated by commas.

(#cameraoverlay <overlay list>)

Using this directive, you can define custom overlay image masks for each camera. It is optional, and if not specified HSYCO will use the standard overlay for PTZ cameras. Parameters:

- **overlay list**: the list of the type of overlay for each camera defined in

hsyco.ini, the elements are separated by commas: “default” - default overlay for the PTZ cameras “” - default overlay for the PTZ cameras “null” - no overlay “filename.png” - customized file, in the www/img directory.

(#include <file name>)

Include in the index.hsm file the content of another file saved in the same directory. The name of the file to include should have the .hsm extension. You can user many (#include) in an index.hsm file, at any position in the file. Parameters:

- **file name**: the name of the file to include, without the .hsm extension.

(header <title>)

The header corresponds to the Web page header, the text that normally appears in bookmarks and at the top of the browser’s window. This object must always be the first in the index.hsm file, immediately after the general directives.

UI Objects

Interfaces are built using objects. Most objects allow interaction with the system, while some have an aesthetic purpose.

Each object has a series of **parameters** to set its position, size, address and so on. While these parameters are set by modifying the project (through the Project Editor or by modifying the index.hsm), some objects have attributes that can be modified without reloading the project. These are called **UI Attributes** and are accessible through UISet calls (from the events environment or Java). Objects with UI Attributes are identified with an **id**.

Syntax

Objects are declared in the **index.hsm** file with the following format:

```
(object_name <parameter1>; ... <parameterN>)
```

and for objects with an id:

```
(object_name!<id> <parameter1>; ... <parameterN>)
```

List of UI Objects

- Hephais (UI Object)
- Text
- Wyrestorm (UI Object)
- Tecnoalarm (UI Object)
- Paradox (UI Object)
- NuvoMini (UI Object)
- Nuvo (UI Object)
- Fp2000small (UI Object)
- Fp2000 (UI Object)
- Bentel (UI Object)
- ButtonIcon
- Button
- 3button
- Submit
- CSx75 (UI Object)
- Inim (UI Object)
- Video
- Weather
- UserList
- UserImage
- User
- Timer
- TimePanel
- Time
- TempMini
- Temp
- SubmitImage
- SliderButton

- Slider
- SelectPanel
- Select
- Selector
- Scheduler
- RadioButton
- Panel
- Marquee
- Link
- KeypadPanel
- Keypad
- IR
- ImageLink
- Image
- Hsycomonitor
- VBar
- Hbar
- DMXRGB
- DMX
- Dimmer
- DatePanel
- Date
- Datalogger
- SMS (UI Object)
- Guardall (UI Object)
- Elmo (UI Object)
- Galaxy (UI Object)
- Input
- Container
- Checkbox
- Chart
- CameraPanel
- CameraLink
- Camera
- ButtonImage
- Background
- Ats (UI Object)
- Aton (UI Object)
- Airzone (UI Object)
- AdminLink

Hephais (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

Text

For a description and a complete list of UI Objects, see UI Objects.

Shows a generic text.

Parameters

- **id**: the object's ID, used by UISets. Required
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **text**: the text. Can contain HTML tags
- **css**: optional, and defines the CSS style

Syntax

```
(text[!<id>] <pos>; <text>; <css>)
```

E.g.

```
(text!mytext r1c1; text; color:blue)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Text attributes

Name	Value	Description

text	string	Set the text

Wyrestorm (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The Wyrestorm object lets you control one output of a Wyrestorm I/O Server.

The object displays the input selected on the addressed output. If an entry is available in wyrestorm.ini for the selected input, then the specified name will be reported, otherwise the object will show the input number. This object is listed in the Project Editor's new object list only when at least one Wyrestorm I/O Server is defined.

Parameters

- **server id**: the Wyrestorm I/O server ID
- **output**: the output number to be addressed
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **CSS style**: the CSS style to apply to the inner text (optional)

Syntax

```
(wyrestorm <server id>; <output>; <position>; <CSS style>)
```

E.g.

```
(wyrestorm serverid; 2; x10y20; font-size:50px)
```

Tecnoalarm (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The user interface for the Tecnoalarm multi-area intrusion detection system:

File:UI Object tecnoalarm.png

The Tecnoalarm object is listed in the Project Editor's new object list only when at least one Tecnoalarm I/O Server is defined.

For further details refer to the Tecnoalarm I/O Server Application Note.

Parameters

- **server id**: the server ID
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(tecnoalarm <server id>; <position>)
```

E.g.

```
(tecnoalarm serverid; x10y20)
```

Paradox (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The user interface for the control and supervision of the Paradox EVO multi-area intrusion detection system:

File:UI Object paradox.png

The Paradox object is listed in the Project Editor's new object list only when at least one Paradox I/O Server is defined.

For further details refer to the Paradox I/O Server Application Note.

Parameters

- **server id**: the server ID
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(paradox <server id>; <position>)
```

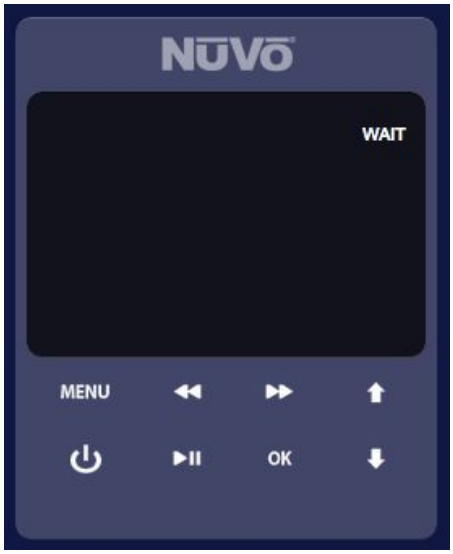
E.g.

```
(paradox serverid; x10y20)
```

NuvoMini (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The user interface for the Nuvo multi-room system:



The NuvoMini object is listed in the Project Editor's new object list only when at least one Nuvo I/O Server is defined. For further details refer to the Nuvo I/O Server Application Note.

Parameters

- **server id**: the server ID
- **zone number**: zone number of the Nuvo multi-room system
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(nuvomini <server id>; <zone number>; <position>)
```

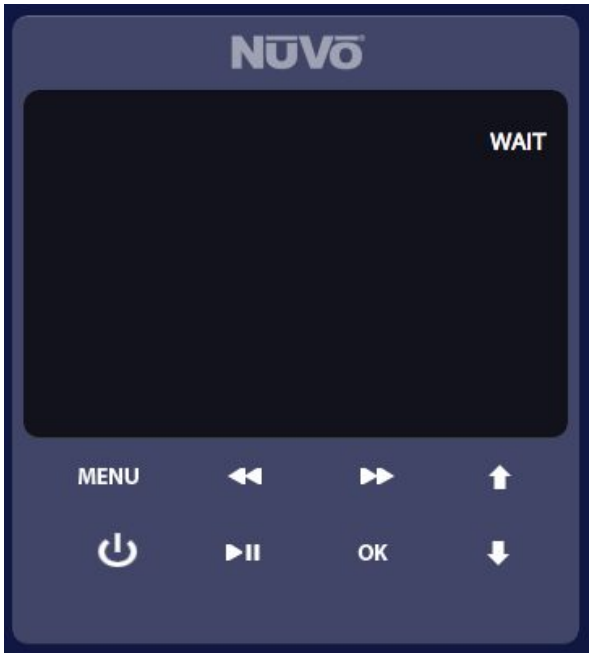
E.g.

```
(nuvomini serverid; 1; x10y20)
```

Nuvo (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The user interface for the Nuvo multi-room system:



The Nuvo object is listed in the Project Editor's new object list only when at least one Nuvo I/O Server is defined. For further details refer to the Nuvo I/O Server Application Note.

Parameters

- **server id**: the server ID
- **zone number**: zone number of the Nuvo multi-room system
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(nuvo <server id>; <zone number>; <position>)
```

E.g.

```
(nuvo serverid; 1; x10y20)
```

Fp2000small (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The user interface for the Aritech Interlogix FP2000 multi-area fire panel:

File:UI Object fp2000small.png

The Fp2000small object is listed in the Project Editor's new object list only when at least one Aritech Interlogix FP2000 I/O Server is defined.

For further details refer to the Fp2000 I/O Server Application Note.

Parameters

- **server id**: the server ID
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(fp2000small <server id>; <position>)
```

E.g.

```
(fp2000small serverid; x10y20)
```

Fp2000 (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The user interface for the Aritech Interlogix FP2000 multi-area fire panel:

File:UI Object fp2000.png

The Fp2000 object is listed in the Project Editor's new object list only when at least one Aritech Interlogix FP2000 I/O Server is defined.

For further details refer to the Fp2000 I/O Server Application Note.

Parameters

- **server id**: the server ID
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(fp2000 <server id>; <position>)
```

E.g.

```
(fp2000 serverid; x10y20)
```

Bentel (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The user interface for the Bentel KYO320 multi-area intrusion detection system:

The Bentel object is listed in the Project Editor's new object list only when at least one Bentel I/O Server is defined.

For further details refer to the Bentel I/O Server Application Note.

Parameters

- **server id**: the server ID
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(bentel <server id>; <position>)
```

E.g.

```
(bentel serverid; x10y20)
```

ButtonIcon

For a description and a complete list of UI Objects, see UI Objects.

Controls a light or automation device, has the same function of the button object.

If the buttonicon object is connected to a device or light that allows it, a long press on the button will open a popup with a slider. A second long press will close the popup. If specified on systemtopo.txt, it will show a thumbnail on mouse over or touch.

Parameters

- **address**: device data point name
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **color**: the text label on the button. Can contain HTML tags

Syntax

```
(buttonicon <address>; <position>; <label>)
```

Eg.

```
(buttonicon autom.1; x10y20; Front Door)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Button

For a description and a complete list of UI Objects, see UI Objects.

Controls a light or automation device.

If the button object is connected to a light or automation device that allows it, a long press on the button will open a popup with a slider. A second long press will close the popup. If specified on systemtopo.txt, it will show a thumbnail on mouse over or touch.

Parameters

- **address**: device data point name
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **label**: the text label on the button. Can contain HTML tags

Syntax

```
(button <address>; <position>; <label>)
```

E.g.

```
(button autom.1; x10y20; Front Door)
```

3button

For a description and a complete list of UI Objects, see UI Objects.

A 3button is similar to button. It is a single button with three function areas: going up/opening, stop and going down/closing.

Parameters

- **address**: device data point name
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **label**: the text label on the button. Can contain HTML tags

Syntax

```
(3button <address>; <position>; <label>)
```

E.g.

```
(3button autom.1; x10y20; Garage Door)
```

Submit

For a description and a complete list of UI Objects, see UI Objects.

A button used to submit the contents of a set of fields. It has several colors (depending on the skin) and three different sizes:

- **submit** (standard size)
- **submitmini** (medium size)
- **submitmicro** (small size)

For a tutorial on using this object, see Working with Forms. For a custom image version, see SubmitImage.

Parameters

- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **color**: the color of the button. It can assume any of the skin's color values (E.g. for the "blue" skin, the possible values are: b - blue; g - green; gr - gray; r - red; y - yellow; glass)
- **label**: the text label on the button. It can contain HTML tags

Syntax

```
(submit!<id> <pos>; <color>; <label>)

(submitmini!<id> <pos>; <color>; <label>)

(submitmicro!<id> <pos>; <color>; <label>)
```

E.g.

```
(submit!submit1 x10y20; r; Submit)

(submitmini!submit1 r0c1; glass; Submit)

(submitmicro!submit1 x0y100; gr; Submit)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*

visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Button attributes

Name	Value	Description
color	r,g,b...	Set the button's color. Colors vary depending on the selected skin
enabled	true	Default. Enable the button
	false	Disable the button

CSx75 (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The user interface for the Aritech CSx75 multi-area security system.

The CSx75 object is listed in the Project Editor's new object list only when at least one Aritech CSx75 I/O Server is defined.

For further details refer to the CSx75 I/O Server Application Note.

Parameters

- **server id**: the server ID
- **partition**: partition number of the Aritech CSx75 multi-area security system
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(csx75 <server id>; <partition>; <position>)
```

E.g.

```
(csx75 serverid; partition; x10y20)
```

Inim (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The user interface for the control and supervision of one area of the system, the zones and the visualization of the security log:

The Inim object is listed in the Project Editor's new object list only when at least one INIM I/O Server is defined.

For further details refer to the Inim I/O Server Application Note.

Parameters

- id - the id assigned to the INIM I/O Server in hsyco.ini
- area - the number of the area to be addressed
- pos - the object's position. Use the pixels or rows and columns coordinates format.

Syntax

```
(inim <server id>; <area>; <position>)
```

E.g.

```
(inim serverid; area; x10y20)
```

Video

For a description and a complete list of UI Objects, see UI Objects.

Displays a video. According to the HTML5 standard, the supported video formats are mp4, webm, and ogv, but actual support depends on the Web browser being used to access the pages. It allows dynamic control of position, visibility, play/pause and mode.

Parameters

- **id**: the object's ID, used by UISets
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **size**:the video's width and height
- **mode**: selects the playback mode, see the mode attribute below

Syntax

```
(video!id <src>; <pos>; <width>; <height>; <mode>)
```

E.g.

```
(video!myvideo video; r5c2; 100; 100; stop)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Video attributes

Name	Value	Description
mode	stop	Default value. The video is paused when the page is shown
	auto	The video is automatically played when the page is shown
	loop	The video is cyclically played when the page is shown
play	true	Play the video
pause	true	Pause the video

Weather

For a description and a complete list of UI Objects, see UI Objects.

Shows a weather panel with current conditions and forecast. Weather information are retrieved using the WXONLINE I/O Server. The id should be equal to the I/O Server's name. For further details refer to the WXONLINE I/O Server Application Note.

Parameters

- **id**: the object's ID, used by UISets. Required
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(weather!<id> <position>)
```

E.g.

```
(weather!wx x10y20)
```

UserList

For a description and a complete list of UI Objects, see UI Objects.

A panel with a list of items that work like user buttons, used to execute customized Java code or generate events for the EVENTS interpreter.

Pressing an item of the list, HSYCO calls the method:

> userCommand(String name, String param)

in the user.class class. HSYCO also executes all actions associated to the event:

> USER name=param

defined in EVENTS.

If the repeat parameter is "true", a long press will repeat the request every second.

For a tutorial on how to use this object see Working with User objects.

Parameters

- **id**: the object's ID, used by UISets
- **name**: name passed to the Java callback method and event in EVENTS
- **parameters**: comma-separated list of the parameters passed to the Java callback method and event in EVENTS
- **labels**: comma-separated list of labels corresponding to the parameters. If missing, parameters will be shown instead
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **size**: the panel's width and height
- **css**: optional. Defines the CSS style for the input field
- **repeat**: if "enabled", the command will be repeat every second

Syntax

```
(userlist!<id> <name>; <parameters>; <labels>; <position>; <width>; <height>; <css>; <repeat>)
```

E.g.

```
(userlist!myuserlist name; p1,p2,p3; user one,user two,user three; r1c1; 220; 224; font-size:16px; enabled)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

UserList attributes

Name	Value	Description
enabled	true	Object is enabled
	false	Object is disabled
focus	true	Focus the object. Only a single object should have this attribute at "true".
	false	Reset focus
error	true	Manually shows an error feedback on the panel.
	false	Hides the error visualization.
name	string	UserList name
parameters	comma-separated list of strings	Parameters
labels	comma-separated list of strings	UserList labels
width	number of pixels	Set the object's width
height	number of pixels	Set the object's height
panel	true	Show background panel
	false	Hide background panel

UserImage

For a description and a complete list of UI Objects, see UI Objects.

A clickable image similar to the user object used to execute customized Java code or generate events for the EVENTS interpreter. Pressing this button, HSYCO calls the method:

> userCommand(String name, String param)

in the user.class class. HSYCO also executes all actions associated to the event:

> USER name=param

defined in EVENTS. For a tutorial on how to use this object see Working with User Objects. A long press will repeat the request every second.

For an object with a list of user commands, see UserList.

Parameters

- **id**: the object's ID, used by UISets
- **image**: image filename
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **size**: the image's width and height
- **name**: name passed to the Java callback method and event in EVENTS
- **parameter**: parameter passed to the Java callback method and event in EVENTS
- **label**: the text label next to the image. It can contain HTML tags
- **css**: the label's css attributes
- **repeat**: if "enabled", the command will be repeat every second

Syntax

```
(userimage[!<id>] <image's filename>; <position>; <width>; <height>; <name>; <parameter>; <label>; <css>;
<repeat>)
```

E.g.

```
(userimage!user1 myimage.png; r8c7; 100; 100; name; param; Label; (font-style:italic; top:-15px))
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*

visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Button attributes

Name	Value	Description
color	r,g,b...	Set the button's color. Colors vary depending on the selected skin
enabled	true	Default. Enable the button
	false	Disable the button

Label attributes

Name	Value	Description
label	<string>	Label's text. Can contain HTML tags
fontfamily	<family name>	A prioritized list of font family names
fontsize	<size>	Font size in pixels. E.g. *15px*
fontcolor	<CSS color>	Font color
fontstyle	normal	Normal font style, default
	italic	Italic font style
	oblique	Oblique font style
fontweight	normal	Normal characters, default
	bold	Thick characters
	bolder	Thicker characters
	lighter	Lighter characters
	100 ... 900	Defines from thin to thick characters. 400 is the same as normal, and 700 is the same as bold

UserImage attributes

Name	Value	Description
name	<string>	Set the user object's name value
parameter	<string>	Set the user object's parameter value
	coordinates	Set the object to send the X,Y coordinates of the point where the user clicks (or drags). E.g. for x=107,y=52 the server will receive *107,52*
repeat	true	Set repeat mode to enabled
	false	Set repeat mode to disabled
interval	<number>	Set the time in milliseconds between repeats (default is 1000)

User

For a description and a complete list of UI Objects, see UI Objects.

A button used to execute customized Java code or generate events for the EVENTS interpreter. Pressing this button, HSYCO calls the method:

> userCommand(String name, String param)

in the user.class class. HSYCO also executes all actions associated to the event:

> USER name=param

defined in EVENTS. For a tutorial on how to use this object see Working with User Objects. A long press will repeat the request every second.

It has several colors (depending on the skin) and three different sizes:

- **user** (standard size)

- **usermini** (medium size)

- **usermicro** (small size)

For a custom image version, see UserImage.

For an object with a list of user commands, see UserList.

Parameters

- **id**: the object's ID, used by UISets. Required
- **name**: name passed to the Java callback method and event in EVENTS
- **parameter**: parameter passed to the Java callback method and event in EVENTS
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **color**: the color of the button. It can assume any of the skin's color values (E.g. for the "blue" skin, the possible values are: b - blue; g - green; gr - gray; r - red; y - yellow; glass)
- **label**: the text label on the button. It can contain HTML tags
- **repeat**: if "enabled", the command will be repeat every second

Syntax

```
(user[!<id>] <name>; <parameter>; <position>; <color>; <label>)

(usermini[!<id>] <name>; <parameter>; <position>; <color>; <label>)
```

```
(usermicro[!<id>] <name>; <parameter>; <position>; <color>; <label>)
```

E.g.

```
(user!user1 name; param; r3c1; gr; label; enabled)

(usermini!user1 name; param; r3c1; b; label;)

(usermicro!user1 name; param; x0y10; r; label;)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Button attributes

Name	Value	Description
color	r,g,b...	Set the button's color. Colors vary depending on the selected skin
enabled	true	Default. Enable the button
	false	Disable the button

Label attributes

Name	Value	Description
label	<string>	Label's text. Can contain HTML tags
fontfamily	<family name>	A prioritized list of font family names

fontsize	<size>	Font size in pixels. E.g. *15px*
fontcolor	<CSS color>	Font color
fontstyle	normal	Normal font style, default
	italic	Italic font style
	oblique	Oblique font style
fontweight	normal	Normal characters, default
	bold	Thick characters
	bolder	Thicker characters
	lighter	Lighter characters
	100 ... 900	Defines from thin to thick characters. 400 is the same as normal, and 700 is the same as bold

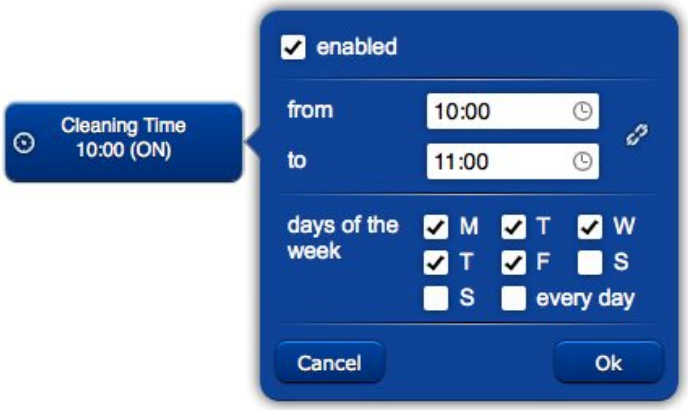
User attributes

Name	Value	Description
name	<string>	Set the user object's name value
parameter	<string>	Set the user object's parameter value
repeat	true	Set repeat mode to enabled
	false	Set repeat mode to disabled
interval	<number>	Set the time in milliseconds between repeats (default is 1000)

Timer

For a description and a complete list of UI Objects, see UI Objects.

A button that controls a timer. The button shows the current status of the timer, and touching it opens the timer set-up pop-up. The timer object generates TIMER events and userTimerEvent() Java callbacks on activation and reset.



Parameters

- **timer id**: the timer id, as defined in the Timers parameter in the hsyco.ini file
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **label**: the text label on the button. Can contain HTML tags

Syntax

```
(timer <timer id>; <position>; <label>)
```

E.g.

```
(timer cleaningtime; x30y30; Cleaning Time)
```

TimePanel

For a description and a complete list of UI Objects, see UI Objects.

Time panel that allows the user to input a time. Can be used with a submit button.

Requires an ID.

For a tutorial on using this object, see Working with Forms

Parameters

- **id**: the object's ID, used by UISets
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(timepanel!id <position>)
```

E.g.

```
(timepanel!mytime x3y20)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed

	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Field attributes

Name	Value	Description
enabled	true	Field is enabled
	false	Field is disabled
autosend	true	Automatically sends the value on changes. It's the default behaviour when there are no submit objects on the page or in a parent container (if the field is inside a container).
	false	Won't send the value on changes. It's the default behaviour when there's a submit object on the page or in a parent container.
focus	true	Focus the field. Only a single object should have this attribute at "true".
	false	Reset focus
rollback	true	Restores the previously saved value.
rollbackenabled	true	Automatically restores the previous value on an error (if the value isn't valid or the remote request resolved in an error).
	false	Won't restore the value on errors.
error	true	Manually shows an error feedback on the field (won't modify the current field value).
	false	Hides the error visualization.

Input attributes

Name	Value	Description
value	time in the "hhmm" format	Time value
notnull	true	Value can't be null
	false	Default. Value can be null
step	number	Amount of minutes time is divided into. E.g. A step value of "30" will only accept "00" or "30" as minutes. A step value of "60" will only accept full hours.

Time

For a description and a complete list of UI Objects, see UI Objects.

Time field that allows the user to input a time. Clicking on the field will open a popup with a time panel. Can be used with a submit button.

Requires an ID.

For a tutorial on using this object, see Working with Forms

Parameters

- **id**: the object's ID, used by UISets
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **css**: optional. Defines the CSS style for the date field

Syntax

```
(time!id <position>; [<css>])
```

E.g.

```
(time!mytime x3y20; width:55px)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object

blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Field attributes

Name	Value	Description
enabled	true	Field is enabled
	false	Field is disabled
autosend	true	Automatically sends the value on changes. It's the default behaviour when there are no submit objects on the page or in a parent container (if the field is inside a container).
	false	Won't send the value on changes. It's the default behaviour when there's a submit object on the page or in a parent container.
focus	true	Focus the field. Only a single object should have this attribute at "true".
	false	Reset focus
rollback	true	Restores the previously saved value.
rollbackenabled	true	Automatically restores the previous value on an error (if the value isn't valid or the remote request resolved in an error).
	false	Won't restore the value on errors.
error	true	Manually shows an error feedback on the field (won't modify the current field value).
	false	Hides the error visualization.

Input attributes

Name	Value	Description
value	time in the "hhmm" format	Time value
notnull	true	Value can't be null
	false	Default. Value can be null

step	number	Amount of minutes time is divided into. E.g. A step value of "30" will only accept "00" or "30" as minutes. A step value of "60" will only accept full hours.

TempMini

For a description and a complete list of UI Objects, see UI Objects.

This object displays a zone panel for temperature control. It has different graphics and features based on the temperature control system being used. See the Application Notes for additional information.

Parameters

- **server**: the I/O Server name
- **address**: the zone address, see the Application Notes
- **label**: the text label on the button. Can contain HTML tags
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(tempmini <server>; <address>; <label>; <position>)
```

E.g.

```
(tempmini bmne500; temp.1; OPENSPACE; r1c1)
```

Temp

For a description and a complete list of UI Objects, see UI Objects.

This object displays a zone panel for temperature control. It has different graphics and features based on the temperature control system being used. See the Application Notes for additional information.

Parameters

- **server**: the I/O Server name
- **address**: the zone address, see the Application Notes
- **label**: the text label on the button. Can contain HTML tags
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(temp <server>; <address>; <label>; <position>)
```

E.g.

```
(temp bmne500; temp.1; OPENSPACE; x3y161)
```

SubmitImage

For a description and a complete list of UI Objects, see UI Objects.

A clickable image similar to the submit object, used to submit the contents of a set of fields.

For a tutorial on using this object, see Working with Forms.

Parameters

- **id**: the object's ID, used by UISets
- **image**: image filename
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **size**:the image's width and height
- **label**:the image's label
- **css**:the label's css attributes

> Image files are searched inside the ***www/<project's name>/img*** directory first, then inside the ***www/img*** directory.
> If the filename is set with a UISet, it's possible to load an image from the web by entering the URL, or a path relative to ***www/<skin's name>/pic*** by specifying ***/pic/*** followed by the filename

Syntax

```
(submitimage<!id> <image's filename>; <position>; <width>; <height>; <label>; <css>)
```

E.g.

```
(submitimage myimage.png; r8c7; 100; 100; Submit; (font-style:italic; top:-15px))
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed

	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

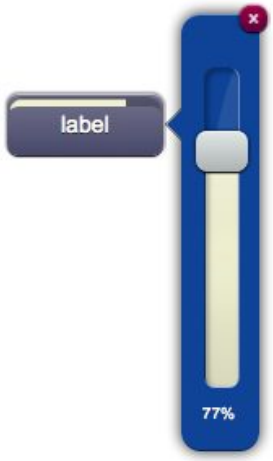
Button attributes

Name	Value	Description
color	r,g,b...	Set the button's color. Colors vary depending on the selected skin
enabled	true	Default. Enable the button
	false	Disable the button

SliderButton

For a description and a complete list of UI Objects, see UI Objects.

A button that opens a popup with a slider object.



Parameters

- **name**: name passed to the Java callback method and event in EVENTS
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **label**: the text label on the button. Can contain HTML tags

Syntax

```
(sliderbutton!<id> <address>; <position>; <label>)
```

E.g.

```
(sliderbutton!myslider slider1; x10y20; Label)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object

	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Slider attributes

Name	Value	Description
value	0-100	Set the slider's value, from 0 to 100
	0%-100%	Set the slider's value, from 0% to 100%
	off	Set the slider's value to 100 (on) or 0 (off)
	0-n/n	Set the slider's value, specifying a base. E.g. 64/256 sets the slider to 25%

Slider

For a description and a complete list of UI Objects, see UI Objects.

A slider object that sends requests to the server.

Using this object you can execute customized Java code or generate events for the EVENTS interpreter, like with the user object, passing the object's name and the selected value.

Three versions are available:

- *slider*: vertical slider with a label and an on/off button

- *sliderv*: minimal vertical slider

- *sliderh*: minimal horizontal slider

Parameters

- **address**: address passed to the Java callback method and event in EVENTS
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **label**: the text label on the button. Can contain HTML tags

Syntax

```
(slider!<id> <address>; <position>; <label>)

(sliderv!<id> <address>; <position>)

(sliderh!<id> <address>; <position>)
```

E.g.

```
(slider!myslider slider1; x10y20; Label)

(sliderv!myslider slider1; x10y20)

(sliderh!myslider slider1; x10y20)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Slider attributes

Name	Value	Description
value	0-100	Set the slider's value, from 0 to 100
	0%-100%	Set the slider's value, from 0% to 100%
	off	Set the slider's value to 100 (on) or 0 (off)
	0-n/n	Set the slider's value, specifying a base. E.g. 64/256 sets the slider to 25%
bar	true	Show the value bar
	false	Hide the value bar

barcolor	CSS color	Set the value bar's color
inverse	true	Inverts the slider's bar verse: on a slider object the cursor is at the top when the value is 0, at the bottom when the value is 100
	false	Default behaviour
enabled	true	Field is enabled
	false	Field is disabled

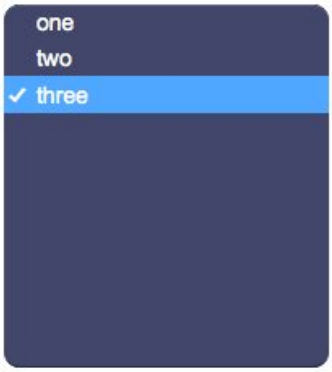
> A **CSS color** can be specified as
> - hex value: *#<rr><gg><bb>* E.g. *#FF1010*
> - color name: red, blue, white...
> - as rgb: *rgb(<r>,<g>,<b>)* E.g. *rgb(255,30,30)*
> - as rgba: *rgba(<r>,<g>,<b>,<alpha>)* E.g. *rgba(255,30,30,0.5)*

SelectPanel

For a description and a complete list of UI Objects, see UI Objects.

Select panel that allows the user to select a value from a list. Can be used with a submit button.

For a tutorial on using this object, see Working with Forms



Parameters

- **id**: the object's ID, used by UISets
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **size**: the panel's width and height
- **values**: comma-separated list of values. When selecting an item from the list, the corresponding value from this list is sent
- **labels**: comma-separated list of labels corresponding to the values. If missing, values will be shown instead
- **css**: optional. Defines the CSS style for the input field

Syntax

```
(selectpanel!<id> <position>; <width>; <height>; <values>; <labels>; <css>)
```

E.g.

```
(selectpanel!myselect r1c1; 220; 224; 1,2,3; one,two,three)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*

visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Field attributes

Name	Value	Description
enabled	true	Field is enabled
	false	Field is disabled
autosend	true	Automatically sends the value on changes. It's the default behaviour when there are no submit objects on the page or in a parent container (if the field is inside a container).
	false	Won't send the value on changes. It's the default behaviour when there's a submit object on the page or in a parent container.
focus	true	Focus the field. Only a single object should have this attribute at "true".
	false	Reset focus
rollback	true	Restores the previously saved value.
rollbackenabled	true	Automatically restores the previous value on an error (if the value isn't valid or the remote request resolved in an error).
	false	Won't restore the value on errors.
error	true	Manually shows an error feedback on the field (won't modify the current field value).
	false	Hides the error visualization.

SelectPanel attributes

Name	Value	Description
value	string	Select value
values	comma-separated list of strings	Select values
labels	comma-separated list of strings	Select labels

multi	true	Can select multiple values. The value sent to the server will be a comma-separated list of the selected values
	false	Default behaviour: only a single value can be selected
notnull	true	Value can't be null
	false	Default. Value can be null
width	number of pixels	Set the panel's width
height	number of pixels	Set the panel's height
panel	true	Show background panel
	false	Hide background panel

Select

For a description and a complete list of UI Objects, see UI Objects.

Select field that allows the user to select a value from a list. Clicking on the field will open a popup with a select panel. Can be used with a submit button.

For a tutorial on using this object, see Working with Forms

Parameters

- **id**: the object's ID, used by UISets
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **values**: comma-separated list of values. When selecting an item from the list, the corresponding value from this list is sent
- **labels**: comma-separated list of labels corresponding to the values. If missing, values will be shown instead
- **css**: optional. Defines the CSS style for the input field

Syntax

```
(select!<id> <position>; <values>; <labels>; <css>)
```

E.g.

```
(select!myselect r1c1; 1,2,3; one,two,three;)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed

	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Field attributes

Name	Value	Description
enabled	true	Field is enabled
	false	Field is disabled
autosend	true	Automatically sends the value on changes. It's the default behaviour when there are no submit objects on the page or in a parent container (if the field is inside a container).
	false	Won't send the value on changes. It's the default behaviour when there's a submit object on the page or in a parent container.
focus	true	Focus the field. Only a single object should have this attribute at "true".
	false	Reset focus
rollback	true	Restores the previously saved value.
rollbackenabled	true	Automatically restores the previous value on an error (if the value isn't valid or the remote request resolved in an error).
	false	Won't restore the value on errors.
error	true	Manually shows an error feedback on the field (won't modify the current field value).
	false	Hides the error visualization.

Input attributes

Name	Value	Description
value	string	Select value
values	comma-separated list of strings	Select values
labels	comma-separated list of strings	Select labels
multi	true	Can select multiple values. The value sent to the server will be a comma-separated list of the selected values

	false	Default behaviour: only a single value can be selected
notnull	true	Value can't be null
	false	Default. Value can be null

Selector

For a description and a complete list of UI Objects, see UI Objects.

The selector allows you to create a graphical behavior that is similar to having multiple panels with tab navigation, only more flexible than that. The graphical appearance of a selector is the same as an Image or ImageLink object, an image and a text label. You should create two or more overlapped containers in a page, each with a unique id. Then create a selector object for each container, with all selectors having the same group name. When a selector object is touched, the corresponding container's visibility will be set to visible, while all other containers associated to selectors with the same group name will become invisible.

Parameters

- **image**: image filename
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **size**: the image's width and height
- **container ID**: the container's id associated to this selector
- **group**: optional group name, identifying a set of selectors. All selectors with no group name defined are part of the same group
- **label**: optional text label next to the button. Can contain HTML tags
- **css**: optional CSS style.

> Image files are searched inside the ***www/<project's name>/img*** directory first, then inside the ***www/img*** directory.
> If the filename is set with a UISet, it's possible to load an image from the web by entering the URL, or a path relative to ***www/<skin's name>/pic*** by specifying ***/pic/*** followed by the filename

Syntax

```
(selector <image>; <position>; <width>; <height>; <container ID>; <group>; <label>; <css>)
```

E.g.

```
(selector apple.png; r3c5; 100; 100; mycontainer;; One)
```

Scheduler

For a description and a complete list of UI Objects, see UI Objects.

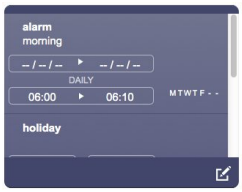
This object provides the user interface to the HSYCO's scheduler system. The scheduler is a powerful function that allows you to define a set of calendar entries to schedule actions at the beginning and end of each time interval.

A normal scheduler rule is defined with a start date/time and an end date/time and is valid within the defined period, even across several days. A daily rule is instead defined to be valid between a start time and end time, on a single day or spanning several days. In addition to the validity dates, you can set a rule to be active only on specific week days. When the current time matches the date/time and week days, the default behavior is that the scheduler generates a TIMER <schedule name> = ON event and the corresponding userTimerEvent() Java callback — just like the (timer) object — and a TIMER <schedule name> = OFF event at the end of the period.

You can also define an interval rule, setting the on and off minutes. An interval rule will go on and off repeatedly during the date/time validity period. Each rule has a group. This group is used to generate the events, and is not unique, as you can define several rules under the same group. In this case the scheduler will evaluate the rules having the same group in the order they appear in the scheduler display, from top to bottom. When a rule matches the date/time validity period, the scheduler stops checking any following rules with the same group. So, rules on top have priority within the rules with the same group. Rules set to work in off mode are used as "blocking rules", because when they are valid they will not become active, and at the same time will prevent the scheduler from processing the following rules that could otherwise match.

A scheduler has two graphical modes: compact and wide, depending on the object's width. A width of 750 pixels or more will toggle the wide mode that will display both the list of rules and the rule's details.

Here's a scheduler in compact mode, showing the list and the details:



Here's a scheduler in wide mode:

Parameters

- **id**: the object's ID, used by UISets
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **size**: panel's width, in pixels
- **rule list mode**: set to “edit” to enable the creation or deletion of individual schedules. Set to “noedit” to only allow editing of already defined schedules
- **groups**: optional, comma separated list of schedule group names that are shown in this object. If the list is not defined or set to "*", the object shows all groups

Syntax

```
(scheduler <position>; <width>; <height>; <mode>; <names>)
```

E.g.

```
(scheduler!myscheduler x3y256; 750; 350; edit;)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object

blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

RadioButton

For a description and a complete list of UI Objects, see UI Objects.

Radio button that allows the user to select a value.

Multiple radio buttons are grouped using the same ID. Can be used with a submit button. Requires an ID.

For a tutorial on using this object, see Working with Forms

label

Parameters

- **id**: the object's ID, used by UISets
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **label**: the text label next to the button. Can contain HTML tags
- **value**: value to be sent if selected
- **css**: optional. Defines the CSS style for the label

Syntax

```
(radiobutton!id <position>; <label>; <value>; [<css>])
```

E.g.

```
(radiobutton!radiogroup1 r1c1; label; one)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking

opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Field attributes

Name	Value	Description
enabled	true	Field is enabled
	false	Field is disabled
autosend	true	Automatically sends the value on changes. It's the default behaviour when there are no submit objects on the page or in a parent container (if the field is inside a container).
	false	Won't send the value on changes. It's the default behaviour when there's a submit object on the page or in a parent container.
focus	true	Focus the field. Only a single object should have this attribute at "true".
	false	Reset focus
rollback	true	Restores the previously saved value.
rollbackenabled	true	Automatically restores the previous value on an error (if the value isn't valid or the remote request resolved in an error).
	false	Won't restore the value on errors.
error	true	Manually shows an error feedback on the field (won't modify the current field value).
	false	Hides the error visualization.

Radiobutton attributes

Name	Value	Description
value	string	Radiobutton value

Panel

For a description and a complete list of UI Objects, see UI Objects.

Displays a background panel with a specific color, based on the skin.

Parameters

- **id**: the object's ID, used by UISets
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **size**: the panel's width and height
- **color**: the color of the button. It can assume any of the skin's color values (E.g. for the "blue" skin, the possible values are: b - blue; g - green; gr - gray; r - red; y - yellow; glass)

Syntax

```
(panel[!<id>] <pos>; <width>; <height>; <color>)
```

E.g.

```
(panel!mypanel r1c1; 200; 200; gr)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object

blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Marquee

For a description and a complete list of UI Objects, see UI Objects.

Shows a generic text just like the text object, but with a scrolling effect.

Parameters

- **id**: the object's ID, used by UISets. Required
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **size**: the text box's width and height
- **direction**: the scrolling direction:
  - left: from right to left
  - right: from left to right
  - up: scrolls up
  - down: scrolls down
- **speed**: scroll speed; minimum speed is 1; larger numbers increase the speed
- **text**: the text. Can contain HTML tags
- **css**: optional, and defines the CSS style

Syntax

```
(marquee[!<id>] <pos>; <width>; <height>; <direction>; <speed>; <text>; <css>)
```

E.g.

```
(marquee!mymarquee r1c1; 200; 50; right; 1; scrolling text; color:blue)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking

opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Marquee attributes

Name	Value	Description
width	number	Text box's width in pixels
height	number	Text box's height in pixels
paused	true	Scrolling paused
	false	Default: scrolling
text	string	Set the text

Link

For a description and a complete list of UI Objects, see UI Objects.

A button used to navigate to another page, a pop-up, or an external Web page. It has several colors (depending on the skin) and four different sizes:

- **link** (standard size)

- **linkmini** (medium size)

- **linkmicro** (small size)

- **dlink** (big size)

For a custom image version, see ImageLink.

Parameters

- **id**: the object's ID, used by UISets. Required
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **color**: the color of the button. It can assume any of the skin's color values (E.g. for the "blue" skin, the possible values are: b - blue; g - green; gr - gray; r - red; y - yellow; glass)
- **page**: the name of the destination page associated to the button. It is possible to specify an absolute or relative URL in the **page** parameter. The tag will work as an <A HREF> tag, loading the new page. If the loaded page is an HSYCO menu, the **back** link will show up in the menu page, to reload the referrer page
- **label**: the text label on the button. It can contain HTML tags

Syntax

```
(link[!<id>] <pos>; <color>; <page>; <label>)

(linkmini[!<id>] <pos>; <color>; <page>; <label>)

(linkmicro[!<id>] <pos>; <color>; <page>; <label>)

(dlink[!<id>] <pos>; <color>; <page>; <label>)
```

E.g.

```
(link!link1 x10y20; r; page1; Page 1)

(linkmini r0c1; glass; http://www.hsyco.com; hsyco)

(linkmicro x0y100; gr; /hsycohsyco/office; HSYCO Office)

(dlink!dlink1 x10y10; b; page1; Larger text<br>with <b>HTML</b> tags)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Label attributes

Name	Value	Description
label	<string>	Label's text. Can contain HTML tags
fontfamily	<family name>	A prioritized list of font family names
fontsize	<size>	Font size in pixels. E.g. *15px*
fontcolor	<CSS color>	Font color
fontstyle	normal	Normal font style, default
	italic	Italic font style
	oblique	Oblique font style
fontweight	normal	Normal characters, default
	bold	Thick characters
	bolder	Thicker characters

	lighter	Lighter characters
	100 ... 900	Defines from thin to thick characters. 400 is the same as normal, and 700 is the same as bold

Button attributes

Name	Value	Description
color	r,g,b...	Set the button's color. Colors vary depending on the selected skin
enabled	true	Default. Enable the button
	false	Disable the button

KeypadPanel

For a description and a complete list of UI Objects, see UI Objects.

Keypad panel that allows the user to input a numeric value.

Can be used with a submit button.

The keypad performs range and number of digits validation of the input data.

Requires an ID.

For a tutorial on using this object, see Working with Forms

Parameters

- **id**: the object's ID, used by UISets
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **min**: the smallest accepted value
- **max**: the largest accepted value
- **digits**: the number of required digits; leave empty to accept a variable number of digits
- **decimals**: the maximum number of decimal digits; leave blank for integer numbers
- **type**: set to "input" for ordinary data; set to "password" to hide the digits
- **label**: the label shown at the top of the keypad panel
- **css**: optional. Defines the CSS style for the keypad field

Syntax

```
(keypadpanel!id <pos>; <min>; <max>; <digits>; <decimals>; <type>; [<label>]; [<css>])
```

E.g.

```
(keypadpanel!mykeypad x3y20; 0; 100; 2;; input; Label;)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Field attributes

Name	Value	Description
enabled	true	Field is enabled
	false	Field is disabled
autosend	true	Automatically sends the value on changes. It's the default behaviour when there are no submit objects on the page or in a parent container (if the field is inside a container).
	false	Won't send the value on changes. It's the default behaviour when there's a submit object on the page or in a parent container.
focus	true	Focus the field. Only a single object should have this attribute at "true".
	false	Reset focus
rollback	true	Restores the previously saved value.
rollbackenabled	true	Automatically restores the previous value on an error (if the value isn't valid or the remote request resolved in an error).
	false	Won't restore the value on errors.
error	true	Manually shows an error feedback on the field (won't modify the current field value).
	false	Hides the error visualization.

KeypadPanel attributes

Name	Value	Description
value	number	Keypad value
min	number	The smallest accepted value
max	number	The largest accepted value
digits	number	The number of required digits; leave empty to accept a variable number of digits
decimals	number	The maximum number of decimal digits; leave blank for integer numbers
ktype	input	Show the digits
	password	Hide the digit
label	string	The label shown at the top of the keypad panel

Keypad

For a description and a complete list of UI Objects, see UI Objects.

Keypad field that allows the user to input a numeric value. Clicking on the field will open a popup with a keypad panel. Can be used with a submit button.

The keypad performs range and number of digits validation of the input data.

Requires an ID.

For a tutorial on using this object, see Working with Forms

Parameters

- **id**: the object's ID, used by UISets
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **min**: the smallest accepted value
- **max**: the largest accepted value
- **digits**: the number of required digits; leave empty to accept a variable number of digits
- **decimals**: the maximum number of decimal digits; leave blank for integer numbers
- **type**: set to "input" for ordinary data; set to "password" to hide the digits
- **label**: the label shown at the top of the keypad panel
- **css**: optional. Defines the CSS style for the keypad field

Syntax

```
(keypad!id <pos>; <min>; <max>; <digits>; <decimals>; <type>; [<label>]; [<css>])
```

E.g.

```
(keypad!mykeypad x3y20; 0; 100; 2;; input; Label;)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Field attributes

Name	Value	Description
enabled	true	Field is enabled
	false	Field is disabled
autosend	true	Automatically sends the value on changes. It's the default behaviour when there are no submit objects on the page or in a parent container (if the field is inside a container).
	false	Won't send the value on changes. It's the default behaviour when there's a submit object on the page or in a parent container.
focus	true	Focus the field. Only a single object should have this attribute at "true".
	false	Reset focus
rollback	true	Restores the previously saved value.
rollbackenabled	true	Automatically restores the previous value on an error (if the value isn't valid or the remote request resolved in an error).
	false	Won't restore the value on errors.

error	true	Manually shows an error feedback on the field (won't modify the current field value).
	false	Hides the error visualization.

Keypad attributes

Name	Value	Description
value	number	Keypad value
min	number	The smallest accepted value
max	number	The largest accepted value
digits	number	The number of required digits; leave empty to accept a variable number of digits
decimals	number	The maximum number of decimal digits; leave blank for integer numbers
ktype	input	Show the digits
	password	Hide the digit
label	string	The label shown at the top of the keypad panel

IR

For a description and a complete list of UI Objects, see UI Objects.

A button used to send infrared (IR) commands through the IRTrans IR emitter/receiver network devices. It has three button versions with several colors (depending on the skin) and an image version with a custom image:

- **ir** (standard size)

- **irmini** (medium size)

- **irmicro** (small size)

- **irimage** (custom image)

Parameters

- **irid**: IRTrans id, as defined in the list of the IRTrans parameter in hsyco.ini
- **com**: the sequence of IR commands to send. Every command is composed by a name that identifies the commands database, followed by a comma and the name of the command in that database. Different commands in sequence can be separated by blank spaces. It is possible to insert the special @N command, which introduces a wait time of N milliseconds before sending the next command. For example:

  `denon-1036,on @4000 denon-1036,tv`
  sends the “on” command of the “denon-1036” remote control, then waits 4 seconds, and finally sends the “tv” command of the “denon-1036”

- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **label**: the text label on the button. It can contain HTML tags
- **repeat**: if true, the command will be repeat every second

For the button versions:

- **color**: the color of the button. It can assume any of the skin's color values (E.g. for the "blue" skin, the possible values are: b - blue; g - green; gr - gray; r - red; y - yellow; glass)

For the image version:

- **image filename**: the image filename

> Image files are searched inside the ***www/<project's name>/img*** directory first, then inside the ***www/img*** directory.
> If the filename is set with a UISet, it's possible to load an image from the web by entering the URL, or a path relative to ***www/<skin's name>/pic*** by specifying ***/pic/*** followed by the filename

Syntax

```
(ir <irid>; <com>; <pos>; <color>; <label>; <repeat>)

(irmini <irid>; <com>; <pos>; <color>; <label>; <repeat>)

(irmicro <irid>; <com>; <pos>; <color>; <label>; <repeat>)

(irimage <image filename>; <pos>; <width>; <height>; <irid>; <com>; <label>; <css>; <repeat>)
```

E.g.

```
(ir 1; denon-1036,on; r1c1; gr; label; enabled)

(irmicro 1; denon-1036,on; x193y0; gr; label)

(irmini 1; denon-1036,on; x113y0; gr; label)

(irimage apple.png; x246y0; 81; 77; 1; denon-1036,on)
```

ImageLink

For a description and a complete list of UI Objects, see UI Objects.

A clickable image from the www/img directory. You can add an optional label text next to the image, and define several optional CSS attributes to control the text's format and position. The "top" CSS attribute is used to move the text from its default position. If top is not defined, the text will be located just below the image. If defined, you can specify a positive or negative offset, in pixels. The zero offset position is at the top corner of the image; a positive offset moves the text down, a negative offset moves the text up.

Parameters

- **id**: the object's ID, used by UISets
- **image**: image filename
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **size**: the image's width and height
- **page**: the name of the destination page when you click the image. It is

possible to specify an absolute or relative URL in the *page* parameter. The tag will work as an <A HREF> tag, loading the new page. If the loaded page is an HSYCO menu, the *back* link will show up in the menu page, to reload the referrer page

- **label**: the image's label
- **css**: the label's css attributes

> Image files are searched inside the ***www/<project's name>/img*** directory first, then inside the ***www/img*** directory.
> If the filename is set with a UISet, it's possible to load an image from the web by entering the URL, or a path relative to ***www/<skin's name>/pic*** by specifying ***/pic/*** followed by the filename

Syntax

```
(imagelink<!id> <image's filename>; <position>; <width>; <height>; <page>; <label>; <css>)
```

E.g.

```
(imagelink myimage.png; r8c7; 100; 100; page1; Label; (font-style:italic; top:-15px))
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*

visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Label attributes

Name	Value	Description
label	<string>	Label's text. Can contain HTML tags
fontfamily	<family name>	A prioritized list of font family names
fontsize	<size>	Font size in pixels. E.g. *15px*
fontcolor	<CSS color>	Font color
fontstyle	normal	Normal font style, default
	italic	Italic font style
	oblique	Oblique font style
fontweight	normal	Normal characters, default
	bold	Thick characters
	bolder	Thicker characters
	lighter	Lighter characters
	100 ... 900	Defines from thin to thick characters. 400 is the same as normal, and 700 is the same as bold

Image attributes

img	<full url (http://...)> | <path relative to /img/>	Changes the image source
draw	<draw instructions>	Draws the image. See the Draw API e.g.: line(0,100,20,50,blue);line(50,50,70,70,'255,10,150');

Image

For a description and a complete list of UI Objects, see UI Objects.

Shows an image file from the www/img directory. You can add an optional label text next to the image, and define several optional CSS attributes to control the text's format and position. The "top" CSS attribute is used to move the text from its default position. If top is not defined, the text will be located just below the image. If defined, you can specify a positive or negative offset, in pixels. The zero offset position is at the top corner of the image; a positive offset moves the text down, a negative offset moves the text up.

Parameters

- **id**: the object's ID, used by UISets
- **image**: image filename
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **size**:the image's width and height
- **label**:the image's label
- **css**:the label's css attributes

> Image files are searched inside the ***www/<project's name>/img*** directory first, then inside the ***www/img*** directory.
> If the filename is set with a UISet, it's possible to load an image from the web by entering the URL, or a path relative to ***www/<skin's name>/pic*** by specifying ***/pic/*** followed by the filename

Syntax

```
(image<!id> <image's filename>; <pos>; <width>; <height>; <label>; <css>)
```

E.g.

```
(image myimage.png; r8c7; 100; 100; Label; (font-style:italic; top:-15px))
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object

blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Label attributes

Name	Value	Description
label	<string>	Label's text. Can contain HTML tags
fontfamily	<family name>	A prioritized list of font family names
fontsize	<size>	Font size in pixels. E.g. *15px*
fontcolor	<CSS color>	Font color
fontstyle	normal	Normal font style, default
	italic	Italic font style
	oblique	Oblique font style
fontweight	normal	Normal characters, default
	bold	Thick characters
	bolder	Thicker characters
	lighter	Lighter characters
	100 ... 900	Defines from thin to thick characters. 400 is the same as normal, and 700 is the same as bold

Image attributes

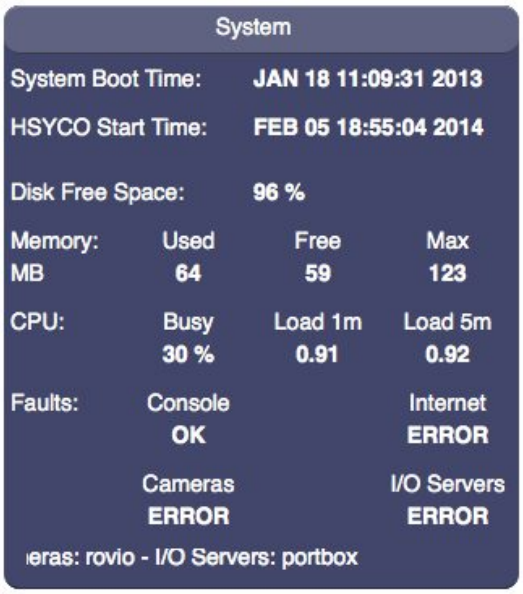
img	<full url (http://...)> | <path relative to /img/>	Changes the image source
draw	<draw instructions>	Draws the image. See the Draw API e.g.: line(0,100,20,50,blue);line(50,50,70,70,'255,10,150');

Hsycomonitor

For a description and a complete list of UI Objects, see UI Objects.

A panel that monitors the system. These parameters are displayed:

- system boot time
- HSYCO start time
- disk free space
- memory in MBs: used, free, max
- CPU usage in %: busy, load 1m, load 5m
- faults: console, internet, cameras, I/O servers
- error messages



Parameters

- **server ID**: the ID of the server as specified in hsyco.ini
- **label**: the text label on the panel. Can contain HTML tags
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(hsycomonitor <server ID>; <label>; <position>)
```

E.g.

```
(hsycomonitor system; System; x10y20)
```

VBar

For a description and a complete list of UI Objects, see UI Objects.

A vertical bar.

Parameters

- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **height**: the height of the bar, in pixels

Syntax

```
(hbar <position>; <height>)
```

UI Attributes

UI Object Attributes (Common)

Hbar

For a description and a complete list of UI Objects, see UI Objects.

A horizontal bar.

Parameters

- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **width**: the width of the bar, in pixels

Syntax

```
(hbar <position>; <width>)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

DMXRGB

For a description and a complete list of UI Objects, see UI Objects.

Controls a group of three adjacent DMX channels for the primary colors: red, green and blue.

Parameters

- **address**: using just one DMX gateway, address should be a number between 1 and 512. If there is more than one gateway, 1000 must be added to the DMX bus number for the second gateway, 2000 for the third one and so on. For example, 2100 stands for the address 100 on the DMX bus controlled by the third gateway
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **label**: the text label on the panel. Can contain HTML tags

Syntax

```
(dmxrgb <address>; <position>; <label>)
```

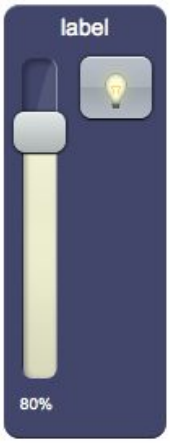
E.g.

```
(dmxrgb 1; x10y20; Open Space)
```

DMX

For a description and a complete list of UI Objects, see UI Objects.

Controls of a single DMX channel.



Parameters

- **address**: using just one DMX gateway, address should be a number between 1 and 512. If there is more than one gateway, 1000 must be added to the DMX bus number for the second gateway, 2000 for the third one and so on. For example, 2100 stands for the address 100 on the DMX bus controlled by the third gateway
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **label**: the text label on the panel. Can contain HTML tags

Syntax

```
(dmx <address>; <position>; <label>)
```

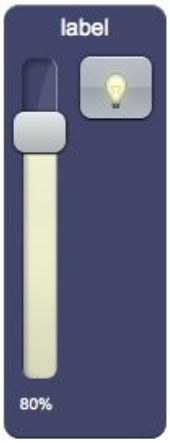
E.g.

```
(dmx 1; x10y20; Light)
```

Dimmer

For a description and a complete list of UI Objects, see UI Objects.

A dimmer light actuator.



Parameters

- **address**: device data point name
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **label**: the text label on the panel. Can contain HTML tags

Syntax

```
(dimmer <address>; <position>; <label>)
```

E.g.

```
(dimmer light.1; x10y20; Light)
```

DatePanel

For a description and a complete list of UI Objects, see UI Objects.

Date panel that allows the user to input a date.

Can be used with a submit button.

Requires an ID.

For a tutorial on using this object, see Working with Forms

Parameters

- **id**: the object's ID, used by UISets
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(datepanel!id <position>)
```

E.g.

```
(datepanel!mydate r1c1)
```

UI Attributes

Common attributes

Name	Value	Description

pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Field attributes

Name	Value	Description
enabled	true	Field is enabled
	false	Field is disabled
autosend	true	Automatically sends the value on changes. It's the default behaviour when there are no submit objects on the page or in a parent container (if the field is inside a container).
	false	Won't send the value on changes. It's the default behaviour when there's a submit object on the page or in a parent container.
focus	true	Focus the field. Only a single object should have this attribute at "true".
	false	Reset focus
rollback	true	Restores the previously saved value.
rollbackenabled	true	Automatically restores the previous value on an error (if the value isn't valid or the remote request resolved in an error).
	false	Won't restore the value on errors.
error	true	Manually shows an error feedback on the field (won't modify the current field value).
	false	Hides the error visualization.

Input attributes

Name	Value	Description

value	date in the "yyyymmdd" format	Date value
notnull	true	Value can't be null
	false	Default. Value can be null
fromdate	date in the "yyyymmdd" format	Minimum selectable date
todate	date in the "yyyymmdd" format	Maximum selectable date

Date

For a description and a complete list of UI Objects, see UI Objects.

Date field that allows the user to input a date. Clicking on the field will open a popup with a date panel. Can be used with a submit button.

Requires an ID.

For a tutorial on using this object, see Working with Forms

Parameters

- **id**: the object's ID, used by UISets
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **css**: optional. Defines the CSS style for the date field

Syntax

```
(date!id <position>; [<css>])
```

E.g.

```
(date!mydate r1c1; width:100px)
```

UI Attributes

Common attributes

Name	Value	Description

pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Field attributes

Name	Value	Description
enabled	true	Field is enabled
	false	Field is disabled
autosend	true	Automatically sends the value on changes. It's the default behaviour when there are no submit objects on the page or in a parent container (if the field is inside a container).
	false	Won't send the value on changes. It's the default behaviour when there's a submit object on the page or in a parent container.
focus	true	Focus the field. Only a single object should have this attribute at "true".
	false	Reset focus
rollback	true	Restores the previously saved value.
rollbackenabled	true	Automatically restores the previous value on an error (if the value isn't valid or the remote request resolved in an error).
	false	Won't restore the value on errors.
error	true	Manually shows an error feedback on the field (won't modify the current field value).
	false	Hides the error visualization.

Date attributes

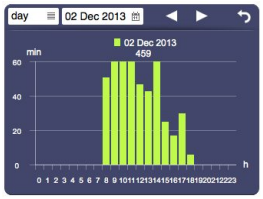
Name	Value	Description

value	date in the "yyyymmdd" format	Date value
notnull	true	Value can't be null
	false	Default. Value can be null
fromdate	date in the "yyyymmdd" format	Minimum selectable date
todate	date in the "yyyymmdd" format	Maximum selectable date

Datalogger

For a description and a complete list of UI Objects, see UI Objects.

Works in combination with the data logger function to present a collection of data using time-based statistical charts. This object automatically adapts the graphical layout and size of the charts based on the overall object's size. The default presentation can be customized using several optional attributes.



Parameters

- **id**: the object's ID, used by UISets. Required
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **size**: the chart's area's width and height
- **label**: the values axis label
- **attributes**: list of UI attribute ids and values defining the chart's aspect and data. Setting this parameter is the same as setting each UI attribute with a UISet

Syntax

```
(datalogger <id>; <pos>; <width>; <height>; <label>; <attributes>)
```

E.g.

```
(chart!chart1 x3y187; 312; 180; (type=line; values=0,10,24,12,7,35,42,1,23,4))
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*

	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Datalogger attributes

Name	Value	Description
type	bars	Draw bars
	gauge	Draw a gauge, for charts with a single value. Use with the "value" attribute
	points	Draws points
	line	Draws a line connecting the values
	spline	Draws a spline connecting the values
mode	live	Live mode
	browser	Browser mode
slot	<number>	Show a specific slot, from 0 to the number of slots-1
scale	year	Set the live scale to year
	month	Set the live scale to month
	day	Set the live scale to day
	hour	Set the live scale to hour
browserscale	year	Set the browser scale to year
	month	Set the browser scale to month
	day	Set the browser scale to day
	hour	Set the browser scale to hour
browserdate	date and time, in the "yyyymmddhh" format	Set the browser date and time. These formats will also be accepted: "yyyy", "yyyymm", "yyyymmdd"
controls	tabs	Show tabs on live mode for scale and slots

	toolbar	Shot a toolbar to control live and browser modes
	none	Don't show any controls
panel	true	Show background panel
	false	Hide background panel
multichart	true	Show multiple charts on live mode
	false	Show a single chart on live mode, based on the selected scale
presentvaluecolor	<CSS color>	A color that applies to all present values
	<CSS color>,<CSS color>,<CSS color>	Colors that apply respectively to minimum, average and maximum present values
pastvaluecolor	<CSS color>	A color that applies to all past values
	<CSS color>,<CSS color>,<CSS color>	Colors that apply respectively to minimum, average and maximum past values
valuelabelcolor	<CSS color>	Value label's color
valuelabeltype	inside	Value labels are shown inside (if "type" is bars, points or gauge) or over (if "type" is line or spline)
	outside	Value labels are shown above or below
	popup	Value labels are shown on mouseover, or on touch (if on a touch-enabled device)
notches	<value>	Number of notches on the x-axis (for vertical charts) or on the y-axis (for horizontal charts)
axiscolor	<CSS color>	Axis color
notchcolor	<CSS color>	Color of the axis notches
labelcolor	<CSS color>	Color of the axis and bars labels
thresholds	<comma-separated list of values>	List of values at which to display a line (threshold)
thresholdcolor	<CSS color>	Color of threshold lines
thresholdcolors	<comma-separated list of CSS colors>	List of colors of threshold lines, one value each line

> A **CSS color** can be specified as
> - hex value: *#<rr><gg><bb>* E.g. *#FF1010*
> - color name: red, blue, white...
> - as rgb: *rgb(<r>,<g>,<b>)* E.g. *rgb(255,30,30)*

- as rgba: *rgba(<r>,<g>,<b>,<alpha>)* E.g. *rgba(255,30,30,0.5)*

SMS (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The user interface for sending SMS messages:

File:UI Object sms.png

The SMS object is listed in the Project Editor's new object list only when at least one GSM I/O Server is defined.

For further details refer to the GSM I/O Server Application Note.

Parameters

- **server id**: the server ID
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(sms <server id>; <position>)
```

E.g.

```
(sms serverid; x10y20)
```

Guardall (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The user interface for the Guardall multi-area intrusion detection system:

The Guardall object is listed in the Project Editor's new object list only when at least one Guardall I/O Server is defined.

For further details refer to the Guardall I/O Server Application Note.

Parameters

- **server id**: the server ID
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(guardall <server id>; <position>)
```

E.g.

```
(guardall serverid; x10y20)
```

Elmo (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The user interface for the EL.MO. multi-area intrusion detection system:

The (elmo) object is listed in the Project Editor's new object list only when at least one EL.MO. I/O Server is defined

For further details refer to the Elmo I/O Server Application Note.

Parameters

- **server id**: the server ID
- **area**:
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(elmo <server id>; <area>; <position>)
```

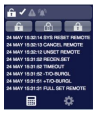
E.g.

```
(elmo elm; 1; x10y20)
```

Galaxy (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The user interfacce for the Galaxy multi-area intrusion detection system:



The Galaxy object is listed in the Project Editor's new object list only when at least one Galaxy I/O Server is defined.

For further details refer to the Galaxy I/O Server Application Note.

Parameters

- **server id**: the server ID
- **group number**:
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(galaxy <server id>; <group number>; <position>)
```

E.g.

```
(galaxy serverid; group number; x10y20)
```

Input

For a description and a complete list of UI Objects, see UI Objects.

Input field that allows the user to input a text. Can be used with a submit button.

Requires an ID.

For a tutorial on using this object, see Working with Forms

input text...

Parameters

- **id**: the object's ID, used by UISets
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **css**: optional. Defines the CSS style for the input field
- **type**: optional. Defines the input type, see inputtype UI attribute

Syntax

```
(input!id <position>; [<css>]; [<type>])
```

E.g.

```
(input!myinput r1c1; color:blue; password)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking

opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Field attributes

Name	Value	Description
enabled	true	Field is enabled
	false	Field is disabled
autosend	true	Automatically sends the value on changes. It's the default behaviour when there are no submit objects on the page or in a parent container (if the field is inside a container).
	false	Won't send the value on changes. It's the default behaviour when there's a submit object on the page or in a parent container.
focus	true	Focus the field. Only a single object should have this attribute at "true".
	false	Reset focus
rollback	true	Restores the previously saved value.
rollbackenabled	true	Automatically restores the previous value on an error (if the value isn't valid or the remote request resolved in an error).
	false	Won't restore the value on errors.
error	true	Manually shows an error feedback on the field (won't modify the current field value).
	false	Hides the error visualization.

Input attributes

Name	Value	Description
inputtype	input	Default behaviour
	multiline	for input fields with more than a line. Use the height attribute on the css parameter to set the size
	password	hides typed characters
value	<string>	Input value
maxlenght	<number>	Specify the maximum length that can be typed
eraseicon	true	Show an erase icon that can be clicked to quickly erase the input's content

	false	Default: no erase icon
style	<CSS style>	Set the field's CSS style
helptext	<string>	Text to show when the field is empty
changedelay	<number>	Sets the delay in milliseconds by which changes are detected. If the field is working in autosend mode, this value determines the delay between the last button press and the value being sent to the server
	off	If the field is working in autosend mode, values aren't sent on a delay, but only when the field loses the focus (also when the tab or enter keys are pressed)
validatekey	<regular expression>	A regular expression to be validated on a keypress. E.g. "[a-z]" will allow only lowercase letters. Won't work on Android devices.
validatevalue	<regular expression>	A regular expression to be validated before the value is sent to the server. If the expression doesn't match the value, the previous value will be restored and the input will flash in red

Container

For a description and a complete list of UI Objects, see UI Objects.

This object is used to group other objects together. All the contained objects' positions are relative to the container's top left corner. Containers can be nested.

Parameters

- **id**: the object's ID, used by UISets
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **visibility**: optional. Set to "hidden" to make the container invisible when the page is loaded

Syntax

```
(container[!<id>] <pos>; [<visibility>])
...
(endofcontainer)
```

E.g.

```
(container!mycontainer x0y0)
...
(endofcontainer)

(container!mycontainer x0y0; hidden)
...
(endofcontainer)
```

UI Attributes

UI Object Attributes (Common)

Checkbox

For a description and a complete list of UI Objects, see UI Objects.

Checkbox button that allows the user to set a value to true or false. Can be used with a submit button.

Requires an ID.

For a tutorial on using this object, see Working with Forms

☑ label

Parameters

- **id**: the object's ID, used by UISets
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **label**: the text label next to the button. Can contain HTML tags
- **css**: optional. Defines the CSS style for the label

Syntax

```
(checkbox!id <position>; <label>; [<css>])
```

E.g.

```
(checkbox!mycheckbox r1c1; label; color:black)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking

opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Field attributes

Name	Value	Description
enabled	true	Field is enabled
	false	Field is disabled
autosend	true	Automatically sends the value on changes. It's the default behaviour when there are no submit objects on the page or in a parent container (if the field is inside a container).
	false	Won't send the value on changes. It's the default behaviour when there's a submit object on the page or in a parent container.
focus	true	Focus the field. Only a single object should have this attribute at "true".
	false	Reset focus
rollback	true	Restores the previously saved value.
rollbackenabled	true	Automatically restores the previous value on an error (if the value isn't valid or the remote request resolved in an error).
	false	Won't restore the value on errors.
error	true	Manually shows an error feedback on the field (won't modify the current field value).
	false	Hides the error visualization.

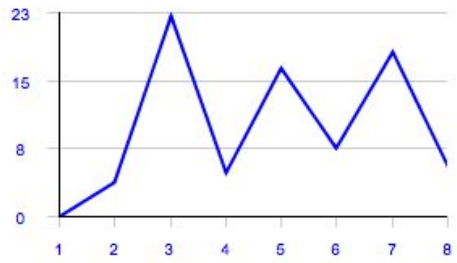
Checkbox attributes

Name	Value	Description
value	true	Checked
	false	Not checked

Chart

For a description and a complete list of UI Objects, see UI Objects.

Draws a chart. Its appearance and data are set with UI Attributes. More than one set of values can be specified.



Parameters

- **id**: the object's ID, used by UISets. Required
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **size**: the chart's area's width and height
- **attributes**: list of UI attribute ids and values defining the chart's aspect and data. Setting this parameter is the same as setting each UI attribute with a UISet

Syntax

```
(chart!<id> <address>; <position>; <width>; <height>; <attributes>)
```

E.g.

```
(chart!chart1 x3y187; 312; 180; (type=line; values=0,10,24,12,7,35,42,1,23,4))
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed

	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Chart attributes

Name	Value	Description
type	bars	Draw bars
	gauge	Draw a gauge, for charts with a single value. Use with the "value" attribute
	points	Draws points
	line	Draws a line connecting the values
	spline	Draws a spline connecting the values
orientation	vertical	Default value. Draws a vertical chart
	horizontal	Draws an horizontal chart
value	single value	Value to be drawn. To be used when "type" is gauge
value[n]		Same as "value", but applies to a specific set of values
values	comma-separated list of values	Values to be drawn. E.g. 2,3,1.2,4
	comma-separated list of arrays	Sets of values to be drawn. E.g. [2,3,1.2,4],[4,1,0,0.2]
values[n]		Same as "values", but applies to a specific set of values
valuerange	[min],[max]	Range of values displayed. E.g. -2,10
axislabels	auto	Default. Labels on the x axis (if vertical) or y axis (if horizontal) will go from 1 to the total number of values.
	none	No label is shown on the x axis (if vertical) or y axis (if horizontal)
	comma-separated list of values	Custom labels on the x axis (if vertical) or y axis (if horizontal)
valuelabeltype	inside	Value labels are shown inside (if "type" is bars, points or gauge) or over (if "type" is line or spline)
	outside	Value labels are shown above or below

	popup	Value labels are shown on mouseover, or on touch (if on a touch-enabled device)
valuelabelcolor	CSS color	Color of the value labels font
valuelabels	comma-separated list of values	Custom values to show in value labels
valuelabels[n]		Same as "valuelabels", but applies to a specific set of values
spacing	0-100	Percentage of spacing between bars or points
spacing[n]		Same as "spacing", but applies to a specific set of values
offset[n]	percentage	Percentage of value offset. Used to shift a specific set on the x axis if vertical, or on the y axis if horizontal. Used to show background sets. E.g. "offset[0]=-20; offset[1]=20"
drawaxis	true	Default. The axes are visible (also notches and labels)
	false	The axes are not visible (also notches and labels)
axisoffset	percentage	The axis offset (y if vertical, x if horizontal) expressed as percentage or as the value series index. E.g. "50%" or "3"
origin	value	Moves the origin to a value different from 0 (e.g. origin=3.2)
notches	value	Number of notches on the x-axis (for vertical charts) or on the y-axis (for horizontal charts)
pointsize	number of pixels	Point size (use for the points chart type)
bgcolor	CSS color	Background color
valuecolor	CSS color	A color that applies to all values
valuecolor[n]		Same as "valuecolor", but applies to a specific set of values
valuecolors	comma-separated list of CSS colors	List of colors, one for each data point in values
valuecolors[n]		Same as "valuecolors", but applies to a specific set of values
barborder	true	Show bars/points border
	false	hide bars/points border
axiscolor	CSS color	Axis color

notchcolor	CSS color	Color of the axis notches
labelcolor	CSS color	Color of the axis and bars labels
thresholds	Comma-separated list of values	List of values at which to display a line (threshold)
thresholdcolor	CSS color	Color of threshold lines
thresholdcolors	comma-separated list of CSS colors	List of colors of threshold lines, one value each line
stroke	number of pixel	Pixel size of the stroke that applies to line and spline chart's types
stroke[n]		Same as "colspan", but applies to a specific set of values

A **CSS color** can be specified as

- hex value: *#<rr><gg><bb>* E.g. *#FF1010*
- color name: red, blue, white...
- as rgb: *rgb(<r>,<g>,<b>)* E.g. *rgb(255,30,30)*
- as rgba: *rgba(<r>,<g>,<b>,<alpha>)* E.g. *rgba(255,30,30,0.5)*

CameraPanel

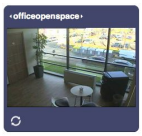
For a description and a complete list of UI Objects, see UI Objects.

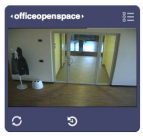
A real-time view of a camera or grid of cameras. Unlike the camera object, camerapanel has two operating modes: live and playback. It includes a panel that allows access to various functions:

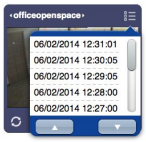
- motions' list: a list of all the motions
- live mode:
  - play last motion
  - PTZ functions: zoom (in, out), pan (up, right, down, left), enlarge, show all, stop
  - cycle mode: cycles through all the cameras (or the cameras specified in the camera list parameter). Stops on a camera if there's a motion being recorded

- playback mode
  - playback controls: play/pause, previous/next motion
  - motion progress bar
  - send screenshot to an email address: sends the current frame to the specified address
  - download video: generates and download a video that includes all the frames from the currently shown one to the last one of the current motion
  - exit playback mode

While in playback mode, hovering the mouse cursor on the lower part of the image (or touching and dragging if using a touch-enabled device) will show a preview of the corresponding frame. Clicking the upper part of the image will toggle back to live mode.







Parameters

- **id**: the object's ID, used by UISets
- **address**: the name of the camera, as in the Cameras parameter in hsyco.ini, or the name of a camera grid, defined as gridN, where N is the grid number assigned in hsyco.ini
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **size**: the camera image's width and height
- **camera list**: optional. If defined, it limits the cameras you can view from this

panel when cycling between cameras (clicking on the camera name, or the camera view), to those listed here

Syntax

```
(camerapanel[!<id>] <address>; <position>; <width>; <height>; <camera list>)
```

E.g.

```
(camerapanel!cam1 livingroom; x3y0; 520; 283)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Camera attributes

Name	Value	Description
camera	<name>	The name of the camera, or the name of a camera grid, defined as grid<n>, where <n> is the grid number
play	true	Play the latest motion
	<yyyymmddhhmmss>	Play the motion from the specified time or the nearest frame available, using the format "yyyymmddhhmmss" (e.g. "20141016125534")
	<timestamp>	Play the motion from the specified timestamp (in seconds) or the nearest frame available
	rwd	Rewind to the previous motion
	fwd	Forward to the next motion
pause	true	Pause the camera if in playback mode
	<yyyymmddhhmmss>	Pause a motion at the specified time or the nearest frame available, using the format

		"yyyymmddhhmmss" (e.g. "20141016125534")
	<timestamp>	Pause a motion at the specified timestamp (in seconds) or the nearest frame available
	rwd	Rewind to the previous motion and pause it
	fwd	Forward to the next motion and pause it
live	true	Toggle live mode

Camerapanel attributes

Name	Value	Description
panel	true	Default value. Shows a background panel
	false	Hides the background panel

CameraLink

For a description and a complete list of UI Objects, see UI Objects.

A button, graphically identical to link, that controls the address of a camera or camerapanel object.

Parameters

- **address**: the name of the camera, as in the Cameras parameter in hsyco.ini, or the name of a camera grid, defined as gridN, where N is the grid number assigned in hsyco.ini
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **color**: button color, see color attribute for possible values
- **label**: the text label on the button. Can contain HTML tags
- **destination panel id**: optional. If set to a panel id, will show the camera on that camera panel

Syntax

```
(cameralink <address>; <position>; <color>; <label>; <destination panel id>)
```

E.g.

```
(cameralink kitchen; x20y20; gr; Kitchen; campanel1)
```

UI Attributes

UI Object Attributes (Common)

Camera

For a description and a complete list of UI Objects, see UI Objects.

A real-time view of a camera or grid of cameras.

Parameters

- **id**: the object's ID, used by UISets
- **address**: the name of the camera, as in the Cameras parameter in hsyco.ini, or the name of a camera grid, defined as gridN, where N is the grid number assigned in hsyco.ini
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **size**: the camera image's width and height
- **destination panel id**: optional. If set to a panel id, will show the camera on that camera panel

Syntax

```
(camera[!<id>] <address>; <position>; <width>; <height>; <destination panel id>)
```

E.g.

```
(camera grid1; x0y10; 312; 180)
(camera livingroom; x0y10; 100; 90; camerapanel)
```

UI Attributes

Common attributes

Name	Value	Description
pos	x<x>y<y>	Position specified as x/y coordinates. E.g. *x-5y10*
	r<r>c<c>	Position specified as row/column. E.g. *r1c2*
visible	true	Default. Show the object
	false	Hide the object
blink	true | slow	Blink the object at a slow speed
	fast	Blink the object at a fast speed
	false	Stop the blinking
opacity	0.0 ... 1.0	Object opacity from 0 (not visible) to 1 (fully visible)

Camera attributes

Name	Value	Description
camera	<name>	The name of the camera, or the name of a camera grid, defined as grid<n>, where <n> is the grid number
play	true	Play the latest motion
	<yyyymmddhhmmss>	Play the motion from the specified time or the nearest frame available, using the format "yyyymmddhhmmss" (e.g. "20141016125534")
	<timestamp>	Play the motion from the specified timestamp (in seconds) or the nearest frame available
	rwd	Rewind to the previous motion
	fwd	Forward to the next motion
pause	true	Pause the camera if in playback mode
	<yyyymmddhhmmss>	Pause a motion at the specified time or the nearest frame available, using the format "yyyymmddhhmmss" (e.g. "20141016125534")
	<timestamp>	Pause a motion at the specified timestamp (in seconds) or the nearest frame available
	rwd	Rewind to the previous motion and pause it
	fwd	Forward to the next motion and pause it
live	true	Toggle live mode

ButtonImage

For a description and a complete list of UI Objects, see UI Objects.

Controls a light or automation device, has the same function of the button object, but you can choose custom graphic images for the different states. You can add an optional label text next to the image, and define several optional CSS attributes to control the text's format and position. The "top" CSS attribute is used to move the text from its default position. If top is not defined, the text will be located just below the image. If defined, you can specify a positive or negative offset, in pixels. The zero offset position is at the top corner of the image; a positive offset moves the text down, a negative offset moves the text up.

If the button object is connected to a device or light that allows it, a long press on the button will open a popup with a slider. A second long press will close the popup. If specified on systemtopo.txt, it will show a thumbnail on mouse over or touch.

Parameters

- **address**: device data point name
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **size**: the image's width and height
- **unknown**: image associated to the "unknown" state
- **on**: image associated to the "on" state
- **off**: image associated to the "off" state
- **up**: image associated to the "up" state
- **down**: image associated to the "down" state
- **off up**: image associated to the "off up" state
- **off down**: image associated to the "off down" state
- **label**: the text label next to the button. Can contain HTML tags. It's optional
- **css style**: the label's css style, optional. See css style

> Image files are searched inside the ***www/<project's name>/img*** directory first, then inside the ***www/img*** directory.
> If the filename is set with a UISet, it's possible to load an image from the web by entering the URL, or a path relative to ***www/<skin's name>/pic*** by specifying ***/pic/*** followed by the filename

Syntax

```
(buttonimage <address>; <pos>; <width>; <height>; <unknown img>; <on img>; <off img>; <up img>; <down img>;
<offup img>; <offdown img>; <text>; <css>)
```

E.g.

```
(buttonimage autom.1; x10y20; 200; 200; door_unknown.png; door_open.pgn; door_closed.png; ; ; ; ; Front
Door; font-size:20px)
```

Background

For a description and a complete list of UI Objects, see UI Objects.

Page's background image.

Parameters

- **id**: the object's ID, used by UISets
- **image**: image filename

> Image files are searched inside the ***www/<project's name>/img*** directory first, then inside the ***www/img*** directory.
> If the filename is set with a UISet, it's possible to load an image from the web by entering the URL, or a path relative to ***www/<skin's name>/pic*** by specifying ***/pic/*** followed by the filename

Syntax

```
(background<!id> <image's filename>)
```

E.g.

```
(background!menubackground menubackground.png)
```

UI Attributes

Image attributes

img	<full url (http://...)> | <path relative to /img/>	Changes the image source
draw	<draw instructions>	Draws the image. See the Draw API e.g.: line(0,100,20,50,blue);line(50,50,70,70,'255,10,150');

Ats (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The user interface for the Aritech ATS multi-area intrusion detection system:

The Ats object is listed in the Project Editor's new object list only when at least one Aritech ATS I/O Server is defined.

For further details refer to the Ats I/O Server Application Note.

Parameters

- **server id**: the server ID
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(ats <server id>; <position>)
```

E.g.

```
(ats serverid; x10y20)
```

Aton (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The user interface for the Aton multi-room system:

File:UI Object aton.png

The Aton object is listed in the Project Editor's new object list only when at least one Aton I/O Server is defined.

For further details refer to the Aton I/O Server Application Note.

Parameters

- **server id**: the server ID
- **module**:
- **zone**: zone number of the Aton multi-room system (max 12 zones)
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(aton <server id>; <module>; <zone>; <position>)
```

E.g.

```
(aton serverid; module; zone; x10y20)
```

Airzone (UI Object)

For a description and a complete list of UI Objects, see UI Objects.

The user interface for the Airzone control framework:

File:UI Object airzone.png

The (airzone) object is listed in the Project Editor's new object list only when at least one Airzone I/O Server is defined.

For further details refer to the Airzone I/O Server Application Note.

Parameters

- **server id**: the server ID
- **zone**: the zone
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(airzone <server id>; <zone>; <position>)
```

E.g.

```
(airzone serverid; zone; x10y20)
```

AdminLink

For a description and a complete list of UI Objects, see UI Objects.

This button allows access to the standard configuration pages.

HSYCO supports four configuration pages: users, clock, network and password.

The users page allows you to manage the users access rights. It is possible to set and revoke the administrator rights, enable or disable a user, modify PIN and PUK, delete a user and define the access rights to sub-directories. If no page is specified through the Web interface, the user has no limits and can access all the existing pages.

The clock page sets the server's clock and time zone, and enables the automatic update of the clock based on Internet time servers.

The network page is used to change the IP parameters of Ethernet ports and DNS configuration. This page is normally not needed for regular use of HSYCO after the initial configuration.

The password page is used to change the system password. The system password is used to access both the operating system's console, directly or with SSH, and the HSYCO files through the shared folder network service. This password is needed to access the HSYCO SERVER for maintenance and to change its configuration.

The management features are active only for users with administrator rights. When a user without administrator rights access these pages, an error message is shown.

Parameters

- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **color**: button color, see color attribute for possible values
- **label**: the text label on the button. Can contain HTML tags

Syntax

(adminlink <position>; <color>; <label>)

The User Interface Configuration Database

The **systemtopo.txt** file should list all the I/O Servers data points that you want to control directly from the user interface.

It also contains the names of the Wi-Fi location zones.

> Several I/O Servers support a discovery mode, where all data points that are suitable for direct control from the user interface can be automatically added to the systemtopo.txt file. Refer to the Application Notes in order to configure the I/O Servers.
>
> systemtopo.txt can be modified at any time; any change is applied within a few seconds without restarting HSYCO. Web pages are automatically reloaded after any change to systemtopo.txt.

The file format is quite simple, it is separated in sections containing the list of all the devices available in the system. For example:

```
(location)
1p : FIRST FLOOR
2p : SECOND FLOOR

(devices)
k.light.14 : LIGHT ; LIGHT ;
k.light.21 : LIGHT ; DIMMER ; DIMMER
k.autom.14 : AUTOM ; VSHUT ;
```

Every section starts with a line where the name of the section is within brackets, followed by the list of devices corresponding to every section. Every line starts with the id, which identifies every specific device, followed by ":" and the list of parameters separated by ";".

The file is read ignoring blank lines and blanks within the lines which are not significant.

Any mistake in the file content will be shown in the log. In case of a general syntax error, the error message is:

```
systemtopo.txt Parser: Syntax Error, line: nnn
```

The number of the file line of the first mistake will always be shown.

💡 **In case of error the file is not fully interpreted, therefore the Web interface would not be able to work. The content of systemtopo.txt must be formally corrected to allow the correct execution of the Web Interface.**

The individual sections can be omitted if empty.

A systemtopo.txt file which is completely empty is a correct file anyway.

Every section must appear only once in the file, repeating the header of an already defined section is an error.

(location)

Contains a line for each Access Point id listed in the LocationBases parameter in the configuration file hsyco.ini.

The format is:

```
id : <zone name>
```

that is, the id as listed in LocationBases, followed by ":" and the user friendly name of the zone. The friendly name is shown on the up right corner of the display, and it is mandatory.

Example:

```
open : OPEN SPACE
```

(devices)

This section contains a line for each I/O Server data point that is used in the Web interface, with the following format:

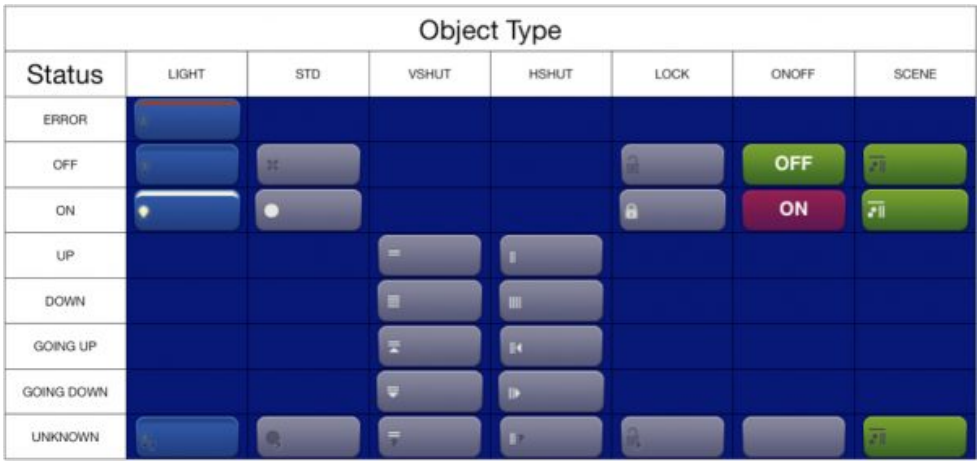
```
id : <function>; <type>; <img>; <options>; <description>
```

id is the data point ID, a point separated string with no spaces. See the Application Notes for data points' formats.

The <function> field can assume the LIGHT or AUTOM values, according to the configuration of the actuator associated to the bus address. For actuators configured in interlocking mode, it must be set to AUTOM.

The <type> field defines the type of function of the command, for the Web interface. It can assume one of the following values:

STD, LIGHT, DIMMER, VSHUT, HSHUT, LOCK, ONOFF, SCENE, corresponding to the graphics shown below (the DIMMER type is not shown in the chart since it is the same as LIGHT, the only difference is that the horizontal bar assumes a different length according to the dimmer level, in 20% steps) when using the standard "blue" skin.



The <img> field is optional. The following line has therefore a valid format:

```
k.12 : AUTOM; VSHUT
```

The <img> field can be used to associate an image or a camera to the device.

By clicking on the button or passing with the mouse in the button area the static image or the video of the camera will be shown in a small, 120x80 pixels pop-up next to the button.

For images, use the filename with extension, of a file located in the www/img directory.

To associate a camera, the cam:N format should be used, where N is the camera index starting from 1, based on the order as listed in the Cameras parameter in hsyco.ini.

The <options> field is also optional. It can assume the NOCLICK value to deactivate, on the Web interface, the possibility to change the status of the device.

The <description> field is an optional comment, not relevant for the correct execution of HSYCO, but shown in the Project Editor's devices list.

Example:

```
k.11 : LIGHT; LIGHT; entrance.jpg; Entrance
```

URL Keywords

The user interface supports some options that can modify its behavior. These options are activated by specifying special keywords in the page URL, they are therefore freely manageable from each Web Browser.

The complete URL format is:

```
https://<name server>/<URLKey>/<subdirectory name>?<options>
```

The options field can assume one or more of the keywords listed below; multiple keywords can be separated with the "." character:

- **nocamstream**: disables MJPEG streaming to the Web browser, even if the browser supports this format
- **nocache**: disables the HTML5 persistent cache for the Web session, overriding the default settings
- **nopics**: disables the display of the images associated to the devices, as set in the systemtopo.txt file. This option is typically used to enhance the performances in case of slow remote connections
- **page=<page name>**: when the user interface is initially loaded by the browser, display the specified page instead of the menu

Manager

The Manager is the Web-based toolkit of applications that you will use to create customized graphic user interfaces and access all system's configuration files.

> The Manager runs on most modern Web browsers, including tablets like Apple's iPad. It also runs on small screen devices like the iPhone or iPod touch, although a larger screen size is advisable for ease of use. Microsoft's Internet Explorer versions 8 and older are not supported, and IE9 is not guaranteed to work, although it should in most cases.

To access the Manager, enter the following URL in your Web Browser:

```
https://192.168.0.50/hsycoserver/manager
```

The Manager's application popup

Touch "about" in the bottom left corner to show HSYCO's current version information.

You need to authenticate using a PIN with administrator privileges to use the Manager.

The Manager has several applications:

- **Settings** - to configure the HSYCO system.
- **Admin** - for basic user and network settings.
- **Project Editor** - the visual editor used to create the user interfaces.
- **File Manager** - provides access to all HSYCO's files, just like the standard file manager you have in your personal computer, only this one is fully Web-based and works directly on HSYCO Server's files.

- **Log Viewer** - a real-time viewer of HSYCO's logs.
- **Status Browser** - a real-time viewer of data points, variables and dynamic user interface attributes.
- **System Monitor** - used to monitor several key hardware and operating system parameter of the HSYCO server.
- **Wiring Editor** - an easy to use end-user logic and scenarios builder.
- **KNX Utility** - accessible if the KNX I/O server is enabled
- **BACnet Utility** - accessible if the BACnet I/O server is enabled

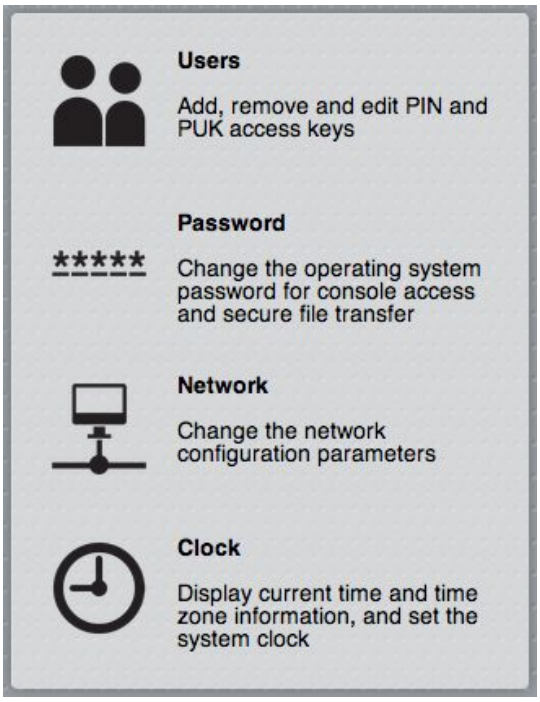
You can switch between applications from any application using the

button on the top bar.

Using a PC with a physical keyboard, you can also press **[Ctrl] and [Shift]** at the same time for fast switching between applications. An application selection panel appears until you keep the [Ctrl] key pressed; select the application you need by repeatedly pressing [Shift], then releasing [Ctrl].

Admin

The Admin application allows management of the Users, System Password, Network and Clock settings functions, described below.

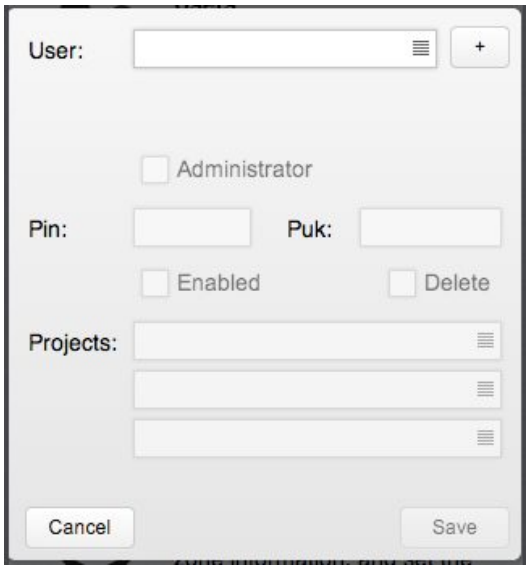


Users

The users page lets you fully manage individual users from the HSYCO Web interface.

You can set and revoke the administrator rights, enable and disable a user, modify PIN and PUK codes, delete a user, and also define the rights of access to the subdirectories.

If no pages have been specified through the Web interface, the user has no limits and can access all the existing pages.



To change the predefined PIN and PUK, select user, then enter the new PIN and PUK codes in the appropriate fields, and

press Save to confirm.

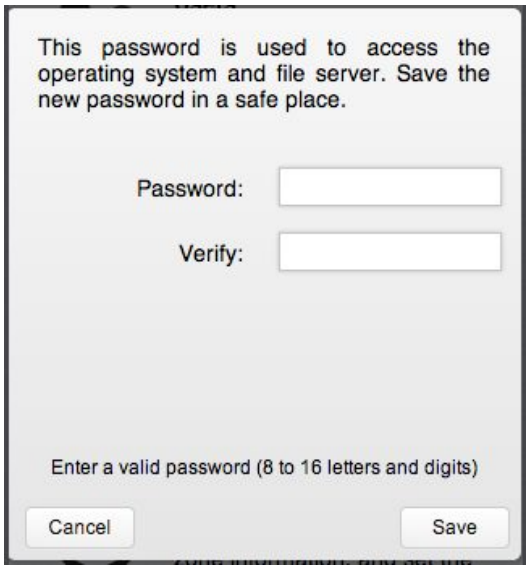
Or you can create one or more new users, at least one of them with administrator privileges, then log in again with the new administrator PIN and PUK and delete the default user.

System Password

The system password is used to access both the operating system's console, directly or with SSH, and the HSYCO files through the files sharing network service.

This password is needed to access the HSYCO SERVER for maintenance and to change its configuration.

To change the system password you should log in with a PIN and PUK corresponding to a user with administrator privileges.



Click the Password button in the Admin menu and enter the new password twice, then press Save to confirm.

In order to guarantee an adequate safety the password should be composed by letters and numbers, should not contain well-known words and needs to be of at least 10-16 characters.

You should save the password in a safe place.

If you forget the password, but still have access to HSYCO, you can simply go to the password page and type a new password.

The system password should be known only by the owner of the system.

If, for maintenance reasons, someone else needs to access HSYCO, you should change the password to a temporary new password, then revert to your secret password.

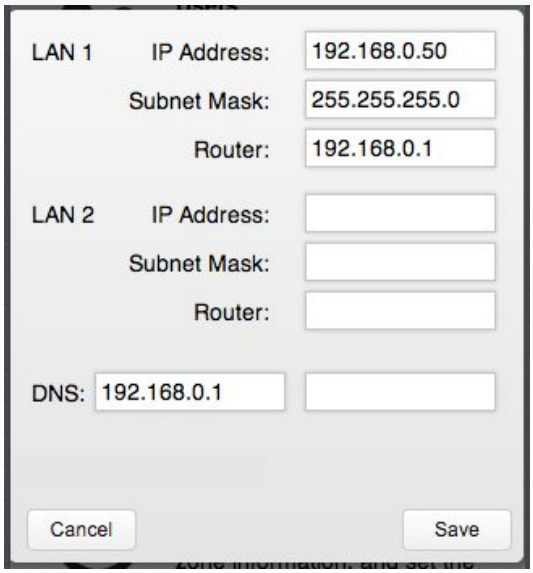
Be aware that the system password provides unrestricted access to the operating system configuration, to the HSYCO configuration, including users administration, and to all files, like recorded images from all the cameras and log files.

Network Settings

The network settings page lets you change the network parameters of HSYCO SERVER.

Be sure to know the meaning and the effects of your changes, as a mistake in these parameters may render HSYCO SERVER not accessible from the network and through the HSYCO Web interface.

> HSYCO implements a safety mechanism that in most cases would automatically revert, after 5 minutes from the change, the network settings to the original configuration if for any reason network access to HSYCO is lost after a change.



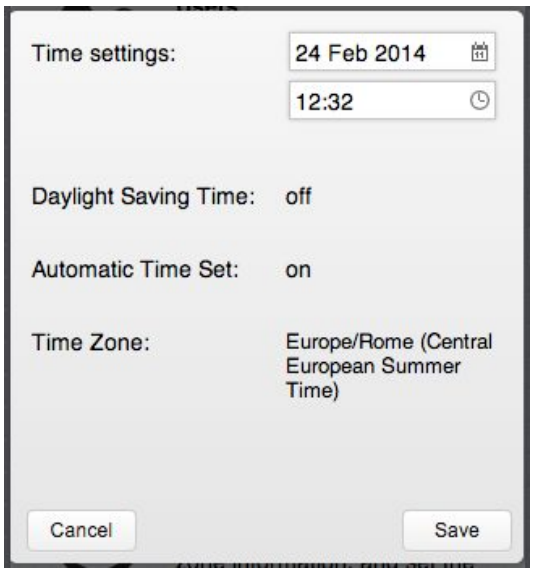
To change the network settings, enter the new parameters and press Save.

After the change, you have 5 minutes to log in again to any page to confirm your changes.

The network settings page is not normally needed after the initial installation, and it is usually deleted from the menu to prevent accidental changes to the network configuration.

Clock Settings

The clock page sets the server's clock and time zone, and enables the automatic update of the clock based on Internet time servers.



To change the time settings, enter the new parameters and press Save.

All the other parameters are read-only from the clock page, and can be set using the Settings menu, as explained in the Configuration Guide.

The time zone information is used to obtain the correct local time and automatically adjust for daylight saving time.

If automatic time is set to on HSYCO automatically sets the local date and time by polling Network Time Protocol (NTP) servers.

Project Editor

The Project Editor is a very powerful Web-based visual editor you can use to create user interfaces.

You will see three tabs in the top-left corner of the page, Projects, Pages, and Objects.

The **Projects** section is used to upload a project, edit its parameters, duplicate, delete or create a new project. **Pages** allows the creation and editing of individual pages o pop-ups within a project. Finally, the **Objects** tab is used to manage individual objects within a single page.

Creating a New Project

To create a new project, press the + button to show a blank project's settings form. Here you should enter a unique name for this project, this is used for both the path under www and the URL to access the project from the Web browser.

The "blue" **skin** is automatically selected. This is the standard skin. If you have other skins installed, they will appear in the Skin selector.

The **Language'** selector offers the supported languages, Italian, English, French and Chinese. The language attribute is used for all text messages in the project you are creating.

Header is a free-text field that simply corresponds to the Web page header (the text that normally appears in bookmarks and at the top of the browser's window). If no header is specified, the default header is the project's name.

The **size** options are discussed in the next paragraph.

When done, press the "create project" button to confirm the new project. The Project Editor will create a directory for this project with the index.hsm skeleton file.

Page Size and Coordinates

When creating a project, you should decide the size of it to fit the screen layout of the devices that you wish to use to access HSYCO.

In the **Size** section, press the iPad or iPhone button to preset a size that properly fits these devices, as well as several others, or choose your custom size by entering the width and height in pixels. You can freely have a larger width, as you would usually do for desktop or portable traditional computers, or a larger height, that normally fits the iPhone, iPod touch or other mobile phones better.

You can also enable the **orientation** support. With orientation enabled and on tablets and mobile phones that support screen rotation based on the device's orientation, you will be able to define distinct pages within the same project to show when the device is oriented vertically or horizontally. In this case, you could also define completely different layouts for the landscape and portrait pages, with a portrait layout optimized for the iPhone, and the landscape for the iPad or personal computers.

If the device doesn't support orientation, the landscape format will be used when showing the project's Web interface.

The general layout and metrics of the page area are shown in the following snapshot, based for example on a 1000x800 window size.

You can use different formats for the position field of objects:

- X-Y pixel coordinates. For example, x:100 y:200 places the object's origin at pixel 100 horizontally and 200 vertically, starting from the top-left corner of the display area
- Row-Column coordinates. r:12 c:14 places the object at row 12, column 14 based on the reference grid, which depends on the size of the object using it

In the index.hsm file, the position value will be formatted as x<x>y<y> for X-Y coordinates and r<r>c<c> for Row-Column.

Editing a Project

Selecting the **edit** tab of an existing project, you have access to several additional parameters that affect the general behavior of the project's Web interface.

Device Image enables or disables the camera or image pop-ups associated to devices in the Web interface, as configured in the systemtopo.txt file.

Kiosk mode is used to remove the menu bar and borders, and is intended for digital signage or kiosks applications where you want to have a full-screen display of your pages. Selecting the "lock" mode also disables the navigation functions.

Scale sets a scaling factor to resize the whole page area from the original size. Values greater than 1.0 create a magnifying effect, proportionally increasing the size of the page and its content; values smaller than 1.0 make the page and its content smaller than the original.

Selecting **Project locked**, disables any action that could be sent pressing buttons and commands in this project's Web interface.

The **Background** parameters is used to select a custom image as the general background for all pages in this project.

Pressing the "..." button in the **Camera List** section, will show a pop-up dialog where you should enter a friendly description for each one of the cameras listed in the hsyco.ini file.

Just leave the description text field blank if you don't want to show a camera in this project.

Pressing the "..." button in the **Camera Overlay** section, will show a pop-up dialog where you could specify custom overlay image masks for each camera. It is optional, and if not specified HSYCO will use the standard overlay for PTZ cameras.

Camera Grid, just like the Camera List section, allows you to enter friendly descriptions for camera grids defined in hsyco.ini.

Scope List is used to associate a scope to the project, allowing specific UISet to be made. For further information about scope, see JavaScript Command and Utility Functions API.

Style allow skin customization based on several attributes. For further information, see Project Style.

Finally, **UISet** allows the execution of a group of UISet instructions. It's useful to initialize UI Attributes. UI Attributes in the list are specified with an Attribute and a corresponding Value. Attributes have the format **<object's id>.<attribute>**. For UISets that apply to the project, the project's id is optional. For UISets that apply to all objects of a certain type, the format **(<object's name>).<attribute>** is used.

E.g.

myimage.pos=x0y10	Changes position for object "myimage"
autoreload=false	Sets the project's autoreload attribute to false
(input).maxlenght=20	Sets the maximum length of characters of all input objects

At the bottom of the panel you have buttons to delete or duplicate the project.

Menus, Pages and Popups

We have three types of pages in the Web interface, menus, regular pages and pop-ups. To create a new page, select the pages tab, then press the + button.

Menus are just like ordinary pages, but a menu is also the home page of a project. In a project you can have only one menu, or two if the orientation mode is enabled: a landscape and a portrait version.

Press **menu** to create a menu, then select the **orientation**, if enabled. The **gestures** attribute defines the gestures that are available on the page. **Scope List** defines the page's scope (for further information see).

Press **page** to create a new page, or pick a page from the selection list to edit.

When a new page is created, the Project Editor automatically assigns it a name (that is the page ID) and a default title. You should change the page name and also its title to some more user friendly values.

Select the orientation, if enabled.

The **Location** field lets you associate a page to one or more location IDs, as defined with the LocationBases parameter in hsyco.ini. Thanks to the association between pages and locations, you can click the location name that is shown in the top-right area of the menu bar to jump to the pages associated to that area.

If **Protected** is set to pin or puk, the user will be forced to input the pin or puk when accessing the page.

Gestures and **Scope List** is the same as the menu page.

A page can be **deleted** or **duplicated** to an identical page, with a different name (ID) using the two buttons at the bottom of the panel. Also it's possible to **copy** and **cut** the page. To paste the page, there's a button on the bottom of the select tab.

Press **popup** to create a new pop-up.

Instead of using the full real estate of the project’s Web interface, a pop-up is a smaller page that shows on top of an existing page or another pop-up.

Just like a page, you should change the name and title of the popup, and also set its **width** and **height** in pixels. The FIT SIZE button automatically adjusts the popup's size based on the objects that you added to the popup.

The popup's **type** defines its behaviour: a simple popup appears linked to the object that generated it, while a modal popup is exclusive and appears in the middle of the page, darkening the background.

Objects

You can add and edit graphic and control objects in a page. Press the OBJ button and you will see the select tab with the list of all objects in the page. Press any object in the list to select it, then press the edit tab to edit its attributes.

Press the + button to add a new object. Pick an object from the list of all available objects, then move it in the page and edit its attributes.

The edit tab shows all attributes of the selected object. Some attributes' fields have selectors that let you choose from a list of values, like the address attribute of button objects. Others, like the image objects, have images selection wizards that let you browse existing files. Finally, objects with the CSS style attribute, have a style wizard.

Touch the object's coordinates to toggle between the row/column grid and pixels.

When you have an object selected, go to the select tab to change the order of the object in the page. You can move an object up or down the list.

Objects on top will graphically appear above others in the Web page.

Containers

A container is used to group other objects together. All the objects' positions within the container are relative to the container's position. Containers can be also nested inside one another.

To create a container, press the + button, then choose container from the list.

A container box appears in the page. You can resize it by touching the bottom right corner and dragging, or open the container touching the bottom left corner.

When you open a container, you have access to the objects inside the container, like in a normal page. When you are done editing the enclosed objects, close the container touching again the bottom left corner.

The toolbar

A top toolbar allows access to functions that are specific of the Project Editor.

Next to the projects, pages, and objects tabs, we have the navigation hierarchy, with project's name, current page name, and selected object. You can click these items to quickly jump back to the objects' listing or pages' listing.

A warning button appears if there are errors in the page, like objects with missing parameters. Press the button to list all warnings, then press a warning message to highlight the object and show the specified attribute.

The tools icon shows a number of useful editing functions:

- **Snap** enables or disables the snap to grid function. Undo allows you to revert changes you've made since last save
- **Bring forward** and **send backward** move the selected objects on top or at the bottom of the others
- **Hide** hides an object from view in the Project Editor, not affecting the final interface, and is used to hide objects that are on top of others, so you can see objects that would otherwise be fully or partially hidden. The **unhide** button shows all hidden objects again
- **Select** toggles the multiple selection mode, that allows more objects to be selected and moved together, copied or cut. The same result can also be achieved holding the **shift** button on a physical keyboard
- **Paste** pastes copied objects on the current page
- **Source** shows the project's index.hsm file. Using the File Manager you can open it for editing and directly modify the project's content working on text. Going back to the Project Editor, if that project was open, you will be asked to reload the changes
- **Undo** undoes the last operation as you would normally expect

Keyboard Shortcuts

- **Ctrl + C**: copy selected object(s)
- **Ctrl + X**: cut selected object(s)
- **Ctrl + V**: paste copied object(s)
- **Ctrl + Z**: undo last operation
- **Ctrl + S** or **Cmd + S**: save project
- hold **Shift**: toggle multiple selection mode, click and drag or click on individual objects to select them
- **Ctrl + Shift**: Quick App switch

JavaScript Command and Utility Functions API

System Functions

dateSet

```
dateSet(year, month, day, hour, minute, second)
```

Sets the HSYCO Server operating system's clock and hardware clock, in the local time zone.

Parameters:

- year: numeric - the year, greater than 1900
- month: numeric - the month, between 1-12
- day: numeric - the day, between 1-31
- hour: numeric - the hour, between 0-23
- minute: numeric - the minute, between 0-59
- second: numeric - the second, between 0-59.

Returns:

- boolean - true if successful, false if not.

getNextSunrise

```
getNextSunrise(now, withoffset)
```

Computes the time of the next sunrise after the time set in the now parameter.

The sunrise time is returned in milliseconds since midnight, January 1, 1970 UTC.

Parameters:

- now: numeric - the reference time in milliseconds
- withoffset: boolean - true to take the sunrise offset into account, as defined with the SunriseOffsetMinutes in hsyco.ini, false to ignore the offset and return the actual sunrise time (Based on civil dawn, begins when the geometric center of the sun is 6° below the horizon).

Returns:

- numeric - the next sunrise time in milliseconds since midnight, January 1, 1970 UTC.

getNextSunset

```
getNextSunset(now, withoffset)
```

Computes the time of the next sunset after the time set in the now parameter.

The sunset time is returned in milliseconds since midnight, January 1, 1970 UTC.

Parameters:

- now: numeric - the reference time in milliseconds
- withoffset: boolean - true to take the sunset offset into account, as defined with the SunriseOffsetMinutes in hsyco.ini, false to ignore the offset and return the actual sunrise time (Based on civil dusk, ends when the geometric center of the sun reaches 6° below the horizon).

Returns:

- numeric - the next sunset time in milliseconds since midnight, January 1, 1970 UTC.

haActiveState

```
haActiveState()
```

When HSYCO is installed in a high availability configuration, this method returns true if the system (master or slave) is active, false if it is not active.

If high availability is not configured, the method returns true.

Returns:

- boolean - high availability current state.

haActive

```
haActive(active)
```

Force a master HSYCO server to become inactive, or return to the active state.

This method should be called on the master only, and has no effect on the slave server.

Parameters:

- active: boolean - set to true to force a master to become inactive, or false to return it to the active state after the master was forced to become inactive.

isDaylight

```
isDaylight()
```

Returns the day or night status at present time.

The result is based on the latitude and longitude data set in hsyco.ini and the SunriseOffsetMinutes and SunsetOffsetMinutes parameters.

Returns:

- boolean - true during the day, false at night.

powerSet

```
powerSet(power)
```

Sets the electric power state variable.

The power value is shown in the navigation or status bar of the Web interface.

Besides setting the power value, this method also triggers the execution of the powerEvent() callback and the POWER event in EVENTS.

Parameters:

- power: numeric - the power in Watt units.

programTimerClear

```
programTimerClear(name)
```

Cancels a program timer.

Parameters:

- name: string - the program timer name.

programTimerRepeat

```
programTimerRepeat(name, seconds)
```

Sets a program timer that is executed repeatedly with an interval equal to the specified number of seconds.

If a program timer with the same name has already been set, the number of seconds to the timer's execution is reset to the new value.

Parameters:

- name: string - the program timer name
- seconds: numeric - the interval in seconds of the timer's execution.

programTimerReset

```
programTimerReset(name, seconds)
```

Sets a program timer to be executed after the specified number of seconds.

If a program timer with the same name has already been set, the number of seconds to the timer's execution is reset to the new value.

Parameters:

- name: string - the program timer name
- seconds: numeric - the number of seconds to the timer's execution.

programTimerSet

```
programTimerSet(name, seconds)
```

Sets a program timer to be executed after the specified number of seconds. If a program timer with the same name has already been set, the action will be ignored.

Parameters:

- name: string - the program timer name
- seconds: numeric - the number of seconds to the timer's execution.

schedulerRegister

```
schedulerRegister(groupname, schedulename, interval)
```

This function registers a new scheduled callback.

Parameters:

- groupname: string - the group name
- schedulename: string - the schedule name
- interval: numeric - interval in milliseconds.

schedulerRemove

```
schedulerRemove(groupname, schedulename)
```

Deletes a schedule.

Parameters:

- groupname: string - the group name
- schedulename: string - the schedule name.

sleep

```
sleep(millis)
```

Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.

Parameters:

- millis: numeric - the length of time to sleep in milliseconds.

varGet

```
varGet(name)
```

Returns the value of a program variable, or null if no variable with the given name has been defined.

Program variables can be set using varSet() in user.java or in the EVENTS programming environment.

To access a variable used in EVENTS, the name used in varGet() should start with the $ character.

Variables names ending with ! are considered persistent and their values are preserved when HSYCO restarts; normal variables are deleted when HSYCO starts.

Parameters:

- name: string - the variable name. Names are case-insensitive.

Returns:

- string - the current value of the program variable.

varSet

```
varSet(name, value)
```

Sets a program variable to the value parameter.

Variables set with varSet() can be read using varGet() and are also available in the EVENTS programming environment, if the variable name starts with $.

If you need to use variables in Java or JavaScript but don't want these variables to be available in EVENTS, just use names that don't start with the $ character.

Variables names ending with ! are considered persistent and their values are preserved when HSYCO restarts; normal variables are deleted when HSYCO starts.

Parameters:

- name: string - the variable name. Names are case-insensitive
- value: string - the value to be assigned to the program variable.

Cameras

cameraCommand

```
cameraCommand(func, action, camera)
```

Sends PTZ and focus commands to cameras.

Parameters:

- function: string - see the table below
- action: string - see the table below
- camera: string - the camera name.

icon	function	action
	focus	far
	focus	near
	zoom	tele
	zoom	wide
	move	left
	move	right
	move	up
	move	down
	move	stop

cameraMode

```
cameraMode(cameraName, enabled)
```

Disables or enables a camera.

When the live view is disabled, recording is also disabled.

Parameters:

- camera: string - the camera name
- enabled: boolean - true to enable the camera, false to disable.

cameraRecMode

```
cameraRecMode(cameraName, enabled)
```

Disables or enables recording from a camera, not affecting live view.

Parameters:

- camera: string - the camera name
- enabled: boolean - true to enable the recording, false to disable.

cameraRecTrigger

```
cameraRecTrigger(cameraName, source, seconds)
```

Records video from a camera.

Parameters:

- camera: string - the camera name
- source: string - a string used to generate a log message, to identify the source that triggered the recording
- seconds: numeric - recording time in seconds. If set on 0, stops the current recording.

cameraRecTriggerFull

```
cameraRecTriggerFull(cameraName, source, seconds)
```

Records video from a camera, like cameraRecTrigger(), but ignores the DroppedFrames parameter in hsyco.ini, so that all frames captured from the camera during the recording period are recorded with no skips.

Parameters:

- camera: string - the camera name
- source: string - a string used to generate a log message, to identify the source that triggered the recording
- seconds: numeric - recording time in seconds. If set on 0, stops the current recording.

Data Logger

dataLoggerClear

```
dataLoggerClear(name)
```

Permanently deletes all historical data of the specified data logger.

Parameters:

- name: string - data logger name.

Returns:

- boolean - true if successful, false if not.

dataLoggerOptions

```
dataLoggerOptions(name, param, value)
```

Sets a specific option for the specified data logger.

So far the only available option is “origin” which sets the x-bar of all the charts linked to the data logger to the passed value.

Parameters:

- name: string - data logger name
- param: string - option to be set (“origin” is the only option you can currently set with this method)
- value: string - value for the specified option.

Returns:

- boolean - true if successful, false if not.

dataLoggerSave

```
dataLoggerSave(type, names, path, timestamp)
```

Creates or updates a CSV file containing the data processed by the specified data loggers.

The possible types are:

- log: append the last acquired value by the data logger to the specified file
- stat: updates the specified file with the currently gathered data by the specified data logger.

Parameters:

- type: string - log type (log or stat)
- names: string array - data logger names
- path: string - path of the file to be created
- timestamp: boolean - if true and the type is log adds a timestamp column in the CSV file for the logged values.

Returns:

- boolean - true if successful, false if not.

Examples:

The following code appends a new line every minute to the userdata/log.csv file with the most recent values of three data loggers, "a", "b" and "c".

```
time : { dataLoggerSave('log', ['a','b','c'], 'userdata/log.csv', true); }
```

dataLoggerUpdate

```
dataLoggerUpdate(name, value)
```

Supplies the specified data logger with a new value to be processed.

Parameters:

- name: string - data logger name

- value: numeric - value to be processed. Setting value to null, with counter data loggers only, resets the internal delta baseline.

Returns:

- boolean - true if successful, false if not.

DMX

dmxGet

```
dmxGet(channel)
```

Returns the current status of a channel on the DMX-512 bus.

Parameters:

- channel: numeric - when using one DMX gateway, address will be a standard DMX-512 address between 1 and 512. If there is more than one gateway, 1000 must be added to the DMX address for the second gateway, 2000 for the third one and so on.

Returns:

- numeric - channel current status, between 0 and 255, or UNKNOWN (-1) if the status is unknown.

dmxOff

```
dmxOff(channel)
```

Sets a channel on the DMX-512 bus to 0. It also saves the previous value, so it can be restored with dmxOn().

Parameters:

- channel: numeric - when using one DMX gateway, address will be a standard DMX-512 address between 1 and 512. If there is more than one gateway, 1000 must be added to the DMX address for the second gateway, 2000 for the third one and so on.

Returns:

- numeric - 1 if successful, 0 if not.

dmxOn

```
dmxOn(channel)
```

Sets a DMX-512 channel to the value preceding the last call to dmxOff().

Parameters:

- channel: numeric - when using one DMX gateway, address will be a standard DMX-512 address between 1 and 512. If there is more than one gateway, 1000 must be added to the DMX address for the second gateway, 2000 for the third one and so on.

Returns:

- numeric - 1 if successful, 0 if not.

dmxMerge

```
dmxMerge(fromchannel, tochannel, merge)
```

Sets all DMX-512 channels between the parameters from and to to merge mode or normal mode. In merge mode, the channels on the DMX OUT bus of the gateway follow the values of the channels of the DMX IN bus.

Parameters:

- fromchannel: numeric - first address of the range. When using one DMX gateway, address will be a standard DMX-512 address between 1 and 512. If there is more than one gateway, 1000 must be added to the DMX address for the second gateway, 2000 for the third one and so on
- tochannel: numeric - last address of the range
- merge: boolean - true to enable merge mode, false to return to normal mode.

Returns:

- numeric - 1 if successful, -1 if to < from, 0 for other errors.

dmxSet

```
dmxSet(channel, state)
```

Sets the value of a single DMX-512 channel.

Parameters:

- channel: numeric - when using one DMX gateway, address will be a standard DMX-512 address between 1 and 512. If there is more than one gateway, 1000 must be added to the DMX address for the second gateway, 2000 for the third one and so on

state: numeric - the new value of the channel, from 0 to 255.

Returns:

- numeric - 1 if successful, 0 if not.

Infrared Control

irtransCommand

```
irtransCommand(irid, command)
```

Sends a command to an IRTrans.

An infrared command is composed by the name that identifies the commands database, followed by a comma and the name of the command in the database.

More commands in a sequence can be separated by ";".

It is possible to insert the special command @N, which introduces a wait time of N milliseconds before sending the next command.

This command:

```
denon1036,on;@4000;denon1036,tv
```

sends the "on" command of the "denon-1036" remote, then waits 4 seconds before sending the "tv" command.

Moreover, the special command @0 allows to send IR commands without wait times before sending the next command.

When the @N delay before an IR command is not specified, there will be a default 500ms pause between commands; with @0 this pause will be reduced to 1ms.

For example:

```
denon,on;@0;nad,on
```

sends the "on" command of the "denon" remote, immediately followed by the "on" command of the "nad" remote.

💡 **Every IRTrans can contain a database of several remote controls, and each database stores the list of the distinct IR commands. If the hsyco/ir directory contains a .ccfhex file corresponding to the database name, then the CCF string in the file is used instead of sending the command stored in the IRTrans internal database.**

Parameters:

- irid: string - identifies the IRTrans, as defined with the IRTrans parameter in hsyco.ini
- command: string - the sequence of IR commands to be sent.

Returns:

- numeric - 1 if successful, 0 if not.

I/O Servers

ioGet

```
ioGet(id)
```

Retrieves the current status of an I/O data point.

Parameters:

- id: string - the full data point name, including the I/O server id prefix.

Returns:

- string - the data point value.

ioServersRestart

```
ioServersRestart(name)
```

Restarts an I/O server that was previously stopped.

This command has no effect on a running I/O server.

Note that this command will return before the I/O server is actually restarted.

Parameters:

- name: string - the I/O server name.

Returns:

- numeric - 1 if successful, -1 if the server name is not defined.

ioServersStop

```
ioServersStop(name)
```

Stops the execution of an I/O server.

Note that this command will return before the I/O server is actually stopped.

Parameters:

- name: string - the I/O server name.

Returns:

- numeric - 1 if successful, -1 if the server name is not defined.

ioSet

```
ioSet(id, value)
```

Writes to I/O servers variables or data points.

Parameters:

- id: string - the full data point name, including the I/O server id prefix
- value: string - the new value.

Leak Detector

The Intelligent Leak Detector is used to generate warning for potential water or other quantities leaks by analyzing any generic flow counter.

The detector should be called at regular intervals, possibly faster than the minimum time needed for the counter to increment its value by one.

It will generate a leak warning when the measured flow remains relatively constant over a certain amount of time.

Use this function at your own risk!

The Intelligent Leak Detector uses a correlation algorithm to distinguish between a constant flow and an irregular flow. A relatively constant flow over a certain amount of time is considered as a potential leak and generates a leak event.

In some conditions, also depending on the type of flow counter used, this algorithm could fail from properly and timely recognizing a real leak, or it could generate false leak warnings.

Use different names to implement multiple independent leak detectors.

leakDetectorClear

```
leakDetectorClear(name)
```

Clears the detector's internal state.

In normal applications, there is no need to call this method, as the detector automatically clears its state from old data.

Parameters:

- name: string - leak detector name.

leakDetectorOptions

```
leakDetectorOptions(name, param, value)
```

The detector's logic uses two parameters to set its sensitivity and time base. You can change the defaults to adapt to your specific conditions.

The "period" parameter changes the time base, expressed in seconds. The default is 1200 seconds. The detector generates a warning if the flow deviation constantly remains below the deviation threshold for the time period.

The "deviation" parameter changes the deviation percentage, and should be a number between 1 and 100. The default is 50. Set the period or deviation to 0 to temporarily disable the detector.

Small deviation values cause the algorithm to activate leak warnings only when the flow is very constant in time, becoming less susceptible to false alarms, but a value that is too low could miss actual leaks. Larger values could easily generate false alarms.

Both period and deviation parameters can be changed anytime, even when the detector is already processing data.

Parameters:

- name: string - leak detector name
- param: string - set to “period” or “deviation”
- value: numeric - the period or deviation value.

leakDetectorUpdate

```
leakDetectorUpdate(name, value)
```

Calls the leak detector passing the current liters counter value.

Parameters:

- name: string - leak detector name
- value: numeric - the flow counter value.

Returns:

- numeric - -1 if the detector status is not changed, 0 if the status has changed to a reset condition, and 1 if the status has changed to a leak condition.

Log

errorLog

```
errorLog(message)
```

Generates an error message in the daily log file.

Parameters:

- message: string - the text of the error message.

fileLog

```
fileLog(pathname, message)
```

Appends a generic text message at the end of a file.

Parameters:

- pathname: string - the file name. You can specify any path, starting from HSYCO’s base directory. If the file or parent directories don’t exist, they will be created automatically
- message: string - the message that will be appended at the last line of the text file.

messageLog

```
messageLog(message)
```

Generates an information message in the daily log file.

Parameters:

- message: string - the text of the error message.

Mail

sendMail

```
sendMail(to, from, subject, body)
```

Sends a multipart email message, mixing text and multiple images from cameras, live or recorded.

HSYCO SERVER sends the mail either directly to the recipient's domain mail server if the SmtpName parameter is not defined in hsyco.ini, or using a specific email account with user authentication and traffic encryption if the SMTP server and account parameters are set.

If the email message is sent directly to the recipient's domain mail server, you need to ensure that the mail server accepts mail to the destination address being sent with the from address and the public IP of the HSYCO SERVER.

💡 **HSYCO does not automatically retry sending the message if the destination mail server is not available when the send mail function is called.**

Parameters:

- to: string - a valid email address of the intended recipient. You can optionally specify the destination SMTP server name or IP address by appending :<server name or address to the recipient's email address, for example: john@example.com:192.168.1.1
- from: string - a valid email address to be used as the sender address
- subject: string - the message subject
- body: string array - the message body. To send an ordinary text, just fill the vector with the desired text strings. To send an image, add a string to the vector with the following format: "cam:<cameraname>[:<seconds_back>]". For example, "cam:door" sends a live frame from the camera called "door"; "cam:door:2" sends a frame that was recorded two seconds before the last recorded frame; "cam:door:0" sends the last recorded frame. To send files as attachments, use the following format: "file:<file name>". The file path is relative to the HSYCO root directory.

Returns:

- numeric - 1: message sent; 0: message not sent; -1: other errors, including address errors.

Examples:

```
... : { sendMail('ulde@hsyco.com', 'test@hsyco.net', 'this is a test', ['Door camera','cam:door:2']); }
```

sends an email message with body containing a short text and a camera snapshot.

Modbus

For detailed information about the Modbus protocol, and the response PDU formats, you should refer to the MODBUS Application Protocol Specification V1.1b documentation at:
http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

modbusReadCoils

```
modbusReadCoils(name, unit, address, quantity)
```

Reads the content of a contiguous block of coils in a MODBUS device, using function code **0x01**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x01, the second to the byte count of the data part, and the data starts from the third byte.

💡 **The coils in the response message are packed as one coil per bit of the data field. Status is indicated as 1= ON and 0= OFF. The LSB of the first data byte contains the output addressed in the query. The other coils follow toward the high order end of this byte, and from low order to high order in subsequent bytes. If the returned output quantity is not a multiple of eight, the remaining bits in the final data byte will be padded with zeros (toward the high order end of the byte). The byte count field in the response PDU specifies the quantity of complete bytes of data.**

If there is an error, the first byte is set to 0x81 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: string - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: numeric - the MODBUS slave unit identifier (1 to 247)
- address: numeric - the starting address of the block of coils
- quantity: numeric - the quantity of coils you are reading (1 to 2000).

Returns:

- numeric array - response PDU (the array elements are set to the unsigned decimal value of each byte).

modbusReadDiscreteInputs

```
modbusReadDiscreteInputs(name, unit, address, quantity)
```

Reads the content of a contiguous block of discrete inputs in a MODBUS device, using function code **0x02**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x02, the second to the byte count of the data part, and the data starts from the third byte.

💡 **The discrete inputs in the response message are packed as one input per bit of the data field. Status is indicated as 1= ON; 0= OFF. The LSB of the first data byte contains the input addressed in the query. The other inputs follow toward the high order end of this byte, and from low order to high order in subsequent bytes. If the returned input quantity is not a multiple of eight, the remaining bits in the final data byte will be padded with zeros (toward the high order end of the byte). The byte count field of the response PDU specifies the quantity of complete bytes of data.**

If there is an error, the first byte is set to 0x82 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: string - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: numeric - the MODBUS slave unit identifier (1 to 247)
- address: numeric - the starting address of the block of discrete inputs
- quantity: numeric - the quantity of discrete inputs you are reading (1 to 2000).

Returns:

- numeric array - response PDU (the array elements are set to the unsigned decimal value of each byte).

modbusReadHoldingRegisters

```
modbusReadHoldingRegisters(name, unit, address, quantity)
```

Reads the content of a contiguous block of holding registers in a MODBUS device, using function code **0x03**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x03, the second to the byte count of the data part, and the data starts from the third byte.

If there is an error, the first byte is set to 0x83 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: string - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: numeric - the MODBUS slave unit identifier (1 to 247)
- address: numeric - the starting address of the block of registers
- quantity: numeric - the quantity of registers you are reading (1 to 125).

Returns:

- numeric array - response PDU (the array elements are set to the unsigned decimal value of each byte).

modbusReadInputRegisters

```
modbusReadInputRegisters(name, unit, address, quantity)
```

Reads the content of a contiguous block of input registers in a MODBUS device, using function code **0x04**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x04, the second to the byte count of the data part, and the data starts from the third byte.

If there is an error, the first byte is set to 0x84 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: string - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: numeric - the MODBUS slave unit identifier (1 to 247)
- address: numeric - the starting address of the block of registers
- quantity: numeric - the quantity of registers you are reading (1 to 125).

Returns:

- numeric array - response PDU (the array elements are set to the unsigned decimal value of each byte).

modbusReadWriteMultipleRegisters

```
modbusReadWriteMultipleRegisters(name, unit, readaddress, readquantity, writeaddress, bytes)
```

Writes a block of contiguous registers in a MODBUS device, then reads the content of a contiguous block of holding registers, using function code **0x17**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x17, the second to the byte count of the data part, and the data starts from the third byte.

If there is an error, the first byte is set to 0x97 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: string - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway

- unit: numeric - the MODBUS slave unit identifier (1 to 247)
- readaddress: numeric - the starting address of the block of registers to read
- readquantity: numeric - the quantity of registers you are reading (1 to 125)
- writeaddress: numeric - the starting address of the block of registers to write
- bytes: numeric array - data to be written in the registers, using big-endian encoding (the most significant byte is sent first), with the array elements set to the unsigned decimal value of each byte.

Returns:

- numeric array - response PDU (the array elements are set to the unsigned decimal value of each byte).

modbusWriteMaskRegister

```
modbusWriteMaskRegister(name, unit, address, mask, bytes)
```

Writes a register with a bit mask in a MODBUS device, using function code **0x16**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x16, the next two bytes to the mask and the last two to register value.

If there is an error, the first byte is set to 0x96 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: string - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: numeric - the MODBUS slave unit identifier (1 to 247)
- mask: numeric array - the AND mask to be applied to the register
- bytes: numeric array - data to be written in the register, using big-endian encoding (the most significant byte is sent first), with the array elements set to the unsigned decimal value of each byte.

Returns:

- numeric array - response PDU (the array elements are set to the unsigned decimal value of each byte).

modbusWriteMultipleCoils

```
modbusWriteMultipleCoils(name, unit, address, coils, bytes)
```

Writes multiple coils in a MODBUS device, using function code **0x0F**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x0F, the next two to the data address of the first coil, and the last two to the number of coils written.

If there is an error, the first byte is set to 0x8F and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: string - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: numeric - the MODBUS slave unit identifier (1 to 247)
- address: numeric - the coil's address
- coils: numeric - the number of coils to be written
- bytes: numeric array - The requested ON/OFF states are specified by contents of the request data field, following the same data format defined in the Modbus specification for function 0x0F. A logical '1' in a bit position of the field requests the corresponding output to be ON. A logical '0' requests it to be OFF. The first byte represents coils with the lowest address, with the least significant bit addressing the lowest address coil. The array elements are set to the unsigned decimal value of each byte.

Returns:

- numeric array - response PDU (the array elements are set to the unsigned decimal value of each byte).

modbusWriteMultipleRegisters

```
modbusWriteMultipleRegisters(name, unit, address, bytes)
```

Writes a block of contiguous registers in a MODBUS device, using function code **0x10**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x10, the next two bytes to the starting address and the last two to the quantity of registers.

If there is an error, the first byte is set to 0x90 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: string - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: numeric - the MODBUS slave unit identifier (1 to 247)
- address: numeric - the starting address of the block of registers
- bytes: numeric array - data to be written in the registers, using big-endian encoding (the most significant byte is sent first), with the array elements set to the unsigned decimal value of each byte.

Returns:

- numeric array - response PDU (the array elements are set to the unsigned decimal value of each byte).

modbusWriteSingleCoil

```
modbusWriteSingleCoil(name, unit, address, value)
```

Writes a single coil in a MODBUS device, using function code **0x05**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x05 and the last two to 0xFF00 if the coil is on, and 0x0000 if the coil is off.

If there is an error, the first byte is set to 0x85 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: string - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: numeric - the MODBUS slave unit identifier (1 to 247)
- address: numeric - the coil's address
- value: boolean - true to turn the coil on, false to turn it off.

Returns:

- numeric array - response PDU (the array elements are set to the unsigned decimal value of each byte).

modbusWriteSingleRegister

```
modbusWriteSingleRegister(name, unit, address, bytes)
```

Writes a single register in a MODBUS device, using function code **0x06**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x06 and the last two to register value.

If there is an error, the first byte is set to 0x86 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: string - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: numeric - the MODBUS slave unit identifier (1 to 247)
- address: numeric - the register's address
- bytes: numeric array - data to be written in the register, using big-endian encoding (the most significant byte is sent first), with the array elements set to the unsigned decimal value of each byte.

Returns:

- numeric array - response PDU (the array elements are set to the unsigned decimal value of each byte).

Network Services

ping

```
ping(host, timeout)
```

Sends an IP network ping command to the host specified.

Parameters:

- host: string - the hostname or IP address
- timeout: numeric - timeout in milliseconds.

Returns:

- boolean - true if the host is reachable, false otherwise.

urlGet

```
urlGet(url, user, password)
```

Sends a GET HTTP or HTTPS request.

The HTTP basic or digest access authentication methods are supported.

Parameters:

- url: string - a full url, including the mandatory http:// or https:// query scheme and optional query string
- user: string - the user for HTTP authentication (set to null if authentication is not needed)
- password: string - the password for HTTP authentication (set to null if authentication is not needed).

Returns:

- string - a string with the HTTP return code, followed by ":" and the response content, or null if a generic error occurred (including malformed URLs).

urlPost

```
urlPost(url, contenttype, data, user, password)
```

Sends a POST HTTP or HTTPS request.

The HTTP basic or digest access authentication methods are supported.

Parameters:

- url: string - a full url, including the mandatory http:// or https:// query scheme and optional query string
- contenttype: string - the MIME content type of the data being sent (e.g. "text/plain")
- data: string - the data being sent with the POST request
- user: string - the user for HTTP authentication (set to null if authentication is not needed)
- password: string - the password for HTTP authentication (set to null if authentication is not needed).

Returns:

- string - a string with the HTTP return code, followed by ":" and the response content, or null if a generic error occurred (including malformed URLs).

wakeOnLan

```
wakeOnLan(broadcast, address)
```

Sends a Wake on LAN command to the LAN specified in the broadcast.

Parameters:

- broadcast: string - the LAN broadcast address. For example “192.168.0.255”
- address: string - the IP or MAC address of the network interface. For example, IP address “192.168.0.1” or MAC address “45-ab-44-00-00-ba”.

💡 **If the IP address is used to identify the destination interface, HSYCO will try to automatically determine the corresponding MAC address, permanently saving this association27. So, when using the IP address, and only the first time this method is called, the device with have to be on to allow HSYCO to determine the MAC address.**

Returns:

- numeric - 1 if successful, -1 if not.

Public Announcement

The audioPlayFile() and audioPlayText() functions are used to play recorded audio files or text-to-speech messages.

Audio can be sent to the Web browser, the server’s audio line out connector, the internal speaker or audio out line of Axis cameras, and the speaker of SNOM phones or PA devices.

See the Audio and Public Announcement section for additional information.

Audio Destination	Description
speaker	the server’s audio line out connector
web	the Web browser’s audio output 💡 **this feature is supported only on Firefox and Chrome browsers**
axis@camera id	audio sent to an Axis camera, using the camera id defined in the Cameras parameter in hsyco.ini

snom@ip:port	audio sent to SNOM phones and public announcement devices, to the multicast IP address and port specified (the IP address and port should be configured as multicast addresses on each phone)

audioPlayFile

```
audioPlayFile(where, file)
```

Plays a pre-recorded audio file.

Parameters:

- where: string - the audio destination; see the table above
- file: string - the audio file name pathname, relative to HSYCO's root directory.

Returns:

- numeric - the index of this message in the play back queue if successful, -1 in case of errors.

audioPlayText

```
audioPlayText(where, voice, text)
```

Converts a text message to audio using the installed text-to-speech engine, and plays the audio.

Parameters:

- where: string - the audio destination; see the table above
- voice: string - the voice name for the text-to-speech engine; see the Audio and Public Announcement section for additional information
- text: string - the text message for text-to-speech conversion.

Returns:

- numeric - the index of this message in the play back queue if successful, -1 in case of errors.

Serial Communication Ports

closeComm

```
closeComm(portname)
```

Closes a serial port connection when the serial port is defined ad a “server” type, and is ignored otherwise.

💡 **When used with serial servers having a fail-over configuration, this method forces a reconnect at the next call of readComm() or writeComm(), so that a switch between the fail-over IPs would happen if the current IP is not responding.**

Parameters:

- portname: string - the name of the communications port, as defined in the CommPorts parameter in hsyco.ini.

readComm

```
readComm(portname, len)
```

Reads a sequence of bytes from a communication port.

The number of bytes read depends on the availability of data in the buffer of the communication port, until the maximum number specified in the len parameter.

💡 **There is a 2 seconds default time-out on reads. If the buffer is empty, this function will wait until the time-out expires before returning.**

If len is set to 0, the function doesn't return data, but it empties the communication port's buffer from older data.

💡 **If both verboseLog and userLog are true, the full trace of received and sent bytes is written to the log file.**

Parameters:

- portname: string - the name of the communications port, as defined in the CommPorts parameter in hsyco.ini
- len: numeric - maximum number of bytes to be read from the port, or 0 to empty the buffer without reading data.

Returns:

- string - null in case of errors, a string with the hexadecimal representation of the received byte sequence, or an empty string if the incoming data buffer is empty. For example "45AB4400BA".

writeComm

```
writeComm(portname, data)
```

Sends a sequence of bytes to a communication port.

💡 **If both verboseLog and userLog are true, the full trace of received and sent bytes is written to the log file.**

Parameters:

- portname: string - the name of the communications port, as defined in the CommPorts parameter in hsyco.ini
- data: string - a string with the hexadecimal representation of the sequence of bytes to be sent. For example "45AB4400BA".

Returns:

- numeric - the number of bytes sent if successful, or 0 in case of errors.

Squeezebox

slimButton

```
slimButton(player, button)
```

Controls the Squeezebox players, simulating a remote control button.

Parameters:

- player: string - the Squeezebox name, as listed in the slimPlayers parameter in hsyco.ini
- button: string - the Squeezebox command.

Returns:

- numeric - 1 if successful, 0 otherwise.

slimCommand

```
slimCommand(player, command)
```

Sends commands to the Squeezebox players.

Parameters:

- player: string - the Squeezebox name, as listed in the slimPlayers parameter in hsyco.ini
- command: string - the Squeezebox command.

Returns:

- string - the protocol answer to the command if successful, or null in case of errors.

User Interface

All identified UI objects support dynamic attributes that can be changed at run-time.

The primary commands used to set dynamic attributes are the UISET action in EVENTS, and the uiSet() Java and JavaScript function.

Set levels

You can set dynamic attributes at different levels:

- global, using UISET and not specifying any session information - will set the attribute to the desired value for all clients
- client session, using UISESSIONSET on USER and PAGE events, or uiSessionSet(session, id, attr, value) in JavaScript and uiSet(String session, String id, String attr, String value) in Java - will set the attribute for a single specific client session
- application scope, using uiSessionSet(session, id, attr, value) in JavaScript and uiSet(String session, String id, String attr, String value) in Java - will set the attribute only for projects and pages that are explicitly bound (subscribed) to the same scope. You can use any string literal to identify a scope, and simply use that string in the session parameter of uiSet or uiSessionSet, instead of the client session id.

Priority rules

Overlapping global, session or scope UI attributes set on the same object is possible. In this case, a session set has the highest priority, followed by global set and finally the scope set. So, if you have a scope set on an object, and a global set is called on the same attribute of the same object, this global set will override the scope set. A session set will override both the scope and global set.

Session lifetime

While a global set will last until HSYCO is restarted, the client sessions have a limited lifetime.

Basically a client session is created whenever you load a URL pointing to an HSYCO project, and is automatically cleared after one hour of inactivity, where session activity is defined as the client being actively connected to the HSYCO server.

Scopes also have the same lifetime of client sessions, but activity in this case is defined as at least one client that is showing a page or project that is bound to the scope.

You can also create persistent scopes, simply using scope literals ending with the ! character. Persistent scopes will not follow be cleared, even after prolonged inactivity, and will last until HSYCO is restarted.

uiClear

```
uiClear(session)
```

Removes all session specific UI object attributes set with uiSet().

uiClear() is also implicitly called after the client session inactivity timeout expires.

The uiClearEvent() callback is called following the uiClear() execution if the client session becomes active again (i.e. the client starts handshaking again with the server with the same session id).

Parameters:

- session: string - session id string that uniquely identifies the client session.

uiGet

```
uiGet(id, attr)
```

Gets the current value of an attribute of an identified UI object, set with the uiSet() function or the UISET action in EVENTS.

Parameters:

- id: string - object id
- attr: string - attribute name.

Returns:

- string - the current value of the attribute of an identified UI object, or null if the object attribute has not been set.

uiSessionGet

```
uiSessionGet(session, id, attr)
```

Gets the session-specific value of an attribute of an identified UI object, set with the uiSessionSet() function or the UISESSIONSET action in EVENTS

Parameters:

- id: string - object id
- attr: string - attribute name.

Returns:

- string - the session value of the attribute of an identified UI object, or null if the object attribute has not been set.

uiSessionSet

```
uiSessionSet(session, id, attr, value)
```

Changes the dynamic attributes of identified user interface (UI) objects for a specific client session.

💡 **Session specific attributes values will override values set globally.**

id is the unique name that identifies the object, specified in the index.hsm file with the extension !id after the object type.

Parameters:

- id: string - object id
- attr: string - attribute name
- value: string - attribute value.

uiSet

```
uiSet(id, attr, value)
```

Changes the dynamic attributes of the identified user interface (UI) objects.

id is the unique name that identifies the object, specified in the index.hsm file with the extension !id after the object type.

Parameters:

- id: string - object id
- attr: string - attribute name
- value: string - attribute value.

user

```
user(name, param)
```

Triggers a USER event, the execution of the userCommand() JavaScript function and of the userCommand() Java method.

It can be used as a calling mechanism between Java, JavaScript and EVENTS.

Parameters:

- name: string - the name parameter, passed to userCommand() and the USER event
- param: string -the param parameter, passed to userCommand() and the USER event.

Audio and Public Announcement

HSYCO can play audio for public announcement, as pre-recorded files or using a text-to-speech engine that converts text to audio. Audio can be sent to the Web browser, the server's audio line out connector, the internal speaker or audio out line of Axis cameras, and the speaker of SNOM phones or PA devices. Playing Audio Files When playing a pre-recorded audio file, the file format should be supported by the audio destination you are using.

Destination	Audio Formats
speaker	most of the audio formats that are supported by Linux or Mac OS X should work
web	use an audio format that is supported by the target Web browser (only the most recent versions of Chrome and Firefox are supported)
axis	use uLaw, 8 KHz, 8 bit, mono, WAV files
snom	use uLaw, 8 KHz, 8 bit, mono, WAV files

The Text-to-Speech Engine

HSYCO uses different text-to-speech engines on Linux and Mac OS X.

On Linux, the default engine is eSpeak, with the possibility to use the optional Acapela engine that offers a wide selection of very high quality voices.

Go to http://espeak.sourceforge.net/ for more information about the eSpeak software.

On Mac OS X, HSYCO uses the integrated text-to-speech service and all its voices.

eSpeak

When using eSpeak, set the voice name to any of its supported voices.

Some voices are listed in the following table, but check the on-line documentation for the complete list.

Voice	Description
voice:en	English
voice:en-us	American English
voice:it	Italian
voice:fr	French

The AudioServerSpeed and AudioServerVolume configuration parameters are supported.

The AudioServerSpeed default is 100: higher numbers increase the speech speed.

The AudioServerVolume default is 10, and should be set between 0 and 20, where 20 generates an audio file with the greatest possible amplitude.

Acapela

When using Acapela, it is mandatory to set the AudioServerTTS configuration parameter to acapela:

```
AudioServerTTS = acapela
```

Moreover, set the voice name to any of its installed voices.

When playing to the SNOM phones, use 8k and 22k voices.

When playing to Axis cameras, use 8kmu voices.

Mac OS X

HSYCO supports text-to-speech on Mac OS X 10.7 or later, using the integrated text-to-speech engine.

You can install additional voices through the speech system preferences panel.

Voice names in HSYCO are the same names you see in this panel.

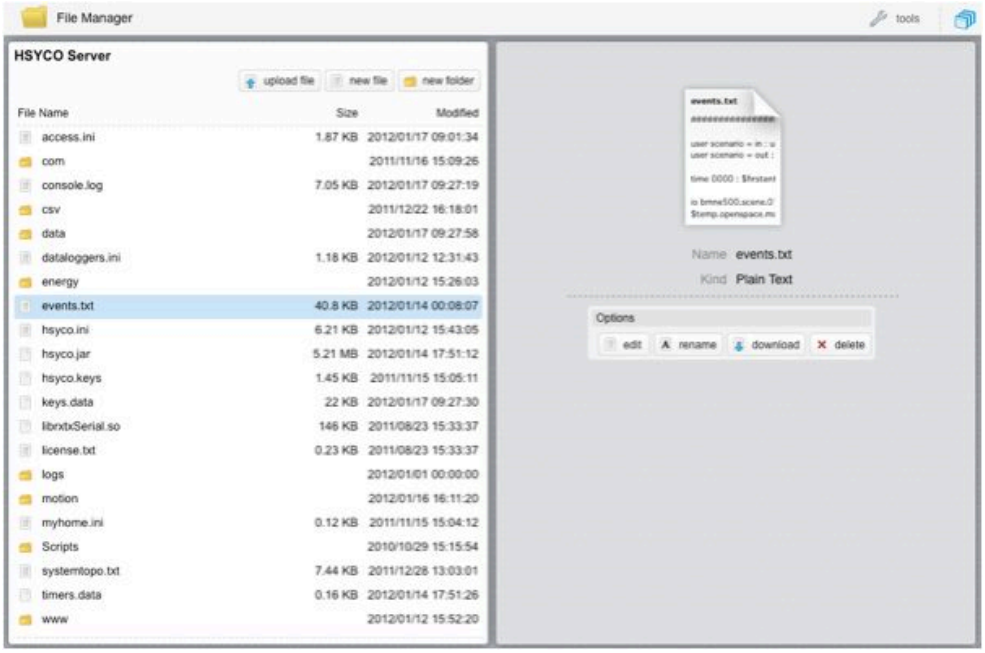
The AudioServerSpeed and AudioServerQuality configuration parameters are supported.

AudioServerSpeed is a positive number that represents speech rate, in words per minute.

AudioServerQuality sets the audio converter quality level between 0 (lowest) and 127 (highest).

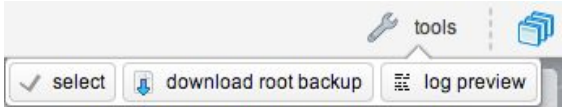
File Manager

The File Manager application offers all the conventional tools used to manage files, including a text editor to modify text files without having to download them to your computer.

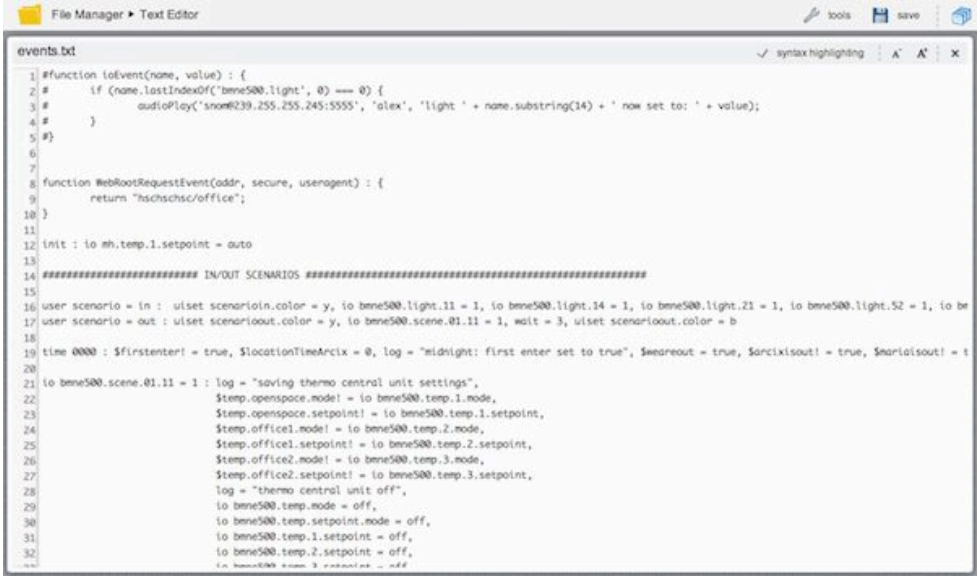


You can download and upload individual files or ZIP directories, and create, rename and delete files and directories.

You can download a full backup of all configuration files (excluding the logs and motion directories), using the download root backup tool. Because of its size, this function could take a considerable amount of time to generate and download the backup.zip file.



You can also activate a small live log view window, quite a useful tool while you are editing files and want to check the logs for debugging.



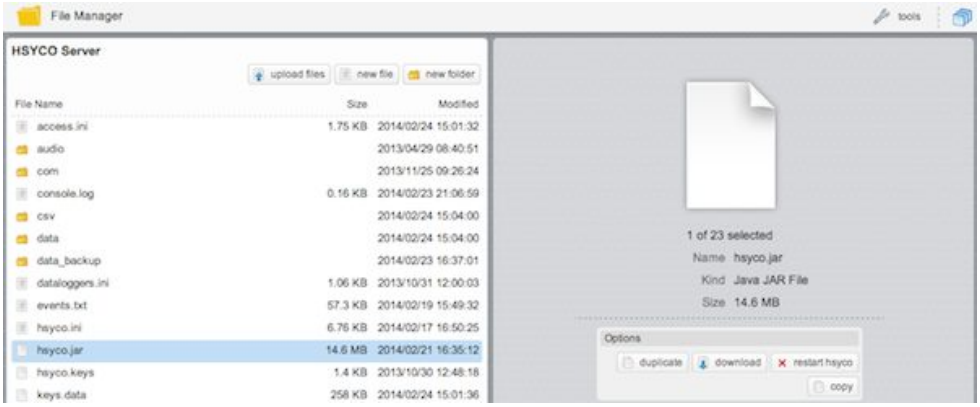
In the text editor, if someone modifies the file you are currently editing, you will be prompted with a warning message, asking to either reload the file with the new version, or ignoring the changes.

Your changes will be lost if you accept to reload the new version of the file.

Upgrading and restarting HSYCO

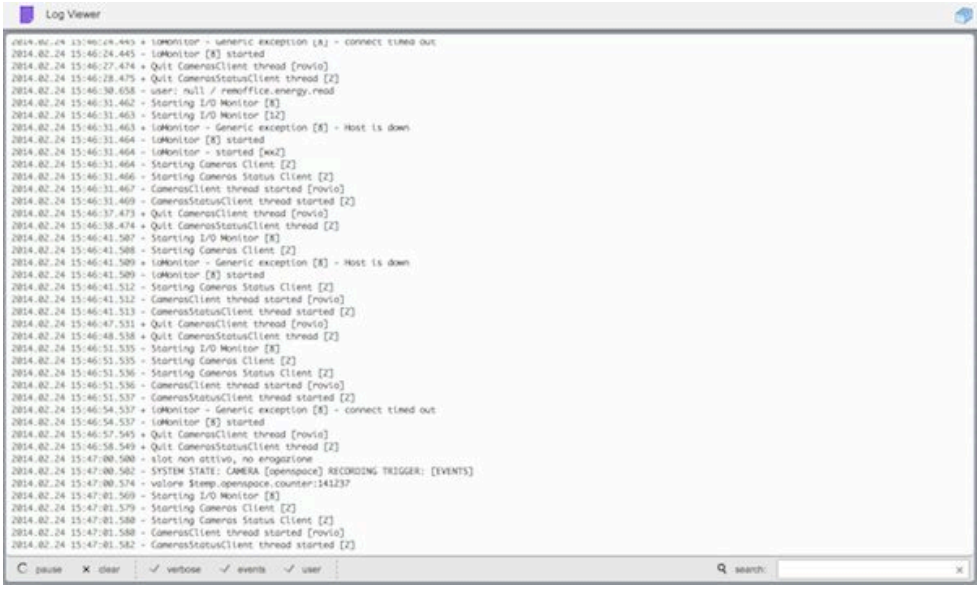
In order to upgrade the HSYCO software version, you should simply upload a new version of the hsyco.jar file to the root directory. After the upload, the server will restart and the Manager will reload on the browser.

You can also manually force an HSYCO restart. To do so, select the hsyco.jar file and press the “restart hsyco” button in the right panel.



Log Viewer

The Log Viewer application shows a scrollable live view of the last 1000 lines from the daily log file.



The tool bar at the bottom lets you filter log lines searching for a text, or to exclude a text in order to de-clutter the log. Type multiple words to select or exclude all lines that contain at least one of the words. Touch “search” to toggle between text search and exclude mode.

Touch “pause” to stop updating the viewer, then press again to restart. New log lines written while paused will not appear in the viewer, but will still be saved in the log file.

Touch “clear” to clear the log window.

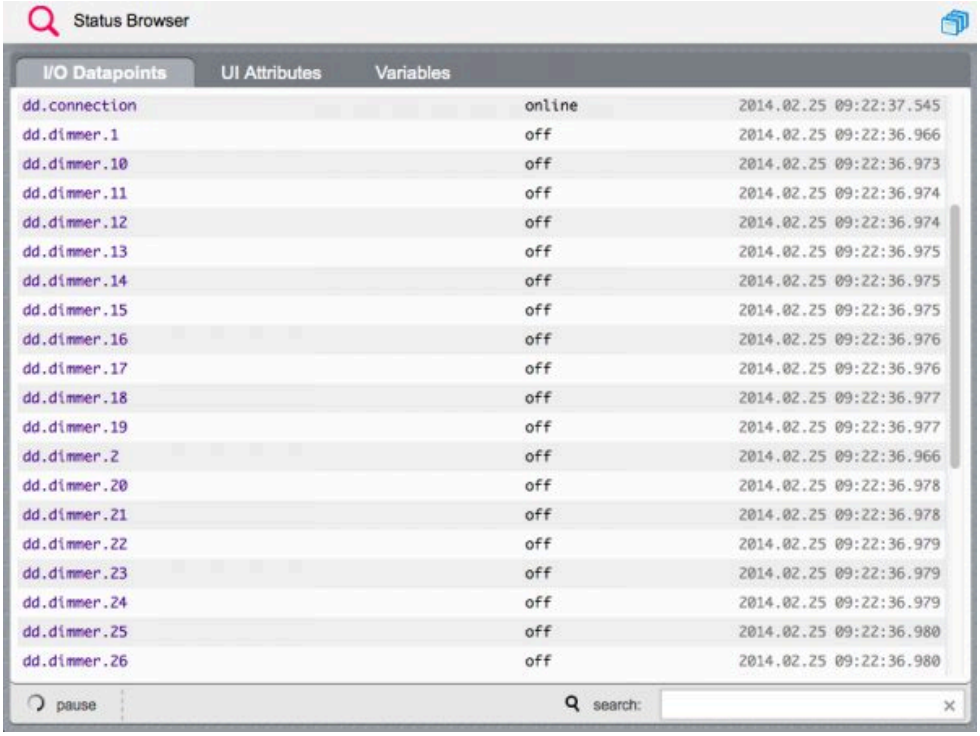
You can also temporarily select which log levels you want, without having to modify the hsyco.ini main configuration file.

Touch “verbose”, “events” or “user” to enable or disable the log levels. Changing log levels also affects what other users will see in the Log Viewer as well as what is actually written in the log file. Changes to the log levels will return to the preset configuration after you leave the Log Viewer application.

Status Browser

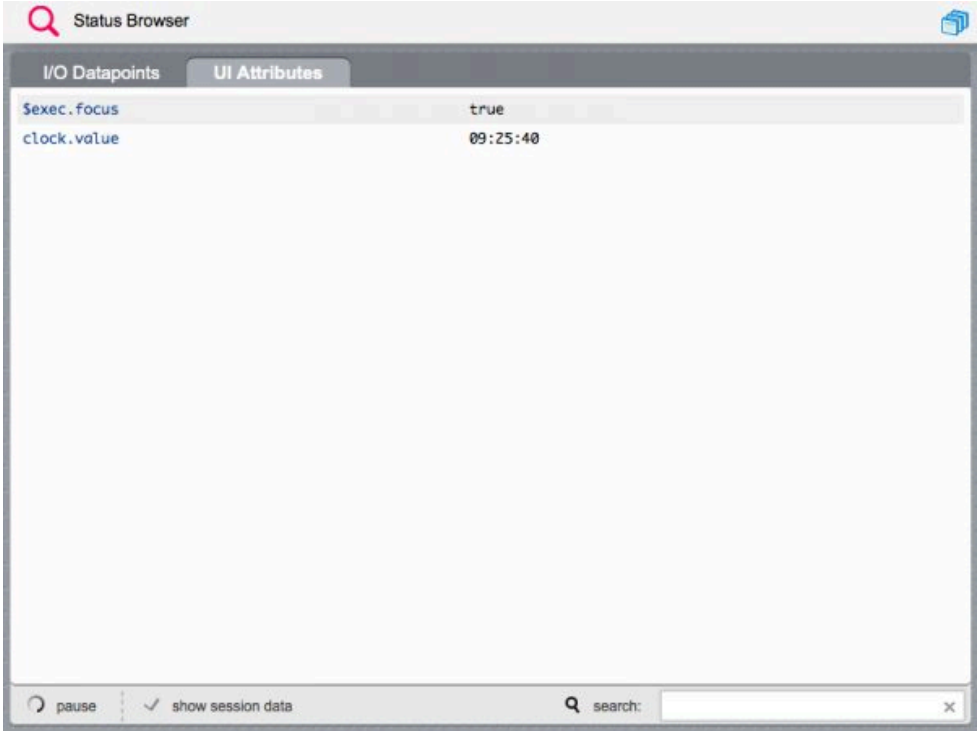
The Status Browser, allows you to show the real time status of data points, user interface attributes and variables.

Open the Manager at **https://192.168.0.50/hsycoserver/manager** and press the Status Browser icon.



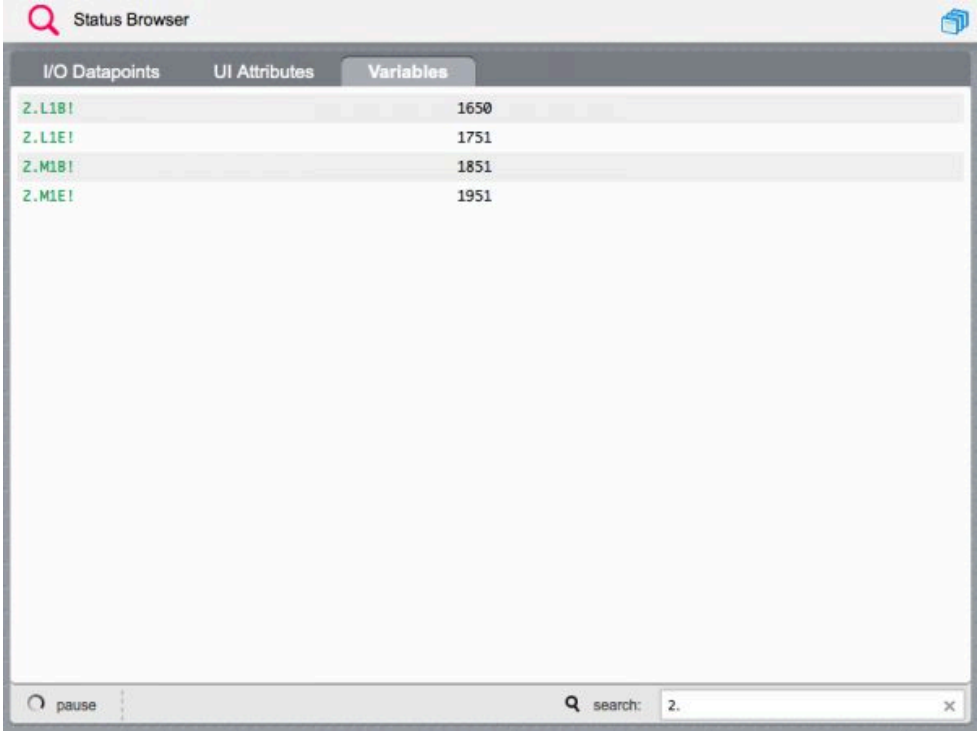
The data point tab shows the real time values of all I/O data points, and the time stamp of each data point last change.

You can pause the real time update, and filter the list by data point name.



The UI attributes tab shows all user interface ids and attributes with values that have been set with the UISET or uiSet() methods.

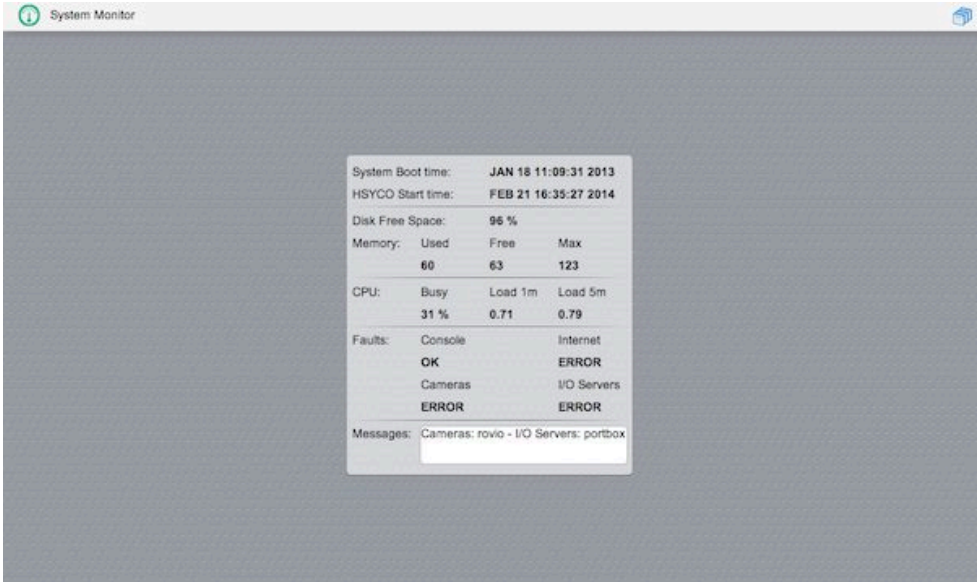
In order to view session specific attributes, you should open a tab or window on the same browser where you are using the Manager, and then select the show session data checkbox in the Status Browser. This will show both global and session data for that tab.



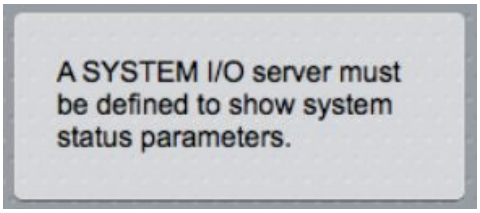
The variables tab shows the values of all volatile and persistent variables.

System Monitor

The System Monitor works in combination with the system I/O server, which must be enabled for this function to work, and monitors several key hardware and operating system parameters of the HSYCO server.



If the system I/O server is not enabled, the following message will be shown.

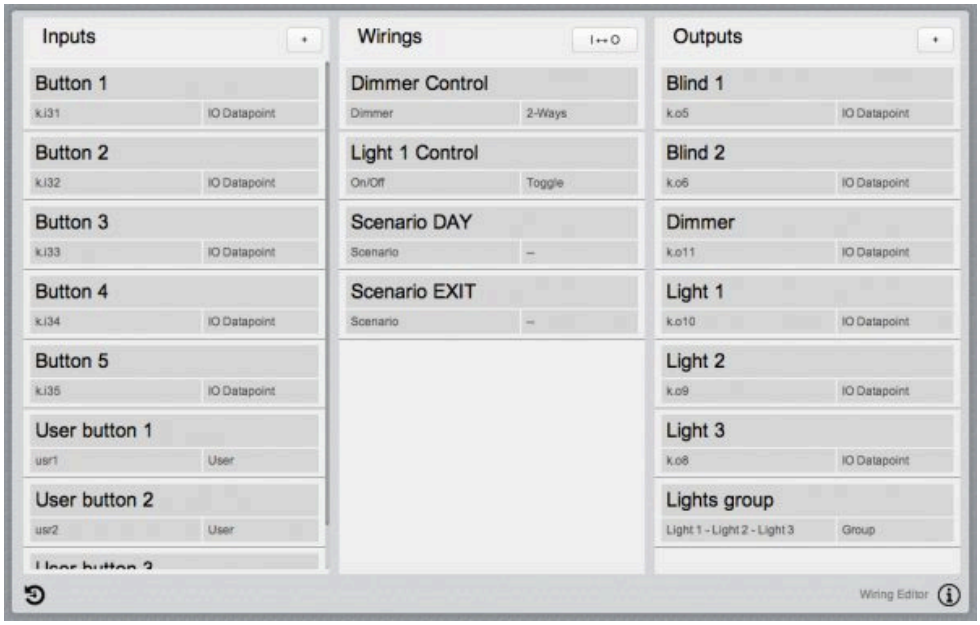


See the system I/O server application note for additional information.

Wiring Editor

The Wiring Editor is a tool to create logical connections (wirings) between HSYCO's inputs and outputs. It can be used, for instance, to quickly and easily make a button turn on/off a light or control a dimmer or an automation or, again, activate a scenario. It provides for a set of functionalities which are often implemented in home automation installation without having to resort to complex programming.

The Wiring Editor is accessible from the Manager or by directly accessing the the "wiring" Project page.



Inputs and Outputs

An **input** is something that triggers an event. It can be: a readable I/O Server's datapoint, a User event (e.g. an event triggered by a User button), a scheduler event or a group of inputs.

Inputs are divided into 2 categories:

Binary (0/1)
: those whose value can be "0" or "1" (e.g. a button), generally used to activate/deactivate an output or a scenario

Level
: those whose value correspond to a general numeric value (e.g. a sensor), used, for instance, to set an output to a specific level.

An **inputs group** corresponds to the logical "or" of the inputs belonging to the group; i.e. whenever an input belonging to a group triggers an event, the group will trigger an event with same value. Input groups can be used, for instance, to activate a light from two different buttons.

An **output** is something on which is possible to perform an action. It can be: a writable I/O Server's datapoint, a User event or a group of outputs.

Outputs are categorized as:

On/Off

they accept a value of "0" or "1". As an example, an On/Off output can be an I/O datapoint representing a light, but also a datapoint to arm/disarm an area of a security system

Dimmer
: they are generally I/O datapoints to control a dimmable light; accept the values "on", "off" and levels values from "1%" to "100%"

Blind
: they are generally I/O datapoints to control blind automations and accept the values "up", "down" and "stop".

An **output group** connects together a set of outputs. Writing a value to an output group will result in writing that value in all the members of the group. As we will see later, output groups are also used as a target for scenarios.

Default Configuration

When the Wiring Editor is first used, it will display a list of inputs and outputs corresponding to the datapoints of the configured I/O Servers. Generally speaking, whatever is commonly used as input or output will be listed in the respective column.

This table shows the I/O Servers for which inputs and outputs are automatically detected:

I/O Server	Inputs	Outputs
Domino	all input datapoints	all output datapoints
Contatto	all input datapoints	all output datapoints
MyHome		light and automation datapoints
KNX	all datapoints excluding virtual ones	all datapoints including virtual datapoints
HwgIO	all input datapoints	all output datapoints
Dummy		all datapoints

For Domino and Contatto I/O Servers make sure to set the “detectevents” option to "true" in their configuration. This allows for the name of the corresponding module to be displayed in the interface for each datapoint.

All inputs and outputs will have their names set to their datapoint ID and will have no value type defined. The first steps to do before creating wirings are: deleting from the lists all datapoints which will not be used here, adding any additional datapoint not originally included and assign a descriptive name and a value type to all inputs and outputs.

After the first time that an input or an output will be added or removed, HSYCO will no longer automatically update the respective list if, for instance, new I/O Servers are added.

Add a New Input/Output Datapoint

To add a new input or output I/O datapoint, click on the “+” button on the top of the corresponding column: a pop-up will show.

Select “IO Server datapoint” as type and insert the ID of the datapoint to be added, an user-friendly description and specify its value type. Press the “Add” button to add it to the list.

Edit or Remove an Input/Output

From the list, long-press on the item to be edited/removed until a pop-up shows up. To remove it click on the “Delete” button; to edit its name or value type, apply the changes and press "Save".

Create an Inputs/Outputs Group

To create a group click on the “+” button above the list: a pop-up will show. Select “Inputs group” as type, select the members of the group from the drop-down list, enter a name and click on “Add”.

Use a User Button as Input

To use a User button as input for the wiring editor, click on the “+” button on top of the inputs list: a pop-up will show. Select “User button” as type and enter the *name* parameter of the user button in the ID field.

If the long-press functionality of the button will be used (e.g. to dim a light or to move a blind automation while it is pressed) remember to set the “Repeat” attribute of the user button to “enabled” in the Project Editor.

Use a Scheduler as Input

It is possible to use a scheduler as input, for instance to have a scenario activated at a specific time every day. Click on the “+” button on top of the inputs list and select “Scheduler” as type.

Specify the ID of the scheduler to use, if the scheduler does not exist yet, just add a new ID that will be assigned to the scheduler. Enter a descriptive name and press on the “Add” button.

The new input will appear in the inputs list. To create or modify the scheduler’s rules, long-press on the item in the list until the edit pop-up will show. From here it is possible to change the descriptive name of the scheduler or, by clicking on the “Edit scheduler” link, access to the linked scheduler and modify it. Close the pop-up once done.

Generate a User Event as Output

It is possible to generate a user event as output for a wiring. This allows for the possibility to execute complex actions when the output is activated.

To this end, click on the “+” button on top of the outputs list: a pop-up will show. Select “User event” as type, input the ID of the event to be generated, assign it a name and specify the value type.

Whenever the output will be activated by a wiring a user event with the specified ID will be triggered and can thus be used in EVENTS, JavaScript or Java programming.

Wirings

A **wiring** represents a logical connection between one or more inputs and one output (which may be a group). It is mainly characterized by its output type, i.e. the value type of the output it acts on, based on which different functions are provided. The activation of a wiring occurs every time one of its inputs triggers an event and, depending on its function, will result in the modification of the output's status.

To create a wiring click on the button labelled as "I ↔ O" on top of the wirings column.

On/Off Wiring

This wiring acts on "On/Off" outputs and provides the following functions:

Toggle
: to invert the value of the output when activating a single input. The option "on" specifies when to activate the wiring: if set to "1" or "0" the wiring will be activated when the input triggers an event with the specified value; if set to "change" the wiring will be activated at each input event

2-Ways
: to control the output using 2 inputs: one for the "on" command and the other for the "off" command

Push-button
: to control the output with a single input used as push-button: the output will be activated only when the input is active (e.g. while the button is kept pressed), the following options can be specified:

Delay: set the delay period, in seconds, after which the command is sent to the output. If the input is active for a period shorter than the delay, no command will be sent
Hold: set the maximum time, in seconds, after which the output is deactivated

Direct
: to have the output follow the state of an input

Dimmer Wiring

This wiring acts on "Dimmer" outputs and provides the following functions:

Cycle
: to control the dimmer with a single input: a short activation of the input (e.g. short press on a button) will send on/off commands, while a long one (e.g. keeping a button held down) will dim up or down

2-Ways
: to control the dimmer using 2 binary inputs: one for the "on" command (short activation) and to dim up (long activation) and the other for the "off" command (short activation) and to dim down (long activation)

3-Ways
: to control the dimmer using 3 binary inputs: one to dim up, one to dim down, and the last one to toggle the on/off commands

Level
: to send a level command to the dimmer. It can be used to have the dimmer following the value of a level input or to set the dimmer to a fixed level when activating a binary input. The "ratio" option can be used to specify the multiplicative factor to apply to the value of the input. Use this option for adjustments between different scales or to set the fixed level if the wiring activation is triggered by a binary input

Blind Wiring

This wiring acts on "Blind" outputs and provides the following functions:

Cycle
: to control the blind automation with a single binary input: a short activation of the input (e.g. short press on a button) will activate the blind (with direction up or down depending on the current position) and will keep the blind to roll until the end is reached or the input is activated again. Keeping the input active for longer than a second (e.g. keeping a button held down) will activate the blind until the input is released. If the "Manned" option is checked the output will be activated only while the input is active, i.e. to move a blind automation it will be required to keep the input button held down; the functionalities activated by short activations of the input will be disabled.

2-Ways
: to control the blind using 2 inputs: one for the "up" command and the other for the "down" command. A short activation of the input will keep the blind to roll until the end is reached or the input is activated again, while keeping the input active for longer than a second will activate the blind until the input is released. As before, the "Manned" option disables the short-activation functions

3-Ways
: to control the blind using 3 inputs: one to send the "up" command, one to send the "down" command, and the last one to stop the movement of the blind

Scenario Wiring

A scenario wiring is used to set an output (most likely an output group) to a pre-recorded state when an input is activated. Follow these steps to create a scenario:

- Most probably you will want to create a scenario involving more than one device, thus we need to create a group with all the outputs we want to act on. It can be an heterogeneous group: its output members can have different value type. For instance a common “exit” scenario will act on all lights and blinds in the house. If you want to create a scenario involving only an output, ignore this step

- Set the output(s) to the state you want to record. For instance, for the “exit” scenario, turn off all the lights and dimmers and close the blinds.
- Create a scenario wiring by choosing the activating input and the target output. If a scheduler is selected as input, it is possible to select wether to activate the scenario when the scheduler is activated (On) or when it is deactivated (Off). This is to ease the process of having two different scenarios associated to the same scheduler: one for its activation and one for its deactivation.

When the scenario wiring is created, the current status of the output(s) is recorded and displayed in a pop-up. To edit the recorded state of the output(s), access the edit panel of the wiring with a long-press and click on "Record" when the output(s) are set to the new wanted state.

Using the Interface

A click on any of the items in the three columns (inputs, outputs or wirings) will show all the elements connected to it. For instance, to see which wirings are acting on an output, click on the output itself: the wirings column will show only the wiring acting on it and the inputs column only the inputs activating those wirings. To reset the interface click on the output again.

Expert/Basic Mode

By default the wiring editor is set to “expert mode”, i.e. all the provided functionalities are enabled. It is possible to disable the possibility of adding, editing and deleting I/O datapoint inputs and outputs to avoid the risk of having the user acting on sensible datapoints. To this end, go to HSYCO's *Settings* > *System* > *Extra* and add the parameter "wiringExpertMode" and set its value to "false".

The user will still be allowed to create groups, but only from the already available inputs and outputs.

Backups

To create a backup of the current configuration click on the icon in the bottom-left corner: a pop-up will show. Click on the “Backup now” button then enter a name for backup to be created.

The backups list will show all the saved backups ordered by date.

To load a previously saved backup, click on the corresponding item in the list and give a confirmation.

Inputs/Outputs Configuration Files

To ease the process of filtering the input and output I/O datapoints to be listed in the wiring editor, it is possible to list the datapoints’ IDs in configuration files to be added in a folder named "wiring" located in the root folder of HSYCO:

whitelist.in.txt: list of the IDs of all the inputs to be added

whitelist.out.txt: list of the IDs of all the outputs to be added

blacklist.in.txt: list of the IDs of all the inputs not to be added

blacklist.out.txt: list of the IDs of all the outputs not to be added

The blacklist files are to be used to have the Wiring Editor list all inputs/outputs automatically detected except for the specified ones.

It is possible to have the whitelist files automatically filled with all the automatically discovered inputs and outputs. To this end just place an empty file in the "wiring" folder named “whitelist.in.txt” and/or “whitelist.out.txt”.

The changes made to these configuration files will only have effect on restart.

The blacklist files have higher priority w.r.t. the whitelist files, i.e. if an ID is listed in both files, the corresponding datapoint won't appear in the wiring editor.

Contatto

The CONTATTO system is DUEMMEGI's proprietary bus architecture for building automation. HSYCO fully integrates with this system through the MCP control and gateway module, and its FXP-XT serial communication protocol.

A serial connection between the MCP and the HSYCO SERVER is required for the integration, either directly through the server's RS-232 port or via the RS-232 port of a supported serial to IP gateway, including the WEBS module.

Communication

Employ a DE-9 (often called DB-9) male-female RS-232 straight cable to connect the RS-232 MCP port to the serial port on HSYCO SERVER or serial gateway.

RS-232 parameters:

Baud rate	9600, 38400, 57600 or 115200 bps (according to the MCP serial port setting)
Data bits	8
Stop bit	1
Parity	none
Flow control	none

💡 **A speed of 115200 bps is recommended to achieve good performance, particularly when the number of devices connected to the bus is large.**

HSYCO Configuration

You can define and connect to more than one CONTATTO bus using several MCP gateways.

The CONTATTO I/O Server also supports dual, redundant connections between HSYCO and MCP.

If the main connection fails, HSYCO automatically re-routes the communication with MCP through the fail-over connection.

Options

ID	Default	Values	Description
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
inputdiscovery	false	true	auto-detects CONTATTO's input devices as configured in the MCP, and automatically creates the list of all detected devices and individual data points in the systemtopo.txt file.

			Should be enabled to allow the automatic update of (button) objects' states
		false	auto-detect for input devices is disabled
outputdiscovery	true	true	auto-detects CONTATTO's output devices as configured in the MCP, and automatically creates the list of all detected devices and individual data points in the systemtopo.txt file. Should be enabled to allow the automatic update of (button) objects' states
		false	auto-detect for output devices is disabled
virtualdiscovery	false	true	auto-detects CONTATTO's virtual points for input and output devices as configured in the MCP, and automatically creates the list of all detected virtual data points in the systemtopo.txt file. Should be enabled to allow the automatic update of (button) objects' states
		false	auto-detect for devices' virtual data points is disabled
virtualpoints	false	1 ... 2033	enables polling for the first n virtual points of the MCP. Enable this option only if you need to generate I/O events based on these virtual points
		true	enables polling for all the 2032 virtual points of the MCP
		false	polling of the MCP virtual data points is disabled
registers	false	1 ... 1024	enables polling for registers R0-Rn of the MCP. Enable this option only if you need to generate I/O events based on these registers
		true	enables polling for registers R0-R127 of the MCP
		false	polling of the MCP registers is disabled
powerdisplay	false	≥ 0 (MODANA address)	enables the automatic display in the GUI of the total real power measured by a DFANA module

			connected to this MCP. Note that, if you have more than one MCP gateway, you should enable this option for one gateway only
		false	power display disabled for this MCP
powersensivity	50	≥ 0	measured power changes are reported only if the difference from last reading is equal or greater than the power sensivity option, in Watts
detectevents	false	true	generate forced events when a device is detected at start-up
		false	do not generate events when a device is detected at start-up
toolspassword		string	set this option with a long string (only letters and numbers) password to allow the remote connection of MCP-IDE and other configuration tools to the MCP gateways that are connected to HSYCO. For additional security, it is recommended to set this option only when required
language	english	en it fr	language for all messages from the CONTATTO system: English, Italian or French

Initialization and Connection Events

Event name	Value	Description
connection	online	connection established to the MCP module
	offline	HSYCO can't connect to the MCP module
detected.input.<n>.	model name	an input device of the type passed as value has been detected at address <n>, using ~~addresses~~
detected.output.<n>.	model name	an output device of the type passed as value has been detected at address <n>, using ~~addresses~~

The Device Configuration Database

The **systemtopo.txt** file contains the list of all devices and their individual input, output and virtual data points that could be directly associated to graphic object in the Web-based user interface. This file can be filled manually or automatically by HSYCO at start-up.

To enable automatic discovery and automatic generation of devices' information in the systemtopo file, use the *inputdiscovery*, *outputdiscovery* and *virtualdiscovery* options in **Settings**. The default behavior is to only discover devices with output data points.

This is an example of an automatically generated systemtopo.txt file:

```
(devices)
contatto.o10.1 : LIGHT ; LIGHT
contatto.o6.1 : LIGHT ; DIMMER
contatto.o7.2 : LIGHT ; LIGHT
contatto.o7.1 : LIGHT ; DIMMER
contatto.o9.2 : AUTOM ; VSHUT
contatto.o5.4 : LIGHT ; LIGHT
contatto.o9.1 : AUTOM ; VSHUT
contatto.o5.3 : LIGHT ; LIGHT
contatto.o5.2 : LIGHT ; LIGHT
contatto.o5.1 : LIGHT ; LIGHT
```

You should then manually add comments and other optional parameters:

```
(devices)
contatto.o10.1 : LIGHT ; LIGHT; main entrance light
contatto.o6.1 : LIGHT ; DIMMER; lobby dimmer
contatto.o7.2 : LIGHT ; LIGHT; kitchen workbench
contatto.o7.1 : LIGHT ; DIMMER; kitchen main dimmer
```

MCP Virtual Data Points and Registers

The MCP implements 2032 virtual points for binary (on/off) data and 1024 registers for positive scalar values (0-65535). All the virtual points and the registers can optionally generate I/O events. You can also write to the virtual data points and registers using the IO action in EVENTS or ioSet() method in Java.

You cannot directly control the MCP virtual data points and registers using GUI objects.

To enable polling of the current state of the MCP's virtual data points, enable the *virtualpoints* options in hsyco.ini.

To enable polling of the current state of the MCP's registers, enable the *registers* options in hsyco.ini.

If you only have to write to virtual points or registers, enabling polling is not strictly required.

ID	Value	R/W	Description
v0.<n>	0	RW	virtual data point <n> off (<n>: 1-2032)
	1	RW	virtual data point <n> on (<n>: 1-2032)
r0.<n>	<x>	RW	register <n> off (<n>: 0-1023) set to value <x> (<x>: 0-65535)

MCP Internal Clock

The MCP module has an integrated real-time clock. You can use the clock datapoint to read the MCP date and time, and set the MCP clock to HSYCO's current time.

ID	Value	R/W	Description
clock	yyyy-mm-dd hh:mm:ss	R	the MCP clock current time
	read	W	read the MCP clock, and the delta with HSYCO's time
	sync	W	set the MCP clock to HSYCO's current time
clock.delta	integer number	R	the delta time in seconds between the MCP and HSYCO clocks. A positive number means that the MCP clock is ahead of HSYCO

CONTATTO Modules Events and Control

Module Code	Description
MOD8I/A 8	digital input module for NO contacts in modular housing
MOD32I/A	32 digital input module for NO contacts in modular housing
MOD8INP2/A	8 digital input module for NO contacts
MOD8INP2/C	8 digital input module for NC contacts
MOAN/I	0÷10V analog input module
MOAN/I4	quad 0÷10V analog input module
MI420	4÷20mA analog input module
MI420-X4	quad 4÷20mA analog input module
MOD4AM12/V/I	4-channel 0÷10V or 4÷20mA (0÷21mA) analog input module
MOD2PT	input module for two PT100 temperature probes
MODCNT	4-channel counter module in modular housing
MODNTC	input module for ambient temperature sensors and potentiometers
MOD4I/A	4-digital input module for NO contacts in modular housing
MOD4I/S	4-digital input module for NO contacts in 2M modular housing
MODLUX	ambient light measurement module with sensor for ceiling mounting
MODMETEO	meteorological data detection module for Contatto bus
MOD8R	8 power relay output module
MODPNP	8 positive logic (PNP) "open collector" digital output module

MODREL	8 relay output compact module in DIN 3M housing
MOAN/U	0÷10V analog output module
MO420	4÷20mA analog output module
MODLC	ambient light regulator module with integrated sensor
MOD4-4S	4 digital input and 4 power relay output multifunction module
MOD2-2R	2 digital input and 2 power relay output module
MOD4-4	4 digital input and 4 digital output module
MOD2DM	2x300W dimmer module
MOD2DV	dual 1-10V output for electronic ballast driving
MOD8IL	8 digital input – 8 LED output module for wall box
MOD4DV	quad 0-10V output module for generic applications or for external dimmers control
MODDMX	DMX gateway
MODDALI	4-channel DALI gateway
CLIMA2	module for the regulation of the ambient temperature
MODANA	network analyzer module for Contatto bus
MODPQ5	tags proximity reader and programmer
MODHT	hotel room controller
MODCA	access control module
MODKB	keypad access control module

MOD8I/A

8 digital input module for NO contacts in modular housing.

It uses one input address in the CONTATTO bus.

ID	Value	R/W	Description
i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address>.3	0	R	input pin 3 off
	1	R	input pin 3 on

i<address>.4	0	R	input pin 4 off
	1	R	input pin 4 on
i<address>.5	0	R	input pin 5 off
	1	R	input pin 5 on
i<address>.6	0	R	input pin 6 off
	1	R	input pin 6 on
i<address>.7	0	R	input pin 7 off
	1	R	input pin 7 on
i<address>.8	0	R	input pin 8 off
	1	R	input pin 8 on

MOD32I/A

32 digital input module for NO contacts in modular housing.

It uses four addresses of the Contatto bus, one for each group of 8 inputs.

ID	Value	R/W	Description
i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address>.3	0	R	input pin 3 off
	1	R	input pin 3 on
i<address>.4	0	R	input pin 4 off
	1	R	input pin 4 on
i<address>.5	0	R	input pin 5 off
	1	R	input pin 5 on
i<address>.6	0	R	input pin 6 off
	1	R	input pin 6 on
i<address>.7	0	R	input pin 7 off
	1	R	input pin 7 on
i<address>.8	0	R	input pin 8 off

	1	R	input pin 8 on
i<address+1>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address+1>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address+1>.3	0	R	input pin 3 off
	1	R	input pin 3 on
i<address+1>.4	0	R	input pin 4 off
	1	R	input pin 4 on
i<address+1>.5	0	R	input pin 5 off
	1	R	input pin 5 on
i<address+1>.6	0	R	input pin 6 off
	1	R	input pin 6 on
i<address+1>.7	0	R	input pin 7 off
	1	R	input pin 7 on
i<address+1>.8	0	R	input pin 8 off
	1	R	input pin 8 on
i<address+2>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address+2>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address+2>.3	0	R	input pin 3 off
	1	R	input pin 3 on
i<address+2>.4	0	R	input pin 4 off
	1	R	input pin 4 on
i<address+2>.5	0	R	input pin 5 off
	1	R	input pin 5 on
i<address+2>.6	0	R	input pin 6 off
	1	R	input pin 6 on
i<address+2>.7	0	R	input pin 7 off

	1	R	input pin 7 on
i<address+2>.8	0	R	input pin 8 off
	1	R	input pin 8 on
i<address+3>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address+3>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address+3>.3	0	R	input pin 3 off
	1	R	input pin 3 on
i<address+3>.4	0	R	input pin 4 off
	1	R	input pin 4 on
i<address+3>.5	0	R	input pin 5 off
	1	R	input pin 5 on
i<address+3>.6	0	R	input pin 6 off
	1	R	input pin 6 on
i<address+3>.7	0	R	input pin 7 off
	1	R	input pin 7 on
i<address+3>.8	0	R	input pin 8 off
	1	R	input pin 8 on

MOD8INP2/A

8 digital input module for NO contacts.

It uses one input address of the Contatto bus.

ID	Value	R/W	Description
i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address>.3	0	R	input pin 3 off
	1	R	input pin 3 on

i<address>.4	0	R	input pin 4 off
	1	R	input pin 4 on
i<address>.5	0	R	input pin 5 off
	1	R	input pin 5 on
i<address>.6	0	R	input pin 6 off
	1	R	input pin 6 on
i<address>.7	0	R	input pin 7 off
	1	R	input pin 7 on
i<address>.8	0	R	input pin 8 off
	1	R	input pin 8 on

MOD8INP2/C

8 digital input module for NC contacts.

It uses one input address of the Contatto bus.

ID	Value	R/W	Description
i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address>.3	0	R	input pin 3 off
	1	R	input pin 3 on
i<address>.4	0	R	input pin 4 off
	1	R	input pin 4 on
i<address>.5	0	R	input pin 5 off
	1	R	input pin 5 on
i<address>.6	0	R	input pin 6 off
	1	R	input pin 6 on
i<address>.7	0	R	input pin 7 off
	1	R	input pin 7 on
i<address>.8	0	R	input pin 8 off

	1	R	input pin 8 on

MOAN/I

0÷10V analog input module.

It uses one input address in the range 1 to 127 of the Contatto bus.

ID	Value	R/W	Description
i<address>.1	0 ... 255	R	voltage value divided in 255 steps

MOAN/I4

Quad 0÷10V analog input module.

It uses four consecutive addresses of the Contatto bus, one for each input.

ID	Value	R/W	Description
i<address>.1	0 ... 255	R	voltage value divided in 255 steps
i<address+1>.1	0 ... 255	R	voltage value divided in 255 steps
i<address+2>.1	0 ... 255	R	voltage value divided in 255 steps
i<address+3>.1	0 ... 255	R	voltage value divided in 255 steps

MI420

4÷20mA analog input module.

It uses one input address in the range 1 to 127 of the Contatto bus.

ID	Value	R/W	Description
i<address>.1	0 ... 255	R	current value divided in 255 steps

MI420-X4

Quad 4÷20mA analog input module.

It uses four consecutive addresses of the Contatto bus, one for each input.

ID	Value	R/W	Description

i<address>.1	0 ... 255	R	current value divided in 255 steps
i<address+1>.1	0 ... 255	R	current value divided in 255 steps
i<address+2>.1	0 ... 255	R	current value divided in 255 steps
i<address+3>.1	0 ... 255	R	current value divided in 255 steps

---

MOD4AM12/V/I

4-channel 0÷10V or 4÷20mA (0÷21mA) analog input module.

It uses one input address of the Contatto bus.

ID	Value	R/W	Description
i<address>.1	0 ... 4095	R	measurement range divided in 4095 steps
i<address>.2	0 ... 4095	R	measurement range divided in 4095 steps
i<address>.3	0 ... 4095	R	measurement range divided in 4095 steps
i<address>.4	0 ... 4095	R	measurement range divided in 4095 steps

---

MOD2PT

Input module for two PT100 temperature probes.

It uses one or two addresses according to how the dip-switch is set.

If both channels are enabled (IN1 and IN2), setting the n “base” address through the FXPRO programmer, the input address and address+1 (consecutive) will be automatically assigned to the module.

ID	Value	R/W	Description
i<address>.1	0 ... 255	R	range –40 ÷ +87.5°C: code 0 corresponds to –40°C and code 255 corresponds to +87.5°C
i<address+1>.1	0 ... 255	R	range –40 ÷ +87.5°C: code 0 corresponds to –40°C and code 255 corresponds to +87.5°C

MODCNT

4-channel counter module in modular housing.

It uses one input address of the Contatto bus.

ID	Value	R/W	Description
i<address>.1	0 ... 65535	R	number of pulses
i<address>.2	0 ... 65535	R	number of pulses
i<address>.3	0 ... 65535	R	number of pulses
i<address>.4	0 ... 65535	R	number of pulses

MODNTC

Input module for ambient temperature sensors and potentiometers.

It uses one input address of the Contatto bus.

The temperature values measured by MODNTC module are reported as Kelvin degrees multiplied by 10; in other words, 0°C will be reported as 2730, 0.1°C will be reported as 2731 and so on.

Concerning the potentiometer channels, the values read from the module will be 0 with the potentiometer at its minimum position and it will be 1000 with the potentiometer at its maximum position.

ID	Value	R/W	Description
i<address>.1	0 ... 4095	R	temperature value
i<address>.2	0 ... 4095	R	temperature value
i<address>.3	0 ... 4095	R	temperature value
i<address>.4	0 ... 4095	R	temperature value

MOD4I/A

4-digital input module for NO contacts in modular housing.

It uses one input address in the range 1 to 127 of the Contatto bus.

ID	Value	R/W	Description
i<address>	0	R	input pin 1 off
	1	R	input pin 1 on
i<address+1>	0	R	input pin 1 off
	1	R	input pin 1 on
i<address+2>	0	R	input pin 1 off
	1	R	input pin 1 on
i<address+3>	0	R	input pin 1 off

	1	R	input pin 1 on

MOD4I/S

4-digital input module for NO contacts in 2M modular housing.

It uses one address of the Contatto bus.

ID	Value	R/W	Description
i<address>	0	R	input pin 1 off
	1	R	input pin 1 on
i<address+1>	0	R	input pin 1 off
	1	R	input pin 1 on
i<address+2>	0	R	input pin 1 off
	1	R	input pin 1 on
i<address+3>	0	R	input pin 1 off
	1	R	input pin 1 on

MODLUX

Ambient light measurement module with sensor for ceiling mounting.

It uses one address of the Contatto bus.

ID	Value	R/W	Description
i<address>.1	0 ... 1023	R	ambient brightness value

MODMETEO

Meteorological data detection module for Contatto bus.

It uses one input address and, if enabled by the configuration panel of MCP IDE, one output address with the same value.

ID	Value	R/W	Description
i<address>.temp	<temp>	R	temperature value (in C/10)
i<address>.lux	<lux*10>	R	daylight value in tens of LUX
i<address>.wind	1 m/s / 10	R	wind speed value
i<address>.rain	0	R	no rain
	1	R	rain

i<address>.night	0	R	day
	1	R	night
i<address>.temp.limit	0	R	measured temperature < threshold
	1	R	measured temperature > threshold
i<address>.lux.limit	0	R	measured daylight < threshold
	1	R	measured daylight > threshold
i<address>.wind.limit	0	R	measured wind speed < threshold
	1	R	measured wind speed > threshold
i<address>.light.south	0	R	light not from South
	1	R	light from South
i<address>.light.west	0	R	light not from West
	1	R	light from West
i<address>.light.east	0	R	light not from East
	1	R	light from East
i<address>.fault	0	R	sensor works correctly
	1	R	lsensor failure
o<address>.temp	<temp>	RW	temperature threshold (in C/10)
	off	RW	temperature threshold off
o<address>.lux	<lux*10>	RW	daylight threshold
	off	RW	daylight threshold off
o<address>.wind	1 m/s / 10	RW	wind speed threshold
	off	RW	wind speed threshold off

MOD8R

8 power relay output module.

It uses one output address of the Contatto bus.

ID	Value	R/W	Description
o<address>.1	0	RW	output pin 1 off

	1	RW	output pin 1 on
o<address>.2	0	RW	output pin 2 off
	1	RW	output pin 2 on
o<address>.3	0	RW	output pin 3 off
	1	RW	output pin 3 on
o<address>.4	0	RW	output pin 4 off
	1	RW	output pin 4 on
o<address>.5	0	RW	output pin 5 off
	1	RW	output pin 5 on
o<address>.6	0	RW	output pin 6 off
	1	RW	output pin 6 on
o<address>.7	0	RW	output pin 7 off
	1	RW	output pin 7 on
o<address>.8	0	RW	output pin 8 off
	1	RW	output pin 8 on

You can also set the value to “on” or “off”, that is equivalent to 1 and 0.

MODPNP

8 positive logic (PNP) “open collector” digital output module.

It uses one output address in the range 1 to 127 of the Contatto bus.

ID	Value	R/W	Description
o<address>.1	0	RW	output pin 1 off
	1	RW	output pin 1 on
o<address>.2	0	RW	output pin 2 off
	1	RW	output pin 2 on
o<address>.3	0	RW	output pin 3 off
	1	RW	output pin 3 on
o<address>.4	0	RW	output pin 4 off
	1	RW	output pin 4 on
o<address>.5	0	RW	output pin 5 off

	1	RW	output pin 5 on
o<address>.6	0	RW	output pin 6 off
	1	RW	output pin 6 on
o<address>.7	0	RW	output pin 7 off
	1	RW	output pin 7 on
o<address>.8	0	RW	output pin 8 off
	1	RW	output pin 8 on

You can also set the value to “on” or “off”, that is equivalent to 1 and 0.

MODREL

8 relay output compact module in DIN 3M housing.

It uses one output address in the range 1 to 127 of the Contatto bus.

ID	Value	R/W	Description
o<address>.1	0	RW	output pin 1 off
	1	RW	output pin 1 on
o<address>.2	0	RW	output pin 2 off
	1	RW	output pin 2 on
o<address>.3	0	RW	output pin 3 off
	1	RW	output pin 3 on
o<address>.4	0	RW	output pin 4 off
	1	RW	output pin 4 on
o<address>.5	0	RW	output pin 5 off
	1	RW	output pin 5 on
o<address>.6	0	RW	output pin 6 off
	1	RW	output pin 6 on
o<address>.7	0	RW	output pin 7 off
	1	RW	output pin 7 on
o<address>.8	0	RW	output pin 8 off
	1	RW	output pin 8 on

You can also set the value to “on” or “off”, that is equivalent to 1 and 0.

MOAN/U

0÷10V analog output module.

It uses one output address in the range 1 to 127 of the Contatto bus.

ID	Value	R/W	Description
o<address>	x/255	R	value in fractional format, where 0 ≤ x ≤ 255
o<address>	<x%>	W	the percentage of signal (0÷100%) applied to the output
	0...255	W	convert the 8 bits digital code received through the bus in the proper analog level.
	x/y	W	fractional format, formatted as “x/y”, where 0 ≤ x ≤ y

---

MO420

4÷20mA analog output module.

It uses one output address in the range 1 to 127 of the Contatto bus.

ID	Value	R/W	Description
o<address>	<x%>	RW	the percentage of signal (0÷100%) applied to the output
	0...255	RW	convert the 8 bits digital code received through the bus in the proper analog level
	x/y	RW	fractional format, formatted as “x/y”, where 0 ≤ x ≤ y

---

MODLC

Ambient light regulator module with integrated sensor.

It uses one input address and, if enabled by the configuration panel of MCP IDE, one output address with the same value.

ID	Value	R/W	Description
i<address>.lux	value equivalent to three times the brightness in lux detected by the sensor	R	the brightness value in lux detected by the MODLC
i<address>.level	calculated value for the automatic brightness regulation	R	the value to be sent to the dimmer in order to execute the automatic regulation
i<address>.mode	auto	R	automatic regulation

	man	R	manual regulation
i<address>.in	0	R	digital input OFF
	1	R	digital input ON
o<address>.setpoint	setpoint value	RW	set the setpoint for the automatic brightness regulation
o<address>.mode	auto	RW	set automatic regulation
	man	RW	set manual regulation
o<address>.in.reset	0	RW	not active
	1	R	force the expiring of the deactivation delay time of the digital input
o<address>.mode.reset	0	RW	not active
	1	R	disable the automatic regulation and it reset the input channel

---

MOD4-4S

4 digital input and 4 power relay output multifunction module.

HSYCO supports two configurations only: four independent outputs, or two shutters.

When set as four independent outputs, the module must be configured with the same base address for input and output.

For example:

```
(I20, O20)
```

The input address must be enabled. Configuration switches must be set as SW1=OFF, SW2=OFF and SW5=ON.

When set as two shutters, the module must be configured with the same base address for input and output.

For example:

```
(I20, O20, O21)
```

The input address must be enabled. Configuration switches must be set as SW1=ON, SW2=ON and SW5=ON.

ID	Value	R/W	Description
i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on

i<address>.3	0	R	input pin 3 off
	1	R	input pin 3 on
i<address>.4	0	R	input pin 4 off
	1	R	input pin 4 on
o<address>.1	up	RW	shutter up command
	down	RW	shutter down command
	stop	RW	shutter stop command
	unknown	R	unknown state
	offup	RW	shutter off, up position
	offdown	RW	shutter off, down position
o<address+1>.1	up	RW	shutter up command
	down	RW	shutter down command
	stop	RW	shutter stop command
	unknown	R	unknown state
	offup	RW	shutter off, up position
	offdown	RW	shutter off, down position
o<address>.1	0	RW	output pin 1 off
	1	RW	output pin 1 on
o<address>.2	0	RW	output pin 2 off
	1	RW	output pin 2 on
o<address>.3	0	RW	output pin 3 off
	1	RW	output pin 3 on
o<address>.4	0	RW	output pin 4 off
	1	RW	output pin 4 on

You can also set the value to “on” or “off”, that is equivalent to 1 and 0.

MOD2-2R

2 digital input and 2 power relay output module.

It uses one input address and one output address in the range 1 to 127 of the Contatto bus.

The input and the output addresses may be the same or they can be different each one to the other.

ID	Value	R/W	Description
i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
o<address>.1	0	RW	output pin 1 off
	1	RW	output pin 1 on
o<address>.2	0	RW	output pin 2 off
	1	RW	output pin 2 on

You can also set the value to “on” or “off”, that is equivalent to 1 and 0.

MOD4-4R

4 digital input and 4 digital output module.

It uses one input address and one output address in the range 1 to 127 of the Contatto bus.

The input and the output addresses may be the same or they can be different each one to the other.

ID	Value	R/W	Description
i<address>.1	0<	R	input pin 1 off
	1	R	input pin 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address>.3	0	R	input pin 3 off
	1	R	input pin 3 on
i<address>.4	0	R	input pin 4 off
	1	R	input pin 4 on
o<address>.1	0	RW	output pin 1 off
	1	RW	output pin 1 on
o<address>.2	0	RW	output pin 2 off
	1	RW	output pin 2 on
o<address>.3	0	RW	output pin 3 off
	1	RW	output pin 3 on

o<address>.4	0	RW	output pin 4 off
	1	RW	output pin 4 on

MOD2DM

2x300W dimmer module.

It uses 2 output addresses (one for each dimmer channel) and 2 input addresses.

HSYCO supports this module only when its dip-switches are set as SW1=ON, SW2=ON, SW3=ON, SW4=ON.

ID	Value	R/W	Description
i<address>	0...100	R	brightness level according to current ramp
	off	R	dimmer off
i<address+1>	0...100	R	brightness level according to current ramp
	off	R	dimmer off
o<address>	0...100	R	brightness level according to current ramp
	off	R	dimmer off
o<address+1>	0...100	R	brightness level according to current ramp
	off	R	dimmer off

The dimmer level can be set using additional formats besides the standard percent values:

- absolute positive integer number between 0 and 100
- percent number, formatted as x%
- fractional format, formatted as “n/m”, where 0 <= n <= m.

MOD2DV

Dual 1-10V output for electronic ballast driving.

It uses 2 output addresses (one for each dimmer channel) and 2 input addresses.

HSYCO supports this module only when its dip-switches are set as SW1=ON, SW2=ON, SW3=ON, SW4=ON.

ID	Value	R/W	Description
i<address>	0...100	R	brightness level according to current ramp
	off	R	dimmer off

i<address+1>	0...100	R	brightness level according to current ramp
	off	R	dimmer off
o<address>	0...100	R	brightness level according to current ramp
	off	R	dimmer off
o<address+1>	0...100	R	brightness level according to current ramp
	off	R	dimmer off

The dimmer level can be set using additional formats besides the standard percent values:

- absolute positive integer number between 0 and 100
- percent number, formatted as x%
- fractional format, formatted as “n/m”, where 0 <= n <= m.

MOD8IL

8 digital input – 8 LED output module for wall box.

It uses 1 input address and 1 output address.

ID	Value	R/W	Description
i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address>.3	0	R	input pin 3 off
	1	R	input pin 3 on
i<address>.4	0	R	input pin 4 off
	1	R	input pin 4 on
i<address>.5	0	R	input pin 5 off
	1	R	input pin 5 on
i<address>.6	0	R	input pin 6 off
	1	R	input pin 6 on
i<address>.7	0	R	input pin 7 off
	1	R	input pin 7 on
i<address>.8	0	R	input pin 8 off

	1	R	input pin 8 on
o<address>.1	0	R	output pin 1 off
	1	R	output pin 1 on
o<address>.2	0	R	output pin 2 off
	1	R	output pin 2 on
o<address>.3	0	R	output pin 3 off
	1	R	output pin 3 on
o<address>.4	0	R	output pin 4 off
	1	R	output pin 4 on
o<address>.5	0	R	output pin 5 off
	1	R	output pin 5 on
o<address>.6	0	R	output pin 6 off
	1	R	output pin 6 on
o<address>.7	0	R	output pin 7 off
	1	R	output pin 7 on
o<address>.8	0	R	output pin 8 off
	1	R	output pin 8 on

You can also set the value to “on” or “off”, that is equivalent to 1 and 0.

MOD4DV

Quad 0-10V output module for generic applications or for external dimmers control.

It uses 4 consecutive output addresses (one for each dimmer channel) and 4 input addresses.

HSYCO supports this module only when its dip-switches are set as SW1=ON, SW2=ON, SW3=ON, SW4=ON, SW5=OFF.

ID	Value	R/W	Description
i<address>	0...100	R	brightness level according to current ramp
	off	R	dimmer off
i<address+1>	0...100	R	brightness level according to current ramp
	off	R	dimmer off
i<address+2>	0...100	R	brightness level according to current ramp

	off	R	dimmer off
i<address+3>	0...100	R	brightness level according to current ramp
	off	R	dimmer off
o<address>	0...100	R	brightness level according to current ramp
	off	R	dimmer off
o<address+1>	0...100	R	brightness level according to current ramp
	off	R	dimmer off
o<address+2>	0...100	R	brightness level according to current ramp
	off	R	dimmer off
o<address+3>	0...100	R	brightness level according to current ramp
	off	R	dimmer off

The dimmer level can be set using additional formats besides the standard percent values:

- absolute positive integer number between 0 and 100
- percent number, formatted as x%
- fractional format, formatted as “x/y”, where 0 <= x <= y
- values in the range 101÷255. Values and commands are described in the MOD4DV Programming Handbook (Operation from the bus, page 3).

MODDMX

ModDMX module allows to handle, through the Contatto bus, up to 32 DMX devices. ModDMX module makes possible the communication on the first 64 of 512 DMX channels allowed by this protocol.

It uses 1 output address, 1-channel 16-bit.

ID	Value	R/W	Description
o<address>.<DMX address>	0...100	RW	Values and commands are described in the MODDMX Programming Handbook

---

MODDALI

ModDALI module allows the management, through the Contatto bus, of 4 DALI lines upon it's possible to connect up to 32 devices for line (or channel). In this way it's possible to control and regulate up to 128 DALI ballasts (or other similar devices).

It uses 1 output address and, if enabled, 1 input address with the same value.

ID	Value	R/W	Description
o<address>.<channel>	refresh	RW	update the state of the DALI devices
o<address>.<channel> o<address>.<channel>.all o<address>.<channel>.0	off	RW	switch off all available DALI devices
	<level>	RW	set a level for all available devices
o<address>.<channel>.group.<DALI group>	off	RW	switch off all DALI devices on that group
	<level>	RW	set a level for all DALI devices in that group
o<address>.<channel>.<DALI address>	off	RW	switch off a single DALI device
	<level>	RW	set a level for a single DALI device
	>100	RW	special functions: possible values are described in the MODDALI Programming Handbook
i<address>.<channel>.polling	off	RW	polling disabled
	on	RW	polling enabled
i<address>.<channel>.test	0	RW	test button not pressed
	1	RW	test button pressed
i<address>.<channel>.dali	nopower	RW	lamp failure
	open	RW	DALI line broken
	short	RW	DALI line short circuit
	on	RW	on DALI line
i<address>.<channel>.1	fault	RW	ballast 1 lamp failure
	unknown	R	ballast 1 lamp unknown state

The channel level can be set using additional formats besides the standard percent values:

- absolute positive integer number between 0 and 100
- percent number, formatted as x%
- fractional format, formatted as “x/y”, where 0 <= x <= y
- values in the range 101÷255. Values and commands are described in the MODDALI Programming Handbook.

CLIMA2

Module for the regulation of the ambient temperature.

It uses 1 input and 1 output address that are equal each one to the other, so only a base address is needed.

ID	Value	R/W	Description
i<address>	on	R	zone on
	off	R	zone off
i<address>.mode	summer	R	summer mode
	winter	R	winter mode
i<address>.status	off	R	off status
	heating	R	heating request
	cooling	R	cooling request
i<address>.fan	off	R	fan off
	min	R	min speed
	med	R	med speed
	max	R	max speed
i<address>.temp	fault	R	NTC probe fault
	<temp>	R	ambient temperature value, in C/10
i<address>.setpoint.temp	<temp>	R	real setpoint value, in C/10
i<address>.knob	0 ... 1000	R	position of rotary knob
o<address>.mode	summer	RW	set summer
	winter	RW	set winter
o<address>	on	RW	zone on
	off	RW	zone off
o<address>.setpoint.temp	0...400	RW	set and read the value of central setpoint (temp in C/10)
o<address>.setpoint.delta.neg	<temp>	RW	set and read the MAX negative delta
o<address>.setpoint.delta.pos	<temp>	RW	set and read the MAX positive delta
o<address>.led.1	red r	RW	LED 1 red
	green g	RW	LED 1 green
	yellow y	RW	LED 1 yellow

	off	RW	set off
o<address>.led.2	red r	RW	LED 2 red
	green g	RW	LED 2 green
	yellow y	RW	LED 2 yellow
	off	RW	set off
o<address>.led.3	red r	RW	LED 3 red
	green g	RW	LED 3 green
	yellow y	RW	LED 3 yellow
	off	RW	set off
o<address>.led.4	red r	RW	LED 4 red
	green g	RW	LED 4 green
	yellow y	RW	LED 4 yellow
	off	RW	set off

You can also set the value to “on” or “off”, that is equivalent to 1 and 0.

MODANA

Network analyzer module for Contatto bus.

It uses, in dynamic mode, up to a maximum of 5 consecutive input addresses and, if enabled, 1 output address equal to the base address.

💡 **The module must be configured with the same base address for input and output.**

For example:

```
(I7, O7)
```

ID	Value	R/W	Description
i<address>.v12	[V]	R	chained voltage phase 1-2
i<address>.v23	[V]	R	chained voltage phase 2-3
i<address>.v31	[V]	R	chained voltage phase 3-1

i<address>.vtm	[V]	R	average chained voltage
i<address>.i1	[A]	R	current phase 1
i<address>.i2	[A]	R	current phase 2
i<address>.i3	[A]	R	current phase 3
i<address>.itm	[A]	R	average current
i<address>.ptot	[W]	R	total active power
i<address>.ptotk	[kW]	R	total active power
i<address>.qtot	[W]	R	total reactive power
i<address>.qtotk	[kW]	R	total reactive power
i<address>.pf	[pf]	R	total power factor
i<address>.frequency	[Hz]	R	frequency
i<address>.v1n	[V]	R	voltage phase 1
i<address>.v2n	[V]	R	voltage phase 2
i<address>.v3n	[V]	R	voltage phase 3
i<address>.p1	[W]	R	active power phase 1
i<address>.p1k	[kW]	R	active power phase 1
i<address>.p2	[W]	R	active power phase 2
i<address>.p2k	[kW]	R	active power phase 2
i<address>.p3	[W]	R	active power phase 3
i<address>.p3k	[kW]	R	active power phase 3
i<address>.q1	[W]	R	reactive power phase 1
i<address>.q1k	[kW]	R	reactive power phase 1
i<address>.q2	[W]	R	reactive power phase 2
i<address>.q2k	[kW]	R	reactive power phase 2
i<address>.q3	[W]	R	reactive power phase 3
i<address>.q3k	[kW]	R	reactive power phase 3
i<address>.pf1	[pf]	R	power factor phase 1
i<address>.pf2	[pf]	R	power factor phase 2
i<address>.pf3	[pf]	R	power factor phase 3
i<address>.s1	[VA]	R	apparent power phase 1

i<address>.s1k	[kVA]	R	apparent power phase 1
i<address>.s2	[VA]	R	apparent power phase 2
i<address>.s2k	[kVA]	R	apparent power phase 2
i<address>.s3	[VA]	R	apparent power phase 3
i<address>.s3k	[kVA]	R	apparent power phase 3
i<address>.stot	[VA]	R	total apparent power
i<address>.hours	[hours]	R	hour-meter
i<address>.temperature	[°C]	R	cabinet temperature
i<address>.energy.active	[Wh]	R	positive active energy
i<address>.energy.activeneg	[Wh]	R	negative active energy
i<address>.energy.reactive	[VARh]	R	positive reactive energy
i<address>.energy.reactiveneg	[VARh]	R	negative reactive energy
i<address>.pm	[W]	R	average positive active power
i<address>.qm	[VAR]	R	average positive reactive power
i<address>.reset.energy	1	RW	reset energies
	0	RW	---
i<address>.reset.hours	1	RW	reset hour-meter
	0	RW	---

MODPQ5

Q5 tags proximity reader and programmer module.

Uses 1 input address on the Contatto bus. Write operations are performed through direct writes to the module's RAM.

ID	Value	R/W	Description
i<address>.data	0 <tag data>	R	30 characters hexadecimal representation of the 15 bytes of tag data; "0" if the module is not seeing any tag
i<address>.data.write	0	R	the tag has been removed from the programmer after writing, or the module is not seeing any tag
	1	R	the tag write operation was successful
i<address>.reader.error	0	R	reader/writer normal status

	1	R	reader/writer module error (normally occurs when the reader is not connected to the MODPQ5 module)
i<address>.tag.error	0	R	the tag is valid
	1	R	the tag is not valid
i<address>.request.error	0	R	no request or handshake errors detected
	1	R	handshake error
i<address>.data.error	0	R	tag data ok
	1	R	tag data errors detected
i<address>.unknown.error	0	R	no unknown errors
	1	R	unknown error detected
i<address>.write	<tag data>	R	tag data write successful
	error	R	tag data write error
o<address>.write	<tag data>	W	30 characters hexadecimal representation of the 15 bytes data to be written on the tag

---

MODHT

MODHT is the hotel room controller module.

It uses 1 input and 1 output address.

Tags configuration data are written to the module's EEPROM (persistent internal memory).

ID	Value	R/W	Description
i<address>.door	0	R	door closed
	1	R	door open
i<address>.window	0	R	window closed
	1	R	window open
i<address>.panic	0	R	panic request not active
	1	R	panic request
i<address>.busy	0	R	room is free
	1	R	room is occupied
i<address>.1	0	R	input pin 1 off

	1	R	input pin 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address>.3	0	R	input pin 3 off
	1	R	input pin 3 on
i<address>.ev	0	R	EV output off
	1	R	EV output on
i<address>.aux	0	R	AUX output off
	1	R	AUX output on
i<address>.dnd	0	R	do not disturb not active
	1	R	do not disturb
i<address>.service	0	R	room service request not active
	1	R	room service request
i<address>.booked	0	R	room is not booked
	1	R	room is booked
i<address>.mode	summer	R	summer mode
	winter	R	winter mode
i<address>.fan	off	R	fan set to off
	auto	R	fan speed set to auto
	min	R	fan speed set to minimum
	med	R	fan speed set to medium
	max	R	fan speed set to max
i<address>.fan.status	0	R	fan is off
	1	R	fan is on (cooling or heating)
i<address>.temp	<temp>	R	ambient temperature value (in C/10)
i<address>.setpoint.temp	<temp>	R	setpoint value (in C/10)
i<address>.tag.door	1...5	R	type of tag that is opening the door (1 to 4 are service tags; 5 is a customer tag). After 5 seconds the value is reset to 0
i<address>.tag.room	0	R	no tag present in room's reader

	1...5	R	type of tag inserted in the room's reader (1 to 4 are service tags; 5 is a customer tag)
i<address>.tag.<n>	<tag data>	R	24 characters hexadecimal representation of the 12 bytes data matching service N tags (N from 1 to 5) stored in module's EEPROM
i<address>.pcam.error	0	R	PCAM module ok
	1	R	PCAM module error
i<address>.tpr.error	0	R	TPR/H module ok
	1	R	TPR/H module error
i<address>.mask	<mask data>	R	24 characters hexadecimal representation of the 12 bytes tag mask stored in module's EEPROM
i<address>.data.door	0 <tag data>	R	30 characters hexadecimal representation of the 15 bytes of tag data being read from the door reader; "0" if the module is not seeing any tag (this data point is available with MODHT firmware 5.3 or newer)
i<address>.data.room	0 <tag data>	R	30 characters hexadecimal representation of the 15 bytes of tag data being read from the room reader; "0" if the module is not seeing any tag (this data point is available with MODHT firmware 5.3 or newer)
i<address>.setpoint.summer.<n> i<address>.setpoint.winter.<n>	<temp>	R	winter/summer setpoint <n> (1...3) current value (in C/10)
i<address>.setpoint.delta.summer.low i<address>.setpoint.delta.summer.high i<address>.setpoint.delta.winter.low i<address>.setpoint.delta.winter.high	<temp delta limit> 1...9	R	the current temperature setpoint low/high limits for winter and summer modes (in C)
o<address>.tag.<n>	<tag data>	R	tag data write successful
	error	R	tag data write error
o<address>.mask	<mask data>	R	mask data write successful
	error	R	mask data write error

o<address>.setpoint.summer.<n> o<address>.setpoint.winter.<n>	<temp>	R	winter/summer setpoint N (1-3) write successful (in C/10)
	error	R	setpoint write error
o<address>.setpoint.delta.summer.low o<address>.setpoint.delta.summer.high o<address>.setpoint.delta.winter.low o<address>.setpoint.delta.winter.high	<temp delta limit> 1...9	R	setpoint min/max limit for winter/summer write successful (in C)
	error	R	setpoint limit write error
o<address>.ev.enable	0	W	EV output disabled
	1	W	EV output enabled
o<address>.aux.enable	0	W	AUX output disabled
	1	W	AUX output enabled
o<address>.panic.reset	1	W	panic request reset
o<address>.busy	0	W	set room status to free
	1	W	set room status to occupied
o<address>.ev	0	W	EV output off
	1	W	EV output on
o<address>.aux	0	W	AUX output off
	1	W	AUX output on
o<address>.dnd.reset	1	W	do not disturb reset
o<address>.service.reset	1	W	do not disturb reset
o<address>.booked	0	W	set room as not booked
	1	W	set room as booked
o<address>.mode	summer	W	set room to summer mode
	winter	W	set room to winter mode
o<address>.fan	off	W	set fan off
	auto	W	set fan to auto mode
o<address>.comfort	0	W	output 3 off
	1	W	output 3 on
o<address>.fan.heating.min	0	W	reset heating speed limit
	1	W	limit heating speed to min

o<address>.fan.cooling.min	0	W	reset cooling speed limit
	1	W	limit cooling speed to min
o<address>.tag.<n>	read	W	read the 12 bytes stored in module's EEPROM for service N tags (<n> from 1 to 5)
	<tag data>	W	24 characters hexadecimal representation of the 12 bytes data matching service N tags (N from 1 to 5), to be stored in module's EEPROM. After writing data, you should wait approximately 150ms before sending a read command
o<address>.mask	read	W	read the 12 bytes mask stored in module's EEPROM
	<mask data>	W	24 characters hexadecimal representation of the 12 bytes tag mask, to be stored in module's EEPROM. After writing data, you should wait approximately 150ms before sending a read command
o<address>.setpoint.summer.<n> o<address>.setpoint.winter.<n>	read	W	read winter/summer setpoint <n> (1-3)
	<temp>	W	set winter/summer setpoint <n> (1-3)
o<address>.setpoint.delta	read	W	read winter and summer, low and high setpoint limits
o<address>.setpoint.delta.summer.low o<address>.setpoint.delta.summer.high o<address>.setpoint.delta.winter.low o<address>.setpoint.delta.winter.high	<temp delta limit>	W	set winter/summer, low/high setpoint limit (in C)

MODCA

MODCA is the access control module.

It uses 1 input and 1 output address.

Tags configuration data are written to the module's EEPROM (persistent external memory).

Site codes and the tags mask are written to the internal EEPROM.

ID	Value	R/W	Description

i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address>.blocked	0	R	access control enabled
	1	R	access blocked
i<address>.tpr.error	0	R	TPR module ok
	1	R	TPR module error
i<address>.access.granted.1	0	R	no tag present at antenna 1
	1	R	access granted to tag at antenna 1
i<address>.access.granted.2	0	R	no tag present at antenna 2
	1	R	access granted to tag at antenna 2
i<address>.access.denied	0	R	access denied flags reset
	site	R	access denied due to site code error
	code	R	access denied due to user code error
	total.limit	R	access denied due to exceeded total number of access grants
	daily.limit	R	access denied due to exceeded daily number of access grants
	time	R	access denied due to time range violation
	day	R	access denied due to weekday violation
	expired	R	access denied due expired tag
i<address>.denied.code	<tag code>	R	the tag code (decimal value of selected two bytes of the tag data) that was denied access
i<address>.granted.1.code	<tag code>	R	the tag code (decimal value of selected two bytes of the tag data) that was granted access at antenna 1
i<address>.granted.2.code	<tag code>	R	the tag code (decimal value of selected two bytes of the tag

			data) that was granted access at antenna 2
i<address>.site.<n>	<site data>	R	16 characters hexadecimal representation of the 8 bytes data representing the Nth site code (N from 1 to 4) stored in module's EEPROM
i<address>.mask	<mask data>	R	16 characters hexadecimal representation of the 8 bytes site code mask stored in module's EEPROM
i<address>.mask	<mask data>	R	16 characters hexadecimal representation of the 8 bytes site code mask stored in module's EEPROM
i<address>.user.<n>	<user data>	R	32 characters hexadecimal representation of the 16 bytes data for user <n> (N from 1 to 2000) stored in module's EEPROM
o<address>.site.<n>	<site data>	R	site data write successful
	error	R	site data write error
o<address>.mask	<mask data>	R	mask data write successful
	error	R	mask data write error
o<address>.user.<n>	<user data>	R	user data write successful
	error	R	user data write error
o<address>.relay	0	W	relay output off
	1	W	relay output on (with firmware versions before 5.3 this is a pulse data point, and resets to 0 immediately after the relay is closed, usually before the relay closure time)
o<address>.block	0	W	enable access control
	1	W	block access block
o<address>.access.granted.code	0	W	reset
	1	W	allow access even when site code is not valid
o<address>.access.granted.total.limit	0	W	reset
	1	W	allow access even when the total limit of access grants has been exceeded

o<address>.access.granted.daily.limit	0	W	reset
	1	W	allow access even when the daily limit of access grants has been exceeded
o<address>.access.granted.time	0	W	reset
	1	W	allow access even outside of the allowed timeframe
o<address>.access.granted.day	0	W	reset
	1	W	allow access even outside of the allowed weekdays
o<address>.access.granted.expired	0	W	reset
	1	W	allow access even when the tag is expired
o<address>.site.<n>	read	W	read the 8 bytes stored in module's EEPROM for site N code (<n> from 1 to 4)
	<site data>	W	16 characters hexadecimal representation of the 8 bytes data of site N code (N from 1 to 4), to be stored in module's EEPROM. After writing data, you should wait approximately 100ms before sending a read command
o<address>.mask	read	W	read the 8 bytes mask stored in module's EEPROM
	<mask data>	W	16 characters hexadecimal representation of the 8 bytes site code mask, to be stored in module's EEPROM. After writing data, you should wait approximately 100ms before sending a read command
o<address>.user.<n>	read	W	read the 16 bytes stored in module's EEPROM for user N (<n> from 1 to 2000)
	<user data>	W	16 characters hexadecimal representation of the first 8 bytes of access control data for user N (N from 1 to 2000) , to be stored in module's EEPROM
	<reset>	W	reset to 0 the total and daily counters for user <n>
o<address>.users	erase	W	deletes all users data. This action requires up to about 27 seconds to be completed. It also

			deletes all i<address>.user.<n> and o<address>.user.<n> data points

---

MODKB

MODKB is the keypad access control module.

It uses 1 input and 1 output address.

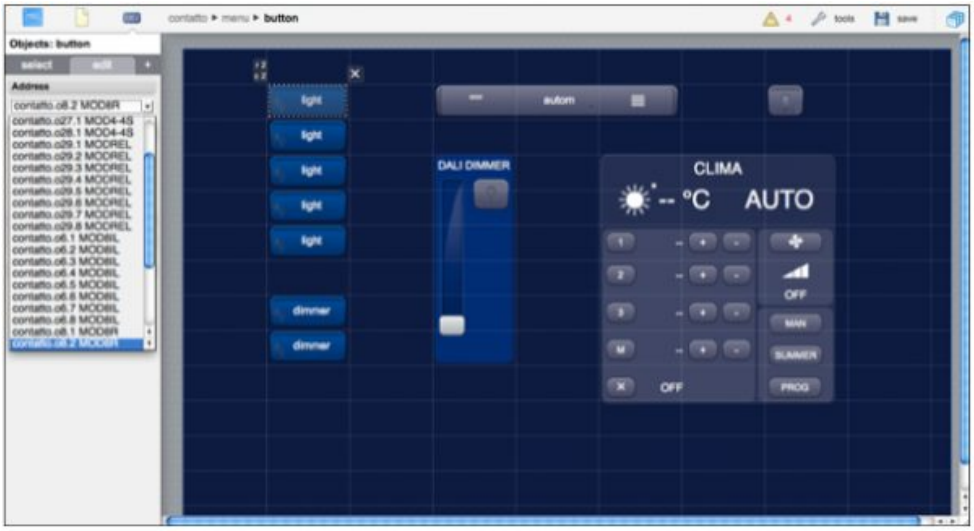
PIN configuration data are written to the module's EEPROM (persistent memory).

ID	Value	R/W	Description
i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address>.access.granted.<n>	0	R	reset
	1	R	access granted to PIN <n> (<n> from 1 to 30)
i<address>.access.pin	<pin code>	R	the last pin code entered on the keypad (from 1 to 65535). Reset to 0 after programmed code persistence time
i<address>.access.granted.<n>	0	R	access denied flags reset
	site	R	access denied due to site code error
	code	R	access denied due to user code error
	time	R	access denied due to time range violation
	day	R	access denied due to weekday violation
i<address>.pin.<n>	<pin data>	R	16 characters hexadecimal representation of the 8 bytes data for PIN <n> (<n> from 1 to 30) stored in module's EEPROM
o<address>.pin.<n>	<pin data>	R	pin data write successful
	error	R	pin data write error
o<address>.relay	0	W	relay output off

	1	W	relay output on (pulse output, resets to 0 immediately after the relay is closed, usually before the relay closure time)
o<address>.lock.all	0	W	reset
	1	W	disables access to all PINs
o<address>.lock.<n>	0	W	reset
	1	W	disables access to PIN <n> (<n> from 1 to 30)
o<address>.pin.<n>	read	W	read the 8 bytes stored in module's EEPROM for PIN <n> (<n> from 1 to 30)
	<pin data>	W	16 characters hexadecimal representation of the 8 bytes data for PIN <n> (<n> from 1 to 30), to be stored in module's EEPROM

User Interface

All CONTATTO devices data points that have been defined in the systemtopo.txt database are automatically listed in the Project Editor. Adding a button to control a device output point requires just a few clicks and no additional EVENTS logic.



Besides the direct association of control buttons and data points, the CONTATTO driver also automatically updates graphical objects that represent values or states of complex devices, like the CLIMA2 temperature control unit. It will also automatically intercept buttons to manually set operation modes and temperature set-points.

CLIMA2

You can use the temp and tempmini objects to control CLIMA2 devices.

The fan button, in the bottom right-hand corner, is not enabled because the manual control of the fan speed is not available. The M button, in the top right-hand corner, allows to switch on/off the module.

You can also easily create customized controls for the CLIMA2, using standard graphic objects that are automatically set to show the device's status and intercepted to send user commands.

UISET Actions

ID	Attribute	Set to
<address>.mode	summer	summer mode (cooling)
	winter	winter mode (heating)
<address>.mode.label.summer	visible	if the CLIMA2 is in summer mode
<address>.mode.label.winter	visible	if the DFCT is in winter mode
<address>.status	off	zone off
	on	zone on
<address>.status.label.off	visible	if the zone is not cooling or heating
<address>.status.label.cooling	visible	if the zone is cooling
<address>.status.label.heating	visible	if the zone is heating
<address>.fan	off	fan off
	min	minimum fan speed
	med	medium fan speed
	max	maximum fan speed
<address>.fan.label.min	visible	the fan speed is min
<address>.fan.label.med	visible	the fan speed is med
<address>.fan.label.max	visible	the fan speed is max
<address>.setpoint	1,2,3	active setpoint

	man	manual setpoint
	off	zone off
<address>.setpoint.label.man	visible	manual setpoint is active
<address>.setpoint.label.off	visible	zone is off
<address>.setpoint.mode	on	if zone is on
	off	if zone is off
<address>.setpoint.temp	<value>	the active setpoint temperature, in Celsius degrees with one decimal digit, followed by " °C"
<address>.temp	<temp>	the measured temperature, in Celsius degrees with one decimal digit, followed by " °C"
	fault	fault/error condition

USER Commands

Name	Param	Action
<address>	on	module on
	off	module off
	mode.summer	set summer mode (cooling)
	mode.winter	set winter mode (heating)
<address>.setpoint	temp.up	increase setpoint temperature in 0.5C steps
	temp.down	decrease setpoint temperature in 0.5C steps
	0...400	set the default setpoint (wrong) to the defined value (in C/10)
	mode	cycle through on and off modes
	mode.on	module on
	mode.off	module off

Release Notes

3.2.2

bug fixes:

- the toolspassword optional parameter was incorrectly converted to lower case
- fixed a bug that prevented reading the last 3 addresses of MOAN/I4 and MI420-X4 modules

3.2.1

- added support for MODHT set-point delta limits read and write
- the lux datapoint name was incorrectly set to ID.i<addr>.1 instead of ID.i<addr>.lux

3.2.0

- support for MODCA, MODHT, MODKB, MODPQ5
- new "detectevents" option, generates forced events when a device is detected at start-up
- support for the toolspassword option, to allow MCP-IDE and other tools secure remote access to the MCP via HSYCO
- new clock datapoint to read the MXP-XT internal clock and set it to HSYCO's time

3.0.3

- optimized performance of MCP's registers status polling
- the server failed to initialized when an unsupported device was configured in the MCP

3.0.1

- you can now limit the number of virtual points and registers read and written by HSYCO, and have access to all of the 1024 registers; defaults are unchanged

3.0.0

- initial release

HSYCO and Home Systems Consulting are registered trademarks of Home Systems Consulting SpA. Java and JavaScript are registered trademarks of Oracle and/or its affiliates. CONTATTO and DUEMMEGI are registered trademarks of DUEMMEGI SRL. Other products or company names can be trademarks or registered trademarks of other companies and are used for demonstrative purposes only, with no violation intent.

Domino

The DOMINO system is DUEMMEGI's proprietary bus architecture for home automation. HSYCO fully integrates with this system through the DFCP control and gateway module, and its FXP-XT serial communication protocol.

A serial connection between the DFCP and the HSYCO SERVER is required for the integration, either directly through the server's RS-232 port or via the RS-232 port of a supported serial to IP gateway, including the WEBS module.

Communication

Employ a DE-9 (often called DB-9) male-female RS-232 straight cable to connect the RS-232 DFCP port to the serial port on HSYCO SERVER or serial gateway.

RS-232 parameters:

Baud rate	9600, 38400, 57600 or 115200 bps (according to the DFCP serial port setting)
Data bits	8
Stop bit	1
Parity	none
Flow control	none

💡 **A speed of 115200 bps is recommended to achieve good performance, particularly when the number of devices connected to the bus is large.**

HSYCO Configuration

You can define and connect to more than one DOMINO bus using several DFCP gateways.

The DOMINO I/O Server also supports dual, redundant connections between HSYCO and DFCP.

If the main connection fails, HSYCO automatically re-routes the communication with DFCP through the fail-over connection.

Options

ID	Default	Values	Description
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
inputdiscovery	false	true	auto-detects DOMINO's input devices as configured in the DFCP, and automatically

			creates the list of all detected devices and individual data points in the systemtopo.txt file. Should be enabled to allow the automatic update of (button) objects' states
		false	auto-detect for input devices is disabled
outputdiscovery	true	true	auto-detects DOMINO's output devices as configured in the DFCP, and automatically creates the list of all detected devices and individual data points in the systemtopo.txt file. Should be enabled to allow the automatic update of (button) objects' states
		false	auto-detect for output devices is disabled
virtualdiscovery	false	true	auto-detects DOMINO's virtual points for input and output devices as configured in the DFCP, and automatically creates the list of all detected virtual data points in the systemtopo.txt file. Should be enabled to allow the automatic update of (button) objects' states
		false	auto-detect for devices' virtual data points is disabled
virtualpoints	false	1...2033	enables polling for the first n virtual points of the DFCP. Enable this option only if you need to generate I/O events based on these virtual points
		true	enables polling for all the 2032 virtual points of the DFCP
		false	polling of the DFCP virtual data points is disabled
registers	false	1...1024	enables polling for registers R0-Rn of the DFCP. Enable this option only if you need to generate I/O events based on these registers
		true	enables polling for registers R0-R127 of the DFCP
		false	polling of the DFCP registers is disabled

powerdisplay	false	true	enables polling for registers R0-Rn of the DFCP. Enable this option only if you need to generate I/O events based on these registers
		≥ 0 (DFANA address)	enables the automatic display in the GUI of the total real power measured by a DFANA module connected to this DFCP. Note that, if you have more than one DFCP gateway, you should enable this option for one gateway only
		false	power display disabled for this DFCP
powersensivity	50	≥ 0	measured power changes are reported only if the difference from last reading is equal or greater than the power sensivity option, in Watts
powerinterval	1	≥ 0	power polling interval for the DFCC meter, in seconds
detectevents	false	true	generate forced events when a device is detected at start-up
		false	do not generate events when a device is detected at start-up
toolspassword		string	set this option with a long string (only letters and numbers) password to allow the remote connection of DFCP-IDE and other configuration tools to the DFCP gateways that are connected to HSYCO. For additional security, it is recommended to set this option only when required
language	english	en it fr	language for all messages from the DOMINO system: English, Italian or French

Initialization and Connection Events

Event name	Value	Description
connection	online	connection established to the DFCP module
	offline	HSYCO can’t connect to the DFCP module
detected.input.<n>.	model name	an input device of the type passed as value has been detected at address <n>, using

		~~addresses~~
detected.output.<n>.	model name	an output device of the type passed as value has been detected at address <n>, using ~~addresses~~

The Device Configuration Database

The **systemtopo.txt** file contains the list of all devices and their individual input, output and virtual data points that could be directly associated to graphic object in the Web-based user interface. This file can be filled manually or automatically by HSYCO at start-up. To enable automatic discovery and automatic generation of devices' information in the systemtopo file, use the *inputdiscovery*, *outputdiscovery* and *virtualdiscovery* options in **Settings**. The default behavior is to only discover devices with output data points.

This is an example of an automatically generated systemtopo.txt file:

```
(devices)
domino.o10.1 : LIGHT ; LIGHT
domino.o6.1 : LIGHT ; DIMMER
domino.o7.2 : LIGHT ; LIGHT
domino.o7.1 : LIGHT ; DIMMER
domino.o9.2 : AUTOM ; VSHUT
domino.o5.4 : LIGHT ; LIGHT
domino.o9.1 : AUTOM ; VSHUT
domino.o5.3 : LIGHT ; LIGHT
domino.o5.2 : LIGHT ; LIGHT
domino.o5.1 : LIGHT ; LIGHT
```

You should then manually add comments and other optional parameters:

```
(devices)
domino.o10.1 : LIGHT ; LIGHT; main entrance light
domino.o6.1 : LIGHT ; DIMMER; lobby dimmer
domino.o7.2 : LIGHT ; LIGHT; kitchen workbench
domino.o7.1 : LIGHT ; DIMMER; kitchen main dimmer
```

DFCP Virtual Data Points and Registers

The DFCP implements 2032 virtual points for binary (on/off) data and 1024 registers for positive scalar values (0-65535).

All the virtual points and the registers can optionally generate I/O events. You can also write to the virtual data points and registers using the IO action in EVENTS or ioSet() method in Java.

You cannot directly control the DFCP virtual data points and registers using GUI objects.

To enable polling of the current state of the DFCP's virtual data points, enable the *virtualpoints* options in hsyco.ini.

To enable polling of the current state of the DFCP's registers, enable the *registers* options in hsyco.ini.

If you only have to write to virtual points or registers, enabling polling is not strictly required.

ID	Value	R/W	Description
v0.<n>	0	RW	virtual data point <n> off (<n>: 1-2032)

	1	RW	virtual data point <n> on (<n>: 1-2032)
r0.<n>	<x>	RW	register <n> off (<n>: 0-1023) set to value <x> (<x>: 0-65535)

DFCP Internal Clock

The DFCP module has an integrated real-time clock. You can use the clock datapoint to read the DFCP date and time, and set the DFCP clock to HSYCO's current time.

ID	Value	R/W	Description
clock	yyyy-mm-dd hh:mm:ss	R	the DFCP clock current time
	read	W	read the DFCP clock, and the delta with HSYCO's time
	sync	W	set the DFCP clock to HSYCO's current time
clock.delta	integer number	R	the delta time in seconds between the DFCP and HSYCO clocks. A positive number means that the DFCP clock is ahead of HSYCO

DOMINO Modules Events and Control

Module Code	Description
DF4I	4 digital inputs
DFCK3	clock
DF8I	2x4 digital inputs
DF2R	2 outputs (relay)
DF4R, DF4RP	4 outputs (power relay)
DF4RP/I	4 generic inputs ON-OFF + 4 outputs
DFDI, DFDI2, DFDI2B, DFDM, DFDT	1 output (dimmer)
DFDMX	DMX interface
DFDALI	DALI interface
DFDV	1 output (1-10V - ballast) + 1 output (power relay)
DFTP	2 outputs (shutters)
DFTP/I	2 outputs (shutters) + 4 digital inputs

DFTR	2 outputs (shutters) + 1 output (relay)
DF4IL	4 digital inputs + 4 outputs (LED)
DF8IL	8 digital inputs + 8 outputs (LED)
DFAI	2 analog inputs
DFLUX, DFSUN	light sensor
DFCT/DFTZ	temperature sensor
DFRHT	temperature and humidity sensor
DFMETEO	weather module
DFCC	electric energy meter and load manager

DF4I

The DF4I is an input module with 4 digital inputs.

It uses one address in the DOMINO bus.

ID	Value	R/W	Description
i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address>.3	0	R	input pin 3 off
	1	R	input pin 3 on
i<address>.4	0	R	input pin 4 off
	1	R	input pin 4 on

DFCK3

The DFCK3 is an input module for managing up to 15 different zones with an integrated clock.

It uses one address in the DOMINO bus.

ID	Value	R/W	Description
i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address>.2	0	R	input pin 2 off

	1	R	input pin 2 on
i<address>.3	0	R	input pin 3 off
	1	R	input pin 3 on
i<address>.4	0	R	input pin 4 off
	1	R	input pin 4 on
i<address>.5	0	R	input pin 5 off
	1	R	input pin 5 on
i<address>.6	0	R	input pin 6 off
	1	R	input pin 6 on
i<address>.7	0	R	input pin 7 off
	1	R	input pin 7 on
i<address>.8	0	R	input pin 8 off
	1	R	input pin 8 on
i<address>.9	0	R	input pin 9 off
	1	R	input pin 9 on
i<address>.10	0	R	input pin 10 off
	1	R	input pin 10 on
i<address>.11	0	R	input pin 11 off
	1	R	input pin 11 on
i<address>.12	0	R	input pin 12 off
	1	R	input pin 12 on
i<address>.13	0	R	input pin 13 off
	1	R	input pin 13 on
i<address>.14	0	R	input pin 14 off
	1	R	input pin 14 on
i<address>.15	0	R	input pin 15 off
	1	R	input pin 15 on

DF8I

The DF8I is an input module with 2x4 digital inputs.

It uses two consecutive addresses in the DOMINO bus.

ID	Value	R/W	Description
i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address>.3	0	R	input pin 3 off
	1	R	input pin 3 on
i<address>.4	0	R	input pin 4 off
	1	R	input pin 4 on
i<address+1>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address+1>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address+1>.3	0	R	input pin 3 off
	1	R	input pin 3 on
i<address+1>.4	0	R	input pin 4 off
	1	R	input pin 4 on

DF2R

The DF2R is an output module with 2 relay outputs.

It uses one address in the DOMINO bus.

ID	Value	R/W	Description
o<address>.1	0	RW	output pin 1 off
	1	RW	output pin 1 on
o<address>.2	0	RW	output pin 2 off
	1	RW	output pin 2 on

You can also set the value to “on” or “off”, that is equivalent to 1 and 0.

DF4R, DF4RP

The DF4R (DF4RP) is an output module with 4 relay outputs.

It uses one address in the DOMINO bus.

ID	Value	R/W	Description
o<address>.1	0	RW	output pin 1 off
	1	RW	output pin 1 on
o<address>.2	0	RW	output pin 2 off
	1	RW	output pin 2 on
o<address>.3	0	RW	output pin 3 off
	1	RW	output pin 3 on
o<address>.4	0	RW	output pin 4 off
	1	RW	output pin 4 on

DF4RP/I

The DF4RP/I is an output module with 4 generic inputs and 4 relay outputs.

It uses one input address and one output address in the DOMINO bus.

ID	Value	R/W	Description
i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
o<address>.1	0	RW	output pin 1 off
	1	RW	output pin 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
o<address>.2	0	RW	output pin 2 off
	1	RW	output pin 2 on
i<address>.3	0	R	input pin 3 off
	1	R	input pin 3 on
o<address>.3	0	RW	output pin 3 off
	1	RW	output pin 3 on
i<address>.4	0	R	input pin 4 off
	1	R	input pin 4 on
o<address>.4	0	RW	output pin 4 off

	1	RW	output pin 4 on

DFDM, DFDI, DFDI2, DFDI2B, DFDT

The DFDM, DFDI, DFDI2, DFDI2B and DFDT are output modules with 1 dimmer output.

They use one address in the DOMINO bus.

ID	Value	R/W	Description
o<address>	program	R	program mode
	fault	R	dimmer fault
	auto	R	auto mode
	man	R	manual mode
o<address>.1	off	RW	dimmer off
	1...100%	RW	percent dimmer level
	on	RW	dimmer on at last level
o<address>.ramp	save	RW	saves the current ramp as default
	1...30	RW	ramp value in seconds (1 sec steps up to 10 and 2 sec steps up to 30)
	40, 50, 60	RW	ramp value in seconds (10 sec steps)
o<address>.setpoint	0...1023	RW	setpoint level
o<address>.hysteresis	0...255	RW	tolerance level
o<address>.time	0...255	RW	period (seconds)

The dimmer level can be set using additional formats besides the standard percent values:

- absolute positive integer number between 0 and 100
- percent number, formatted as x%
- fractional format, formatted as “n/m”, where 0 <= n <= m.

DFDMX

The DFDMX is a DMX-512 protocol output module.

It uses one address in the DOMINO bus.

ID	Value	R/W	Description

o<address>.<command>	0...255	RW	commands and values according to the DFDMX programming manual

---

DFDALI

The DFDALI is a DALI interface module. Using the DFDALI you can control up to 32 DALI devices.

You can control individual devices, groups or broadcast commands to all devices.

It uses one input (if enabled) and one output address in the DOMINO bus.

ID	Value	R/W	Description
o<address>	refresh	RW	queries the DFDALI module to update the current status of all DALI devices
o<address>	off	RW	broadcast off command
o<address>.all o<address>.0	<level>	RW	broadcast level command
o<address>.group.<DALI group>	off	RW	broadcast off command
	<level>	RW	group level command
o<address>.<DALI address>	off	RW	turn off a single device
	<level>	RW	set a single device to a specific level
	> 100	RW	special functions, having values between 101 and 255, as defined in the DFDALI manual
i<address>.polling	off	RW	polling mode is disabled
	on	RW	polling mode is enabled
i<address>.test	0	RW	test button not pressed
	1	RW	test button pressed
i<address>.dali	nopower	RW	power failure on the DALI bus
	open	RW	DALI bus is open
	short	RW	DALI bus is shorted
	on	RW	DALI bus on
i<address>.1	fault	RW	DALI device 1 reporting a lamp failure
	unknown	R	DALI device 1 status unknown

The DALI level can be set using additional formats besides the standard percent values:

- absolute positive integer number between 0 and 100
- percent number, formatted as x%
- fractional format, formatted as “n/m”, where 0 <= n <= m
- special functions, having values between 101 and 255, as defined in the DFDALI manual.

DFDV

The DFDV is an output module with 1 output (1-10V for ballast) + 1 generic power relay output. It uses one address in the DOMINO bus.

ID	Value	R/W	Description
0<address>.1	off	RW	dimmer off
	1 ... 100%	RW	percent dimmer level
	on	RW	dimmer on at last level
0<address>.2	0	RW	output pin 1 off
	1	RW	output pin 1 on

The dimmer level can be set using additional formats besides the standard percent values:

- absolute positive integer number between 0 and 100
- fractional format, formatted as “n/m”, where 0 <= n <= m.

You can also set the value to “on” or “off”, that is equivalent to 1 and 0.

DFTP

The DFTP is an output module for 2 shutters.

It uses one output address in the DOMINO bus.

ID	Value	R/W	Description
0<address>.1	up	RW	shutter up command
	down	RW	shutter down command
	stop	RW	shutter stop command
	unknown	R	unknown state
	offup	RW	shutter off, up position
	offdown	RW	shutter off, down position
0<address>.2	up	RW	shutter up command
	down	RW	shutter down command
	stop	RW	shutter stop command

	unknown	R	unknown state
	offup	RW	shutter off, up position
	offdown	RW	shutter off, down position

DFTP/I

The DFTP is an input/output module for 2 shutters and 4 generic inputs.

It uses one input and one output address in the DOMINO bus.

ID	Value	R/W	Description
i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
o<address>.1	up	RW	shutter up command
	down	RW	shutter down command
	stop	RW	shutter stop command
	unknown	R	unknown state
	offup	RW	shutter off, up position
	offdown	RW	shutter off, down position
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
o<address>.2	up	RW	shutter up command
	down	RW	shutter down command
	stop	RW	shutter stop command
	unknown	R	unknown state
	offup	RW	shutter off, up position
	offdown	RW	shutter off, down position
i<address>.3	0	R	input pin 3 off
	1	R	input pin 3 on
i<address>.4	0	R	input pin 4 off
	1	R	input pin 4 on

DFTR

The DFTR is an output module with 1 shutter output and 1 relay output.

It uses one address in the DOMINO bus.

ID	Value	R/W	Description
o<address>.1	up	R	shutter up
	down	R	shutter down
	unknown	R	unknown state
	offup	R	shutter offup
	offdown	R	shutter offdown
o<address>.2	0	R	output pin 2 off
	1	R	output pin 2 on

DF4IL

The DF4IL is an input/output module with 4 digital inputs and 4 digital open collector outputs for LEDs.

It uses one input and one output address in the DOMINO bus.

ID	Value	R/W	Description
i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
o<address>.1	0	RW	LED 1 off
	1	RW	LED 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
o<address>.2	0	RW	LED 2 off
	1	RW	LED 2 on
i<address>.3	0	R	input pin 3 off
	1	R	input pin 3 on
o<address>.3	0	RW	LED 3 off
	1	RW	LED 3 on
i<address>.4	0	R	input pin 4 off
	1	R	input pin 4 on
o<address>.4	0	RW	LED 4 off
	1	RW	LED 4 on

You can also set the value to “on” or “off”, that is equivalent to 1 and 0.

DF8IL

The DF8IL is an input/output module with 8 digital inputs and 8 digital open collector outputs for LEDs.

It uses 4 input and 4 output addresses in the DOMINO bus.

ID	Value	R/W	Description
i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address>.3	0	R	input pin 3 off
	1	R	input pin 3 on
i<address>.4	0	R	input pin 4 off
	1	R	input pin 4 on
i<address+1>.1	0	R	input pin 5 off
	1	R	input pin 5 on
i<address+1>.2	0	R	input pin 6 off
	1	R	input pin 6 on
i<address+1>.3	0	R	input pin 7 off
	1	R	input pin 7 on
i<address+1>.4	0	R	input pin 8 off
	1	R	input pin 8 on
i<address+2>.1	0	R	LED 1 is off
	1	R	LED 1 is on
i<address+2>.2	0	R	LED 2 is off
	1	R	LED 2 is on
i<address+2>.3	0	R	LED 3 is off
	1	R	LED 3 is on
i<address+2>.4	0	R	LED 4 is off
	1	R	LED 4 is on
i<address+3>.1	0	R	LED 5 is off

	1	R	LED 5 is on
i<address+3>.2	0	R	LED 6 is off
	1	R	LED 6 is on
i<address+3>.3	0	R	LED 7 is off
	1	R	LED 7 is on
i<address+3>.4	0	R	LED 8 is off
	1	R	LED 8 is on
v<address+2>.1	0	RW	LED 1 is off
	1	RW	LED 1 is on
v<address+2>.2	0	RW	LED 2 is off
	1	RW	LED 2 is on
v<address+2>.3	0	RW	LED 3 is off
	1	RW	LED 3 is on
v<address+2>.4	0	RW	LED 4 is off
	1	RW	LED 4 is on
v<address+3>.1	0	RW	LED 5 is off
	1	RW	LED 5 is on
v<address+3>.2	0	RW	LED 6 is off
	1	RW	LED 6 is on
v<address+3>.3	0	RW	LED 7 is off
	1	RW	LED 7 is on
v<address+3>.4	0	RW	LED 8 is off
	1	RW	LED 8 is on

You can also set the value to “on” or “off”, that is equivalent to 1 and 0.

💡 **Note that the first two output addresses are reserved and should not be used. Use address + 2 and address + 3 to control the LEDs.**

DFIGLASS

The DFIGLASS is an input/output module with 6 touch buttons with addressable LEDs and a buzzer for audio feedback.

It uses 1 input and 1 output addresses in the DOMINO bus.

ID	Value	R/W	Description

i<address>.1	0	R	button 1 off
	1	R	button 1 on
i<address>.2	0	R	button 2 off
	1	R	button 2 on
i<address>.3	0	R	button 3 off
	1	R	button 3 on
i<address>.4	0	R	button 4 off
	1	R	button 4 on
i<address>.5	0	R	button 5 off
	1	R	button 5 on
i<address>.6	0	R	button 6 off
	1	R	button 6 on
i<address>.fault	0	R	no fault
	1	R	touch keypad fault
i<address>.led.1	0	RW	led 1 off
	1	RW	led 1 on
i<address>.led.2	0	RW	led 2 off
	1	RW	led 2 on
i<address>.led.3	0	RW	led 3 off
	1	RW	led 3 on
i<address>.led.4	0	RW	led 4 off
	1	RW	led 4 on
i<address>.led.5	0	RW	led 5 off
	1	RW	led 5 on
i<address>.led.6	0	RW	led 6 off
	1	RW	led 6 on
i<address>.backlight	0	RW	backlight off
	1	RW	backlight on
i<address>.buzzer	0	RW	buzzer disabled
	1	RW	buzzer enabled

You can also set the value to “on” or “off”, that is equivalent to 1 and 0.

DF4I/V

The DF4I/V is an input/output module with 4 digital inputs and 12 digital virtual outputs.

It uses four consecutive addresses in the DOMINO bus.

ID	Value	R/W	Description
i<address>.1	0	R	input pin 1 off
	1	R	input pin 1 on
i<address>.2	0	R	input pin 2 off
	1	R	input pin 2 on
i<address>.3	0	R	input pin 3 off
	1	R	input pin 3 on
i<address>.4	0	R	input pin 4 off
	1	R	input pin 4 on
v<address+1>.1	0	RW	virtual pin 1 off
	1	RW	virtual pin 1 on
v<address+1>.2	0	RW	virtual pin 2 off
	1	RW	virtual pin 2 on
v<address+1>.3	0	RW	virtual pin 3 off
	1	RW	virtual pin 3 on
v<address+1>.4	0	RW	virtual pin 4 off
	1	RW	virtual pin 4 on
v<address+2>.1	0	RW	virtual pin 1 off
	1	RW	virtual pin 1 on
v<address+2>.2	0	RW	virtual pin 2 off
	1	RW	virtual pin 2 on
v<address+2>.3	0	RW	virtual pin 3 off
	1	RW	virtual pin 3 on
v<address+2>.4	0	RW	virtual pin 4 off
	1	RW	virtual pin 4 on
v<address+3>.1	0	RW	virtual pin 1 off

	1	RW	virtual pin 1 on
v<address+3>.2	0	RW	virtual pin 2 off
	1	RW	virtual pin 2 on
v<address+3>.3	0	RW	virtual pin 3 off
	1	RW	virtual pin 3 on
v<address+3>.4	0	RW	virtual pin 4 off
	1	RW	virtual pin 4 on

You can also set the value to “on” or “off”, that is equivalent to 1 and 0.

DFAI

The DFAI is an input module (0-10V) with 2 analog inputs.

It uses two consecutive input addresses in the DOMINO bus.

ID	Value	R/W	Description
i<address>	0...1000	R	input voltage in Volt/100
i<address+1>	0...1000	R	input voltage in Volt/100

DFLUX, DFSUN

The DFLUX and DFSUN are input modules with 1 analog light sensor.

They use one input address in the DOMINO bus.

ID	Value	R/W	Description
i<address>	0...1023	R	lux level (raw level, not translated to the lux range)

DFCT

The DFCT is an input/output temperature sensor module.

It uses two consecutive input addresses and five consecutive output addresses in the DOMINO bus.

ID	Value	R/W	Description
i<address>.mode	winter	R	winter mode
	summer	R	summer mode
i<address>.fan	min	R	min fan speed
	med	R	medium fan speed

	max	R	max fan speed
	off	R	fan off
i<address>.fan.mode	man	R	manual fan mode
	auto	R	auto fan mode
i<address>.setpoint	1	R	setpoint 1
	2	R	setpoint 2
	3	R	setpoint 3
	off	R	off
	man	R	manual setpoint
i<address>.setpoint.mode	man	R	manual setpoint mode
	auto	R	auto setpoint mode
i<address>.temp	<temp>	R	temperature value (in C/10)
	fault	R	temperature sensor fault
i<address>.status	off	R	off
	cooling	R	cooling status mode
	heating	R	heating status mode
o<address>.mode	winter	RW	winter mode
	summer	RW	summer mode
o<address>.fan	min	RW	min fan speed
	med	RW	medium fan speed
	max	RW	max fan speed
	off	RW	fan off
o<address>.fan.mode	man	RW	manual fan mode
	auto	RW	auto fan mode
o<address>.setpoint	1	RW	setpoint 1
	2	RW	setpoint 2
	3	RW	setpoint 3
	0	RW	setpoint off
	man	RW	manual setpoint

o<address>.setpoint.temp.1	<temp>	RW	temperature setpoint 1 value (in C/10)
o<address>.setpoint.temp.2	<temp>	RW	temperature setpoint 2 value (in C/10)
o<address>.setpoint.temp.3	<temp>	RW	temperature setpoint 3 value (in C/10)
o<address>.setpoint.temp.man	<temp>	RW	temperature manual setpoint 1 value (in C/10)
o<address>.program.summer o<address>.program.winter	refresh	RW	forces to daily summer/winter program for all days of the week
o<address>.program.summer.<day> o<address>.program.winter.<day>	refresh	RW	forces to daily summer/winter program for the reported day (mon:1, sun:7)
	<s0>...<s47>	RW	48 character represents the daily setpoint program divided in time slots of 30 minutes

---

DFTZ

The DFTZ is an input/output temperature sensor module.

It uses 3 consecutive input addresses and 4 consecutive output addresses in the DOMINO bus.

ID	Value	R/W	Description
i<address>.mode	winter	R	winter mode
	summer	R	summer mode
i<address>.setpoint	comfort	R	comfort setpoint
	eco	R	eco setpoint
	off	R	off
i<address>.temp	<temp>	R	temperature value (in C/10)
	fault	R	temperature sensor fault
i<address>.status	off	R	off
	cooling	R	cooling status mode
	heating	R	heating status mode
o<address>.mode	winter	RW	winter mode
	summer	RW	summer mode
o<address>.setpoint	comfort	RW	comfort setpoint
	eco	RW	eco setpoint

	0	RW	off
o<address>.setpoint.temp.comfort	<temp>	RW	comfort setpoint value (in C/10)
o<address>.setpoint.temp.eco	<temp>	RW	eco setpoint value (in C/10)
o<address>.setpoint.temp.limit	<temp>	RW	summer/winter setpoint temperature limit (in C/10)

DFRHT

The DFRHT is a temperature and humidity sensor.

It uses four consecutive input addresses and two consecutive output addresses in the DOMINO bus.

ID	Value	R/W	Description
i<address>.humidity	0...100	R	relative percent umidity
i<address>.temp	<temp>	R	temperature value
i<address>.dewpoint	<temp>	R	dew point value
i<address>.dewpoint.limit.1	0	R	dew point is lower than limit 1
	1	R	dew point is higher than limit 1
	<temp>	RW	dew point limit 1
	off	RW	limit 1 not set
i<address>.dewpoint.limit.2	0	R	dew point is lower than limit 2
	1	R	dew point is higher than limit 2
	<temp>	RW	dew point limit 2
	off	RW	limit 2 not set

DFMETEO

The DFMETEO is the weather sensor module.

It uses four consecutive input addresses and three consecutive output addresses in the DOMINO bus.

ID	Value	R/W	Description
i<address>.temp	<temp>	RW	temperature value (in C/10)
	off	RW	limit not set
i<address>.lux	<lux*10>	R	lux level according to the DFLUX range
	<lux>	RW	lux limit

	0	RW	limit not set
i<address>.wind	1 m/s /10	R	wind value
	m/s	RW	wind limit
	0	RW	limit not set
i<address>.rain	0	R	no rain
	1	R	rain
i<address>.night	0	R	day
	1	R	night
i<address>.temp.limit	0	R	measured temp is less than limit
	1	R	measured temp is greater than limit
i<address>.lux.limit	0	R	measured lux is less than limit
	1	R	measured lux is greater than limit
i<address>.wind.limit	0	R	measured wind is less than limit
	1	R	measured wind is greater than limit
i<address>.light.south	0	R	light is not coming from south
	1	R	light is coming from south
i<address>.light.west	0	R	light is not coming from west
	1	R	light is coming from west
i<address>.light.east	0	R	light is not coming from east
	1	R	light is coming from east
i<address>.fault	0	R	sensor not fault
	1	R	sensor fault

DFCC

The DFCC is an energy meter and load manager module.

In the following table, <N> is the sequential index (1 to 3) of the DFCC module.

ID	Value	R/W	Description
energy.<n>.power.real	0...65535	R	real power (Watt)
energy.<n>.power.reactive	-32768...+32767	R	reactive power (var)

energy.<n>.power.apparent	-32768...+32767	R	apparent power (VA)
energy.<n>.power.realavg	0...65535	R	average real power (Watt)
energy.<n>.power.reactiveavg	-32768...+32767	R	average reactive power (var)
energy.<n>.cos	-1000...+1000	R	cos(φ) * 1000
energy.<n>.load.1	0	R	load 1 disabled
	1	R	load 1 enabled
energy.<n>.load.2	0	R	load 2 disabled
	1	R	load 2 enabled
energy.<n>.load.3	0	R	load 3 disabled
	1	R	load 3 enabled
energy.<n>.load.4	0	R	load 4 disabled
	1	R	load 4 enabled
energy.<n>.load.5	0	R	load 5 disabled
	1	R	load 5 enabled
energy.<n>.load.6	0	R	load 6 disabled
	1	R	load 6 enabled
energy.<n>.load.7	0	R	load 7 disabled
	1	R	load 7 enabled
energy.<n>.load.8	0	R	load 8 disabled
	1	R	load 8 enabled

DFANA

Network analyzer module for Domino bus.

Uses up to 20 consecutive input addresses and, if enabled, 1 output address equal to the base input address.

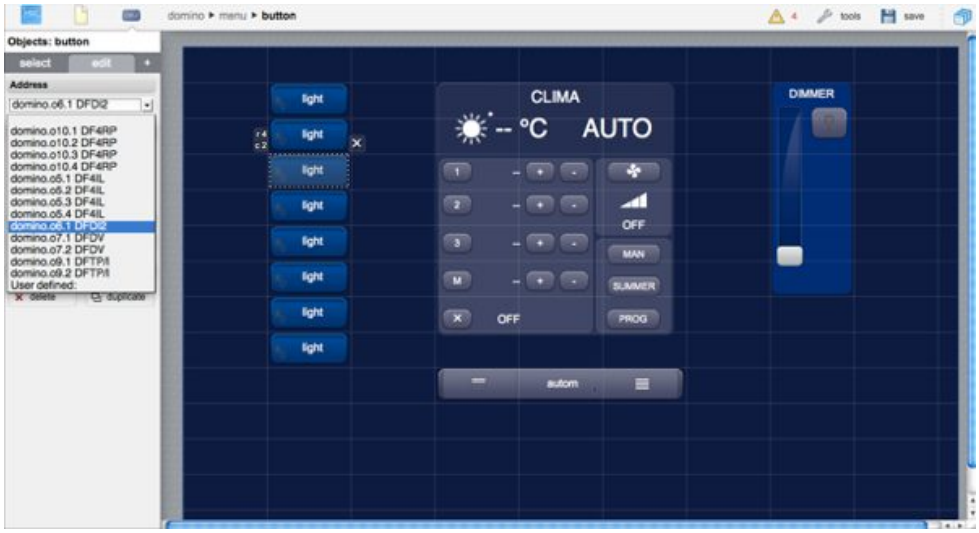
ID	Value	R/W	Description
i<address>.v12	[V]	R	chained voltage phase 1-2
i<address>.v23	[V]	R	chained voltage phase 2-3
i<address>.v31	[V]	R	chained voltage phase 3-1
i<address>.vtm	[V]	R	average chained voltage
i<address>.i1	[A]	R	current phase 1

i<address>.i2	[A]	R	current phase 2
i<address>.i3	[A]	R	current phase 3
i<address>.itm	[A]	R	average current
i<address>.ptot	[W]	R	total active power
i<address>.ptotk	[kW]	R	total active power
i<address>.qtot	[W]	R	total reactive power
i<address>.qtotk	[kW]	R	total reactive power
i<address>.pf	[pf]	R	total power factor
i<address>.frequency	[Hz]	R	frequency
i<address>.v1n	[V]	R	voltage phase 1
i<address>.v2n	[V]	R	voltage phase 2
i<address>.v3n	[V]	R	voltage phase 3
i<address>.p1	[W]	R	active power phase 1
i<address>.p1k	[kW]	R	active power phase 1
i<address>.p2	[W]	R	active power phase 2
i<address>.p2k	[kW]	R	active power phase 2
i<address>.p3	[W]	R	active power phase 3
i<address>.p3k	[kW]	R	active power phase 3

User Interface

All DOMINO devices data points that have been defined in the systemtopo.txt database are automatically listed in the Project Editor.

Adding a button to control a device output point requires just a few clicks and no additional EVENTS logic.



Besides the direct association of control buttons and data points, the DOMINO driver also automatically updates graphical objects that represent values or states of complex devices, like the DFCT temperature control unit. It will also automatically intercept buttons to manually set operation modes and temperature set-points.

DFDMX

You can define a standard HSYCO DMX server for each DFDMX module, then use the dmx and dmxrgb objects to control channels 1-64 of the DMX bus connected to the DFDMX module.

For example:

```
dmxServers = dmx
dmxServersId.dmx = domino.o8
```

Note that, instead of defining the dmxServersIP parameter, the dmxServersId is used to associate the DMX server with the id of the DFDMX module.

DFCC

Setting the *powerdisplay* option to true in **Settings** enables the automatic display in the GUI of the total real power measured by all DFCC modules connected to this DFCP.

Note that, if you have more than one DFCP gateway, you should enable this option for one gateway only.

UISET Actions

You can use any object that accepts a text attribute, usually text but also marquee and others, to automatically display the average real power measured by all DFCC modules connected to the DOMINO bus. These objects are updated even when the *powerdisplay* option is false.

ID	Attribute	Set to
energy.<n>.power	value	real power, followed by " W". <n> is the sequential index (1 to 3) of the DFCC module.

DFCT

You can use the temp and tempmini objects to control DFCT devices.

UISET Actions

You can use any object that accepts a text attribute, usually text but also marquee and others, like images, to automatically display the relevant information of all DFCT modules.

ID	Attribute	Set to
<address>.mode	summer	summer mode (cooling)
	winter	winter mode (heating)
<address>.mode.label.summer	visible	the DFCT is in summer mode
<address>.mode.label.winter	visible	the DFCT is in winter mode
<address>.status	off	zone off
	on	zone on
<address>.status.label.cooling	visible	if the zone is cooling
<address>.status.label.heating	visible	if the zone is heating
<address>.fan	off	fan off
	min	minimum fan speed
	med	medium fan speed
	max	maximum fan speed
<address>.fan.label.min	visible	the fan speed is min
<address>.fan.label.med	visible	the fan speed is med

<address>.fan.label.max	visible	the fan speed is max
<address>.fan.mode	man	manual fan mode
	auto	auto fan mode
<address>.setpoint	1,2,3	active setpoint
	man	manual setpoint
	off	zone off
<address>.setpoint.label.1	visible	setpoint 1 is active
<address>.setpoint.label.2	visible	setpoint 2 is active
<address>.setpoint.label.3	visible	setpoint 3 is active
<address>.setpoint.label.man	visible	manual setpoint is active
<address>.setpoint.label.off	visible	zone is off
<address>.setpoint.mode	man	manual setpoint mode
	auto	auto setpoint mode
<address>.setpoint.temp	<value>	the active setpoint temperature, in Celsius degrees with one decimal digit, followed by " °C"
<address>.setpoint.temp.1	<value>	the active setpoint 1 temperature, in Celsius degrees with one decimal digit, followed by " °C"
<address>.setpoint.temp.2	<value>	the active setpoint 2 temperature, in Celsius degrees with one decimal digit, followed by " °C"
<address>.setpoint.temp.3	<value>	the active setpoint 3 temperature, in Celsius degrees with one decimal digit, followed by " °C"
<address>.setpoint.temp.man	<value>	the manual setpoint temperature, in Celsius degrees with one decimal digit, followed by " °C"
<address>.temp	<temp>	the manual setpoint temperature, in Celsius degrees with one decimal digit, followed by " °C"
	fault	fault/error condition

USER Commands

There are several predefined buttons. Use any ordinary user object, setting the *name* and *param* fields based on this table.

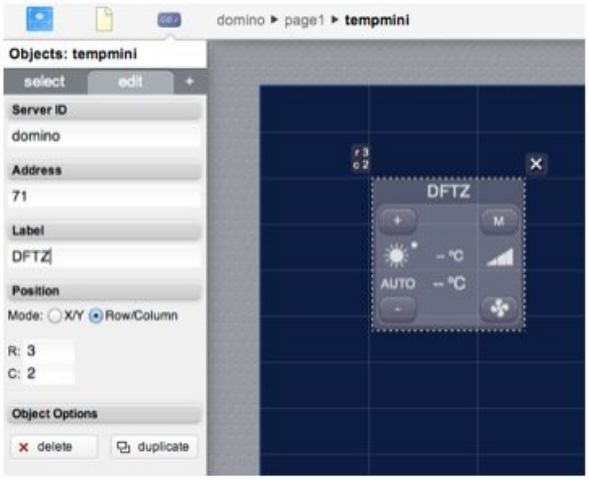
Name	Param	Action
<address>	mode	cycle through summer and winter mode
	mode.summer	set summer mode (cooling)
	mode.winter	set winter mode (heating)
	fan	cycle through fan speeds and modes (auto, off, man/min, man/med, man/max, auto)
<address>.setpoint	mode	cycle through the manual, automatic and off operation modes
	mode.man	set manual operation mode
	mode.auto	set automatic operation mode
	mode.off	off mode
	temp.1.up	increase setpoint 1 temperature in 0.5C steps
	temp.1.down	decrease setpoint 1 temperature in 0.5C steps
	temp.1.<t>	setpoint 1 set to temperature t, in C/10 (0 <= t <= 355)
	temp.2.up	increase setpoint 2 temperature in 0.5C steps
	temp.2.down	decrease setpoint 2 temperature in 0.5C steps
	temp.2.<t>	setpoint 2 set to temperature t, in C/10 (0 <= t <= 355)
	temp.3.up	increase setpoint 3 temperature in 0.5C steps
	temp.3.down	decrease setpoint 3 temperature in 0.5C steps
	temp.3.<t>	setpoint 3 set to temperature t, in C/10 (0 <= t <= 355)
	temp.man.up	increase manual setpoint temperature in 0.5C steps
	temp.man.down	decrease manual setpoint temperature in 0.5C steps
	temp.man.<t>	manual setpoint set to temperature t, in C/10 (0 <= t <= 355)
<address>.fan	mode	cycle through the manual and automatic fan speed modes
	mode.man	set manual fan speed mode

	mode.auto	set automatic fan speed mode
	up	increase fan speed
	down	decrease fan speed
	min	set min fan speed
	med	set med fan speed
	max	set max fan speed
	off	fan off

DFTZ

You can use the tempmini object to control DFTZ devices.

You could also use the larger temp object, but some of its controls are not used with the DFTZ module.



UISET Actions

You can use any object that accepts a text attribute, usually text but also marquee and others, like images, to automatically display the relevant information of all DFTZ modules.

ID	Attribute	Set to
<address>.mode	summer	summer mode (cooling)
	winter	winter mode (heating)
<address>.mode.label.summer	visible	the DFTZ is in summer mode
<address>.mode.label.winter	visible	the DFTZ is in winter mode
<address>.status	off	zone off
	on	zone on
<address>.status.label.off	visible	if the zone is not cooling or heating

<address>.status.label.cooling	visible	if the zone is cooling
<address>.status.label.heating	visible	if the zone is heating
<address>.setpoint	com	comfort setpoint
	man	manual setpoint
	off	zone off
<address>.setpoint.label.1	visible	comfort setpoint is active
<address>.setpoint.label.2	visible	eco setpoint is active
<address>.setpoint.label.off	visible	zone is off
<address>.setpoint.temp	<value>	the active setpoint temperature, in Celsius degrees with one decimal digit, followed by " °C"
<address>.setpoint.temp.1	<value>	the comfort setpoint temperature, in Celsius degrees with one decimal digit, followed by " °C"
<address>.setpoint.temp.2	<value>	the eco setpoint temperature, in Celsius degrees with one decimal digit, followed by " °C"
<address>.setpoint.temp.man	<value>	the summer or winter setpoint temperature limit, in Celsius degrees with one decimal digit, followed by " °C"
<address>.temp	<temp>	the manual setpoint temperature, in Celsius degrees with one decimal digit, followed by " °C"
	fault	fault/error condition

USER Commands

There are several predefined buttons. Use any ordinary user object, setting the *name* and *param* fields based on this table.

Name	Param	Action
<address>	mode	cycle through summer and winter mode
	mode.summer	set summer mode (cooling)
	mode.winter	set winter mode (heating)
<address>.setpoint	mode	cycle through the manual, automatic and off operation modes
	mode.comfort	set comfort operation mode
	mode.eco	set eco operation mode

	mode.off	off mode
	temp.up	increase the active setpoint temperature in 0.5C steps
	temp.down	ddecrease the active setpoint temperature in 0.5C steps
	temp.<t>	active setpoint set to temperature t, in C/10 (0 <= t <= 355)
	temp.1.up	increase comfort setpoint temperature in 0.5C steps
	temp.1.down	decrease comfort setpoint temperature in 0.5C steps
	temp.1.<t>	comfort setpoint set to temperature t, in C/10 (0 <= t <= 355)
	temp.2.up	increase eco setpoint temperature in 0.5C steps
	temp.2.down	decrease eco setpoint temperature in 0.5C steps
	temp.2.<t>	eco setpoint set to temperature t, in C/10 (0 <= t <= 355)
	temp.man.up	increase setpoint temperature limit in 0.5C steps
	temp.man.down	decrease setpoint temperature limit in 0.5C steps
	temp.man.<t>	set-point temperature limit set to temperature t, in C/10 (0 <= t <= 355)

Release Notes

3.2.2

bug fixes:

- the toolspassword optional parameter was incorrectly converted to lower case
- fixed a bug that prevented reading the input pins of DF4RP/I and DF4RPR/I

3.2.1

- support for DFTZ

3.2.0

- support for DFIGLASS
- new "detectevents" option, generates forced events when a device is detected at start-up
- support for the toolspassword option, to allow DFCP-IDE and other tools secure remote access to the DFCP via HSYCO
- new clock datapoint to read the DFCP internal clock and set it to HSYCO's time

3.1.2

bug fixes:

- the (temp) object's DFCT configuration function was broken in HSYCO 3.1.0
- energy metering data point were not updated

3.1.1

- added support for DF4RP/I and DFDT

3.1.0

- added support for DFANA energy meter module

3.0.3

- optimized performance of DFCP's registers status polling

3.0.1

- you can now limit the number of virtual points and registers read and written by HSYCO, and have access to all of the 1024 registers; defaults are unchanged
- added support for DF8IL, DFDI2B and DFRHT modules

3.0.0

- update DFDMX state at startup
- support for failover mode

1.2.0

- fixed bugs related to the DFDM and DFDI modules
- integration with the DMX server engine
- enhanced GUI support for DFCC and DFCT

1.1.0

- automatic generation of the device map in systemtopo.txt
- Web Editor support
- support for DFCP's virtual data points and registers
- support for serial connection to the DFCP through the serial ports of HWg PortBox and ERxx network devices

1.0.0

- initial release

HSYCO and Home Systems Consulting are registered trademarks of Home Systems Consulting SpA. Java and JavaScript are registered trademarks of Oracle and/or its affiliates. DOMINO and DUEMMEGI are registered trademarks of DUEMMEGI SRL. Other products or company names can be trademarks or registered trademarks of other companies and are used for demonstrative purposes only, with no violation intent.

Dummy

The DUMMY I/O Server is a virtual server that doesn't actually connect to any external system, but simulates basic lighting and automation devices.

It could be used for project's development, demonstrations, and to easily implement virtual devices with complex logic connecting to actual systems.

HSYCO Configuration

Add a DUMMY I/O Server in the I/O Servers section of the Settings and set its parameters:

High Availability

- **Shutdown when inactive**: defaults to false.

Options

ID	Default	Values	Description
discovery	true	true	automatically lists all virtual devices in the systemtopo.txt file
		false	auto-detect for devices is disabled
lights	10	0 ... 10000	number of on/off light devices to generate
dimmers	10	0 ... 10000	number of dimmer devices to generate
automations	10	0 ... 10000	number of automation devices to generate

The Device Configuration Database

The **systemtopo.txt** file contains the list of all virtual devices that should be directly associated to graphic objects in the Web-based user interface.

This file can be filled manually or automatically by HSYCO at start-up. To enable automatic discovery and automatic generation of devices' information in the systemtopo.txt file, use the *discovery* option in **Settings**.

This is an example of an automatically generated systemtopo.txt file:

```
(devices)
dummy.autom.1 : AUTOM ; VSHUT ; DUMMY
dummy.autom.2 : AUTOM ; VSHUT ; DUMMY
dummy.autom.3 : AUTOM ; VSHUT ; DUMMY
dummy.autom.4 : AUTOM ; VSHUT ; DUMMY
dummy.autom.5 : AUTOM ; VSHUT ; DUMMY
dummy.dimmer.1 : LIGHT ; DIMMER ; DUMMY
dummy.dimmer.2 : LIGHT ; DIMMER ; DUMMY
```

```
dummy.dimmer.3 : LIGHT ; DIMMER ; DUMMY
dummy.dimmer.4 : LIGHT ; DIMMER ; DUMMY
dummy.dimmer.5 : LIGHT ; DIMMER ; DUMMY
dummy.light.1 : LIGHT ; LIGHT ; DUMMY
dummy.light.2 : LIGHT ; LIGHT ; DUMMY
dummy.light.3 : LIGHT ; LIGHT ; DUMMY
dummy.light.4 : LIGHT ; LIGHT ; DUMMY
dummy.light.5 : LIGHT ; LIGHT ; DUMMY
```

You should then manually add comments and other optional parameters:

```
(devices)
dummy.autom.1 : AUTOM ; VSHUT ; Dummy automation 1
dummy.autom.2 : AUTOM ; VSHUT ; Dummy automation 2
```

Datapoints

ID	Value	R/W	Description
connection	online	R	I/O Server ready
light.<n>	0	R	light <n> is off
		W	switch off light <n>
	off	W	
	1	R	light <n> is on
		W	switch on light <n>
	on	W	
dimmer.<n>	0	R	dimmer <n> is off
		W	switch off dimmer <n>
	off	W	
	1	W	switch on dimmer <n>
	on	W	
	1% ... 100%	R	the light level of dimmer <n> corresponds to the reported value
		W	set the light level of dimmer <n> to the specified value
autom.<n>	offup	R	autom <n> is stopped in upper position (set to this value 30 seconds after a 'up' command)
	offdown	R	autom <n> is stopped in lower position

			(set to this value 30 seconds after a 'down' command)
	stop	W	stop autom <n>
	off		
	0		
	up	R	autom <n> is moving upwards (opening)
		W	move autom <n> upwards
	down	R	autom <n> is moving downwards (closing)
		W	move autom <n> downwards

Virtual automation devices automatically go to the offup or offdown state 30 seconds after the up or down command.

User Interface

All the devices that have been added in systemtopo.txt (automatically or manually) are listed in the Project Editor and can be directly associated to button and dimmer objects.

For example, adding a slider for a dimmer requires just a few clicks and no additional EVENTS logic.

Release Notes

3.0.0

- initial release

HwgIO

The HW group I/O Controller 2 is a general purpose digital input-output Ethernet controller, with 8 inputs and 8 outputs.

This I/O Server also supports the legacy PortBox, ER84 and ER88 conrollers.

HSYCO Configuration

Add an HWGIO I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: host name or IP address of the I/O Controller
- **IP Port**: TCP/IP port to use, leave blank to use default port 23.

High Availability

- **Shutdown when inactive**: defaults to true (the I/O Controller only accepts one connection and can't be accessed by two HSYCO servers simultaneously).

Options

There are no available configuration options for the HWGIO I/O Server.

I/O Controller Configuration

TCP/IP mode needs to be configured and NVT (Network virtual terminal) enabled in order to work with the digital inputs and outputs.

Refer to the I/O Controller manual for detailed configuration information.

Datapoints

The data points associated to the input pins are named “i1” to “i8”.

The data points associated to the output pins are named “o1” to “o8”.

The I/O Server generates asynchronous events based on the status of the I/O Controller's input and output pins. The output data points can be written to control the output pins.

ID	Value	R/W	Description
i<n>	0	R	input <n> is low

	1	R	input <n> is high
o<n>	0	R	output <n> is low
		W	set output <n> to off
	1	R	output <n> is high
		W	set output N to on

Release Notes

3.0.1

- Fixed a bug that could block the update of the systemtopo.txt file at start-up.

2.8.3

- Support of the serial port in HWg I/O Controller.

2.8.0

- Initial support.

HWg is a registered trademark of HW group s.r.o.

KNX

The KNX I/O Server provides for the control of a KNX installation compliant with the KNX Standard version 2.0.

The communication between HSYCO and the KNX network must employ a KNXnet/IP gateway supporting the Tunneling service.

The KNX framework must be programmed to work properly using the ETS tool.

Each defined KNX group is seen as a datapoint whose value is formatted with respect to its DPT type. Supported DPT types range from DPT 1 to DPT 18. Moreover, it is possible to have virtual datapoints that link a set of datapoints used for the control of devices such as lights, dimmers, automations and thermostats which require several KNX groups for commands and feedback.

HSYCO Configuration

Add a KNX I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: IP address of the KNX/IP gateway
- **IP Port**: TCP/IP port to use, leave blank to use default port 3671.

High Availability

- **Shutdown when inactive**: defaults to false.

Options

ID	Default	Values	Description
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
discovery	true	true	automatically create the list of all detected datapoints with type DPT 1 and the defined virtual datapoints in the systemtopo.txt file. Should be enabled to allow the automatic update of (button) objects' states
		false	auto-detect for devices is disabled
fromevents	false	true	generate events for commands sent on the KNX BUS reporting the source address and the value of the command
		false	disable command events

readaftercommand	false	true	force the driver to read the value of a datapoint after a command is sent to it
		false	the driver will assume that commands detected on the BUS are always successful, thus no re-reading is performed to set the value of a datapoint
localport		<port_number>	the port number used locally by HSYCO to communicate with the KNXnet/IP gateway. This option might be necessary when employing network address translation (NAT)
framedelay	50	n ≥ 0	the delay in milliseconds between each frame sent to the KNX bus

Datapoints Declaration

It is required to provide the I/O Server with the KNX framework configuration. To this end, the KNX Utility provides a graphical interface to import ETS4 project files and to configure the datapoints.

It is also possible to manually define the datapoints in the knxtopo.txt file.

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established to the gateway and initial reading executed
	offline	R	HSYCO can't connect to the gateway
<datapoint_id>	<value>	R	this datapoint has value <value>
		W	set this datapont to value <value>
<datapoint_id>.from.<source>	<value>	R	a command with value <value> from the device with individual address <source> has been sent to the addressed datapoint. The source address has the format “x.y.z” (e.g. 1.2.87)
light.<n>	0	R	the light is off
		W	turn off the light

		1	R	the light is on
			W	turn on the light
		flip	W	invert the state of the light
		unknown	R	the state of the light is unknown
dimmer.<n>		off	R	the dimmer is off
			W	turn off the dimmer
		on	W	turn on the dimmer
		1% ... 100%	R	the dimming level is set to the reported level
		0% ... 100%	W	set the dimming level to the specified value
		unknown	R	the state of the dimmer is unknown
autom.<n>	if use position = false	up	R	the automation is moving upwards
		down	R	the automation is moving downwards
		offup	R	the automation is stopped in a open position
		offdown	R	the automation is stopped in the closed position
	if use position = true	0% ... 100%	R	the automation is stopped at the reported position
		0%+ ... 100%+	R	the automation is at the reported position and is closing
		0%- ... 100%-	R	the automation is at the reported position and is opening
		up	W	move the automation upwards
		down	W	move the automation downwards
		0% ... 100%	W	move the automation to the specified position

		stop	W	stop the automation
		unknown	R	the state of the automation is unknown
thermo.<n>.setpoint.temp		<float_value>	W	set the setpoint to the specified value
		up	W	increase the setpoint by 0.5
		down	W	decrease the setpoint by 0.5
thermo.<n>.setpoint		mode	W	switch to the next setpoint mode
thermo.<n>.setpoint.mode		comf	R	the setpoint mode of the thermostat is set to "comfort"
		night	R	the setpoint mode of the thermostat is set to "night"
		stb	R	the setpoint mode of the thermostat is set to "stand-by"
		unknown	R	the setpoint mode of the thermostat is unknown
gateway		next	W	kill and restart the I/O Server and switch to the next gateway as listed in the IP settings
		<n>	R	the I/O Server is using the gateway with index <n> with respect to the list of addresses in the IP settings, starting from 0
			W	kill and restart the I/O Server and switch to the gateway with index <n>

Datapoint types and values

The KNX I/O Server supports all datapoint types (DPTs) with main number ranging from 1 to 18.

The value of a datapoint is formatted with respect to its DPT main number. The actual behavior of the datapoint is anyhow dependent on its complete type (i.e. main number and sub number). Examples for the most common cases will be provided, but for a complete reference of all available datapoint types and allowed values we strongly advice to refer to chapter 3/7/2 of the KNX Standard v2.0.

The following paragraphs will show the KNX data format for each DPT as described in the protocol as well as the allowed datapoint values and the corresponding conversion.

Datapoint Type 1 (1 bit, Boolean)

KNX bit-level format:

8	7	6	5	4	3	2	1
							b

b = {0, 1}

Datapoint values:

Value	Converts to
0	b = 0
1	b = 1

Example:

For a datapoint with DPT 1.001 (e.g. a switch) a value of '0' corresponds to the 'off' state, while for a datapoint with DPT 1.007 (e.g. a step indicator for a dimmer) it would correspond to the state 'decrease'.

Datapoint Type 2 (2 bits, Control)

KNX bit-level format:

8	7	6	5	4	3	2	1
						c	v

c = {0, 1}, v = {0, 1}

Datapoint values:

Value	Converts to
disabled	c = 0, v = 0
0	c = 1, v = 0
1	c = 1, v = 1

Datapoint Type 3 (4 bits, Dimming, Blinds)

KNX bit-level format:

8	7	6	5	4	3	2	1
				c	step		

c = {0, 1}, step = [000b ... 111b]

Datapoint values:

Value	Converts to
-7 ≤ v < 0	c = 0, step = |v|

0 ≤ v ≤ 7	c = 1, step = v

Example:

For a datapoint with DPT 3.007 (control dimming) a value of '3' corresponds to a 3-steps light increase. The same value for a DPT 3.008 (control blinds) datapoint corresponds to a 3-steps downwards movement. In both cases a value of '0' indicates the 'break' state.

Datapoint Type 4 (8 bit, Character set)

KNX bit-level format:

8	7	6	5	4	3	2	1
char							

char = [0 ... 255] (ISO 8859-1 encoding of a character)

Datapoint values:

Value	Converts to
0 ≤ v ≤ 255	char = v

A DPT 4.001 datapoint will only support the values corresponding to ASCII characters (a subset of the ISO 8859-1 standard values), thus only values ranging from 0 to 127 are allowed.

Datapoint Type 5 (8 bits, Unsigned Value)

KNX bit-level format:

8	7	6	5	4	3	2	1
val							

val = [0 ... 255]

Datapoint values:

Value	Converts to
0 ≤ v ≤ 255	val = v

Example:

For a datapoint with DPT 5.001 (scaling) a value of '255' corresponds to '100%'. For a datapoint with DPT 5.003 (angle) the same value corresponds to '360°'.

Datapoint Type 6 (8 bits, Signed Value)

KNX bit-level format:

8	7	6	5	4	3	2	1
val							

val = [-128 ... 127] (two’s complement notation)

Datapoint values:

Value	Converts to
-128 ≤ v ≤ 127	val = v

Datapoint Type 7 (2 Bytes, Unsigned Value)

KNX bit-level format:

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
val															

val = [0 ... 65535]

Datapoint values:

Value	Converts to
0 ≤ v ≤ 65535	val = v

Datapoint Type 8 (2 Bytes, Signed Value)

KNX bit-level format:

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
val															

val = [-32768 ... 32767] (two's complement notation)

Datapoint values:

Value	Converts to
-32768 ≤ v ≤ 32767	val = v

Datapoint Type 9 (2 Bytes, Float Value)

KNX bit-level format:

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
m	e						m								

e = [0 ... 15], m = [-2048 ... 2047] (two's complement notation)

float value = (0.01 * m) * 2^e = [-671088.64 ... 670760.96]

Datapoint values:

Value	Converts to
-671088.64 ≤ v ≤ 670760.96	float value = v

Datapoint Type 10 (3 Bytes, Time)

KNX bit-level format:

24 23 22 21 20 19 18	17	16	15 14 13 12 11 10 9 8	7	6 5	4 3 2 1
day	hour	0	0	min	0	0	sec

day = [0 ... 7] (0 = no day, 1 = Monday, ..., 7 = Sunday)

hour = [0 ... 23]

min = [0 ... 59]

sec = [0 ... 59]

Datapoint values:

Value	Converts to
<day>.<hour>.<minutes>.<seconds>	day = <day>, hour = <hour>, min = <min>, sec = <sec>

Example:

A vale of '2.12.46.23' corresponds to 'Tuesday 12:46:23'.

Datapoint Type 11 (3 Bytes, Date)

KNX bit-level format:

24 23 22 21 20 19 18			17	16	15 14 13 12	11 10 9 8	7	6 5 4 3 2 1
0	0	0	day	0	0	0	0	month	0	year

day = [1 ... 31]

month = [1 ... 12]

year = [0 ... 99] (if ≥ 90 interpret as 20th century, otherwise interpret as 21st century)

Datapoint values:

Value	Converts to
<year>/<month>/<day> (1990 ≤ <year> ≤ 2089)	year = last two digits of <year>, month = <month>, day = <day>

Example:

A vale of '2011/2/8' corresponds to 'February 8th 2011'.

Datapoint Type 12 (4 Bytes, Unsigned Value)

KNX bit-level format:

32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
val

val = [0 ... 4294967295]

Datapoint values:

Value	Converts to
0 ≤ v ≤ 4294967295	val = v

Datapoint Type 13 (4 Bytes, Signed Value)

KNX bit-level format:

32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
val																															

val = [-2147483648 ... 2147483647] (two's complement notation)

Datapoint values:

Value	Converts to
-2147483648 ≤ v ≤ 2147483647	val = v

Datapoint Type 14 (4 Bytes, Float Value)

KNX bit-level format:

32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
s	e								f																						

s = {0, 1}, e = [0 .. 255], f = [0 ... 8388607]

float value = 0, [1.40129846432481707E-45, 3.40282346638528860E+38] (IEEE 754 floating point format)

Datapoint values:

Value	Converts to
v = 0 | 1.40129846432481707E-45 ≤ v ≤ 3.40282346638528860E+38	float value = v

Datapoint Type 15 (4 Bytes, Access)

KNX bit-level format:

32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
d6				d5				d4				d3				d2				d1				e	p	d	c	index			

d1, d2, d3, d4, d5, d6 = [0 ... 9]

index = [0 ... 15]

e = {0, 1} (0 = no error, 1 = reading of access information was not successful)

p = {0, 1} (0 = not accepted, 1 = accepted)

d = {0, 1} (0 = left to right, 1 = right to left)

c = {0, 1} (0 = no encryption, 1 = access information encrypted)

Datapoint values:

Value	Converts to
<d6>.<d5>.<d4>.<d3>.<d2>.<d1>.<e>.<p>.<d>.<c>.<index>	d6 = <d6>, d5 = <d5>, d4 = <d4>, d3 = <d3>, d2 = <d2>, d1 = <d1>, e = <e>, p = <p>, d = <d>, c = <c>, index = <index>

Example:

A valid value would be: '1.2.3.4.5.7.0.1.0.0.13'

Datapoint Type 16 (14 Bytes, String)

KNX bit-level format:

112 111 110 109 108 107 106 105	...	8 7 6 5 4 3 2 1
char1	...	char14

These Datapoint Types are used to transmit strings of textual characters. The length is fixed to 14 octets. The contents are filled starting from the most significant octet. Each octet shall be encoded as specified for the chosen character set. If the string to be transmitted is smaller than 14 octets, unused trailing octets in the character string shall be set to NULL (00H).

Example: “KNX is OK” is encoded as follows: 4B 4E 58 20 60 73 20 4F 4B 00 00 00 00 00

Datapoint values:

Value	Converts to
<string> (max 14 characters)	ISO 8859-1 encoded bytes padded with trailing NULL bytes

Example:

A valid value would be: 'Hello world'

Datapoint Type 17 (1 Byte, Scene Number)

KNX bit-level format:

8 7 6 5 4 3 2 1		
0	0	sn

sn = [0 ... 63]

Datapoint values:

Value	Converts to
0 ≤ v ≤ 63	sn = v

Datapoint Type 18 (1 Byte, Scene Control)

KNX bit-level format:

8 7 6 5 4 3 2 1		
c	0	sn

c = {0, 1} (0 = activate the scene, 1 = learn the scene)

sn = [0 ... 63]

Datapoint values:

For this case the datapoint syntax is slightly different. The scene number (sn) must be suffixed to the datapoint ID as follows: <datapoint_id>.<scene_number>.

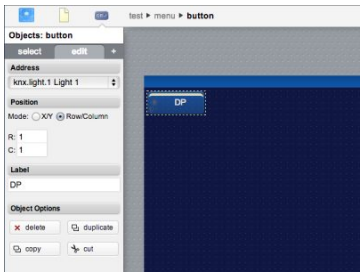
Value	Converts to
record	c = 1
1	c = 0

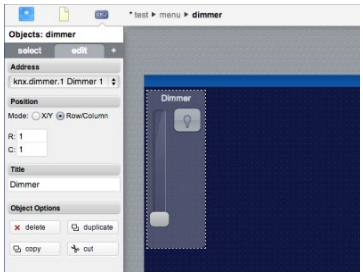
Example:

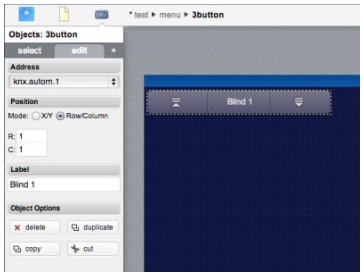
The activation of scenario 12 on a datapoint would correspond to: <datapoint_id>.12 = 1

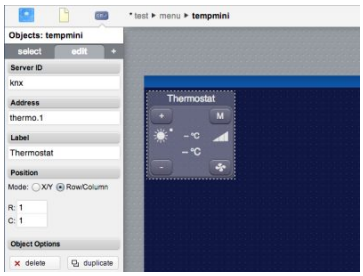
User Interface

All the DPT 1 datapoints and virtual datapoints that have been defined are automatically listed in the Project Editor. DPT 1 datapoints and lights can be directly linked to a "button" object, dimmers to a "button" or a "dimmer" object, automations to "button" or "3button" objects and thermostats can be controlled though a "tempmini" object.









Release Notes

3.3.0

- added support for light virtual datapoints

3.2.1

- fixed bug on auto-discovery of AUTOM datapoints
- fixed bug on loading of datapoints' thumbnails

3.2.0

- initial release

KNX is a registered trademark of KNX Association.

Knxtopo.txt

The knxtopo.txt is a configuration file for the KNX I/O Server. The KNX Utility provides a graphical interface to modify the content of this file.

The file knxtopo.txt contains the declaration of the used datapoints of all the defined KNX I/O Servers and their functional attributes.

For each KNX datapoint we have a corresponding line with the following syntax:

```
<io_server_id>.<group_address> : dpt = {1 ... 18}; id = <id>; [priority = {low|normal| urgent};] [read=
{true|false};] [description = <description_text>]
```

For details on the attributes see: KNX_Utility#Datapoints

For instance:

```
knx.0/0/1 : dpt = 1; id = dpt1; priority = low; read=true; description = Datapoint X
```

For virtual datapoints use the following syntax.

For a light:

```
<io_server_id>.light.<id> : [fb = <feedback_datapoint_address>;] [cmd = <command_datapoint_address>;] [on =
{1|0};] [description = <description_text>]
```

For a dimmer:

```
<io_server_id>.dimmer.<id> : [fb.level = <level_feedback_datapoint_address>;] [fb.onoff =
<on_off_feedback_datapoint_address>;] [cmd.level = <level_command_datapoint_address>;] [cmd.onoff =
<on_off_command_datapoint_address>;] [cmd.step = <step_command_datapoint_address>;] [on = {1|0};] [stepval =
{1|2|3|4|5|6|7};] [trip = {auto|<trip_time>};] [description = <description_text>]
```

For an automation:

```
<io_server_id>.autom.<id> : [fb.position = <position_feedback_datapoint_address>;] [fb.updown =
<up_down_feedback_datapoint_address>;] [fb.stopup = <stop_up_feedback_datapoint_address>;] [fb.stopdown =
<stop_down_feedback_datapoint_address>;] [fb.moving = <moving_feedback_datapoint_address>;] [cmd.updown =
<up_down_command_datapoint_address>;] [cmd.position = <position_command_datapoint_address>;] [cmd.stop =
<stop_command_datapoint_address>;] [up = {0|1};] [closed = {255|0};] [position = {true|false};] [trip =
<trip_time>;] [description = <description_text>]
```

For a thermostat:

```
<io_server_id>.thermo.<id> : [fb.setpoint = <setpoint_feedback_datapoint_address>;] [fb.status =
<status_feedback_datapoint_address>;] [fb.temp = <temperature_feedback_datapoint_address>;]
[fb.cooling.status = <cooling_status_feedback_datapoint_address>;] [fb.heating.status =
<heating_status_feedback_datapoint_address>;] [fb.mode = <mode_feedback_datapoint_address>;] [cmd.setpoint =
<setpoint_command_datapoint_address>;] [cmd.setpoint.mode = <setpoint_mode_command_datapoint_address>;]
[unit = {C|F|K};] [description = <description_text>]
```

For details on the attributes see: KNX_Utility#Virtual_datapoints

For instance:

```
knx.light.1 : fb = 0/1/1; cmd = 0/1/2; on = 1; description = Light 1
```

```
knx.dimmer.1 : fb.level = 0/1/3; fb.onoff = 0/1/4; cmd.level = 0/1/5; cmd.onoff = 0/1/6
```

```
knx.autom.1 : fb.position = 0/1/17; fb.updown = 0/1/13; fb.stopup = 0/1/14; fb.stopdown = 0/1/22; fb.moving
= 0/1/28; cmd.updown = 0/1/20; cmd.position = 0/1/21; cmd.stop = 0/1/19; up = 0; closed = 255; position =
false; trip = 60; description = Autom 1
```

```
knx.thermo.1 : fb.setpoint = 0/1/31; fb.status = 0/1/0; fb.temp = 0/1/33; fb.cooling.status = 0/1/32;
fb.heating.status = 0/1/39; fb.mode = 0/1/38; cmd.setpoint = 0/1/37; cmd.setpoint.mode = 0/1/36; unit = C;
description = Thermo 1
```

When the knxtopo.txt configuration is complete and the KNX I/O Server is started, for each defined DPT 1 datapoint and virtual datapoint a corresponding entry is automatically added to the systemtopo.txt devices list so to be able to directly associate them to graphical objects in the Web user interface.

This is an example of an automatically generated systemtopo.txt file:

```
(devices)
knx.light.1 : LIGHT ; LIGHT ; Light 1
knx.dimmer.1 : LIGHT ; DIMMER ; Dimmer 1
knx.autom.1 : AUTOM ; VSHUT ; Autom 1
knx.dpt1 : LIGHT ; LIGHT ; Datapoint X
```

You can then add/modify graphical attributes such as the icon types or the thumbnails for any device listed in this file.

MyHome

MyHome is a proprietary bus system for home automation developed by BTicino. HSYCO fully integrates with MyHome using the OpenWebNet (OWN) protocol.

HSYCO supports all MyHome devices and features that the OWN protocol officially supports, including lighting, automation, scenes, AUX and CEN addressing, the temperature control unit and burglar alarm unit.

The bus architecture can be arranged as a single bus or with a riser and several local busses.

Communication

HSYCO supports the Ethernet IP connection to MyHome through IP-based OWN gateways, including the F452, F453, F454, F453AV, MH200 and MH200N.

HSYCO also supports the L4686SDK module through an USB connection and an associated virtual serial port.

Gateway Configuration

The OWN gateway must be configured to allow open OWN connections from the IP address assigned to the HSYCO server.

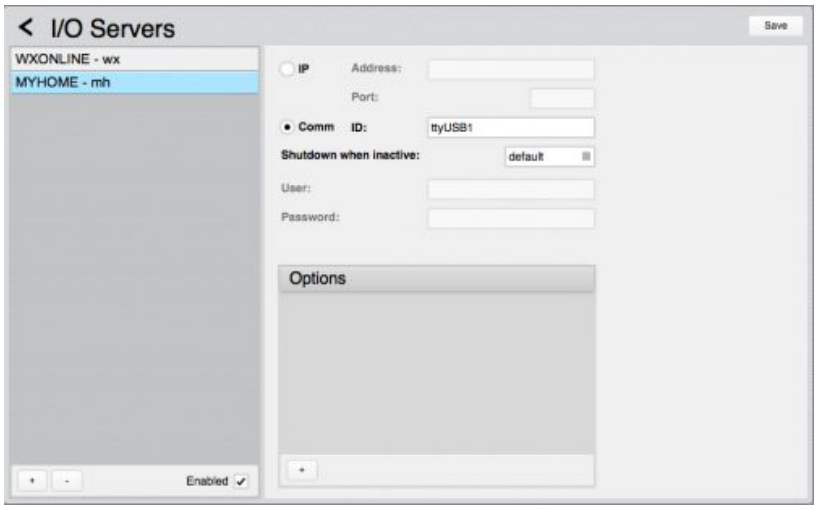
L4686SDK Module

The L4686SDK interface module is connected to HSYCO using a USB cable.

The operating system creates a virtual serial port; its name depends on your HSYCO server configuration, and should be ttyUSB0 if there are no other devices connected to the USB ports.

You could also use the ftdi-n-n.n alias, that is unique for each physical USB port on the server. In this case, you should also list the full device name in the *CommPortsList* parameter.

For example, in this screenshot we can take a look to the configuration with an L4686SDK connected to the second USB port (ttyUSB1)



L4686SDK Limitations

- L4686SDK firmware version 1.20.0 or later is required
- Not all protocol frames are supported. The I/O server has been tested with the following functions: scenarios, lights, automation, temperature control, AUX and CEN plus
- Check the L4686SDK documentation to know which functions are actually supported, also based on the L4686SDK firmware version
- The clock read and sync functions are not supported
- HSYCO only support the L4686SDK module on a single main bus; private raiser buses are not supported
- the USB connection cable should never be unplugged during normal operations. If the cable is unplugged, HSYCO will not be able to restore the connection to the L4686SDK until it is manually restarted.

HSYCO Configuration

Add a MYHOME I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: IP address of the gateway
- **IP Port**: TCP/IP port to use, leave blank to use default port 20000.

High Availability

- **Shutdown when inactive**: defaults to false.

Options

ID	Default	Values	Description
gui	true	true	enables automatic support of Web GUI objects to show the status of the system and intercept control buttons
		false	disables support for the Web GUI
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
discovery	true	true	auto-detects MyHome lighting and automation devices that are connected to the bus and are fully operational, and automatically creates the list of all detected devices and individual data points in the systemtopo.txt file
		false	auto-detect is disabled
tempzones[Note 1]		z<n>	if you want to monitor and control temperature zones, all zone numbers should be listed with the tempzones option. <n>

			is the zone number. Multiple zones should be separated by semicolons
language	en	en, it, fr	the language used for automatically generated GUI text fields

Note 1

The 4695 four zone temperature control unit uses a zone address for the control unit itself. This address has to be marked as a special address in the tempzones parameter, enclosing it in round brackets.

For example:

```
tempZones=(32);33
```

defines a central unit having address 32, and an additional zone with address 33.

The MyHome I/O Server supports only one central unit for each SCS riser bus.

myhome.ini

The myhome.ini file is a specific configuration file located in the main directory (same directory as hsyco.ini or hsyco.jar).

This file is optional. It is only used to assign friendly names to the burglar alarm zones and technical alarms, so that these names are used in the GUI log and the security.log files.

If you have multiple My Home gateways with multiple burglar alarm panels, the parameters need to be defined for each system, using unique ids, according to the following table:

ID	Description
security.zone.<n>	the friendly name for burglar alarm zone <n>
security.aux.<n>	the friendly name for burglar alarm technical alarm <n>

The myhome.ini is automatically detected at start-up and automatically reloaded whenever it is modified.

For example, the following myhome.ini file is used to define a few friendly names of two distinct My Home systems:

```
mh1.security.zone.1.name=entrance
mh1.security.zone.2.name=lobby
mh1.security.aux.1.name=flooding
mh2.security.zone.3.name=windows
mh2.security.zone.5.name=driveway
```

The Device Configuration Database

The **systemtopo.txt** file contains the list of all lighting and automation devices that could be directly associated to graphic object in the Web-based user interface.

This file can be filled manually or automatically by HSYCO at start-up. To enable automatic discovery and automatic generation of devices' information in the systemtopo file, use the *discovery* option in **System Settings**. The discovery feature is enabled by default.

This is an example of an automatically generated systemtopo.txt file:

```
(devices)
mh.autom.82 : AUTOM ; VSHUT ;
mh.autom.88 : AUTOM ; VSHUT ;
mh.autom.93 : AUTOM ; VSHUT ;
mh.autom.96 : AUTOM ; VSHUT ;
mh.light.11 : LIGHT ; LIGHT ;
mh.light.12 : LIGHT ; LIGHT ;
mh.light.13 : LIGHT ; LIGHT ;
mh.light.16 : LIGHT ; LIGHT ;
mh.light.25 : LIGHT ; DIMMER ; DIMMER
```

Note that HSYCO can automatically detects dimmer modules if they are on when discovery is performed. If a dimmer is off, it will be listed as an ordinary on/off light module, and should be manually changed to DIMMER.

You should manually add comments and other optional parameters:

```
(devices)
mh.autom.82 : AUTOM ; VSHUT ; driveway gate
mh.autom.88 : AUTOM ; VSHUT ; awning
mh.autom.93 : AUTOM ; VSHUT ; kitchen shutter
mh.autom.96 : AUTOM ; VSHUT ; bedroom shutter
```

Also, if you want to use (button) objects in the Project Editor to control scenes, you should manually add scenes definitions in systemtopo.txt, as HSYCO can't automatically detect scene modules and stored scenes:

```
(devices)
mh.scene.94.5 : LIGHT ; SCENE ; close all shutters
mh.scene.94.10 : LIGHT ; SCENE ; night mode
```

Datapoints

The naming convention HSYCO uses for the events and to control all MyHome devices is closely related to the MyHome addressing standards and the OpenWebNet protocol implementation.

In OWN, the standard monitor and command frames have the following format:

```
*<who>*<what>*<where>##
```

The <who> field defines the module's family or function. <what> is the state of a device, for example 0 for off and 1 for the on state of an on/off module. Finally, <where> is the module's address, including the local bus number if the module is not on the riser bus.

With few exceptions, this frame format translates to the following event format inside HSYCO:

```
IO servername.<who>.<where> = <what>
```

The WHO field in OWN frames is a number, while we use a literal string in HSYCO to make names easier to read.

The table below lists all supported <who> functions and the HSYCO's corresponding literals that form the prefix of all IO data points.

OWN WHO	HSYCO Prefix	Function Description
0	scene	scenario module

1	light	lighting module
2	autom	automation module (inter-locking)
3	load	load control
4	temp	temperature control
5	security	burglar alarm panel
9	aux	auxiliary address
15	cen	cen modules / addresses
25	cenplus	cen plus modules and dry contacts / IR modules

The <where> field in OWN is, in most cases, the numeric address of a module, eventually followed by #4#<b> for modules on local buses (<b> is the local bus numerical id). Group addressing has the <where> field formatted as #<g>, where <g> is the group's numerical id.

The table below defines the conversion rules between the OWN <where> format and HSYCO's data points format.

OWN WHERE	HSYCO Format	Description
<n>	<n>	if <n> is a plain number, it is used as-is in HSYCO. Note that you should retain the leading zero when present. 01 is not equivalent to 1 in OWN, and HSYCO simply passes the where field back and forth with no changes
#<n>	g<n>	group addresses are represented in HSYCO as a lowercase g followed by the group numerical id, from 1 to 9
<n>#4#	l.**<n>**	a lowercase "l" and the local bus id are prefixed to the base address
#<n>#4#	l.**g<n>**	group addressing on local buses

The <what> field represents the value or action, and corresponds to the data point value in HSYCO. Each function supports a specific set of <what> values.

Gateway Module

ID	Value	R/W	Description
connection	online	R	connection established to the OWN gateway
	offline	R	HSYCO can't connect to the OWN gateway
frame	OWN raw frame	R	a frame event is generated on incoming frames, based on the monitor setting (see below)

		W	send a raw OWN frame to the gateway
monitor	none	W	the frame datapoint is not set with monitored frames
	ignored	W	the frame datapoint is set with monitored frames that are not recognized and translated to specific events
	all	W	the frame datapoint is set with any monitored frame
clock	sync	W	set the gateway clock to HSYCO's current time
	read	W	read the gateway clock, and the delta with HSYCO's time
	<yyyy>-<mm>-<hh>:<mm>:<ss>	R	the gateway clock current time
clock.delta	integer	R	the delta time in seconds between the gateway and HSYCO clocks. A positive number means that the gateway clock is ahead of HSYCO

Scenario Modules

ID	Value	R/W	Description
scene.<module> scene.<bus>.<module>	erase	R	all scenarios erased
		W	erase all scenarios
	lock	R	module is locked (cannot erase or record scenarios)
		W	lock the scenario module
	unlock	R	module is unlocked (can erase and record scenarios)
		W	unlock the scenario module
	unavailable	R	the scenario module is busy
	full	R	the scenario module memory is full
scene.<module>.<scene> scene.<bus>.<module>.<scene> [Note 2]	1	R	scenario activated
		W	execute scenario
	on	W	execute scenario
	0	R	scenario deactivated or cleared

		W	deactivate scenario
	off	W	deactivate scenario
	clear	W	clear scenario status, without sending the deactivate command to the module
	record	R	recording
		W	start recording
	end	R	end of recording
		W	stop recording
	erase	R	the scenario has been erased
		W	erase scenario

Note 2

<module> is the scenario module address, from 01 to 99 (retain the leading 0).

<bus> is the local bus ID, from 1 to 9, when the scenario module is on a local bus and not on the riser bus.

<scene> is the scenario number, from 1 to 32 (do not add a leading 0 for numbers less than 10).

The unavailable event is generated when the scenario module cannot accept commands. This usually happens when the module is recording a scenario and at the same time is asked to execute another scenario.

Lighting: on/off Modules

ID	Value	R/W	Description
light.<where>	1	R	on status
		W	turn on
	on	W	turn on
	0	R	off status
		W	turn off
	off	W	turn off
	lock	W	lock (disable)
	unlock	W	unlock (enable)

When a module is locked, it doesn't respond to any bus command except the unlock command.

Lighting: Dimmer Modules

ID	Value	R/W	Description

light.<where>	1	W	turn on at last level
	on	W	turn on at last level
	0	R	off status
		W	turn off
	off	W	turn off
	<x>%	R	<x>% level
		W	set to <x>% level
	<x>	W	set to <x>% level
	<x>/<y>	W	set to <x>/<y> level
	fault	R	load fault detected
	lock	W	lock (disable)
	unlock	W	unlock (enable)

Dimmers level can be set between 20% and 100%, in 10% steps.

Automation: Inter-locking Modules

ID	Value	R/W	Description
autom.<where>	unknown	R	the module's status is unknown at this time
	offup	R	off in up position
	offdown	R	off in down position
	stop off 0	W	stop
	up	R	going up
		W	up/open command
	down	R	going down
		W	down/close command
	lock	W	lock (disable)
	unlock	W	unlock (enable)

When the MyHome driver starts, and until an up or down command is sent to the module, HSYCO sets the data point value to unknown.

The offup and offdown status values do not represent the fully up/opened or down/closed position of the mechanical device controlled by the module, but just the fact that the module was commanding an up or down movement before being

stopped.

Temperature Control: Central Unit

ID	Value	R/W	Description
temp.control	0	R	supervisor control of central unit disabled
	1	R	supervisor control of central unit enabled
temp.zone.off [forced event]	1	R	at least one zone is set to off mode
temp.zone.protection [forced event]	1	R	at least one zone is set to protection mode
temp.zone.manual [forced event]	1	R	at least one zone is set to manual mode
temp.fault [forced event]	1	R	generic fault
temp.battery.fault [forced event]	1	R	central unit battery fault
temp.mode	winter	R	winter mode
		W	set winter mode
	summer	R	summer mode
		W	set summer mode
temp.setpoint	off	R	off mode
	protection	R	temperature protection mode
	<t>	R	when setpoint.mode=man, <t> is the temperature setpoint in °C/10
		R	when setpoint.mode=auto, is the weekly program, from 1 to 3
		R	when setpoint.mode=scenario, ~~is the scenario number, from 1 to 16~~
	<d>	R	when setpoint.mode=away, <d> is the number of days, from 1 to 255
		R	when setpoint.mode=holiday, is the weekly program to be set at the end of the day

temp.setpoint.mode	off	R	central unit is off
	protection	R	central unit is in temperature protection mode
	man	R	central unit is in manual mode
	auto	R	central unit is in automatic mode
	scenario	R	central unit is in scenario mode
	away	R	central unit is in away mode
	holiday	R	central unit is in holiday mode
temp.command	off	W	turn the system off
	protection	W	set temperature protection mode
	1	W	set weekly program 1
	2	W	set weekly program 2
	3	W	set weekly program 3
	<t>	W	set to manual mode, temperature <t> (in C/10), for example t=215 is 21.5 °C
	up	W	set to manual mode, raising the setpoint by 0.5 °C
	down	W	set to manual mode, lowering the setpoint by 0.5 °C
	holiday	W	set holiday program (returns to weekly program 1 at midnight)
	holiday.<n>	W	set holiday program (returns to weekly program <n> at midnight)
	away.<yyyymmdd>	W	set away mode until the day passed as parameter, returning to weekly program 1 at midnight
	away.<yyyymmdd> .<n>	W	set away mode until the day passed as parameter, returning to weekly program <n> at midnight
	away.<d>	W	set away mode for <d> days, returning to weekly program 1 at midnight
	away.+<d>	W	increases the number of away days by <d>, starting from 2 days

	away.-<d>	W	decreases the number of away days by <d>
	away.<d>.<n>	W	set away mode for <d> days, returning to weekly program <n> at midnight
	scenario.<n>	W	set scenario mode <n>

Temperature Control: Zones

In the following table, <z> is the zone number, from 1 to 99.

ID	Value	R/W	Description
temp.<z>.cooling.status	0	R	zone actuator off in summer mode
	1	R	zone actuator on in summer mode
temp.<z>.heating.status	0	R	zone actuator off in winter mode
	1	R	zone actuator on in winter mode
temp.<z>.fan	off	R	fan is off
	min	R	slow fan speed
	med	R	medium fan speed
	max	R	fast fan speed
temp.<z>.mode	winter	R	winter mode
	summer	R	summer mode
temp.<z>.setpoint	0	W	turn zone off
	off	W	turn zone off
	auto	W	set zone to automatic mode
	protection	W	set zone to temperature protection mode
	up	W	set zone to manual mode, raising the setpoint by 0.5 °C
	down	W	set zone to manual mode, lowering the setpoint by 0.5 °C
	<t>	R	current temperature setpoint <t> (in °C/10), corrected with local offset
		W	set zone to manual mode, temperature <t> (in °C/10), for example t=215 is 21.5 °C

temp.<z>.setpoint.local	-3 ... +3	R	zone local offset is set to -3 ... +3
	protection	R	zone locally set to protection mode
	off	R	zone locally set to off
temp.<z>.setpoint.mode	off	R	zone is off
	protection	R	protection mode
	man	R	manual mode
	auto	R	automatic mode
temp.<z>.temp	<t>	R	measured temperature <t> (in °C/10), for example t=183 is 18.3 °C
temp.<z>	unlock	W	zone unlock

Burglar Alarm: Main Panel

ID	Value	R/W	Description
security.active	0	R	the burglar alarm panel is not active
	1	R	the burglar alarm panel is active (not necessarily armed)
security.alarm	0	R	no alarm
	1	R	ongoing alarm condition
security.delay [forced event]	end	R	end of arming delay (system now fully armed)
security.lowbattery	0	R	battery ok
	1	R	low battery or battery fault condition
security.maintenance	0	R	not in maintenance mode
	1	R	maintenance mode
security.nopower	0	R	AC power ok
	1	R	fault condition on the AC power supply
security.programming	0	R	not in programming mode
	1	R	programming mode
security.status	off	R	system not armed

	armed	R	system armed

💡 **Note that, because the MyHome burglar alarm panel cannot be directly controlled via OpenWebNet commands, there are no write-enabled data points for this system.**

Burglar Alarm: Zones and Technical (aux)

In the following table, <z> is the zone number.

ID	Value	R/W	Description
security.aux.<z>.alarm [forced event]	0	R	technical alarm reset
	1	R	technical alarm zone <z>
security.sensor.<z>.fault [forced event]	1	R	sensor <z> fault
security.zone.<z>.active	0	R	zone <z> is not active
	1	R	zone <z> is active
security.zone.<z>.alarm	0	R	zone <z> intrusion alarm reset
	1	R	zone <z> intrusion alarm
security.zone.<z>.panic	0	R	zone <z> panic alarm reset
	1	R	zone <z> panic alarm
security.zone.<z>.tamper	0	R	zone <z> tamper alarm reset
	1	R	zone <z> tamper alarm

Auxiliary Addresses

ID	Value	R/W	Description
aux.<where>	1	R	on status
		W	turn on
	on	W	turn on
	0	R	off status
		W	turn off
	off	W	turn off

CEN Modules

ID	Value	R/W	Description
cen.<where>.<bt>	on	R	button <bt> (00-31) pressed

		W	virtual press of button <bt>
	off	R	button <bt> (00-31) released after short time (<= 0.5s)
		W	virtual short time release of button <bt>
	offlong	R	button <bt> (00-31) released after long time (> 0.5s)
		W	virtual long time release of button <bt>
	onrepeat	R	button <bt> (00-31) continues to be pressed (this event repeats every 0.5s)
		W	virtual long time pressure of button <bt>

CENPLUS Modules

ID	Value	R/W	Description
cenplus.<where>.<bt> (used for CEN PLUS command modules) [Note 3]	on	R	button <bt> (00-31) pressed
		W	virtual press of button <bt>
	off	R	button <bt> (00-31) released after short time (<= 0.5s)
		W	virtual short time release of button <bt>
	offlong	R	button <bt> (00-31) released after long time (> 0.5s)
		W	virtual long time release of button<bt>
	onrepeat	R	button <bt> (00-31) continues to be pressed (this event repeats every 0.5s)
		W	virtual long time pressure of button <bt>
cenplus.<where> (used for DRY CONTACT and IR modules) [Note 3]	on	R	on status
		W	turn on
	off	R	off status
		W	turn off
	request	W	request a status update

Note 3

The <where> field used for command modules is prefixed by the “2” character in front of the address, while dry contact and IR modules have the <where> field prefixed with “3”.

WHERE	Value	Description
2	0-2047	CEN PLUS modules configured with Advanced Virtual Configuration
3	1-201	for automation dry contact interface (3477 and F428) <where> should be configured using the Virtual Configurator Software
3	[1-9][1-9]	for alarm dry contact interface and IR (3480, F482, IR 4610, 4611, 4640) <where> is configured using <z> and <n> with physical configuration pins

F421 Load Control Unit

ID	Value	R/W	Description
load.voltage	refresh	W	requests the measured voltage
	V	R	measured voltage, in Volts
load.current	refresh	W	requests the measured current
	I	R	measured current, in Amperes
load.power	refresh	W	requests the measured power
	P	R	measured power, in Watts
load.energy	refresh	W	requests the measured energy
	E	R	measured energy

💡 **Note that reading a measured value to the F421 load control unit requires a relatively long time (up to about 4 seconds, based on our tests) and, during this time, the My Home gateway is unable to process any other request. This can result in temporary slowdown of the execution of other commands.**

User Interface

Temperature Control Unit

UISET Actions

ID	Attribute	Set to
temp.status	value	summer/winter followed by the current status, for example: WINTER MAN 21.0 °C

USER Commands

Name	Param	Action
temp.command	off	turn the system off
temp.command	protection	set temperature protection mode
temp.command	1	set weekly program 1
temp.command	2	set weekly program 2
temp.command	3	set weekly program 3
temp.command	<t>	set to manual mode, temperature <t> (in °C/10), for example t=215 is 21.5 °C
temp.command	up	set to manual mode, raising the setpoint by 0.5 °C
temp.command	down	set to manual mode, lowering the setpoint by 0.5 °C
temp.command	holiday	set holiday program (returns to weekly program 1 at midnight)
temp.command	holiday.<n>	set holiday program (returns to weekly program <n> at midnight)
temp.command	away.<yyyymmdd>	set away mode until the day passed as parameter, returning to weekly program 1 at midnight
temp.command	away.<yyyymmdd>.<n>	set away mode until the day passed as parameter, returning to weekly program <n> at midnight
temp.command	away.<d>	set away mode for D days, returning to weekly program 1 at midnight
temp.command	away.+<d>	increases the number of away days by <d>, starting from 2 days
temp.command	away.-<d>	decreases the number of away days by <d>
temp.command	away.<d>.<n>	set away mode for <d> days, returning to weekly program <n> at midnight
temp.command	scenario.<n>	set scenario mode <n>

Temperature Zones

The standard TempMini object can be used to control each zone. The fan speed indicator is not visible if the zone is not configured for fan control.

Set the I/O server id in the *Server ID* field, and temp.<z> where <z> is the zone number from 1 to 99 in the *Address* field.

> The Temp object is not supported for the MyHome system.

Zone unlock

If a zone has been locked in the off or protection modes using the local dial, you can create a User object to unlock that zone.

Name	Param	Action
temp.<z>	unlock	unlock zone <z>

Burglar Alarm Unit

UISET Actions

ID	Attribute	Set to
security.status	value	shows system status: ARMED, DISARMED or MAINTENANCE
security.alarm	value	shows ALARM if there is an ongoing alarm condition, empty otherwise
security.alarm.type	value	shows the alarm type, if not an intrusion alarm: TAMPER, PANIC or TECHNICAL
security.zones	value	shows zones 1-8 activation status, for example: 12-----8 means that only zones 1, 2 and 8 are active
security.log.<n>	value	log line <n> of the events log, where <n> is between 1 and 20. Line 1 lists the most recent event
security.status.label.maintenance	visible	true only when the burglar alarm panel is in maintenance mode
security.status.label.active	visible	true only when the burglar alarm panel is active (not necessarily armed)
security.lowbattery.label	visible	true only when there is a low battery or battery fault condition
security.acfault.label	visible	true only when there is a fault condition on the AC power supply
security.status.label.armed	visible	true only when the system is armed
security.alarm.label.intrusion	visible	true only when there is an ongoing intrusion alarm condition
security.alarm.label.tamper	visible	true only when there is an ongoing tamper alarm condition

security.alarm.label.panic	visible	true only when there is an ongoing panic alarm condition
security.alarm.label.aux	visible	true only when there is an ongoing technical alarm condition

USER Commands

Because the MyHome burglar alarm panel cannot be directly controlled via OpenWebNet commands, there are no standard control objects for this system.

Release Notes

3.3.0

- added support of the L4686SDK interface module

3.2.2

- clock read and sync are now based on local time

3.2.1

- improved handling of connection errors

3.2.0

- new clock datapoint to read the gateway’s internal clock and set it to HSYCO's time

3.0.0

- initial release

MyHome and BTicino are registered trademarks of BTicino SpA.

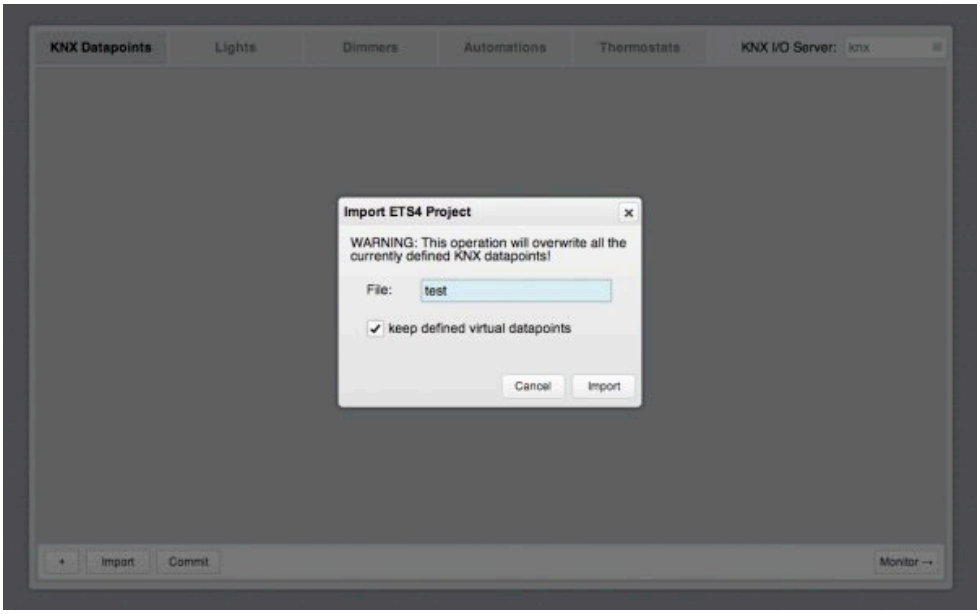
KNX Utility

The KNX Utility application provides a graphical interface to configure a KNX I/O Server. Whenever a KNX I/O Server is defined in HSYCO, the KNX Utility will appear among the applications of the manager.

Open the application and select the KNX I/O Server to be configured in the select field in the top-right corner. If only one KNX I/O Server is defined it will be automatically selected.

Import an ETS4 project

To import an ETS4 project, upload the .knxproj file on HSYCO server, then click on the "Import" button at the bottom of the page and enter the file name (the ".knxproj" extension can be omitted).

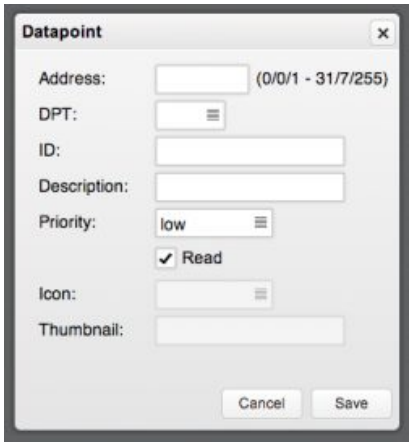


The checkbox option "keep defined virtual datapoints" is needed when updating an old KNX installation to an extended one and you need to retain the previously created virtual datapoints.

This procedure will create a datapoint for each group defined in the ETS project with the corresponding DPT type and the default attributes.

Datapoints

A datapoint is the equivalent of a group of the KNX framework. New datapoints can be added employing the import procedure described above or manually, by clicking on the "+" button in the "KNX Datapoints" tab. To modify the attributes of a datapoint click on the corresponding item in the list.



Datapoint.

Attributes

- **Address:** this is the KXN group address corresponding to this datapoint ranging from 0/0/1 to 31/7/255. It must be unique for each defined datapoint
- **DPT:** the DPT type (ranging from 1 to 18) used to format the datapoint value
- **ID:** the ID to assign to the corresponding datapoint of the KNX I/O Server. It can be any unique string without spaces
- **Description:** a textual description of the datapoint. It will be visible in the Project Editor when configuring GUI objects
- **Priority:** KNX priority to be used when sending frames to this group (low, normal or urgent)
- **Read:** check this option if this group
- **Icon:** attribute for DPT 1 datapoints. It specifies the icon to be used for button objects associated to this datapoint (the values correspond to the ones used in the devices list of systemtopo.txt)
- **Thumbnail:** attribute for DPT 1 datapoints. It can be used to optionally associate an image or a camera to the datapoint. For images, use the filename of a file located in the www/img directory. To associate a camera, the “cam:<n>” format should be used, where <n> is the camera index starting from 1, based on the order of the camera list in the settings page

Virtual datapoints

KNX lights, dimmers, automations (e.g. blinds) and thermostats need a set of datapoints for control and feedback. It is possible to create a virtual datapoint that links these datapoints so to be treated as a single one which, in turn, can be directly associated to an object in the Web user interface.

Light

To add a "light" virtual datapoint select the “Lights” tab and click on the “+” button in the bottom bar.

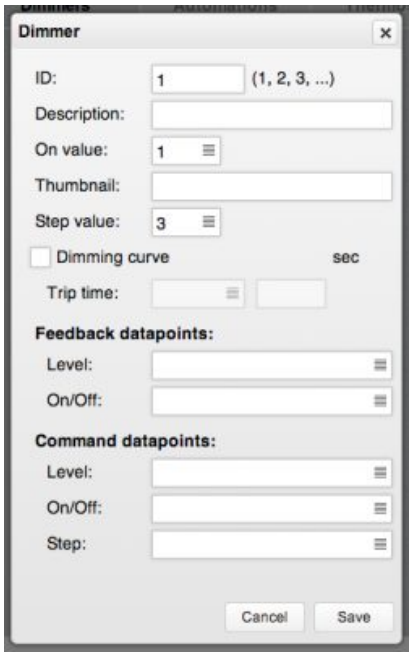
Light virtual datapoint.

Attributes

- **ID:** an identification number for this datapoint. It will be used to compose the ID of the corresponding datapoint of the KNX I/O Server (e.g. if ID is set to 1 the corresponding datapoint will be 'light.1')
- **Description:** a textual description of the datapoint. It will be visible in the Project Editor when configuring GUI objects
- **On value:** specifies if the 'on' state for this light is represented by a value of '1' or '0'
- **Icon:** It specifies the icon to be used for button objects associated to this datapoint (the values correspond to the ones used in the devices list of systemtopo.txt)
- **Thumbnail:** It can be used to optionally associate an image or a camera to the datapoint. For images, use the filename of a file located in the www/img directory. To associate a camera, the “cam:<n>” format should be used, where <n> is the camera index starting from 1, based on the order of the camera list in the settings page
- **Feedback datapoint:** datapoint representing the on/off feedback. It must be a DPT 1 datapoint
- **Command datapoint:** datapoint to be addressed when sending on/off commands. It must be a DPT 1 datapoint

Dimmer

To add a "dimmer" virtual datapoint select the “Dimmers” tab and click on the “+” button in the bottom bar.



Dimmer virtual datapoint.

Attributes

- **ID:** an identification number for this datapoint. It will be used to compose the ID of the corresponding datapoint of the KNX I/O Server (e.g. if ID is set to 1 the corresponding datapoint will be 'dimmer.1')
- **Description:** a textual description of the datapoint. It will be visible in the Project Editor when configuring GUI objects
- **On value:** specifies if the 'on' state for this light is represented by a value of '1' or '0'
- **Step value:** specifies the step value to send to the Step command datapoint (if defined) when using a 3button to dim the light
- **Dimming curve:** enable this option if the dimmer has a timed dimming curve. The buttons connected to this datapoint will show an animated slider when a new brightness level is set.
- **Trip time:** this option is available only if 'Dimming curve' is enabled. If set to 'Auto' the trip time of the dimming curve will be automatically calculated and adjusted by HSYCO; otherwise it is possible to specify a fixed trip time in seconds.
- **Thumbnail:** It can be used to optionally associate an image or a camera to the datapoint. For images, use the filename of a file located in the www/img directory. To associate a camera, the “cam:<n>” format should be used, where <n> is

the camera index starting from 1, based on the order of the camera list in the settings page

Feedback datapoints:

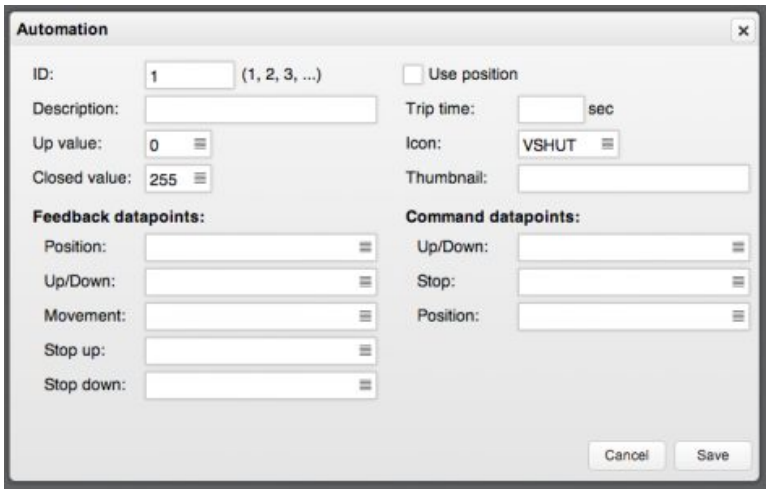
- **Level:** datapoint representing the dimming level feedback. It must be a DPT 5 datapoint
- **On/Off:** datapoint representing the on/off feedback. It must be a DPT 1 datapoint

Command datapoints:

- **Level:** datapoint to be addressed when sending a dimming level command. It must be a DPT 5 datapoint
- **On/Off:** datapoint to be addressed when sending an on/off command. It must be a DPT 1 datapoint
- **Step:** datapoint to be addressed when sending a step command. It must be a DPT 3 datapoint, generally DPT 3.007 (control dimming)

Automation

To add an "automation" virtual datapoint select the “Automations” tab and click on the “+” button in the bottom bar.



Automation virtual datapoint.

Attributes

- **ID:** an identification number for this datapoint. It will be used to compose the ID of the corresponding datapoint of the KNX I/O Server (e.g. if ID is set to 1 the corresponding datapoint will be 'autom.1')
- **Description:** a textual description of the datapoint. It will be visible in the Project Editor when configuring GUI objects
- **Up value:** specifies if the 'up' direction for this automation is represented by a value of '1' or '0'
- **Closed value:** specifies if the 'closed' state for this automation is represented by a value of '255' or '0' of the position feedback
- **Use position:** specifies wether to use percentage values or not for the read values of the datapoint of the KNX I/O Server. If position is used a button linked to the datapoint will have a slider to control and visualize the position of the automation
- **Trip time:** if position is used, specifies the trip time (in seconds) from position close to open. If the field is left blank or set to '0' the trip time will be automatically calculated
- **Icon:** the icon to be used for the buttons associated to this automation datapoint. Available choices are 'vshut' (vertical shut) or 'hshut' (horizontal shut)
- **Thumbnail:** It can be used to optionally associate an image or a camera to the datapoint. For images, use the filename of a file located in the www/img directory. To associate a camera, the “cam:<n>” format should be used, where <n> is the camera index starting from 1, based on the order of the camera list in the settings page

Feedback datapoints:

- **Position:** datapoint representing the position feedback. It must be a DPT 5 datapoint

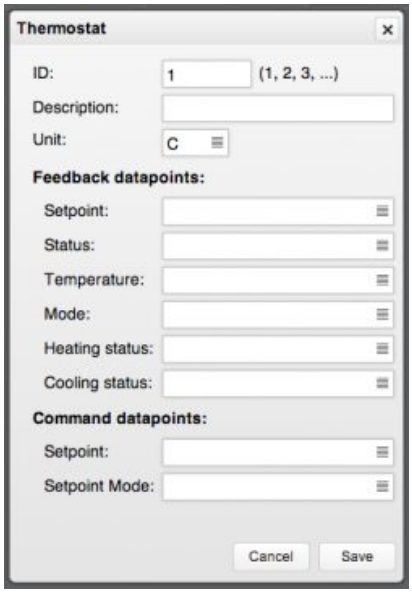
- **Up/Down:** datapoint representing the up/down direction feedback. It must be a DPT 1 datapoint
- **Movement:** datapoint representing the movement feedback. It must be a DPT 1 datapoint
- **Stop up:** datapoint representing the “stop in upper position” state feedback. It must be a DPT 1 datapoint
- **Stop down:** datapoint representing the “stop in lower position” state feedback. It must be a DPT 1 datapoint

Command datapoints:

- **Up/Down:** datapoint to be addressed when sending an up/down command. It must be a DPT 1 datapoint
- **Stop:** datapoint to be addressed when sending a stop command. It must be a DPT 1 datapoint
- **Position:** datapoint to be addressed when sending a position command. It must be a DPT 5 datapoint

Thermostat

To add an "thermostat" virtual datapoint select the “Thermostat” tab and click on the “+” button in the bottom bar.



Thermostat virtual datapoint.

Attributes

- **ID:** an identification number for this datapoint. It will be used to compose the ID of the corresponding datapoint of the KNX I/O Server (e.g. if ID is set to 1 the corresponding datapoint will be 'thermo.1')
- **Description:** a textual description of the datapoint. It will be visible in the Project Editor when configuring GUI objects
- **Unit:** The temperature unit to show on the associated tempmini Web object: C, F or K.

Feedback datapoints:

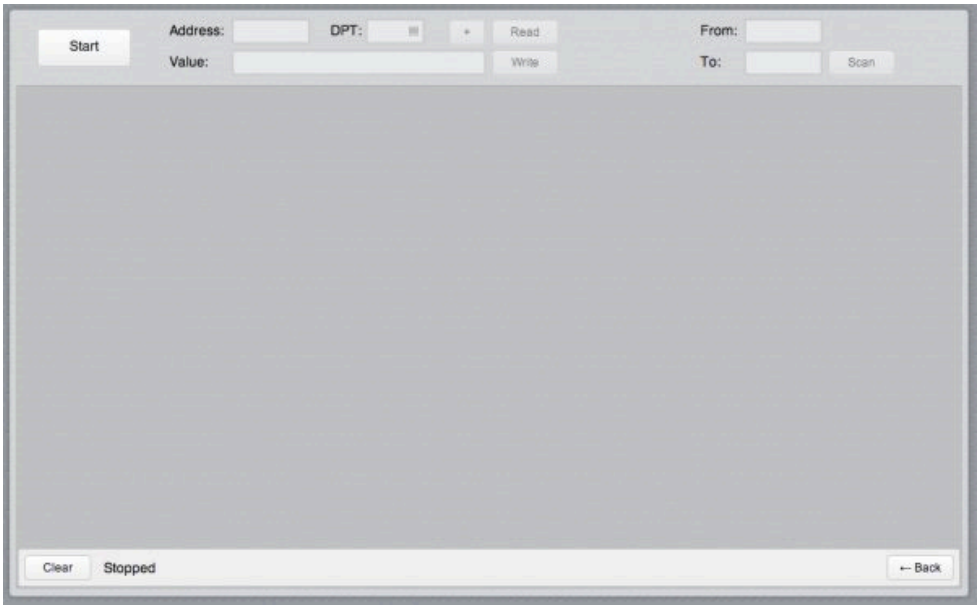
- **Setpoint:** datapoint representing the temperature setpoint feedback. It must be a DPT 9 datapoint
- **Status:** datapoint representing the working status feedback of the thermostat (e.g. "comfort", "stand-by", ...). It must be a DPT 5 datapoint
- **Temperature:** datapoint representing the measured temperature. It must be a DPT 9 datapoint
- **Mode:** datapoint representing working mode (heating or cooling). It must be a DPT 1 datapoint
- **Heating status:** datapoint representing the heating level feedback. It must be a DPT 5 datapoint
- **Cooling status:** datapoint representing the cooling level feedback. It must be a DPT 5 datapoint

Command datapoints:

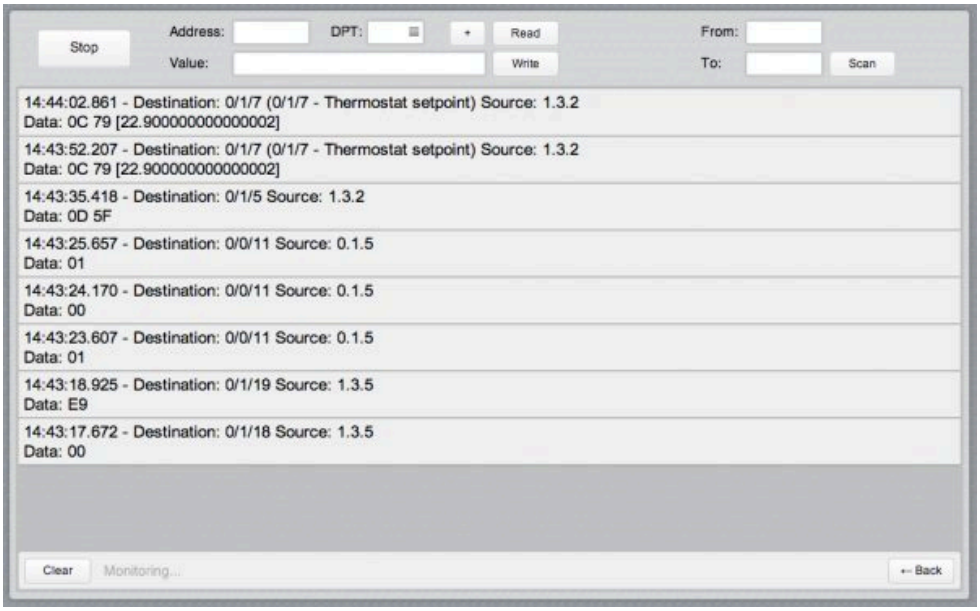
- **Setpoint:** datapoint to be addressed when sending a temperature setpoint command. It must be a DPT 9 datapoint
- **Setpoint mode:** datapoint to be addressed when sending a command to change the working status. It must be a DPT 5 datapoint

KNX Bus Monitor

The KNX Utility provides an interface for monitoring the KNX bus which is useful for testing purposes and helps detect the underlying KNX framework structure if unknown. To access the monitor, click on the "Monitor" button in the bottom-right corner and enable the monitor mode by clicking on the “Start” button.



Whenever a frame is detected on the bus, it will be listed in the central panel:



For each frame the following data will be reported:

- The time when the frame has been detected
- The destination group address the frame was sent to (with corresponding ID and description if a datapoint with that address is defined)
- The individual address of the device which generated the frame (souce)
- Hexadecimal representation of the user data contained in the frame (with the value formatted according to the DPT of the datapoint, if defined)

Using the read and write functions you can respectively recall the value currently assigned to a specific group address or write a value to it (if allowed by the KNX framework settings). When reading or writing, the DPT field must be specified so

that the value can be interpreted accordingly. The value to be written or shown after a reading, is expressed as a value formatted for the specified DPT.

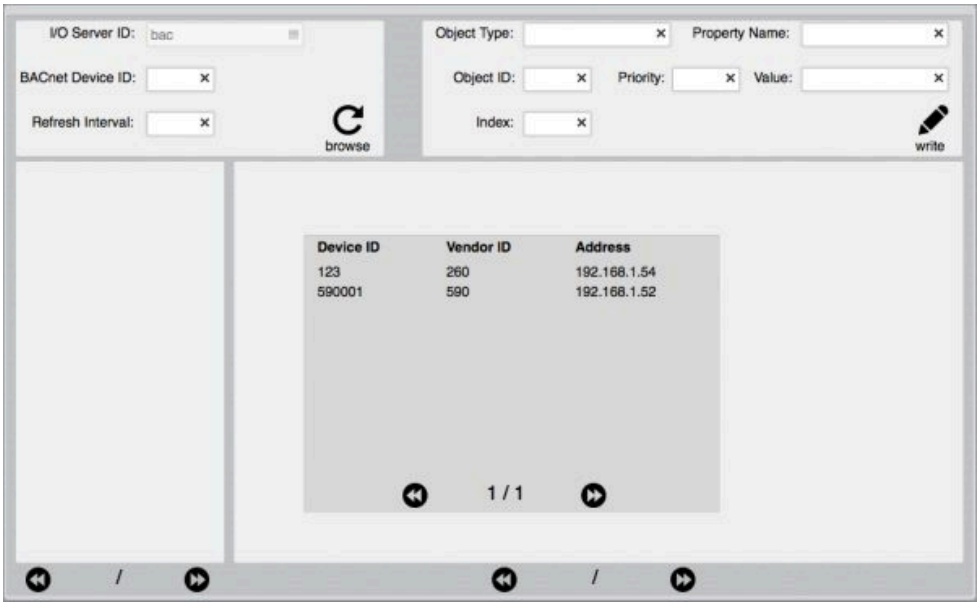
For details on how a value is formatted for a specific DPT see the datapoint types and values section of the KNX I/O Server page.

The “Scan” button on the top right corner lets you send consecutive read requests to a set of group addresses. Specify the address range using the fields “From” and “To”, press “Scan” and wait until the scanning procedure is complete to analyze the received responses. To interrupt the scanning press the "Stop" button.

BACnet Utility

The BACnet Utility application allows you to find all BACnet/IP devices available on the network, to show all objects of a device, and all properties and values of any object instance. It also supports writing values to writable properties. Whenever a BACnet I/O Server is defined in HSYCO, the BACnet Utility will appear among the applications of the manager.

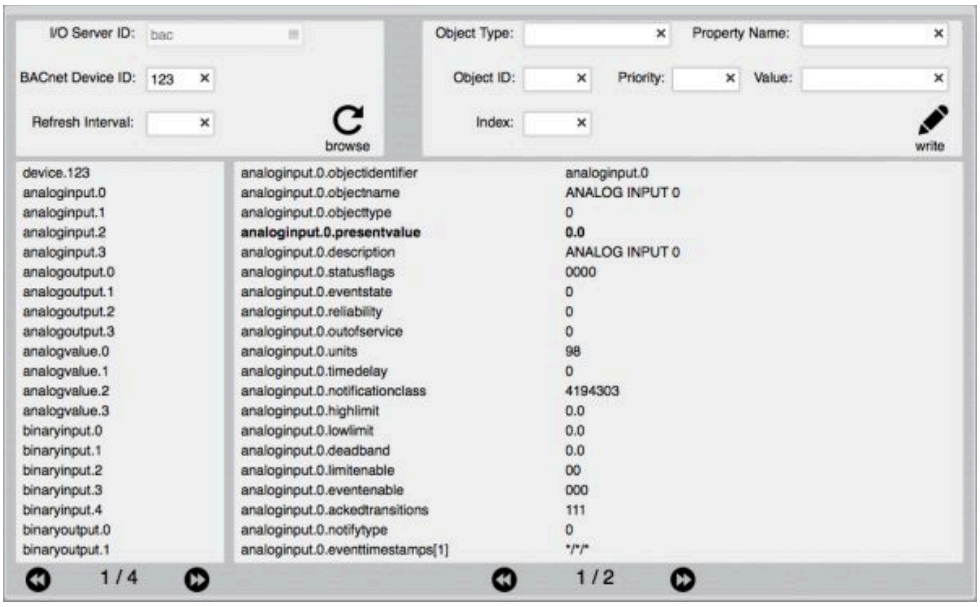
To discover all BACnet/IP devices, simply leave the BACnet Device ID field empty and press the browse icon. If the browser finds any BACnet devices, they will be listed in a pop-up selection box, with device ID, vendor ID and IP address.



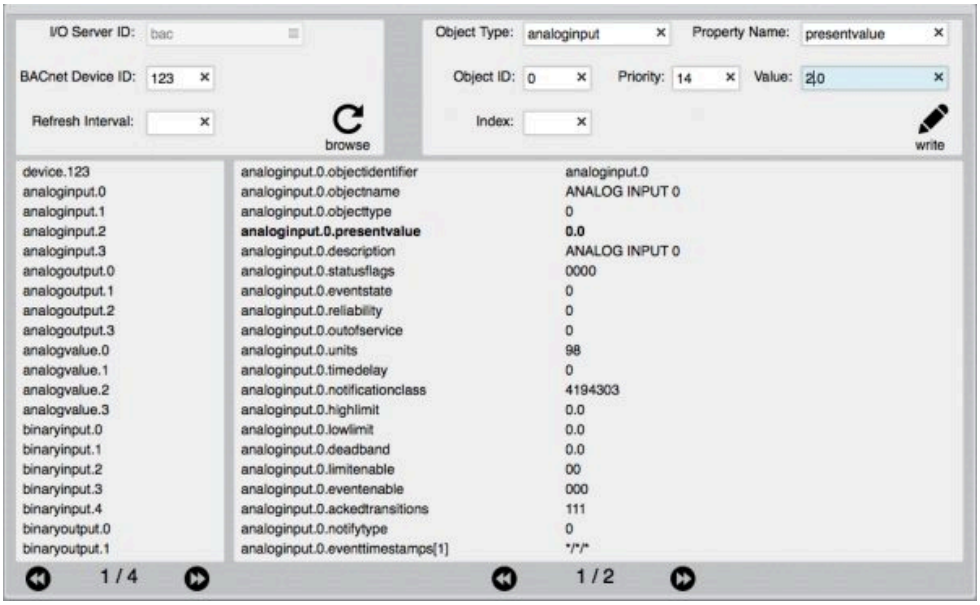
Touch a line to select that device, and the browser will show all object instances for that device.

The BACnet I/O Server doesn't support segmented messages. Segmented messages could be required to return the list of all object instances of a device. Because of this, the BACnet Utility may not be able to browse devices having a lot of object instances.

Touch any object instance to show all properties and values for that object. To manually update the values, simply press the browse icon again. You can also enter a refresh interval in seconds and press browse to automatically refresh values.



Properties that are writable, are highlighted. Touch any writable property to fill the fields in the write box, then set the new value and priority if needed, then press the write icon to write the new value to the device.



Products

HSYCO is a software platform that integrates many third-party hardware products, but doesn't require any specialized hardware itself.

Because HSYCO is a server solution that needs a reliable platform to run unattended and continuously, we sell it as a hardware/software appliance, with a pre-configured Ubuntu Linux operating system installed on high-quality computer hardware.

We offer several different versions, running on different hardware and with sotfware license limits on the number of drivers and external systems that each version supports:

- **HSYCO PRO** - The unparalleled, powerful, scalable & cost effective solution for Building Automation
- **HSYCO MINI** - Powerful and flexible for Home & Small Business and Distributed Sites
- **HSYCO Pi** - A powerful, customizable and cost effective engine for local and remote control solutions
- **HSYCO TOUCH** - The integrated touch-PC based solution for plant supervision and VoiP intercom and telephony.

Please go to the HSYCO main site for additional sales information.

If you need to install HSYCO on other hardware platforms, including Apple Mac OS X or virtualized architectures, you should contact us to share your requirements. HSYCO is a very flexible and scalable solution, that will very likely fit your needs better than any other solution.

Support

We offer free basic email technical support to all registered developers.

You can submit your requests using the tech support form.

If you would like to register, please fill the HSYCO Developer's registration form.

For other general requests, please visit the contacts page.

We also offer customized training programs, at our headquarters or at customer's locations worldwide, from basic topics to advanced programming. Please contact us for more details.

FAQ

General questions

Is HSYCO free?

No. HSYCO requires a paid license, that is associated to the server's hardware. You can still install and run HSYCO without a license, but all I/O and core functions will be disabled.

Please go to the HSYCO main site for sales information.

Can I buy a software-only version of HSYCO, and install on my PC?

We sell HSYCO as a hardware-software bundle. We use industrial-grade low power servers, that we ship pre-installed with a stripped version of Ubuntu Linux.

We can address specific deployment needs of large organizations, for example to install HSYCO on virtual servers in large data centers. Feel free to contact us and tell us about your requirements.

Installation and configuration questions

Do I need an app to use HSYCO from a smartphone?

No. The HSYCO client is a pure Web application. You only need your standard Web browser on your smartphone or any other device. We have no app for iOS or Android but, due to the way HSYCO is built, you could even build your own HSYCO client app if you need to do so.

Which browser do I need to access the HSYCO control interface?

HSYCO requires JavaScript, HTML5 and CSS3 on the browser, and is compatible with most modern Web browsers, including Chrome, Firefox, Safari, Opera and the latest versions of Internet Explorer.

Does HSYCO require the Adobe Flash player, Java, or any other plug-in on my browser?

No. While HSYCO is built in Java on the server, the client side is pure HTML5 and JavaScript.

Does HSYCO require an Internet connection?

Only if you need to use HSYCO from a remote location or if you need Internet access from HSYCO, for example to retrieve the weather information or to automatically adjust the internal clock.

How can I access my HSYCO server remotely?

You will need to directly access your HSYCO server, usually via an HTTPS (secure HTTP) connection. You will need a public IP (static or dynamic) and configure your router/firewall to forward the HTTPS requests to the HSYCO server in your LAN.

Is HSYCO a plug-and-play supervision system?

While HSYCO is quite easy to configure, it is an extremely flexible and powerful supervision solution. You will need to manually define the configuration parameters and design your own control pages. The process is effectively supported by several integrated development tools.

Is HSYCO a cloud service?

Absolutely not. HSYCO is a local dedicated server, that is completely under your own control and ownership. HSYCO doesn't depend on any portal or external infrastructure.

Can I access the HSYCO server's Ubuntu Linux operating system and make configuration changes?

Yes. HSYCO is not a closed appliance, but a Java core software running on a standard Linux operating system. You have full root access to the operating system, with no restrictions. Of course you are fully responsible for any configuration or version changes to the operating system. Tweaking the operating system will not void the hardware warranty, but we may charge you to restore a server to its original settings if asked to do so.

Does HSYCO run with any free Java Virtual Machine?

HSYCO is developed and tested using the Oracle Java Embedded Virtual Machine. This is a commercial product that we license from Oracle and install on each server. We do not test HSYCO with other virtual machines.

Is HSYCO automatically upgraded to the latest version?

No. Because HSYCO is used to implement critical supervision systems and runs continuously and unattended, it would be impossible for us to support an automatic upgrade process without risking the integrity and continuity of its services. You are responsible to upgrade the HSYCO software version, as well as the underlying operating system, based on your needs.

Development questions

Do I need to be a Java developer to create my supervision application using HSYCO?

No. You don't need any programming language at all to design a fully functional user interface. HSYCO has friendly Web-based tools to create logic rules between inputs and outputs. And you can use a simple declarative language called EVENTS, or write code in JavaScript and have access to most features. You only really need Java for advanced programming tasks, like writing your own device driver.

If HSYCO doesn't support a specific product I would like to integrate, is there anything I can do?

HSYCO offers a standard API to let you write device drivers. This is the same API we, the developers of HSYCO, also use when writing drivers. So you can do pretty much all we can do. Of course you can also ask us to develop custom drivers for you.

Can I heavily customize the user interface design?

Sure. Based on your level of HTML5, CSS and JavaScript skills you should be able to completely redesign the user interface appearance and behavior. Theoretically, with a little bit of extra support from us (this is beyond the scope of the official development documentation), you could even rewrite the Web-based client yourself.

Can I write code in HSYCO to access an external SQL database or other application services?

Yes. Anything you can do in a standard Java application, is also perfectly feasible in HSYCO. HSYCO doesn't restrict in any way what you can do with the standard JavaScript and Java languages. So, talking SQL, if your database system has a JDBC API, you will be able to use the SQL server from your Java, or JavaScript, code in HSYCO.

Installation Guide

HSYCO SERVER ships pre-installed with the latest version of the HSYCO software, on a Linux Ubuntu 12.04 LTS Server distribution stripped of all not-essential services.

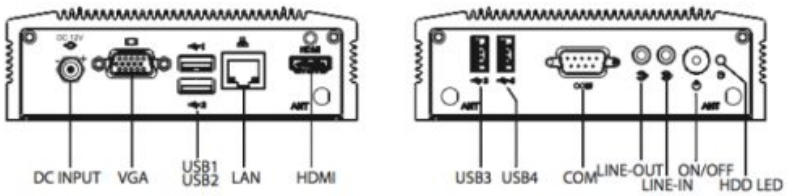
The Oracle Java SE Runtime Environment version 7 is also installed. No other software is required.

This chapter describes the access procedures for the basic management of the system and the structure of HSYCO's installation and configuration files.

Initial Set-Up

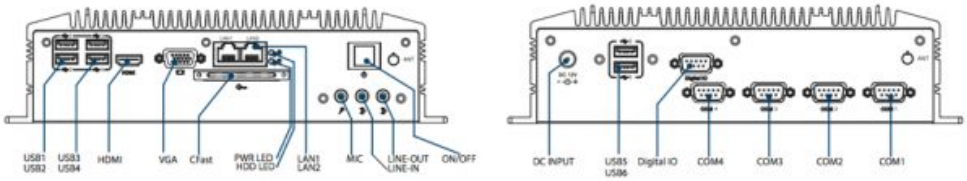
Hsyco comes with different hardware configurations:

- **HSYCO MINI - ARK1122**



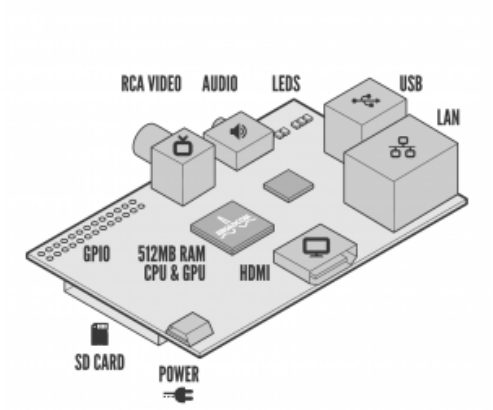
To start the server simply plug the power supply. HSYCO automatically starts, the execution of HSYCO is signalled by three consecutive high-pitch beeps.

- **HSYCO PRO - ARK2120**



To start the server simply plug the power supply. HSYCO automatically starts, the execution of HSYCO is signalled by three consecutive high-pitch beeps.

- **HYSCO PI - RASPBERRY PI**



To start the server simply plug the power supply. HSYCO automatically starts in two minutes (approx).

HSYCO SERVER is configured with the following default network configuration:

- IP address: 192.168.0.50
- Net mask: 255.255.255.0
- Default Gateway: 192.168.0.1
- DNS: 192.168.0.1

The IP address 192.168.0.50 will be used from now on in all the examples as the HSYCO SERVER address.

On the two Ethernet ports server, the port which is usually configured is the one next to the USB ports, shown as LAN1. The Ethernet port LAN2, on the right, works but it is not configured.

Access to the Manager

HSYCO SERVER is pre-configured with a Web-based toolkit of applications, the Manager, that gives you access to configuration and administration functions, including the ability to change the network settings and the operating system's password, HSYCO users management and the tools to create customized graphic user interfaces.

To access the system's management application, simply called Manager from now on, enter the following URL in your Web Browser:

```
https://192.168.0.50/hsycoserver/manager
```

The Manager runs on most modern Web browsers, including tablets like Apple’s iPad. It even runs on small screen devices like the iPhone or iPod touch, although a larger screen size is advisable for ease of use. Microsoft’s Internet Explorer versions 8 and older are not supported, and IE9 is not guaranteed to work, although it should in most cases.

HSYCO default configuration only accepts Web requests through the HTTPS secure protocol, while access via HTTP is disabled.

Furthermore, to protect against malicious service discovery robots, HSYCO does not answer to Web requests where only the server address is defined, for example https://192.168.0.50, but requires an extended URL, which must include an access key, called URLKey, defined in the general system configuration parameters.

> The factory default URLKey is hsycoserver.

The URLKey is not a secret password, but only an additional protection feature.

Because the SSL certificate used to secure the communication between the Web Browser and HSYCO SERVER is self-generated, you will be asked to accept the server's certificate.

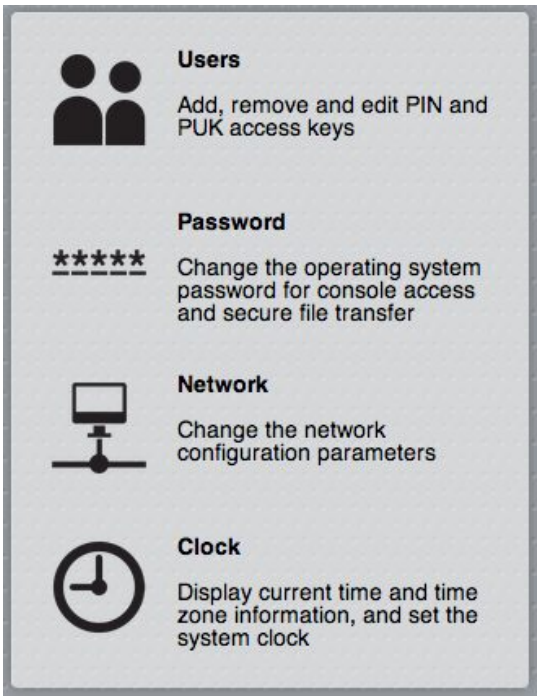
On the iPod touch, iPhone or iPad press Continue, on other browsers follow the dialog boxes to permanently accept the certificate.

You will see the PIN and PUK authentication page. Enter the default PIN and PUK, and press OK:

```
PIN: 0000
PUK: 00000 0000 00000
```

> The default pre-configured PIN and PUK are the same for all HSYCO SERVERS, to let you conveniently access the user interface during the initial installation. Both the PIN and PUK should be changed immediately to different, hard to guess numbers.

Press the Admin icon in the Manager's main menu to access the Users, System Password, Network and Clock settings functions.



HSYCO is now ready to be configured!

For further information about the Manager toolkit usage please visit the Configuration Guide.

SSH Access to the Operating System

The configuration of the operating system only needs one administrator user, whose login is **hsyco**.

The password initially configured for the **hsyco** user is **hsycoserver**.

> In order to guarantee the system safety and to prevent any unauthorized access it is essential to modify the access password of the HSYCO user before installing the system onto any not-secure LAN network.

The SSH secure protocol is used to access the operating system's console and, if needed, to manage files. The SSH protocol encrypts all traffic and can be safely used for remote management. If the network has a firewall it might be necessary to open port **TCP 22** and configure traffic forwarding towards the HSYCO SERVER IP address.

HSYCO is installed in the home directory of the **hsyco** user, **/home/hsyco**.

In this directory, the subdirectory **./hsyco** contains the software, the configuration files, custom Java programs, logs and recorded images from cameras.

Several software programs are available for the SSH access on Microsoft Windows, in particular Putty to access the console, and Bitvise Tunnelier [1] and WinSCP to access files through SFTP.

On Apple Mac OS X the SSH access is integrated in the operating system and directly usable through Terminal, while Cyberduck is one of the most widespread utilities to manage files through SFTP.

On Ubuntu OS the SSH access is integrated in the operating system and accessible through Terminal, while Filezilla is a good solution for managing file through SFTP.

Start and Stop the HSYCO Server

HSYCO starts automatically with the operating system when the server is turned on. The execution of HSYCO is signaled by three consecutive high-pitch beeps.

HSYCO also restarts automatically with every change to the **hsyco.ini**, **hsyco.jar** and **com/hsyco/user.class** files. In this case the restart is also signaled by three beeps.

To manually stop HSYCO, access the console via SSH:

```
ssh hsyco@192.168.0.50
```

and execute the command:

```
sudo stop hsyco
```

To start HSYCO, execute:

```
sudo start hsyco
```

Finally, to verify the current state of HSYCO execute:

```
sudo status hsyco
```

If HSYCO is running, the answer to the command will be something like:

```
hsyco (start) running, process ...
```

SMB Access to Files

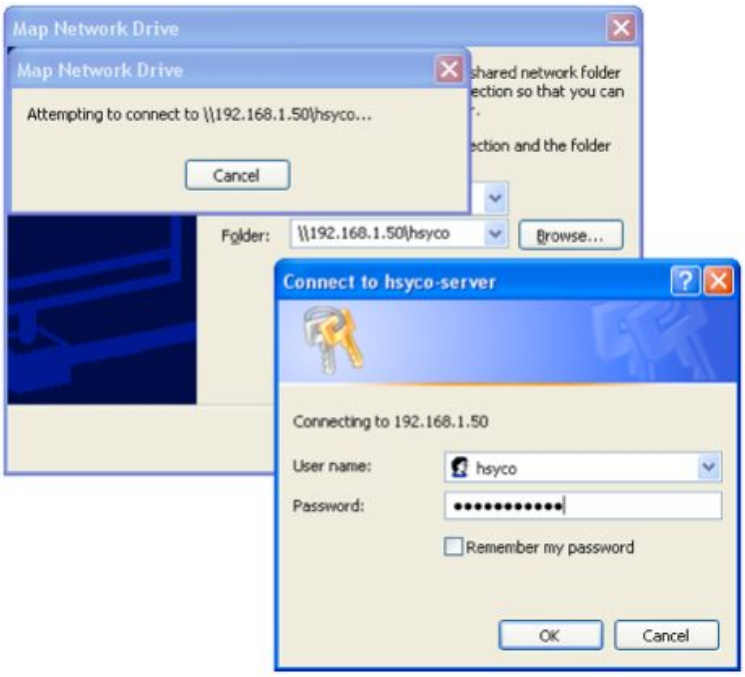
To guarantee an easy access to files, HSYCO SERVER makes the main directory available through a standard *file server* (network disk) service. The name of the service is *hsyco*. To access the file server you should use the same hsyco user and password provided to access the operating system.

Access to Files on Microsoft Windows

In order to connect a Microsoft Windows system to the HSYCO SERVER shared network folder, select **Tools > Map network drive** from the Explorer menu.

Type *\\192.168.0.50\hsyco* in the field **Folder** and press **Finish**.

Then, insert the user name (*hsyco*, as before) and password (initially *hsycoserver*, that you can modify as already described) and press OK.

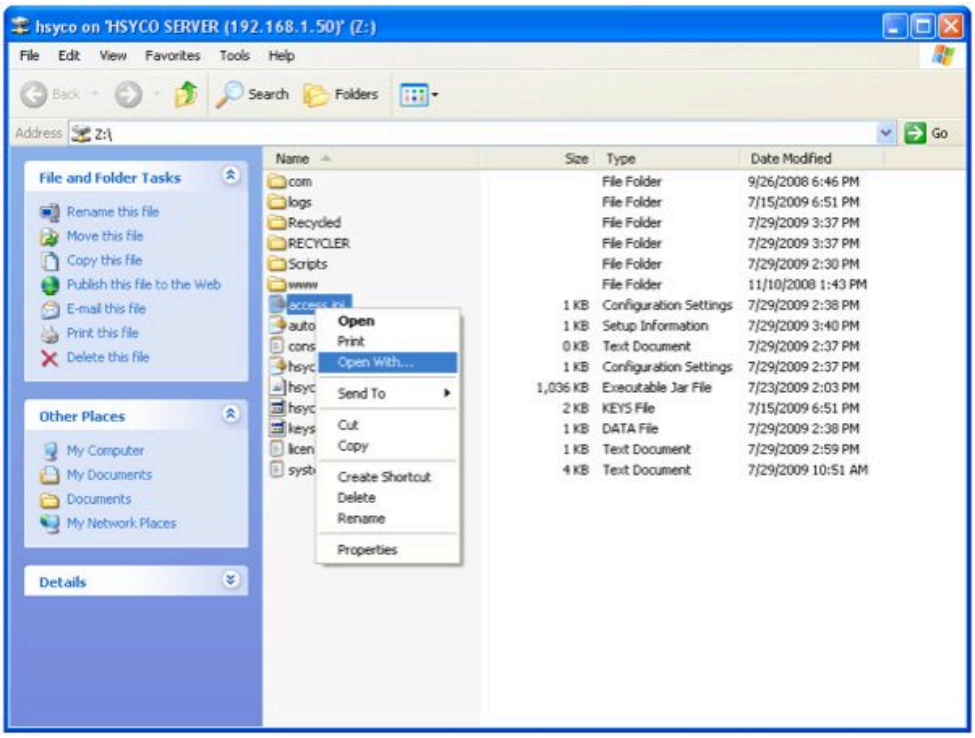


All the HSYCO files are available and accessible in the shared folder window, and you have full read and write permissions on them.

> In order to guarantee the highest safety and protection of HSYCO from any unauthorized access, you should disconnect the shared folder when not in use and avoid saving[Note 8] the password for automatic access.

To manually edit the content of the files directly from the shared folder, use the **WordPad** application, and not the **Notepad** application, the latter cannot correctly manage the text files format.

To open the files using *WordPad*, right click on the file, then select **Open with...** and *WordPad*, if it appears in the menu, or go to *Programs* and choose WordPad in the list of available applications.

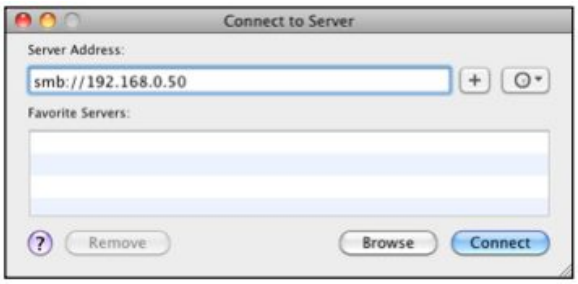


Note 8

Do not select the *Remember my password* option in the window where the username and password are required.

Access to Files on Mac OS X

In order to connect an Apple Mac OS X system to the HSYCO SERVER file server, select **Go > Connect to Server** from the **Finder** menu.



In the **Server Address** field insert *smb://192.168.0.50* and press **Connect**.

Then, insert the username (*hsyco*) and password (initially *hsycoserver*, that you can modify as already described) and press **Connect**.

> In order to guarantee the highest safety and protection of HSYCO from any unauthorized access, we advise you to disconnect the shared folder when not in use and not to save the password in the Mac *keychain*.

Software License Key

The license.txt file contains the software license key needed to unlock HSYCO. The first time HSYCO is started, or in case this file is removed before start, HSYCO creates a new file containing the encrypted code that univocally identifies the hardware where the software runs. In order to allow the correct execution of HSYCO the file should contain the software license key, issued by Home Systems Consulting, specific to every single computer.

> The hardware identification code is based on the hardware components installed on the server. For example, by replacing or adding network cards the code changes and therefore the software license key will be no longer valid for the new configuration.

If the license key is not in the file, or if it is not correct, HSYCO disables all the access functions to the field devices, keeping only the Web Server working. It will be therefore possible to access the Web pages, but not to execute any action on or interface to the systems connected to HSYCO.

> The possible error messages related to the software license are written in the daily log files.

The content of license.txt, generated by HSYCO and containing only the hardware identification code is the following:

```
#Generated by HSYCO CONTROLLER 3.3.0
#Tue Dec 10 11:49:06 CET 2013
hwid=2080334b6283f450283bd4bbc3389bf66dc03dda
```

To unlock HSYCO features, replace the license.txt file generated by HSYCO at the time of its first execution with the license.txt version with the key provided by Home Systems Consulting. The file and the key are automatically recognized within a few seconds, it is therefore not necessary to restart HSYCO.

This is an example of the complete software license key file:

```
#Generated by HSYCO LICENSE MANAGER 3.3.0
#Tue Dec 10 12:20:00 CET 2013
name=Company Name
hwid=2080334b6283f450283bd4bbc3389bf66dc03dda
key=302D021500898B20F13ABE66DF3D2577809C6A53343A23A5CD0214540FD580A1E3DA1CBEB18D11E98CAAE453D402BF
```

The Configuration Console

HSYCO Server provides a simple low-level maintenance console, used to reboot the system, reset its settings or restore the whole system to the original factory configuration.

This console is not used for normal operations.

The console is available using a keyboard and a monitor connected to the VGA/DVI/HDMI video out connector.

```
Server Configuration Console

1) Reboot the system
2) Halt the system
3) Stop HSYCO Server
4) Start HSYCO Server
5) Restore network configuration to factory defaults (192.168.0.50)
6) Delete all PINs and file server password and restore default values
7) Restore database to factory defaults. All data will be lost
8) Restore system software to factory defaults. All data will be lost
Type a function number and press Enter [1, 2, 3, 4, 5, 6, 7, 8]:
```

Type the desired function number and Enter, then press Y to confirm after the beeps (the system will sound a number of beeps, corresponding to the function number; this should allow you to access the console functions using the keyboard and audio feedback only, without a monitor).

Function 1 performs the operating system shutdown, followed by a reboot.

Function 2 performs the operating system shutdown and halts the system. You should turn it off then back on to restart.

Function 3 and 4 are used to temporarily stop the HSYCO Server process, and to restart it.

> Use functions 5 to 8 with caution. These function will delete configuration settings, user data, and security settings of the system. They could allow a local user with physical access to the server, unrestricted access to your data.

Function 5 restores the factory default network configuration. If you make a mistake while setting the network parameters, preventing access to the server via the network connection, you can use this function to reset the configuration and regain access to the server.

Function 6 removes all users' PIN and PUK codes, restoring the default 00000 administrator enabled PIN. It also reset the operating system password to its default.

Function 7 deletes all data in the HSYCO database, including persistent variables.

Function 8 performs a full factory reset, returning the system to the HSYCO version and settings as shipped, removing all custom data, configuration and programs.

Settings

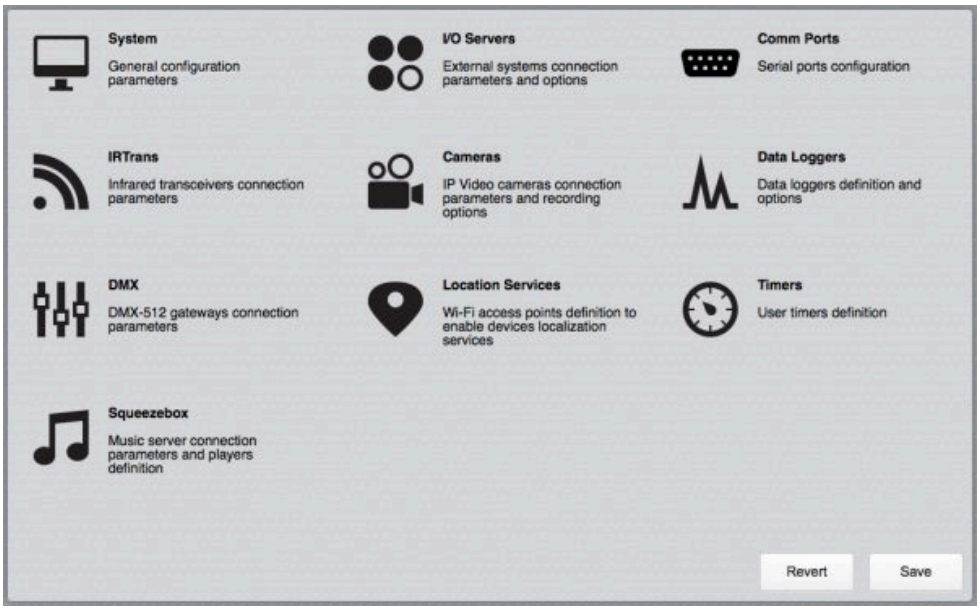
HSYCO can be configured using the Settings application in the Manager.

Settings stores all configuration parameters in the hsyco.ini file.

You can copy this file to save the current configuration, or manually change it. In this case, be sure to reopen the Settings application after the changes have been saved.

The configuration is read at start-up, so any changes become effective only after restarting the HSYCO process.

HSYCO is factory configured to automatically restart when hsyco.ini is saved.



When you make changes using Settings, the hsyco.ini file will be overwritten when you press the Save button.

You can't return to the previous configuration once it is saved.

The Revert button allows you to reload the current configuration if you have made changes in Setting that have not yet been saved.

Settings parameters are grouped in several sections.

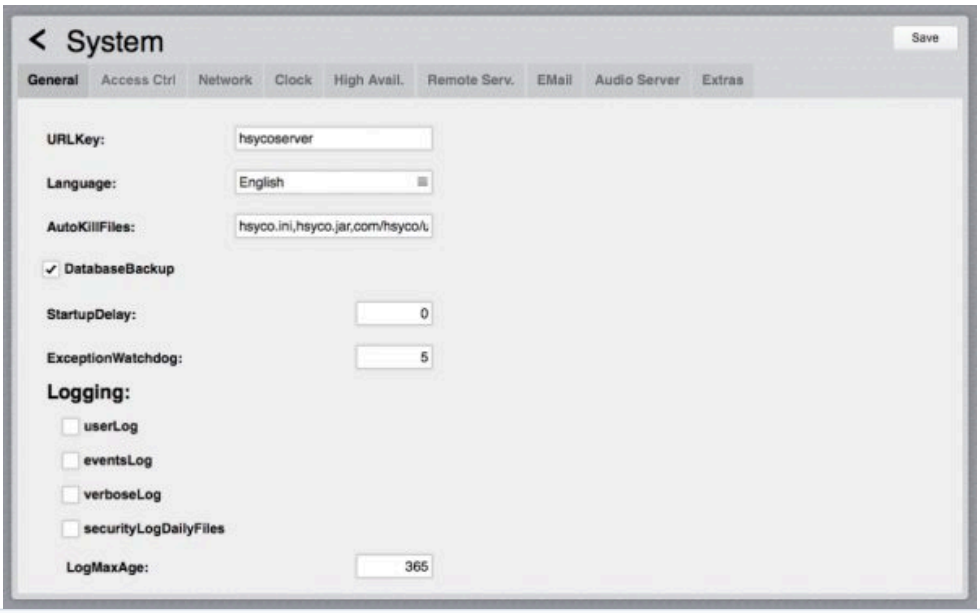
You can make changes to any parameter, even in different sections, and then save all changes together.

System

The Systems section contains all general configuration parameters, including vital parameters affecting system's security and reliability.

These parameters are further split in several sub-sections.

General




```
URLKey
```

Default: hsycoserver

Format: string of at least 8 characters

To protect against malicious service discovery robots, HSYCO does not answer to Web requests where only the server address is defined, for example https://192.168.0.50, but requires an extended URL, which must include an access key, called URLKey.

💡 **The URLKey must be at least 8 characters long.**

It is possible to specify more than one URLKEY, separated by a comma; in this case all the specified URLKEYs will be valid for the web access to HSYCO.

The factory default URLKey is hsycoserver.

The URLKey is not a secret password, but only an additional protection feature.

```
Language
```

Default: en

Format: cn | en | fr | it

Some I/O Servers and other core services use localized text messages. This parameter defines the default system language for these services.

```
AutoKillFiles
```

Default:

Format: list of file names separated by commas

This parameter is usually set as: "hsyco.ini,hsyco.jar,com/hsyco/user.class" forcing the automatic restart after the changes

of the three files listed.

```
DatabaseBackup
```

Default: false

Format: false | true

When set to true, a hot backup of the core database is automatically executed on a daily basis, saving data in the data_backup directory, and overwriting the previous backup files.

```
StartupDelay
```

Default: 0

Format: 0 or positive integer number of seconds

When set to a positive number, the HSYCO server will wait for the specified number of seconds at start-up before becoming active.

This start-up delay could be useful to prevent HSYCO from starting before other peripherals or devices, like external storage systems, are completely initialized.

```
ExceptionWatchdog
```

Default: 5

Format: positive integer number

After N uncaught Java execution exceptions, the HSYCO server will be killed and restarted.

Set to 0 to disable the exception watchdog.

```
EventsLoadTimeout
```

Default: 20

Format: positive integer number of seconds

When events.txt is saved, If any code section takes more than the specified amount of time to execute during the test run and INIT events sections, HSYCO will abort the JavaScript engine, rename events.txt to events_unsafe.txt and, a few seconds later, will restart the whole HSYCO server engine to protect the integrity of the system.

If you have code in events.txt that requires more than the default 20 seconds to run, you should set a larger value for EventsLoadTimeout.

```
userLog
```

Default: false

Format: false | true

Set to true to enable the log of the Java methods called in user.java or of the commands defined in the EVENTS programming environment.

```
eventsLog
```

Default: false

Format: false | true

If true, the log of events received from field devices is enabled, for example the events related to the IO Servers.

```
verboseLog
```

Default: false

Format: false | true

If true, the extended log is enabled.

It is useful for debugging, or during the advanced customization phase or the development of Java code.

If silent, only the most serious errors will be written on the file and all the other messages will be disabled, including the events received from field devices.

```
securityLogDailyFiles
```

Default: false

Format: false | true

When set to true, the security logs are written in daily files named MMDD-security.log.

```
LogMaxAge
```

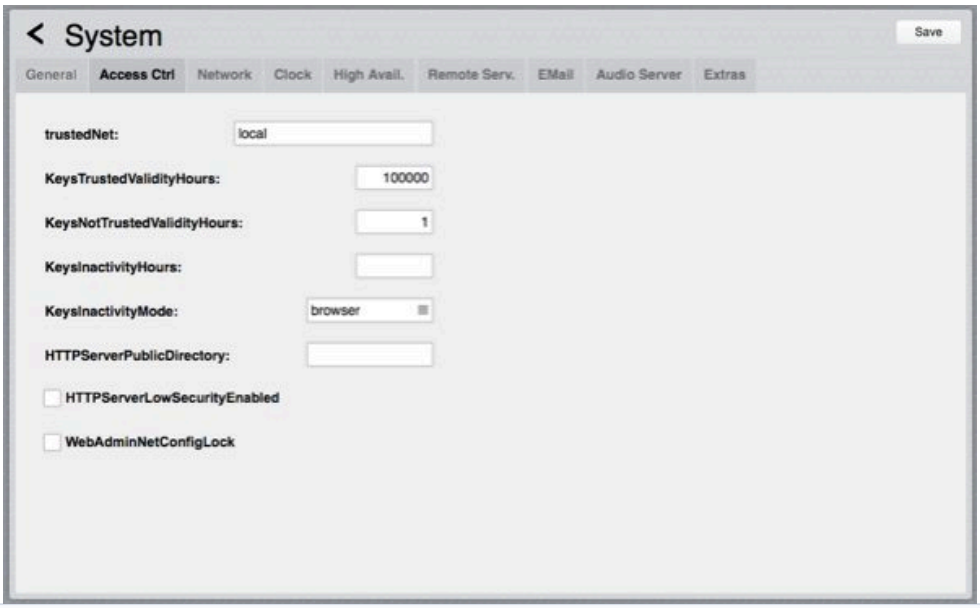
Default: 0

Format: 0, or positive integer number

Log files are automatically deleted when older than the number of days defined with this parameter.

When set to 0 or not defined, log files are not deleted automatically.

Access Control




```
trustedNet
```

Default: local

Format: local, or nn.nn.nn.nn-nn.nn.nn.nn

IP addresses that define the group of network addresses belonging to the secure local network.

It is possible to enter multiple, comma-separated IP ranges or individual IP addresses, or simply enter “local”, so that HSYCO will assume all IP addresses in the LAN as trusted.

The Web clients connecting to HSYCO from these addresses are subject to the time-out defined in KeysTrustedValidityHours, which is usually longer than the one used for all the other IP addresses, defined in KeysNotTrustedValidityHours.

```
KeysTrustedValidityHours
```

Default: 24

Format: integer > 0 or hh:mm

Login time-out in hours for the connections from trusted IP addresses.

Set on a very high value, e.g. 100000, to practically avoid the time-out of sessions.

It can also be set using the hh:mm hours and minutes format.

```
KeysNotTrustedValidityHours
```

Default: 1

Format: integer > 0 or hh:mm

Login time-out in hours for not trusted IP addresses.

It can also be set using the hh:mm hours and minutes format.

```
KeysInactivityHours
```

Default: 1

Format: integer > 0 or hh:mm

Inactivity time-out in hours.

This is an optional parameter; if not set there will be no inactivity time-out and the user session will be automatically logged out based on the login time-out only.

It can also be set using the hh:mm hours and minutes format.

```
KeysInactivityMode
```

Default: browser

Format: browser | cameras | commands

Inactivity time-out mode.

This is an optional parameter.

It is significant only when KeysInactivityHours is also defined.

In browser mode, having the Web browser open on the HSYCO page will keep the session alive.

In cameras mode the session will remain alive only when sending commands or watching cameras.

In commands mode the session will stay alive only if commands are sent before the inactivity timer expires.

```
HTTPServerPublicDirectory
```

Default: browser

Format: directory name

If defined, enables a simple Web server that serves files, without any parsing, via HTTP and HTTPS, only to clients in the trusted range of IP addresses.

This parameter sets the name of the directory, under the www root Web directory, used for the public files.

If, for example, you have HTTPServerPublicDirectory=public, and an HTML file named home.html in the public directory, then the https://192.168.0.50/public/home.html URL will point to that file.

As you see, you shouldn't add the URLKey in public URLs.

```
HTTPServerLowSecurityEnabled
```

Default: false

Format: true | false

Usually, the HTTP server is only active to let cameras and PBX systems send motion detection and calls notifications.

To avoid the authentication keys, PIN and PUK codes and all traffic to be transmitted in the clear, the HTTP protocol is not normally used for Web access to HSYCO.

Set the parameter to true only when you want to enable the not-secure HTTP protocol for Web access to HSYCO.

```
WebAdminNetConfigLock
```

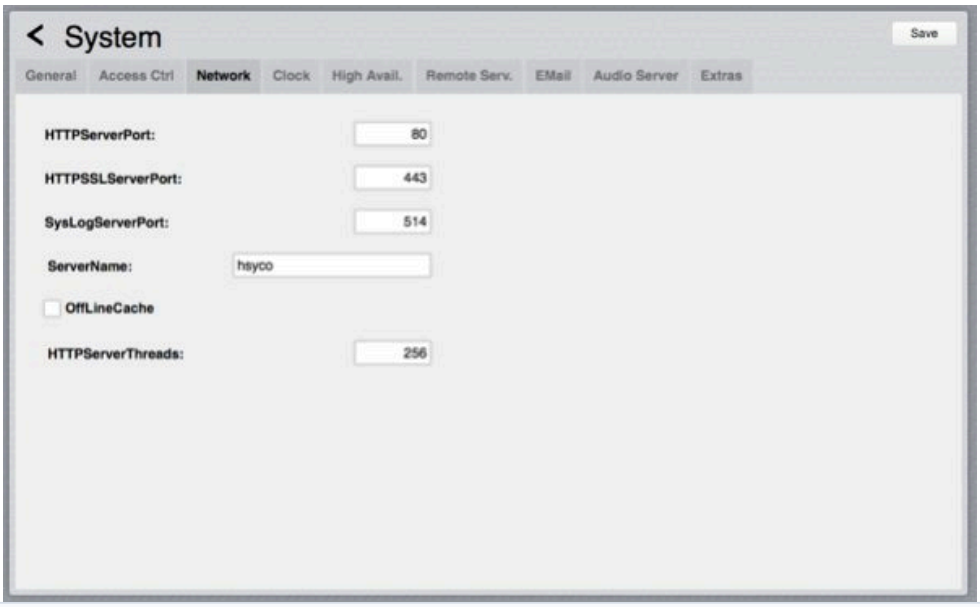
Default: false

Format: true | false

If true, the Web network settings functions are disabled.

This parameter should be set to true once you expect no changes in the network configuration of HSYCO Server.

Network




```
HTTPServerPort
```

Default:

Format: integer number < 65535

TCP/IP port of HSYCO HTTP Web Server. If not defined, the HTTP server is not enabled.

Normally set to 80.

```
HTTPSSLServerPort
```

Default:

Format: integer number < 65535

TCP/IP port of HSYCO HTTPS Web Server.

If not defined, the HTTPS server is not enabled.

Normally set to 443.

```
SysLogServerPort
```

Default:

Format: integer number < 65535

TCP/IP port of HSYCO SYSLOG Web server.

The SYSLOG server is used to receive from the Access Points (AP) the information to determine which AP every user in the Wi-Fi LAN is using to connect.

If not defined, the SYSLOG server is not enabled.

It is usually set to 514.

```
ServerName
```

Default: hsyco

Format: domain name

Name used to generate the SSL certificate.

It is necessary for the cryptography of the HTTPS Web traffic and should correspond to the domain name through which the HSYCO server is accessed via the Internet.

The certificate is saved in the hsyco.keys file.

When HSYCO is started, if this file is not available or its name doesn't match the ServerName parameter in hsyco.ini, a new SSL certificate is automatically generated according to the name defined in ServerName.

```
OffLineCache
```

Default: false

Format: false | true

HSYCO implements the off-line cache feature of the Web browsers that support the HTML 5.0 standard.

To enable the off-line storage on the Web browser of all HSYCO HTML static content, set this parameter to true.

```
HTTPServerThreads
```

Default: 256

Format: positive integer number

Sets the total number of processing threads for the HTTP and HTTPS internal web servers.

You should change this parameter only when you have a very large number of active web clients.

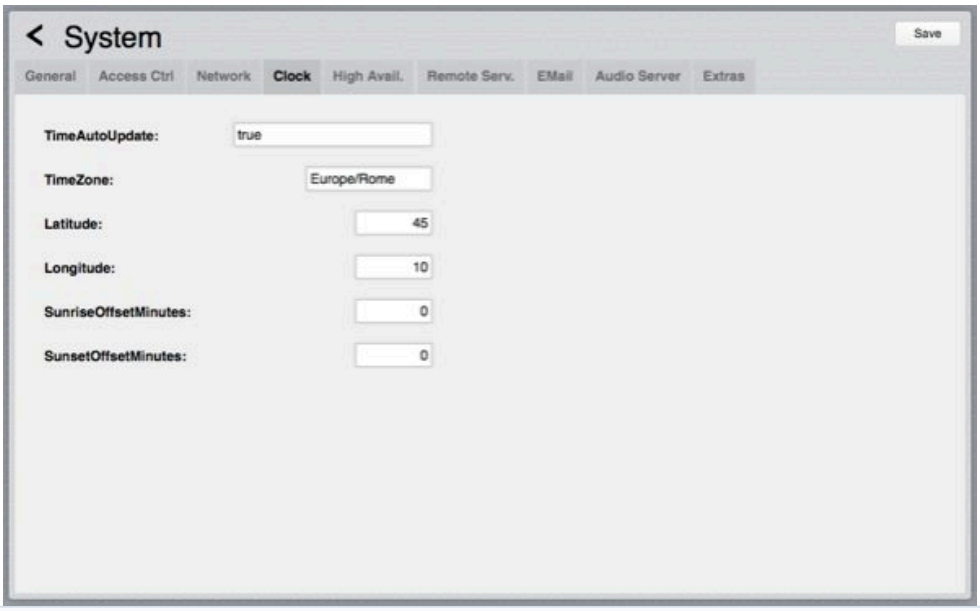
Web browsers normally use only a few (up to about 6) concurrent connections for each session, and the default should be fine to support up to at least 40 active clients.

When the HTTP and HTTPS servers run low of threads, some clients could see their requests rejected, causing page load errors or becoming unresponsive.

In this case you should see “too many connections” error messages in HSYCO’s log files.

💡 **Increasing this parameter also increases memory and I/O resources requirements to the underlying operating system.**

Clock




```
TimeAutoUpdate
```

Default: true

Format: false | true | NTP server name/address

HSYCO automatically sets the local date and time by polling Network Time Protocol (NTP) servers.

To disable this feature, set this parameter to false.

To use a specific NTP server, set this parameter to the name or IP address of the NTP server.

```
TimeZone
```

Default: true

Format: the time zone id

Set the current time zone.

The time zone information is used to obtain the correct local time and automatically adjust for daylight saving time.

If this parameter is not set, HSYCO will use the operating system's time zone settings.

The time zone id is either in the form "Area/Location", or a generic "GMT+NN", "GMT-NN" format that simply defines the hours offset from UTC time.

When the "Area/Location" format is used, the clock is automatically adjusted for daylight saving time.

The TimeZone value should have no spaces; replace spaces in area or location names with underscores, for example "America/New_York".

When HSYCO starts, you can check the message.log file to see the actual time zone and DST setting, just after the start-up message, for example:

```
2013.07.10 15:28:30.113 - HSYCO Ver. 3.3.0 Build 0112 (USER Ver. No User Code) started
2013.07.10 15:28:30.425 - Time zone is: Europe/Rome (Central European Summer Time)
```

```
Latitude
```

Default: 0

Format: positive or negative numerical value, decimals separated by "."

Latitude in decimal degrees (positive for the Northern Hemisphere), to calculate the sun position and sunrise/sunset time.

```
Longitude
```

Default: 0

Format: positive or negative numerical value, decimals separated by "."

Longitude in decimal degrees (positive for the Eastern Hemisphere), to calculate the sun position and sunrise/sunset time.

```
SunriseOffsetMinutes
```

Default: 0

Format: positive or negative integer number

Deviation in minutes from the astronomical sunrise time (in advance if negative, delayed if positive).

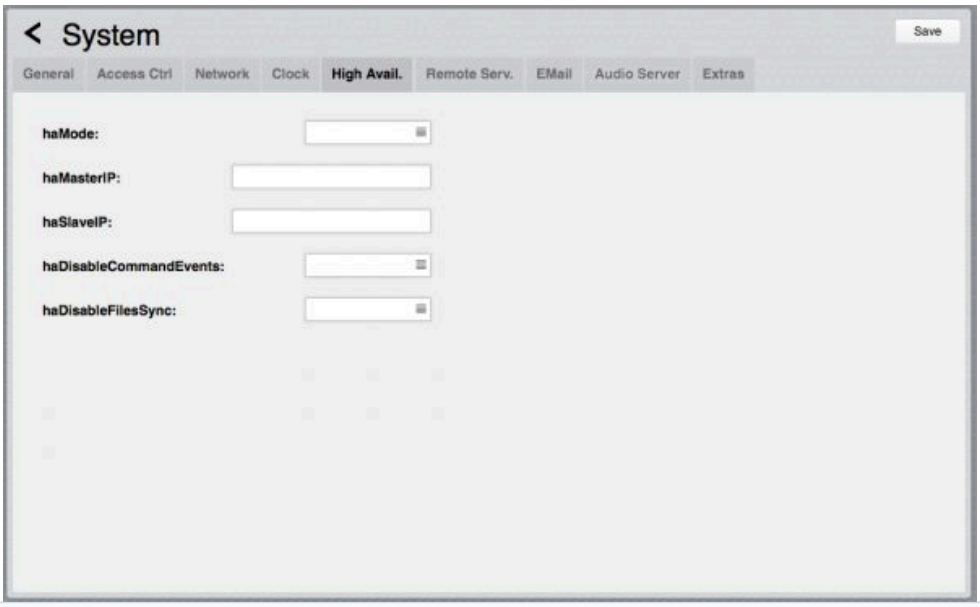
```
SunsetOffsetMinutes
```

Default: 0

Format: positive or negative integer number

Deviation in minutes from the astronomical sunset time (in advance if negative, delayed if positive).

High Availability




```
haMode
```

Default:

Format: master | slave

Enables the high availability configuration of HSYCO.

The high availability configuration uses two different HSYCO systems, one is configured as the master system, the other as slave.

```
haMasterIP
```

Default:

Format: IP address: nnn.nnn.nnn.nnn

Mandatory parameter for the high availability configuration.

It specifies the IP address of the HSYCO master system.

```
haSlaveIP
```

Default:

Format: IP address: nnn.nnn.nnn.nnn

Mandatory parameter for the high availability configuration.

It specifies the IP address of the HSYCO slave system.

```
haDisableCommandEvents
```

Default: true

Format: false | true

Optional parameter for the high availability configuration.

If true, command methods and actions in Java and EVENTS are automatically disabled on the HSYCO Server that is not active (for example the system defined as slave is not active when the master system works properly) for the following systems: Audio, Cameras, Comm ports, DMX, I/O Servers, IRTrans, Modbus (write command are disabled) Squeezebox, URL.

```
haDisableFilesSync
```

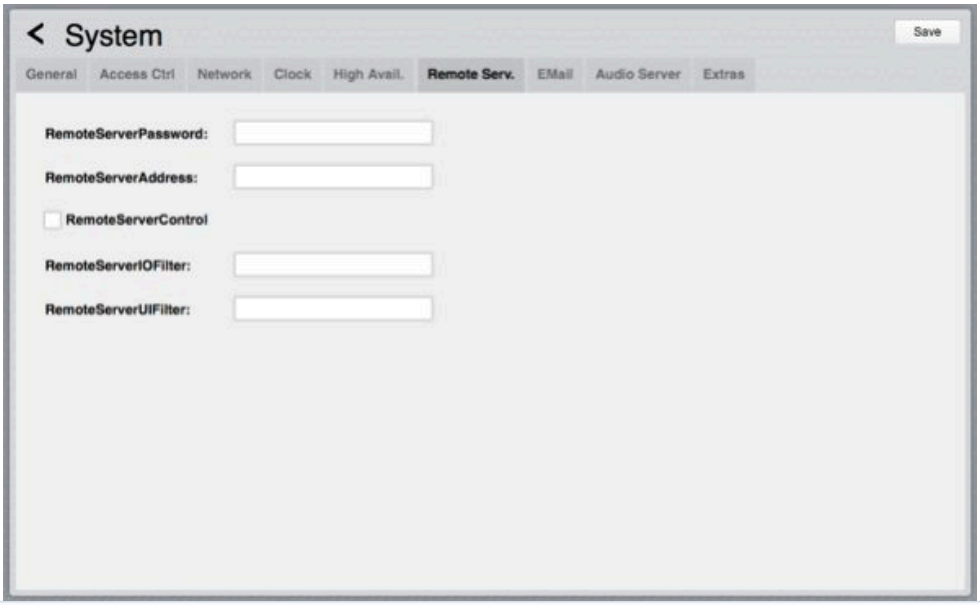
Default: false

Format: false | true

Optional parameter for the high availability configuration.

If true, files changes on the master will not be synched to the slave server (changes to access.ini, scheduler.ini and timers.data will not be synched too).

Remote HSYCO Servers




```
RemoteServerPassword
```

Default:

Format: string (letters and numbers)

Set this password to enable this HSYCO server to be accessed by another HSYCO server via the HSYCO Remote I/O Server (see the HSYCO Remote I/O Server Application Note for further information).

```
RemoteServerAddress
```

Default:

Format: comma separated list of IP addresses, or *

Set to the IP addresses of the remote HSYCO servers that should be allowed to access this HSYCO server.

Set to * to allow any remote IP address.

```
RemoteServerControl
```

Default: false

Format: false | true

Set to true to allow the remote server to modify the data points values of this HSYCO server.

```
RemoteServerIOFilter
```

Default:

Format: comma separated list of I/O Servers IDs

If you want to limit remote access and control to just some of the I/O Servers defined in this HSYCO, use this parameter to list the I/O Servers that should be available to the remote HSYCO system.

Filtering out I/O Servers that are not used remotely could have a positive impact on performance and network traffic.

```
RemoteServerUIFilter
```

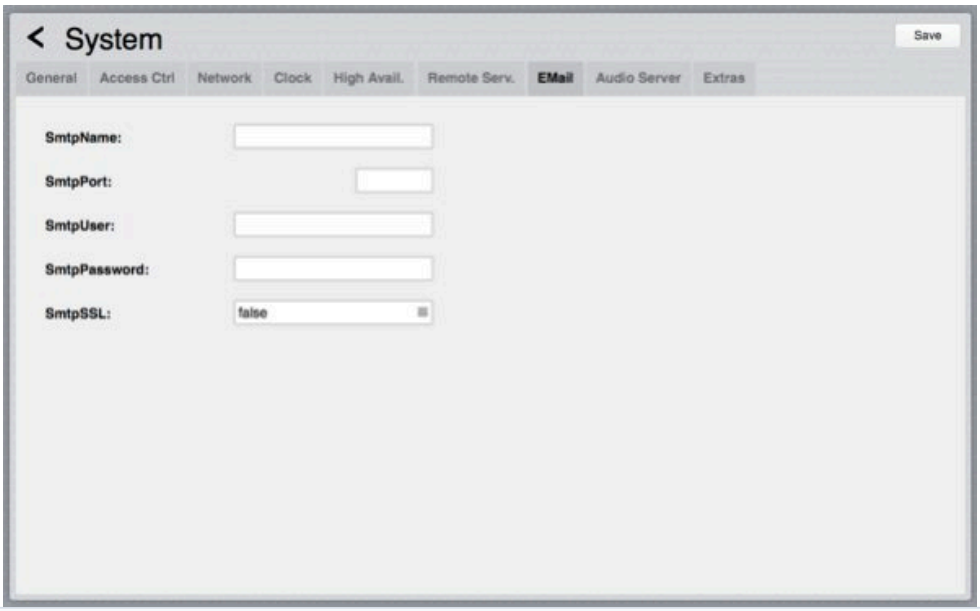
Default:

Format: comma separated list of I/O Servers IDs

If you want to limit remote access and control to just some of the user interface objects defined in this HSYCO, use this parameter to list the objects' IDs prefixes (a prefix is the first part of the ID, before the first dot character, if you are using a dot-separated notation; use the full ID if not using the dot notation) that should be available to the remote HSYCO system.

Filtering out UI objects that are not used remotely could have a positive impact on performance and network traffic.

Email




```
SmtpName
```

Default:

Format: SMTP server name or IP address

SMTP server name or numeric IP address.

This parameter is used by the MAIL action and sendMail() method to send email messages through a specific authenticated and secured SMTP account rather than directly to the recipient's SMTP server.

```
SmtpPort
```

Default: 25, 465 or 587

Format: positive integer number

SMTP server's port number, if different from the default port.

The default port is:

- 25 for not encrypted traffic
- 465 for SMTP over SSL
- 587 for ESMTP.

```
SmtpUser
```

Default:

Format: string

Account's username to authenticate the connection to the SMTP server.

```
SmtpPassword
```

Default:

Format: string

Account's password to authenticate the connection to the SMTP server.

```
SmtpSSL
```

Default: false

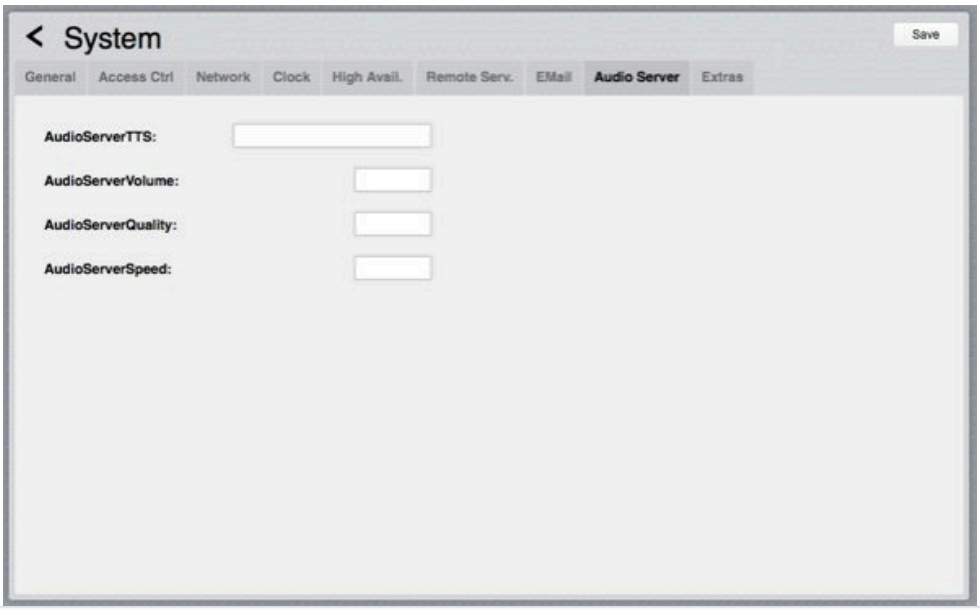
Format: false | true | esmtp

Set to "true" for SMTP server that support SMTP over SSL (default port is 465).

Set to "esmtp" for SMTP server that support ESMTP (default port is 587).

Set to "false" for SMTP servers that don't support traffic encryption.

Audio Server




```
AudioServerTTS
```

Default:

Format: string

This parameter is only needed if you are using a text-to-speech engine that is not the default one for the operating system.

See the Audio and Public Announcement section for additional information.

```
AudioServerVolume
```

Default:

Format: integer number

Change the default output volume for pre-recorded audio files and the text-to-speech engine.

```
AudioServerQuality
```

Default:

Format: integer number

Change the default quality for the text-to-speech engine.

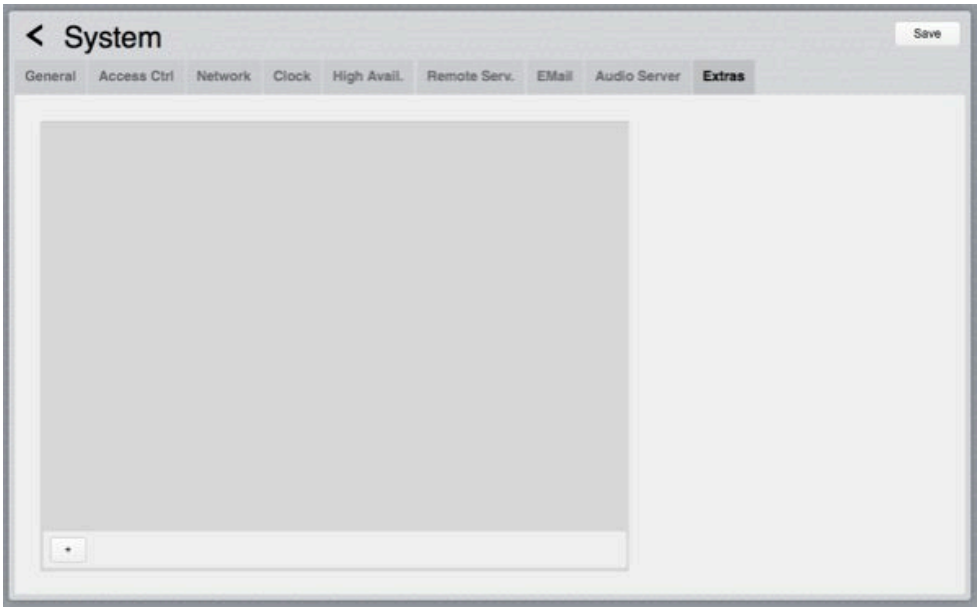
```
AudioServerSpeed
```

Default:

Format: integer number

Change the default speed for pre-recorded audio files and the text-to-speech engine.

Extras

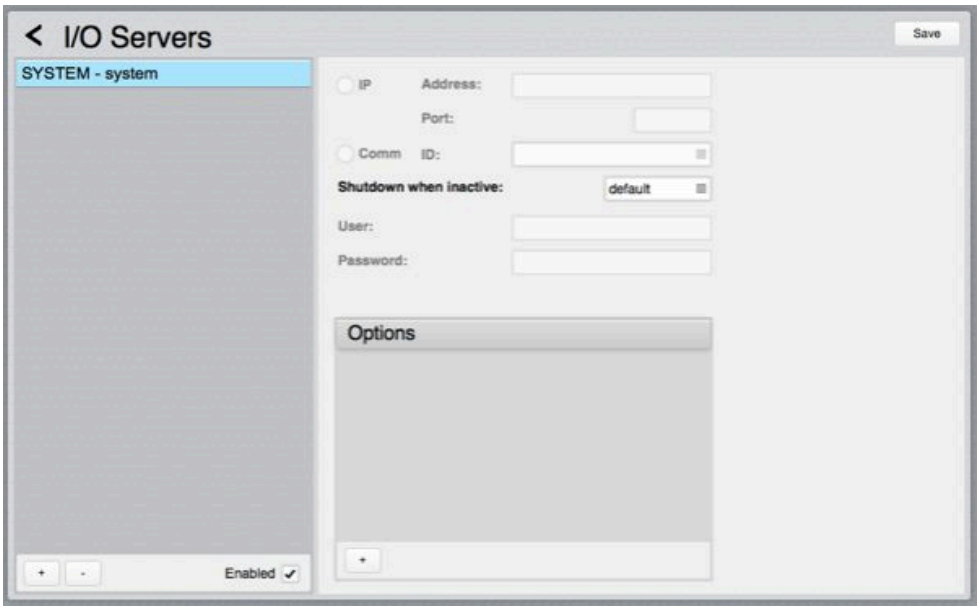


This section is used to set other not standard parameters that should be saved in hsyco.ini, like application-specific parameters.

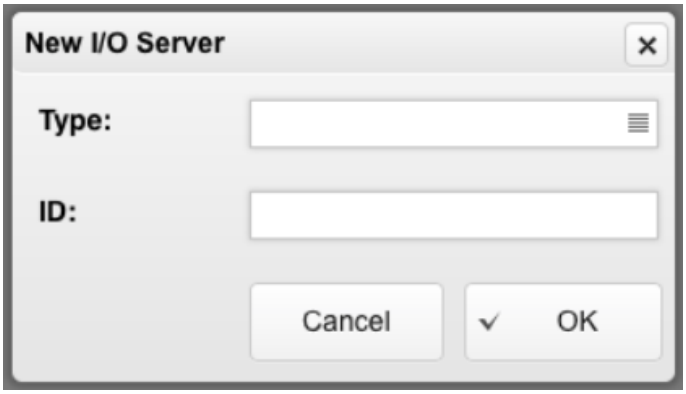
I/O Servers

HSYCO Server interfaces to several field sub-systems, including lighting and automation systems, video cameras, HVAC controllers, security systems, and other special-purpose devices. HSYCO supports several standard protocols, like BACnet, DALI, DMX, KNX, Modbus, as well as quite a few proprietary protocols.

Most of these field systems are represented inside HSYCO as “I/O servers”. An I/O server is a software module that offers a standardized abstraction layer so that even very different systems can be interfaced using a consistent naming convention and programming model.



Each I/O server must have a unique ID. You can add a new I/O server pressing the + button.



Select the type and enter the server's unique ID.

Already defined I/O servers could be deleted with the - button or disabled without removing their configuration parameters by unchecking the Enabled checkbox.

Each server type may require different configuration parameters, like IP address and port, serial port id, or authentication information.

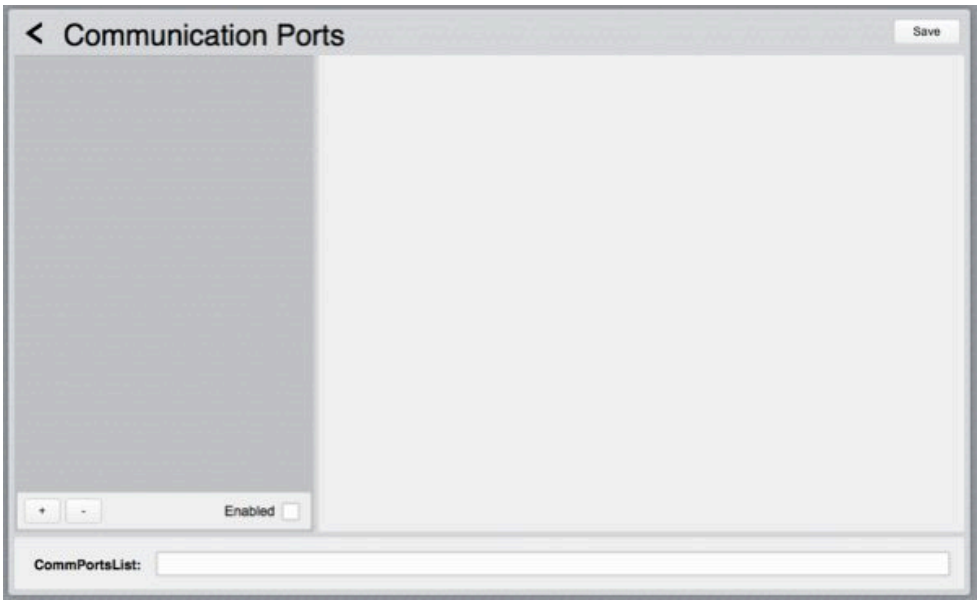
Refer to the Application Notes documentation for detailed configuration and interfacing information for I/O servers.

The "Shutdown when inactive" parameter, that is available for all I/O server types, is only used in high availability configurations.

Set it to true to automatically stop the I/O server when the HSYCO server (master or slave) is not the active system. Set to false to keep the I/O server running even when the HSYCO server is not the active system.

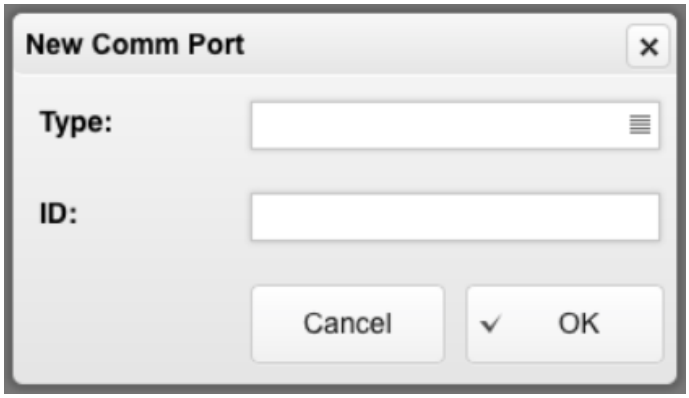
File:ToDo LINK ALL'ELENCO DEGLI I/O SERVERS

Communication Ports



The Comm Ports section is used to define all serial ports and their configuration parameters.

Each port must have a unique ID. You can add a new ports pressing the + button.



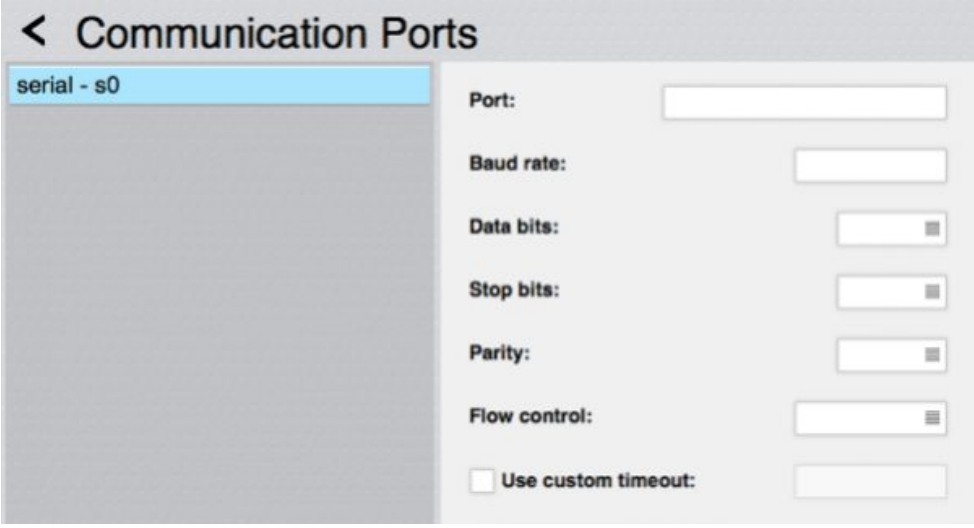
Select the type and enter the port's unique ID.

Already defined ports could be deleted with the - button or disabled without removing their configuration parameters by unchecking the Enabled checkbox.

The different types of comm ports have different configuration parameters.

The CommPortsList is an optional parameter, and should not be required unless the operating system fails to automatically detect the local serial ports. In this case, list all local serial ports using the full operating system port names, for example: /dev/ttyS0,/dev/ttyS1.

Serial ports



To configure one of the real serial ports of the server, or a USB serial adapter, create a port of type “serial”.

```
Port
```

Default: mandatory

Format:

Complete or partial system name of the associated communication port. In the Linux based architecture of the HSYCO servers this is the naming convention:

- COM1: ttyS0 or /dev/ttyS0
- COM2: ttyS1 or /dev/ttyS1
- COM3: ttyS2 or /dev/ttyS2
- COM4: ttyS3 or /dev/ttyS3

For USB connected devices:

- 1st USB device: ttyUSB0 or /dev/ttyUSB0
- 2nd USB device: ttyUSB1 or /dev/ttyUSB1
- ...etc etc

```
Baud rate
```

Default: mandatory

Format:

Port speed.

```
Data bits
```

Default: mandatory

Format:

The number of bits can be 8, 7, 6 or 5. It should be set to 8 in most cases.

```
Stop bits
```

Default: mandatory

Format:

The number of stop bits can be 1, 1.5 or 2.

```
Parity
```

Default: mandatory

Format:

Parity can be none, odd, even, mark or space.

```
Flow control
```

Default: mandatory

Format:

Flow control can be none, XON/XOFF or RTS/CTS.

```
Timeout
```

Default: 2000

Format:

The timeout parameter is optional and could be omitted. It defines the serial port receive timeout, in milliseconds. Defaults to 2 seconds if not defined.

Serial Gateways

Generic serial gateways that provide bi-directional serial port communication via a TCP connection are supported by HSYCO.

To configure a comm port to send data to the serial port of a serial gateway, you should set the comm port type to “server”, and set the gateway’s IP address and TCP port number.

The serial port handshake parameters must be set on the gateway using its configuration utilities.

HSYCO supports a high-availability, failover configuration for serial gateways.

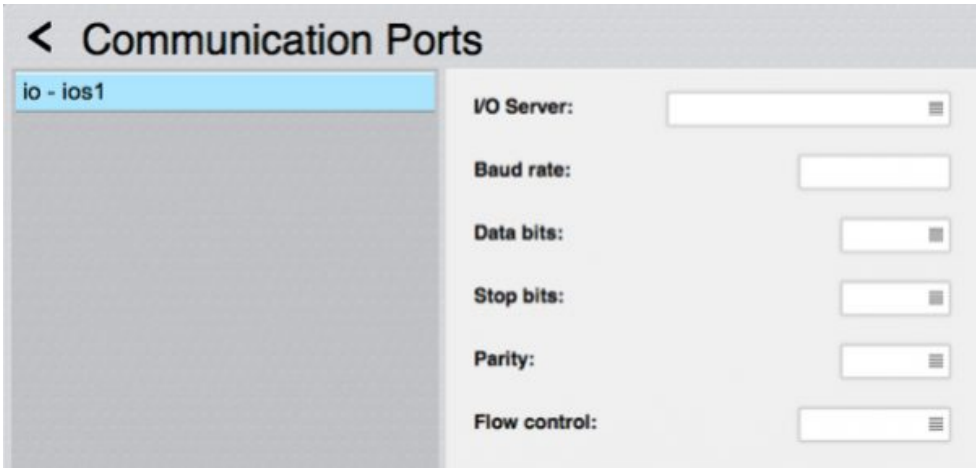
HSYCO supports a high-availability, failover configuration for serial gateways. You enable failover by adding a second IP address to the IP parameter.

HSYCO will try to establish a connection using the first IP address, automatically switching to the failover address if a communication error occurs.

In this configuration, the JavaScript or Java closeComm() function can be used to force a reconnect on the next call to readComm() or writeComm(), so that a switch between the fail-over IPs would happen if the current IP is not responding.

HWG I/O Server ports

Some network I/O controllers produced by HWG have a serial port, that could be used as fully functional serial port by HSYCO. To configure these ports, select type “io”, then select the I/O server id of the HWG controller and enter the normal serial port configuration parameters.



IRTrans ports

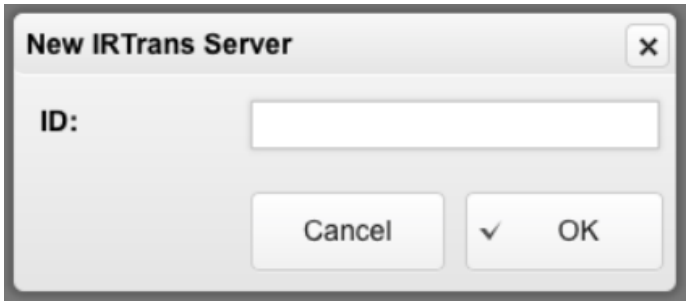
Some IRTrans network IR controllers have a local serial port. HSYCO can use this port as an output-only serial port.

To configure an IRTrans serial port, select type “irtrans”, and select the id of the IRTrans device. The serial port communication parameters, like speed, flow control etc., should be configured on the IRTrans.

IRTrans Servers

The IRTrans Servers section is used to define the IRTrans network infrared devices.

Each IRTrans must have a unique ID. You can add a new devices pressing the + button.

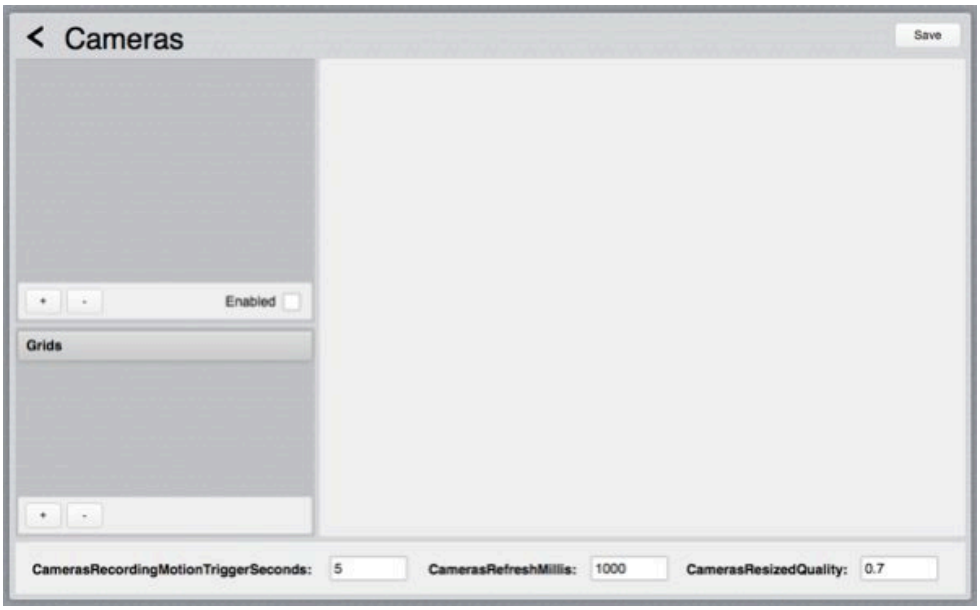


Already defined devices could be deleted with the - button or disabled without removing their configuration parameters by unchecking the Enabled checkbox.

The only configuration parameter is the IRTrans IP address.

Cameras

This section is used to define individual IP cameras, to set general configuration parameters for video processing, and to define camera grids for the user interface.



General parameters

```
CamerasRecordingMotionTriggerSeconds
```

Default: 5

Format: positive integer number

Number of seconds during which HSYCO keeps on recording from a camera after a recording trigger event for that camera. It is possible to set this parameter at 0, disabling the triggered recording of events; this can be useful in case it is preferred to control recording only through the Java user code or EVENTS.

```
CamerasRefreshMillis
```

Default: 1000

Format: positive integer number

Camera frames acquisition interval, in milliseconds. For example, for a frequency of 4 frames a second, set the value to 250. A low value corresponds to a higher number of frames acquired and recorded in the time unit, with a heavier load in terms of CPU resources and disk space.

```
CamerasResizedQuality
```

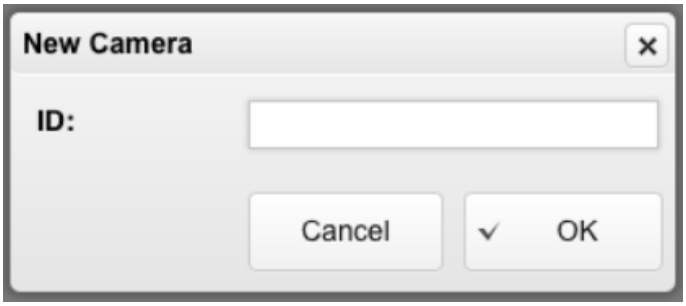
Default: 0.7

Format: decimal number between 0 and 1 (decimals separated by “.”)

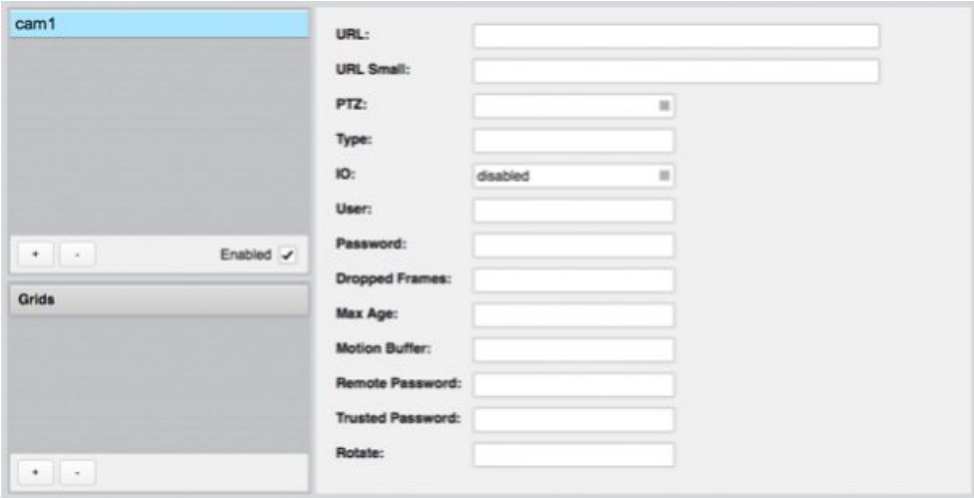
Image quality for images processing and resizing. Numbers close to 1 produce a better quality, but heavier load and larger size for each frame.

Cameras parameters

Each camera must have a unique ID. You can add a new cameras pressing the + button in the cameras list panel.



Already defined cameras could be deleted with the - button or disabled without removing their configuration parameters by unchecking the Enabled checkbox.




```
URL
```

Default: mandatory

Format: complete URL

Complete URL to acquire single frames in JPEG format, or MJPEG streams, in the desired resolution. Mandatory for each camera. Recording is based on frames acquired with this URL.

When using MJPEG streams, it is advisable to configure the number of frames per second the camera streams to a value that is very close to HSYCO's frame rate as set with CamerasRefreshMillis.

```
URL Small
```

Default:

Format: complete URL

In order to optimize camera processing performance, set this optional parameter. The URL fetches frames at a lower resolution. When serving frames to the Web interface, HSYCO will automatically choose the most appropriate frame resolution to use, based on the size of the frame in the Web interface.

Small frames are also normally used to create grids, improving processing performance.

> For a list of supported URLs check here: Camera URLs and here: Encoder URLs.

```
PTZ
```

Default:

Format: name of the PTZ driver

Defines the PTZ features of a camera and family of PTZ driver. The PTZ control of a camera is automatically added to the Web interface.

You can click on the edges of the camera image to control the camera: up, down, right, left, zoom and focus.

Supported types are:

Camera.*id*.PTZ = axis (AXIS PTZ cameras) Camera.*id*.PTZ = axis-vptz (AXIS fixed cameras with virtual PTZ support) Camera.*id*.PTZ = panasonic (Panasonic BB and BL Series cameras) Camera.*id*.PTZ = panasonic-wv (Panasonic WV Series cameras) Camera.*id*.PTZ = mobotix (Mobotix cameras) Camera.*id*.PTZ = rovio (the WowWee Rovio camera) Camera.*id*.PTZ = SNV-3120, SNP-3120, SNP-3120V, SNP-3120H (Samsung cameras) Camera.*id*.PTZ = vptz (fixed cameras with virtual PTZ support implemented by HSYCO) Camera.*id*.PTZ = user (to associate custom Java code to the control areas of the image).

```
Type
```

Default:

Format: string (letters and numbers)

Defines the camera model type for cameras that support this option. For example, *Camera.samsung.Type = SND-5080*.

```
IO
```

Default:

Format: *enabled*

For some camera models, you can enable I/O and VA (Visual Analysis) option for a camera. For example, Camera.<camid>.IO=enabled. If you want to enable I/O only, set it to “enabled:io”, or to “enabled:va” to enable VA only.

```
User
```

Default:

Format: string (letters and numbers)

When this parameter is defined, the access requests to the frames and the commands for PTZ control of the camera are sent with user and password authentication.

```
Password
```

Default:

Format: string (letters and numbers)

When this parameter is defined, the access requests to the frames and the commands for PTZ control of the camera are sent with user and password authentication.

```
Dropped Frames
```

Default: 0

Format: integer number ≥ 0

This parameter specifies the number of frames that should be discarded during recording.

For example, setting the value to 1, half of the frames will be recorded out of those normally acquired from the camera; setting the value to 3, 3 frames will be discarded for each frame acquired, thus recording only 1/4 of the acquired frames. Dropping frames reduces space used to store frames on disk, and causes accelerated playback of recorded video.

```
Max Age
```

Default: 30d

Format: positive integer number followed by "*d*" character, for days, "*h*" for hours or "*m*" for minutes

Period of time, from when each frame acquired by the camera corresponding is recorded, during which the frames remain on HSYCO's hard disk.

When this time expires the frames are automatically deleted from the HSYCO motion directory. The older frames can be automatically erased even before their expiration time, in case the space available on disk drops below a safe margin.

```
Motion Buffer
```

Default: 0

Format: positive integer number

Recording normally starts exactly when a recording trigger or command is issued. Using this parameter you can have HSYCO start recording some time ahead of the recording command. This is particularly important to capture video a few seconds before an event happens.

When this parameter is set, HSYCO constantly acquires images from the camera, not only when necessary to display or record video. Moreover, the temporary recording of this buffer requires memory resources, and it is therefore convenient not to use high values (values included between 10 and 20, with an acquisition rate of about 400 milliseconds, can be considered normal).

Enable motion buffering only when actually needed.

```
Remote Password
```

Default: 0

Format: string (letters and numbers)

HSYCO can optionally serve live images of its cameras through a password protected HTTP request, for example:

```
https://<hsycoserver>/x/camera/<cameraid>?password=<password>&size=<width>x<height>
```

(size is optional) for single frames or:

```
https://<hsycoserver>/x/camerastream/<cameraid>?size=<width>x<height>&password=<pwd>[&period=<millis>]
```

to retrieve a MJPEG stream.

It is also possible to define different passwords for clients inside the trusted range of IP addresses and outside.

```
Trusted Password
```

Default: 0

Format: string (letters and numbers)

Similar to RemoteRequestPassword, you can use this optional parameter to set remote passwords for clients inside the trusted range of IO addresses.

Note that requests with the TrustedRequestPassword value will be also served via HTTP, without SSL.

```
Rotate
```

Default: 0

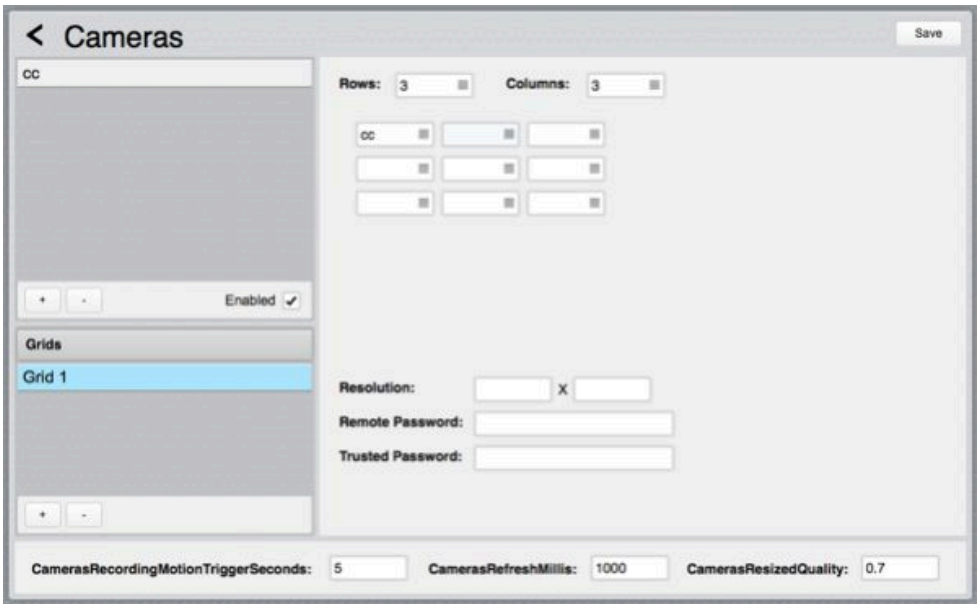
Format: integer number

Allows the rotation of the images acquired by a camera. Set to the number of decimal degrees of rotation. A positive number causes a clockwise rotation. Because of the rotation, the images could be cut or have bars at their edges to adapt the rotated picture to the frame size. Setting this parameter could cause a noticeable impact of performance, especially when processing high resolution images.

Grids

You can define up to 99 different combinations of cameras to create matrix displays of multiple cameras. The id is automatically generated when the + button is pressed, starting from 1 and up to 99 with consecutive numbers.

Grids could be deleted using the - button.



Select an item in the grids list to define the grid's number of rows and columns and select the cameras for each position. You can leave some positions empty. You should also define the other grid parameters.

```
Resolution
```

Default: 0

Format: <width> x <height> in pixels

This optional parameter is used to optimize grid processing performance, especially on large grids when using large size skins for the Web interface.

You can define a maximum resolution for the grid. HSYCO will always deliver this grid with a resolution that is never larger than the width and height defined with this parameter.

The effect is to reduce the byte size of the resulting image, improving performance on remote connections, and to offload the CPU processing required to generate the grid, having a potential benefit on frame rate.

```
Remote Password
```

Default: 0

Format: string (letters and numbers)

HSYCO can optionally serve live images of its grids through a password protected HTTP request, for example:

```
https://<hsycoserver>/x/camera/grid<N>?password=<password>&size=<width>x<height>
```

(size is optional) for single frames or:

```
https://<hsycoserver>/x/camerastream/grid<N>?size=<width>x<height>&password=<pwd>[&period=<millis>]
```

to retrieve a MJPEG stream.

It is also possible to define different passwords for clients inside the trusted range of IP addresses and outside.

```
Trusted Password
```

Default: 0

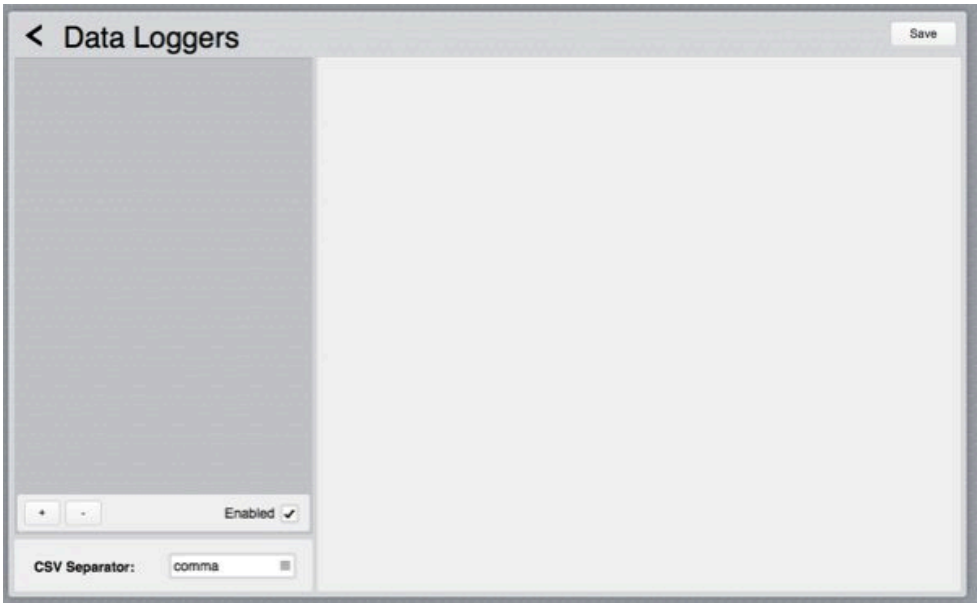
Format: string (letters and numbers)

Similar to RemoteRequestPassword, you can use this optional parameter to set remote access passwords for clients inside the trusted range of IP addresses.

Note that requests with the TrustedRequestPassword value will be also served via HTTP, without SSL.

Data Loggers

The Data Loggers section is used to define HSYCO's data loggers and their configuration parameters.



There is only one general configuration parameter, to set the CSV files separator character.

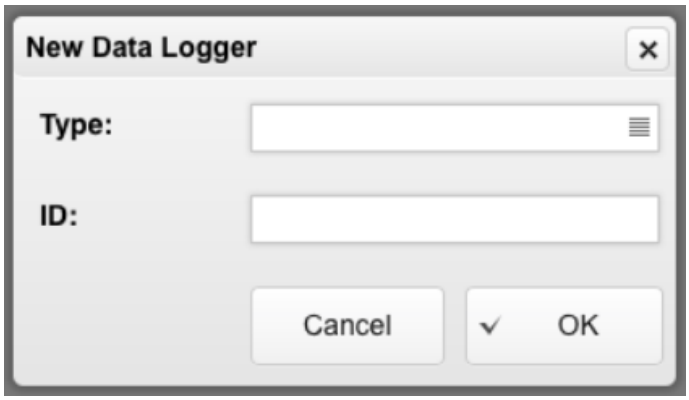
```
CSV Separator
```

Default: comma

Format: tab | comma | semicolon

This parameter defines the field separator type for CSV files generated by the data loggers.

Each data logger must have a unique ID. You can add a new data loggers pressing the + button.

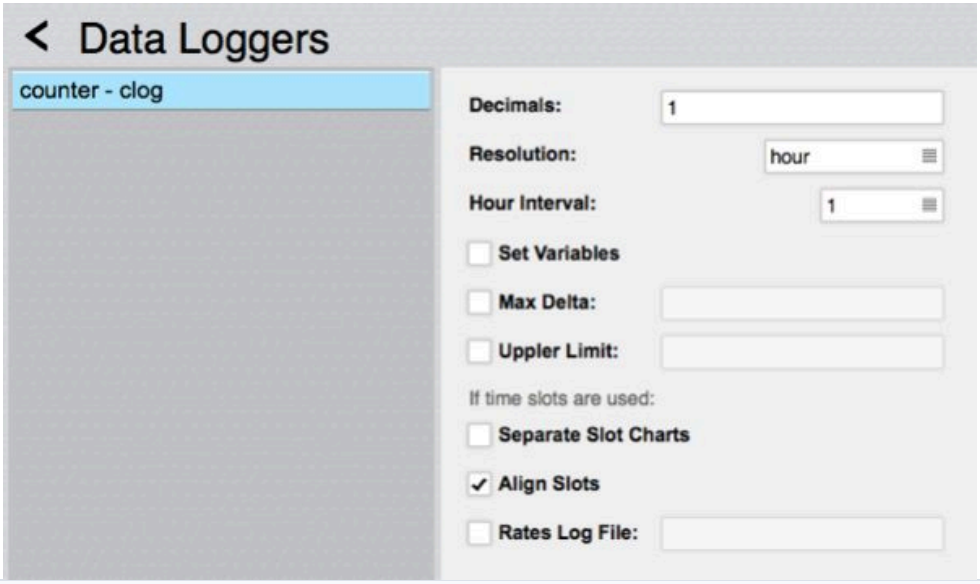


Select the type, range or counter, and enter the data logger unique ID.

Already defined data loggers could be deleted with the - button or disabled without removing their configuration parameters by unchecking the Enabled checkbox.

Counter and range data loggers have different configuration parameters.

Counter Data Loggers




```
Decimals
```

Default: 1

Format: integer number

The number of decimal digits to be used for the input data.

```
Resolution
```

Default: hour

Format: minute | hour | day | month

Data aggregation maximum resolution.

```
Hour Interval
```

Default: 1

Format: divisor of 24

Specifies the number of hours to be grouped into a single interval. For example, if set to 6, the day will be divided in groups of 6 hours: 1-6, 7-12, 13-18, and 19-24.

```
Set Variables
```

Default: false

Format: true | false

If set to true (checked), the following variables will be generated:

- `$dlog.<datalogger_id>.hour.tot`: its value is updated to the total value of the current hour
- `$dlog.<datalogger_id>.hour.past.tot`: its value is updated to the total value of the past hour
- `$dlog.<datalogger_id>.day.tot`: its value is updated to the total value of today
- `$dlog.<datalogger_id>.day.past.tot`: its value is updated to the total value of yesteday
- `$dlog.<datalogger_id>.month.tot`: its value is updated to the total value of the current month
- `$dlog.<datalogger_id>.month.past.tot`: its value is updated to the total value of the past month
- `$dlog.<datalogger_id>.year.tot`: its value is updated to the total value of the current year
- `$dlog.<datalogger_id>.year.past.tot`: its value is updated to the total value of the past year

If slots are used, the following variables will be generated for each slot:

- `$dlog.<datalogger_id>.s<slot_id>.hour.tot`: its value is updated to the total value of the current hour for this slot
- `$dlog.<datalogger_id>.s<slot_id>.hour.past.tot`: its value is updated to the total value of the past hour for this slot
- `$dlog.<datalogger_id>.s<slot_id>.day.tot`: its value is updated to the total value of today for this slot
- `$dlog.<datalogger_id>.s<slot_id>.day.past.tot`: its value is updated to the total value of yesteday for this slot
- `$dlog.<datalogger_id>.s<slot_id>.month.tot`: its value is updated to the total value of the current month for this slot
- `$dlog.<datalogger_id>.s<slot_id>.month.past.tot`: its value is updated to the total value of the past month for this slot
- `$dlog.<datalogger_id>.s<slot_id>.year.tot`: its value is updated to the total value of the current year for this slot
- `$dlog.<datalogger_id>.s<slot_id>.year.past.tot`: its value is updated to the total value of the past year for this slot

```
Max Delta
```

Default:

Format: positive float number

Sets the maximum valid delta between consecutive readings. If the calculated delta exceed such value, the read value is ignored. If not specified, all values are accepted.

💡 **It is highly recommended to set the delta limit in order to prevent the data series from being corrupted in case of false readings.**

```
Upper Limit
```

Default: 0

Format: positive float number

Specifies the maximum value that can be reached by the input value. This value will be used in case of counter roll-over. A value of 0 indicates that no upper limit is specified.

```
Separate Slot Charts
```

Default: false

Format: true | false

Option for counter data loggers using time slots. Specifies wether or not to use different monthly and yearly charts for each defined time slot.

```
Align Slots
```

Default: true

Format: true | false

Option for counter data loggers using time slots with separate charts. Specifies wether or not to align the scale of the slot charts belonging to the same period.

```
Rates Log File
```

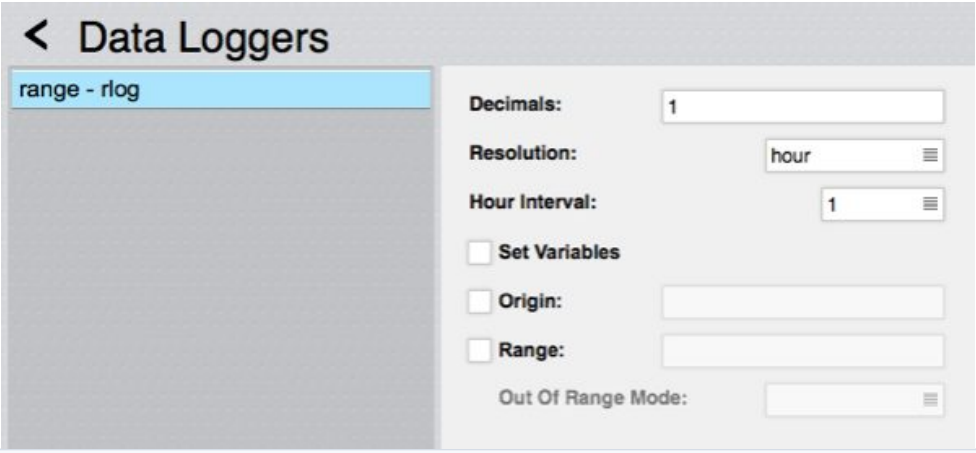
Default:

Format: file path. e.g.: *hsyco/rates.csv*

Option for counter data loggers using time slots. Specifies the path of the file where the processed data will be logged. If omitted, no log file will be created.

If the path includes the strings “%Y”, “%M”, or “%D” they will be replaced respectively by the current year, month, or day.

Range Data Loggers




```
Decimals
```

Default: 1

Format: integer number

The number of decimal digits to be used for the input data.

```
Resolution
```

Default: hour

Format: minute | hour | day | month

Data aggregation maximum resolution.

```
Hour Interval
```

Default: 1

Format: divisor of 24

Specifies the number of hours to be grouped into a single interval. For example, if set to 6, the day will be divided in groups of 6 hours: 1-6, 7-12, 13-18, and 19-24.

```
Set Variables
```

Default: false

Format: true | false

If set to true (checked), the following variables will be generated:

- `$dlog.<datalogger_id>.hour.min.tot`: its value is updated to the minimum value of the current hour
- `$dlog.<datalogger_id>.hour.max.tot`: its value is updated to the maximum value of the current hour
- `$dlog.<datalogger_id>.hour.avg.tot`: its value is updated to the average value of the current hour
- `$dlog.<datalogger_id>.hour.past.min.tot`: its value is updated to the minimum value of the past hour
- `$dlog.<datalogger_id>.hour.past.max.tot`: its value is updated to the maximum value of the past hour
- `$dlog.<datalogger_id>.hour.past.avg.tot`: its value is updated to the average value of the past hour

- `$dlog.<datalogger_id>.day.min.tot`: its value is updated to the minimum value of today
- `$dlog.<datalogger_id>.day.max.tot`: its value is updated to the maximum value of today
- `$dlog.<datalogger_id>.day.avg.tot`: its value is updated to the average value of today
- `$dlog.<datalogger_id>.day.past.min.tot`: its value is updated to the minimum value of yesterday
- `$dlog.<datalogger_id>.day.past.max.tot`: its value is updated to the maximum value of yesterday
- `$dlog.<datalogger_id>.day.past.avg.tot`: its value is updated to the average value of yesterday

- `$dlog.<datalogger_id>.month.min.tot`: its value is updated to the minimum value of the current month
- `$dlog.<datalogger_id>.month.max.tot`: its value is updated to the maximum value of the current month
- `$dlog.<datalogger_id>.month.avg.tot`: its value is updated to the average value of the current month
- `$dlog.<datalogger_id>.month.past.min.tot`: its value is updated to the minimum value of the past month
- `$dlog.<datalogger_id>.month.past.max.tot`: its value is updated to the maximum value of the past month
- `$dlog.<datalogger_id>.month.past.avg.tot`: its value is updated to the average value of the past month

- `$dlog.<datalogger_id>.year.min.tot`: its value is updated to the minimum value of the current year
- `$dlog.<datalogger_id>.year.max.tot`: its value is updated to the maximum value of the current year
- `$dlog.<datalogger_id>.year.avg.tot`: its value is updated to the average value of the current year
- `$dlog.<datalogger_id>.year.past.min.tot`: its value is updated to the minimum value of the past year
- `$dlog.<datalogger_id>.year.past.max.tot`: its value is updated to the maximum value of the past year
- `$dlog.<datalogger_id>.year.past.avg.tot`: its value is updated to the average value of the past year

```
Origin
```

Default: 0

Format: float number

Specifies the origin of the charts.

```
Range
```

Default:

Format: range: low:up

Specifies the range of the charts as “low:up”, where “low” indicates the lower limit and “up” the upper one. The lower and upper limits can be specified as rigid or flexible: if the value of a limit is followed by a “!” (e.g. 0!:20!) then it is rigid, meaning it will not change even if the data logger has processed a value exceeding the range. On the contrary, flexible ranges (e.g. 0:20) will be modified whenever a value exceed it.

If the parameter is omitted, the charts will range from the lowest processed value to the highest one.

```
Out of Range Mode
```

Default: cut

Format: cut | ignore

Option for range data loggers with a predefined range. Specifies the behavior of the data logger when updated with a value which exceed the set value range: if “cut” is specified then the value is adjusted to the limits, else, if this parameter is set to “ignore”, the passed value is not taken into account. In both cases, an error message is reported for out of range values.

DMX

The DMX section is used to define the DMX-512 gateways.

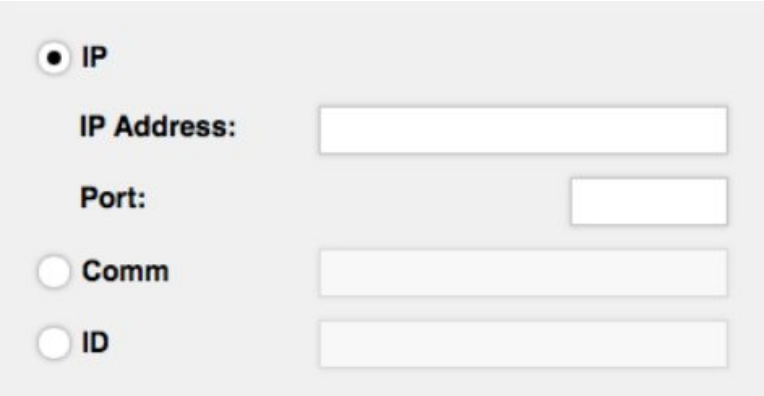
Each DMX gateway must have a unique ID. You can add a new devices pressing the + button.

Already defined devices could be deleted with the - button or disabled without removing their configuration parameters by unchecking the Enabled checkbox.

HSYCO supports three different DMX solutions, using different configuration parameters.

KissBox DMX Network Gateways

For KissBox network gateways, check IP and enter the IP address and port number.



ENTTEC DMX USB PRO DMX Gateways

For the ENTTEC DMX USB PRO DMX gateways, check Comm and enter the virtual serial port name assigned to the USB port. See the ENTTEC DMX USB PRO application note for additional information.

Domino and Contatto DMX devices

When using Doemmegi's Domino or Contatto systems DMX devices, check ID and enter the full device name, including the I/O server prefix. See the Domino or Contatto application notes for additional information.

Location Services

The Location Services section is used to define the WiFi access points used to locate the WiFi clients position.

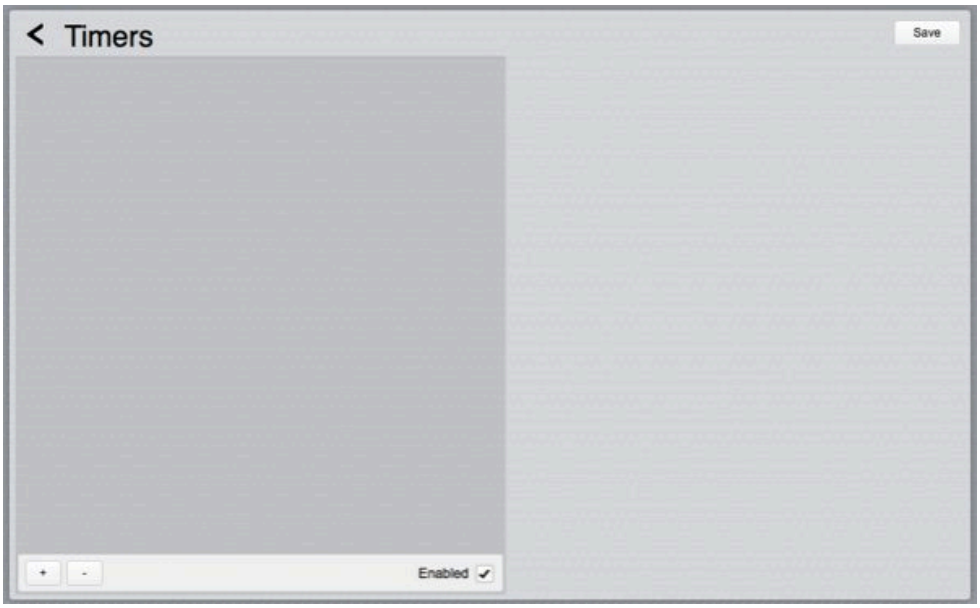
Each access point must have a unique ID. You can add a new devices pressing the + button.

Already defined devices could be deleted with the - button or disabled without removing their configuration parameters by unchecking the Enabled checkbox.

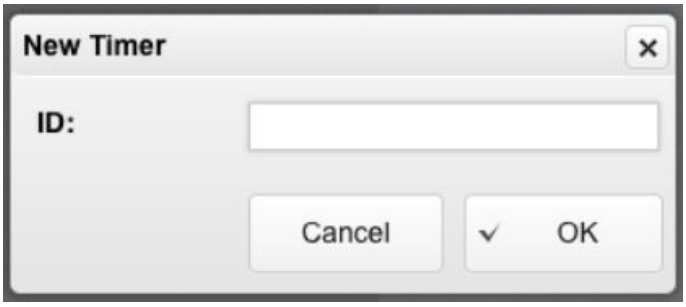
The only configuration parameter is the access point IP address.

Timers

The Timers section is used to define user timers.



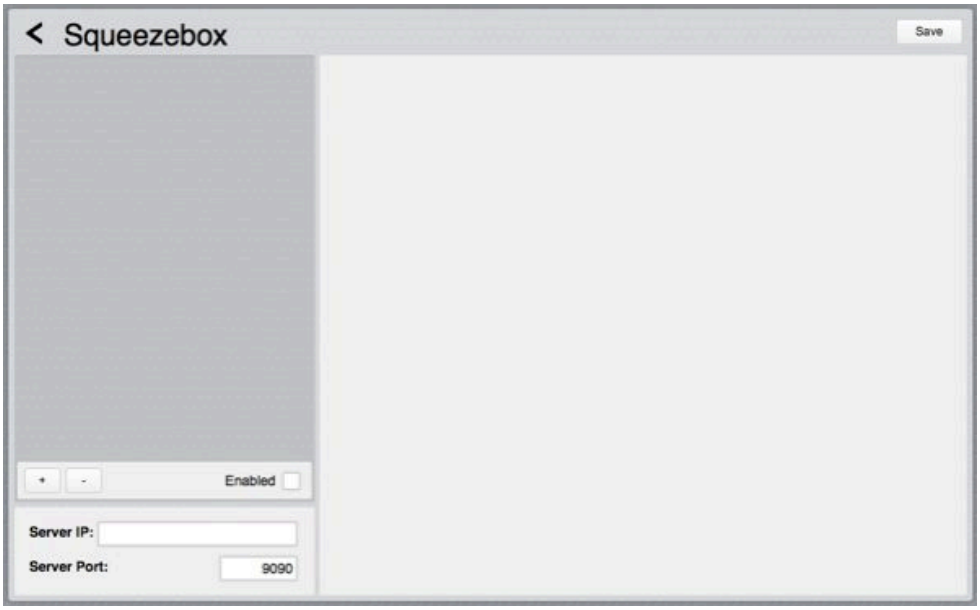
Each timer must have a unique ID. You can add a new timers pressing the + button.



Already defined timers could be deleted with the - button or disabled by unchecking the Enabled checkbox. Timers don't require any configuration parameter besides the id.

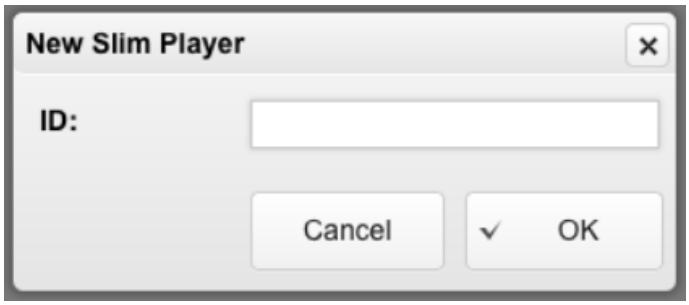
Squeezebox

This section is used to define the Squeezebox server and music players.

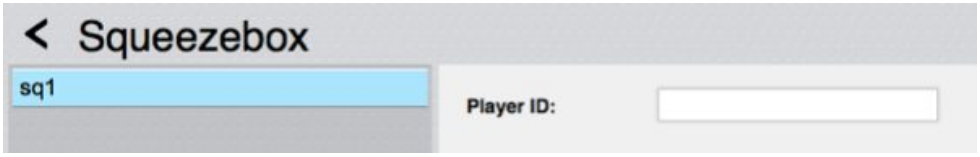


You should set the Squeezebox server's IP address or network name and port number.

Each player must have a unique ID. You can add a new devices pressing the + button.



Already defined devices could be deleted with the - button or disabled without removing their configuration parameters by unchecking the Enabled checkbox.



The only configuration parameter is the player id, usually its MAC address.

Camera URLs

List of HTTP camera URL that have been tested with HSYCO.

Brand	Type	URL
Arecont	jpg	http://CAMERAIPADDRESS/image?res=half&x0=0&y0=0&x1=640&y1=480&quality=1&doublescan=0
Axis	mjpg	http://CAMERAIPADDRESS/axis-cgi/mjpg/video.cgi?compression=50&resolution=640x480&fps=7
Axis	jpg	http://CAMERAIPADDRESS/axis-cgi/jpg/image.cgi
Bosch	jpg	http://CAMERAIPADDRESS/snap.jpg
Dahua	mjpg	http://CAMERAIPADDRESS/video/mjpg.cgi?profileid=1
Dallmeier	mjpg	http://CAMERAIPADDRESS/live/image0.jpg
Dlink	mjpg	http://CAMERAIPADDRESS/cgi-bin/mjpg/video.cgi
Foscam	mjpg	http://CAMERAIPADDRESS/videostream.cgi
Hikvision	mjpg	http://CAMERAIPADDRESS/Streaming/channels/1/picture
Mobotix	mjpg	http://CAMERAIPADDRESS/cgi-bin/image.jpg?size=640x480&quality=60
Mobotix (alternative)	mjpg	http://CAMERAIPADDRESS/faststream.jpg?&preview&stream=full&needlength&fps=1
Panasonic	jpg	http://CAMERAIPADDRESS/SnapshotJPEG?Resolution=640x480
Samsung	mjpg	http://CAMERAIPADDRESS/cgi-bin/video.cgi?msubmenu=mjpg
Sony	mjpg	http://CAMERAIPADDRESS/mjpeg?speed=5
Vivotek	mjpg	http://CAMERAIPADDRESS/video.mjpg

Encoder URLs

List of HTTP URL that have been tested with HSYCO.

Brand	Type	URL
Axis (channel1)	mjpg	http://ENCODERIPADDRESS/axis-cgi/mjpg/video.cgi?fps=7&camera=1
Axis (channel2)	mjpg	http://ENCODERIPADDRESS/axis-cgi/mjpg/video.cgi?fps=7&camera=2
Axis (channel3)	mjpg	http://ENCODERIPADDRESS/axis-cgi/mjpg/video.cgi?fps=7&camera=3
Axis (channel4)	mjpg	http://ENCODERIPADDRESS/axis-cgi/mjpg/video.cgi?fps=7&camera=4
Aviosys (channel1)	jpg	http://ENCODERIPADDRESS/usr/yoics0.jpg
Aviosys (channel2)	jpg	http://ENCODERIPADDRESS/usr/yoics1.jpg
Aviosys (channel3)	jpg	http://ENCODERIPADDRESS/usr/yoics2.jpg
Aviosys (channel4)	jpg	http://ENCODERIPADDRESS/usr/yoics3.jpg
Mesa DVW series (channel1)	jpg	http://ENCODERIPADDRESS/cgi-bin/webra_fcgi.fcgi?api=get_jpeg_raw&chno=0
Mesa DVW series (channel2)	jpg	http://ENCODERIPADDRESS/cgi-bin/webra_fcgi.fcgi?api=get_jpeg_raw&chno=1
Mesa DVW series (channel3)	jpg	http://ENCODERIPADDRESS/cgi-bin/webra_fcgi.fcgi?api=get_jpeg_raw&chno=2
Mesa DVW series (channel4)	jpg	http://ENCODERIPADDRESS/cgi-bin/webra_fcgi.fcgi?api=get_jpeg_raw&chno=3

Enttec

Enttec DMX USB PRO is a USB DMX adapter that allows HSYCO to control any DMX-512 standard bus.

This device connects to a USB port on HSYCO Server, and is powered through the USB, so there is no need for an external power supply.

The circuitry implements 1500 V full electrical isolation (data & power lines to protect the HSYCO Server from surges).

HSYCO supports the DMX out port only.

Check the ENTTEC site for additional information.

Connections

The ENTTEC DMX USB PRO connects to HSYCO using a USB port with the FTDI virtual com driver.

The device driver will assign a serial port, with a name that depends on the operating system and the presence of virtual com devices on other USB ports.

You can connect multiple ENTTEC DMX USB PRO devices or other DMX controllers to an HSYCO Server.

HSYCO Running on Linux

The FTDI driver is already present on HSYCO's standard Ubuntu-based distribution.

Simply connect the ENTTEC DMX USB PRO to a USB port on HSYCO Server and look in the /var/log/syslog file to find the serial device name assigned to the device.

You should see something like this:

```
Jul 25 12:22:07 hsycoserver kernel: [1206668.247487] usb 2-1: FTDI USB Serial Device converter now attached
to ttyUSB0
```

In this case the port name is ttyUSB0. You should expect this to be the name that is assigned in most cases if there are no other devices attached to the USB ports.

HSYCO Running on Mac OS X

On Mac OS X you will have to install the FTDI Virtual COM (VCP) driver. You can download the appropriate driver from FTDI's drivers page, then follow the installation instructions for Mac OS X.

Once the driver is installed, and the device connected to a USB port, the driver will assign a port name that contains part of the device's serial number, so it should be quite easy to find. Look at the files in the /dev directory, and you should see something like this:

```
crw-rw-rw-  1 root  wheel   18,  49 Jul 25 13:06 /dev/cu.usbserial-EN099618
crw-rw-rw-  1 root  wheel   18,  48 Jul 25 13:06 /dev/tty.usbserial-EN099618
```

You can simply use the last part of the name, 099618 in this case, to configure the port name in System Settings.

HSYCO Configuration

Once the ENTTEC DMX USB PRO is configured in System Settings, you will be able to control the DMX-512 bus as with any other supported DMX controller on HSYCO.

There is no need to explicitly define an HSYCO comm port to use the ENTTEC DMX USB PRO.

Release Notes

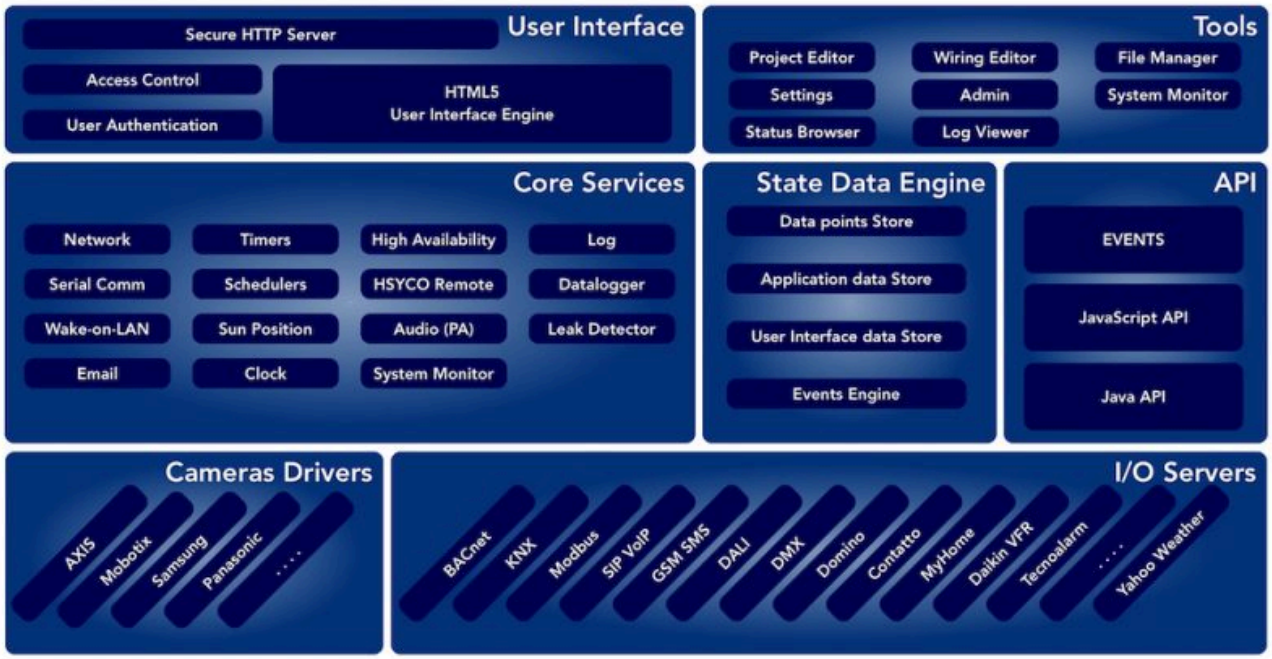
3.1.0

- initial release

ENTTEC is a registered trademark of ENTTEC Pty Ltd. FTDI is a registered trademark of Future Technology Devices International Ltd.

Programming

HSYCO offers an extremely powerful programming framework, allowing you to easily create complex programming logic that can be quickly integrated with the user interface functions, the field systems and HSYCO's native features.



At its core, HSYCO is an events based architecture, optimized to quickly process any status change from the low-level drivers or its internal services, and allows you to write custom code that is triggered by these events.

You can write code in Java, JavaScript and a simple, proprietary language called EVENTS.

Java is precompiled, offers the best performance and powerful programming features of a fully standard Java language, but requires restarting HSYCO every time you make changes to the code.

On the other hand, EVENTS is a very simple declarative language, that associates one or more actions to a field event or combination of conditions.

Programming EVENTS is as simple as editing a text file, and all changes are immediately effective the moment you save the file. It is also a very effective way to quickly associate actions to events, faster than Java in many simple cases, and easier to read.

Finally, JavaScript is the middle ground, combining some of the best features of a powerful, standard programming language like Java, and the simplicity and efficiency of EVENTS. In fact you can write JavaScript code seamlessly mixed with EVENTS.

These diversified programming tools should not be considered to be confusing, but a flexible approach to address different problems with different solutions.

An experienced Java developer may decide to code all its custom logic in Java, because everything that can be done in JavaScript or EVENTS can also be implemented in Java, but will most likely appreciate the convenience of both EVENTS and JavaScript for some tasks.

But, if software development is not your business, you will definitely stay with EVENTS and possibly some JavaScript, and quite likely never need to use Java.

Anyways, EVENTS, JavaScript and Java are fully interoperable and integrated, with specific solutions to let you merge code components of the different languages into a seamless application.

Using the Status Browser

An HSYCO application, no matter whether it is written in EVENTS, JavaScript or Java, interacts with external systems through I/O data points, and with the user by dynamically setting user interface attributes.

It also uses variables to save and retrieve internal state information.

When testing an application it is extremely useful for the developer to be able to access this information.

The **Status Browser**, in the **Manager** toolkit, allows you to show the real time status of data points, user interface attributes and variables.

Events

Events

Event Keywords

Action Keywords

JavaScript

JavaScript

JavaScript Callback Functions API

JavaScript Command and Utility Functions API

Java

Java

Java Callback Methods API

Java Command and Utility Methods API

Creating User Interfaces

Working with User Objects

Working with Forms

Draw API

Action Keywords

An action is defined as a keyword that identifies the action type, always followed by one or more parameters and sometimes the equal (assignment) operator.

Every keyword uses a specific syntax that interprets parameters according to its own rules.

System Keywords

DATESET

Set the HSYCO Server's system clock.

Action	Description
DATESET = datetime	set the HSYCO Server operating system's clock and hardware clock

Parameters:

- datetime - date and time formatted as "YYYY-MM-DD hh:mm:ss" or "YYYYMMDDhhmmss", in the local time zone

Examples:

```
DATESET "2014-10-31 20:00:00"
DATESET "20141031200000"
```

HAACTIVE

Force a master HSYCO server to become inactive, or return to the active state.

Action	Description
HAACTIVE = false	force a master to become inactive (has no effect on the slave system)
HAACTIVE = true	used after the master was forced to become inactive, returns it to the active state (has no effect on the slave system)

Parameters:

- active - true or false

Examples:

```
IO mygate.connection = offline : HAACTIVE = false
```

POWER

Sets the electric power state variable.

The power value is shown in the navigation or status bar of the Web interface.

Besides setting the power value, this method also triggers the execution of the powerEvent() callback and the POWER event in EVENTS.

Action	Description
POWER = power	sets the electric power state variable to a specific value, in Watts

Parameters:

- power - the power in Watt units.

Examples:

```
POWER = IO meter.power
```

PROGRAMTIMER

Creates or deletes a program timer.

Action	Description
PROGRAMTIMER name = SET seconds	a program timer will be executed after the number of seconds specified in SET. If a program timer with the same name has already been set, the action will be ignored
PROGRAMTIMER name = CLEAR	deletes a timer
PROGRAMTIMER name = RESET seconds	similar to SET, but if a program timer with the same name has already been set, the timer settings will be changed to the new timeout
PROGRAMTIMER name = REPEAT seconds	sets a program timer that is executed repeatedly with an interval equal to the specified number of seconds

Parameters:

- name - the program timer name
- seconds - the timeout, in seconds.

Examples:

```
SECURITY evo = ON 1  : PROGRAMTIMER presence = REPEAT 3600
SECURITY evo = OFF 1 : PROGRAMTIMER presence = CLEAR
```

WAIT

Pauses before the execution of the next action.

Action	Description

WAIT = s	causes a pause in the execution of actions for the specified number of seconds. It can be also set in decimal format

Parameters:

- s - wait time in seconds

Examples:

```
WAIT = 5
WAIT = 0.3
```

Cameras

CAMERA

The CAMERA action is used to temporarily disable cameras' live view and recording features.

Action	Description
CAMERA name = ON	live view enabled
CAMERA name = OFF	live view and recording disabled
CAMERA name = RECON	recording enabled
CAMERA name = RECOFF	recording disabled

Parameters:

- name - the camera's id. You can specify a list of space separated ids.

Examples:

- CAMERA entrance = ON

CAMERAREC

Records video from a camera.

Action	Description
CAMERAREC name = sec	records for the number of seconds defined with the set attribute

Parameters:

- name - the camera's id. You can specify a list of space separated ids.

Examples:

```
CAMERAREC openspace = 30
```

CAMERARECFULL

Records video from a camera, like CAMERAREC, but ignores the DroppedFrames parameter in hsyco.ini, so that all frames captured from the camera during the recording period are recorded with no skips.

Action	Description
CAMERARECFULL name = sec	records for the number of seconds defined with the set attribute

Parameters:

- name - the camera's id. You can specify a list of space separated ids.

Examples:

```
CAMERARECFULL openspace = 30
```

Data Logger

Data loggers collect and display statistical data on the variation of a specified value.

They can be used to automatically generate trend charts and log the data as CSV files.

See the Data Logger documentation for additional information.

DATALOGGER

Updates and performs operations on a data logger.

Action	Description
DATALOGGER name = value	Supplies the data logger with a new value to be processed. It will also trigger the refresh procedure
DATALOGGER name = CLEAR	Clears the data gathered in the data logger
DATALOGGER name = ERROR	Used with counter data loggers, resets the internal delta baseline
DATALOGGER name = FILE LOG filename [TIMESTAMP]	Appends the last acquired value to the specified file (filename) using the CSV format. If the option TIMESTAMP is specified, the timestamp of the record will be reported too
DATALOGGER name = FILE STAT filename	Creates or updates the specified file (filename) with the currently gathered data using the CSV format

Parameters:

- name - identifier of the data logger. It is possible to use a list of names separated by spaces to address several data loggers. Variables can be used
- filename - the file pathname. You can use multiple distinct strings, including variables, that will be concatenated to create the file name.

Examples:

```
$value : DATALOGGER energy = $value,
  DATALOGGER energy = FILE LOG hsyco/energylog.csv TIMESTAMP,
  DATALOGGER energy = FILE STAT hsyco/energystat.csv
```

```
TIME : DATALOGGER temperature = IO ste.1
TIME : DATALOGGER humidity = IO ste.2
```

DMX

DMX

Controls the value of DMX-512 channels.

Action	Description
DMX address = ON	sets the DMX address to the value preceding the last OFF command
DMX address = OFF	sets the DMX address to value = 0
DMX address = FLIP	flips the status of a DMX channel (if the status is ON it executes the OFF command and vice versa). When more channels are specified, if the previous status of at least one of the channels is different from OFF, the OFF command will be sent to all the channels, otherwise the ON command will be sent
DMX address = value	assigns a value to the DMX channel. Valid values are numbers between 0 and 255
DMX addressX = DMX addressY	the DMX channel addressX assumes the current status of the DMX channel addressY
DMX address = MERGE	the channel or channels on the DMX OUT bus of the gateway follow the same channels of the DMX IN bus
DMX address = UNMERGE	disables merge mode

Parameters:

- address - when using one DMX gateway, address will be a standard DMX-512 address between 1 and 512. If there is more than one gateway, 1000 must be added to the DMX address for the second gateway, 2000 for the third one and so on. For example, 2100 stands for the address 100 on the DMX bus controlled by the third gateway. You can also use a from-to range format to specify a contiguous block of channels.

Examples:

```
DMX 100 101 102 = ON
DMX 100-120  200-220 = DMX 40
DMX 33 = MERGE
```

Infrared Control

IR

Sends a command to an IRTrans.

Action	Description
IR name = command	the IR code must be in the local IRTrans FLASH memory database or, If the hsyco/ir directory contains a .ccfhex file corresponding to the database name, then the CCF string in that file is used instead of sending the command stored in the IRTrans internal database

Parameters:

- name - identifies the IRTrans, as defined with the IRTrans parameter in hsyco.ini
- command - the command format is remote.command; that is the remote control database name followed by a dot and the command name.

Examples:

```
IR theater = dvd.play
```

I/O Servers

IO

Writes to I/O servers data points.

Action	Description
IO name = value	sets the data point to a value. Values are specific to the type of I/O server
IO name = FLIP	inverts the output of the I/O data point. If the status of the output is 0 the new status will be 1, if different from 0 the new status will be 0
IO name1 = IO name2	sets the I/O interface output name1 to the current value of the I/O interface name2

Parameters:

- name - the server id, followed by a single dot and the data point suffix. The EVENTS interpreter converts data points names to lower case
- value - you can use multiple unquoted words, quoted strings and variables, that will be appended to the value string.

Examples:

```
IO contacts.c1 contacts.c3= 1
IO sensors.c1 = FLIP
IO contacts.o1 contacts.o2 = IO sensors.flood
```

Leak Detector

The Intelligent Leak Detector is used to generate warning for potential water or other quantities leaks by analyzing any generic flow counter.

The detector should be called at regular intervals, possibly faster than the minimum time needed for the counter to increment its value by one.

It will generate a leak warning when the measured flow remains relatively constant over a certain amount of time.

Use this function at your own risk!

The Intelligent Leak Detector uses a correlation algorithm to distinguish between a constant flow and an irregular flow. A relatively constant flow over a certain amount of time is considered as a potential leak and generates a leak event.

In some conditions, also depending on the type of flow counter used, this algorithm could fail from properly and timely recognizing a real leak, or it could generate false leak warnings.

Use different names to implement multiple independent leak detectors.

LEAK

Controls the integrated leak detector.

The detector's logic uses two parameters to set its sensitivity and time base.

The detector generates a warning if the flow deviation constantly remains below the deviation threshold for the time period.

You can change the defaults to adapt to your specific conditions.

Action	Description
LEAK name = value	calls the leak detector passing the current liters counter value. Using variables or references to IO data points is supported
LEAK name = PERIOD t	changes the time base. The default is 1200 seconds. Set the period to 0 to temporarily disable the detector. The time base can be changed anytime, even when the detector is already processing data
LEAK name = DEVIATION d	changes the deviation percentage. d should be a number between 1 and 100. The default is 50. Set the deviation to 0 to temporarily disable the detector. Lower values cause the algorithm to activate leak warnings only when the flow is very constant in time, becoming less susceptible to false alarms, but a value that is too low could miss actual leaks. Higher values could easily generate false alarms. The deviation can be changed anytime

LEAK name = CLEAR	clears the detector's internal state. In normal applications, there is no need to call this function, as the detector automatically clears its state from old data

Parameters:

- name - the name of a leak detector. Using different names you can implement several independent leak detectors
- t - time base
- d - deviation percentage

Examples:

```
HSYCOSTART : PROGRAMTIMER modbusread = repeat 4
PROGRAMTIMER modbusread :  IO modbus.2.768 = readholdingregisters:uint, LEAK water = IO modbus.2.768
LEAK water = ON : MAIL "john@example.com" = "hsyco@example.com" "Urgent Message from HSYCO" "Leak Detected"
LEAK water = OFF : MAIL "john@example.com" = "hsyco@example.com" "Message from HSYCO" "Leak Reset"
```

Log

FILELOG

Appends a generic text message at the end of a file.

Action	Description
LOG filename = text	you can use multiple unquoted words, quoted strings and variables, that will be appended at the last line of the text file

Parameters:

- filename - the file name. You can specify any path, starting from HSYCO's base directory. If the file or parent directories don't exist, they will be created automatically. The filename parameter can be a variable
- text - the message that will be appended at the last line of the text file.

Examples:

```
HSYCOSTART : FILELOG logs/mylog.txt = "log message example"
```

LOG

Generates an information message in the daily log file.

Action	Description
LOG = text	you can use multiple unquoted words, quoted strings and variables, that will be appended to the log message

Parameters:

- text - the log message
- from - the sender e-mail address.

Examples:

```
LOG = "log message example"
LOG = "Time: " $TIME:H-M-S$ " DATE: " $DATE:D/M/Y$
```

Mail

MAIL

Sends an email message.

HSYCO SERVER sends the mail either directly to the recipient's domain mail server if the SmtpName parameter is not defined in hsyco.ini, or using a specific email account with user authentication and traffic encryption if the SMTP server and account parameters are set.

If the email message is sent directly to the recipient's domain mail server, you need to ensure that the mail server accepts mail to the destination address being sent with the from address and the public IP of the HSYCO SERVER.

💡 **HSYCO does not automatically retry sending the message if the destination mail server is not available when the send mail function is called.**

Action	Description
MAIL to = from subject body	sends text and camera images. The body part of the mail is the last attribute. You can email the same message to multiple recipients

Parameters:

- to - the recipient address. It is possible to send the same email to a space separated list of email addresses. You can optionally specify the destination SMTP server name or IP address by appending :server name or address to the recipient's email address, for example: john@example.com:192.168.1.1
- from - the sender e-mail address
- subject - a quoted string or variable with the message object
- body - the message body. To send an ordinary text, just enter a quoted text string. To send an image, append strings with the following format: "cam:cameraname[:seconds_back]". For example, "cam:door" sends a live frame from the camera called "door"; "cam:door:2" sends a frame that was recorded two seconds before the last recorded frame; "cam:door:0" sends the last recorded frame. To send files as attachments, use the following format: "file:file name". The file path is relative to the HSYCO root directory.

Examples:

In this example, we send a message every minute, with the measured power load, to two email addresses:

```
TIME : $BODY = "Energy consumption for " $DATE:y/m/d$ " at " $TIME:h:m:s$ " is " $power$ " Watt"
$BODY : MAIL email1@hsyco.com email2@hsyco.com = hsyco@hsyco.net "Sent from HSYCO" $BODY
```

40 seconds after the alarm event, an email from the address hsyco@hsyco.net is sent to emal1@hsyco.com, with three images attached: the first is a real time image, the second is the last recorded image, the third one is the frame that was recorded 15 seconds before:

```
user "security" = "alarm" : CAMERAREC entrance = 30, PROGRAMTIMER rec = 40
PROGRAMTIMER rec : MAIL email1@hsyco.com = hsyco@hsyco.net “Sent from HSYCO” “Camera entrance”
“cam:entrance” “cam:entrance:0” “cam:entrance:15”
```

At midnight, we are sending the energy.csv file in the data sub-directory as an attachment:

```
TIME = 0000: MAIL email1@hsyco.com = hsyco@hsyco.net "Sent
from HSYCO” "Energy daily report." file:data/energy.csv
```

Network Services

PING

Tests the reachability of one or more hosts, identified by their hostnames or IP addresses, within the optional timeout defined in milliseconds (or using a default timeout of 200 ms).

This test generates PING events like PING hostname = ON if the host is reachable, or PING hostname = OFF if not reachable.

Action	Description
PING hostname	test the reachability of hostname, with 200ms response time-out
PING hostname = time	test the reachability of hostname, with a specific response time-out

Parameters:

- hostname - the IP device name or IP address
- time - response time-out, in milliseconds.

Examples:

```
TIME : PING 192.168.1.1
TIME : PING 192.168.1.200 192.168.1.201 = 1000
```

URL

Sends a GET or POST HTTP or HTTPS request to the specified url.

The HTTP basic or digest access authentication methods are supported.

Responses can be checked using the corresponding URL event.

Action	Description
URL GET "url"	sends a GET request without authentication
URL GET "user:password" "url"	sends a GET request with authentication

URL POST "url" "content_type" "data"	sends a POST request without authentication
URL POST "user:password" "url" "content_type" "data"	sends a POST request with authentication

Parameters:

- url - a full url, including the mandatory http:// or https:// query scheme and optional query string
- user:password - user and password for basic or digest access authentication
- content_type - the MIME type descriptor for the POST data
- data - the URL-encoded POST data sent with the request.

Examples:

```
HSYCOSTART: URL POST "http://10.0.0.5/cgi-bin/cmd.sh" "text/xml" "%3C%3Fxml%20version%3D%22..."
```

```
TIME 0800: URL GET "usr1:qi3qw" "http://10.0.0.5/cgi-bin/cmd.sh?id=relay1"
```

```
URL "http://10.0.0.5/cgi-bin/cmd.sh?id=relay1" : LOG = "relay opened"
```

Public Announcement

AUDIO

The AUDIO action is used to play text-to-speech messages or recorded audio files.

Audio can be sent to the Web browser, the server's audio line out connector, the internal speaker or audio out line of Axis cameras, and the speaker of SNOM phones or PA devices.

See the Audio and Public Announcement section for additional information.

Action	Description
AUDIO to = FILE filename	plays a pre-recorded audio file
AUDIO to = VOICE:voicename message	converts a text message to speech and plays the audio

Parameters:

- to - the audio destination (see table below)
- filename - the pathname of the audio file, relative to the HSYCO's main directory
- voicename - the voice name for the text-to-speech engine; see the Audio and Public Announcement section for additional information
- message - the text message for text-to-speech conversion; you can use multiple strings that will be automatically appended.

Audio Destination	Description
speaker	the server's audio line out connector

web	the Web browser's audio output 💡 **this feature is supported only on Firefox and Chrome browsers**
axis@camera id	audio sent to an Axis camera, using the camera id defined in the Cameras parameter in hsyco.ini
snom@ip:port	audio sent to SNOM phones and public announcement devices, to the multicast IP address and port specified (the IP address and port should be configured as multicast addresses on each phone)

Examples:

The gong.mp3 audio file is played every minute through the server's audio line out or internal speaker:

```
TIME : audio speaker = file "audio/gong.mp3"
```

Same as above, but played at the same time on all phones registered to the 239.255.255.245 multicast address and port 5555:

```
TIME : audio snom@239.255.255.245:5555 = file "audio/gong.mp3"
```

Same as above, but played to the Axis camera with id "cam1":

```
TIME : audio axis@cam1 = file "audio/gong.mp3"
```

Converts a text message to speech using the "en" voice of the text-to-speech engine, and play the audio on an Axis camera:

```
TIME : audio axis@cam1 = "voice:en" "the time is " $time:h:m$
```

Serial Communication Ports

COMM

Sends a sequence of bytes to the specified communication port.

Action	Description
COMM name = hexbytes	sends bytes to a serial port

💡 **If both verboseLog and userLog are set to true, the full trace of sent bytes is written to the log file.**

Parameters:

- name - the name of the communications port, as defined in the CommPorts parameter in hsyco.ini
- hexbytes - string with the hexadecimal representation of the sequence of bytes to be sent. You can use multiple unquoted words, quoted strings and variables, that will be appended to generate the byte stream sent to the communication port.

Examples:

```
COMM serialport =  FE03C9104B27
COMM serialport =  FE $body 27
```

Squeezebox

MUSIC

Controls the Squeezebox players.

Action	Description
MUSIC address = ON	turns the player on
MUSIC address = OFF	turns the player off
MUSIC address = PLAY	starts playing music in the active playlist

Parameters:

- address - player number, starting from 0 for the first player, based on the listing order of the slimPlayers parameter in hsyco.ini.

Examples:

```
MUSIC 0 = ON
MUSIC 1 = PLAY
```

User Interface

UISESSIONSET

Changes the dynamic attributes of an identified GUI object, just like UISET, but only affects the client session that generated the event that triggered this action.

UISESSIONSET can only be used with PAGE and USER events, and will override settings made with UISET.

Action	Description
UISESSIONSET id.attr = value	id is the unique name that identifies the object, specified in the index.hsm file with the extension !id after the object type

Parameters:

- id - object id
- attr - attribute name
- value - attribute value. You can use multiple unquoted words, quoted strings and variables, that will be appended to form the value parameter.

Examples:

```
USER mybutton = 1 : UISESSIONSET mytext.style = italic
```

UISET

Changes the dynamic attributes of an identified GUI object.

Action	Description
UISET id.attr = value	id is the unique name that identifies the object, specified in the index.hsm file with the extension !id after the object type

Parameters:

- id - object id
- attr - attribute name
- value - attribute value. You can use multiple unquoted words, quoted strings and variables, that will be appended to form the value parameter.

Examples:

```
UISET mytext.style = italic
UISET mychart.barcolor =”#000000”
```

USER

Triggers a USER event and the userCommand(String name, String param) Java method and JavaScript function.

Can be used like a function call in EVENTS, and as a calling mechanism between EVENTS, JavaScript and Java.

Action	Description
USER name = param	executes the userCommand(name, param) Java method and JavaScript function, and the USER name = param event. Either name or param can be omitted, and will be passed as empty strings, but not both

Parameters:

- name - passed as the name parameter to USER and userCommand(). You can use multiple unquoted words, quoted strings and variables, that will be appended to form the name parameter
- param - passed as the param parameter to USER and userCommand(). You can use multiple unquoted words, quoted strings and variables, that will be appended to form the param parameter.

Examples:

```
USER hometheater = "on"
USER hometheater = "changed" $newvalue
```

Draw API

The Draw API allows to create graphics through various drawing instruction. These instructions are grouped together (separated as semicolons) and set as the value of the **draw UI attribute** of any image object.

E.g. The following instructions:

```
fill('150,150,150');rectangle(10,10,100,100,red,green);line(10,10,110,110,black);
```

can be drawn from the events environment withe the following line of code:

```
uiset myimage.draw = "fill('150,150,150');rectangle(10,10,100,100,red,green);line(10,10,110,110,black);"
```

In any of the functions, the coordinates of the top left pixel are **0,0**.

Functions

Arc

```
arc(x, y, w, h, start, extent, type, border_color, fill_color)
```

Draws an arc.

Parameters:

- x: the x coordinate of the upper-left corner of the arc's framing rectangle
- y: the y coordinate of the upper-left corner of the arc's framing rectangle
- w: the overall width of the full ellipse of which this arc is a partial section
- h: the overall height of the full ellipse of which this arc is a partial section
- start: the starting angle of the arc in degrees
- extent: the angular extent of the arc in degrees
- type: the closure type for the arc: "open", "chord" or "pie"
- border_color: see the note below
- fill_color: see the note below

Ellipse

```
ellipse(x, y, w, h, border_color, fill_color)
```

Draws an ellipse or a circle.

Parameters:

- x: the x coordinate of the upper-left corner of the ellipse's framing rectangle
- y: the y coordinate of the upper-left corner of the ellipse's framing rectangle
- w: the overall width of the ellipse
- h: the overall height of the ellipse
- border_color: see the note below
- fill_color: see the note below

Fill

```
fill(color)
```

Fills the whole area with given color.

Parameters:

- color: see the note below

Image

```
image(file, x, y)
```

Parameters:

- file: the full pathname, relative to the HSYCO home directory
- x: the x coordinate of the location where the String should be rendered - y: the y coordinate of the location where the String should be rendered

Line

```
line(x1, y1, x2, y2, ...., color)
```

Draws a sequence of connected lines.

Parameters:

- x1: X axis coordinate of first point
- y1: Y axis coordinate of first point
- x2: X axis coordinate of second point
- y2: Y axis coordinate of second point
- xN: X axis coordinate of N-th point
- yN: Y axis coordinate of N-th point
- color: see the note below

Rectangle

```
rectangle(x, y, w, h, border_color, fill_color)
```

Parameters:

- x: the x coordinate of the upper-left corner of the rectangle
- y: the y coordinate of the upper-left corner of the rectangle
- w: the width of the rectangle
- h: the height of the rectangle
- border_color: see the note below
- fill_color: see the note below

Spline

```
spline(x1, y1, x2, y2, ...., color)
```

draws a natural cubic spline interpolation of a sequence of points.

Parameters:

- x1: X axis coordinate of first point
- y1: Y axis coordinate of first point
- x2: X axis coordinate of second point
- y2: Y axis coordinate of second point
- xN: X axis coordinate of N-th point
- yN: Y axis coordinate of N-th point
- color: see the note below

String

```
string(text, x, y, font, style, size, color)
```

Parameters:

- text: quoted text
- x: the x coordinate of the location where the String should be rendered - y: the y coordinate of the location where the String should be rendered - font: font name, like “arial”, “helvetica”, “courier” etc.
- style: “bold”, “italic”, or “bolditalic”
- color: see the note below

Stroke

```
stroke(w)
```

Set the stroke width for all subsequent drawing commands.

Parameters:

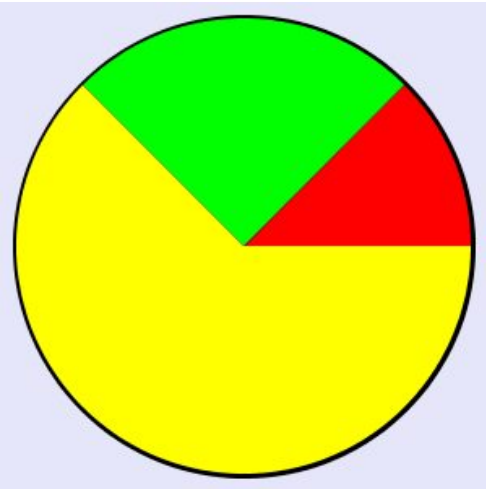
- w: the width of the stroke

> A **color** can be set using:
> - an RGB comma separated numeric notation (delimited by quotes), with values from 0 to 255 (for example, **"255,0,0"** is red)
> - an RGBA comma separated numeric notation (delimited by quotes), with values from 0 to 255 for R,G and B and from 0 to 100 for the alpha channel (for example, **"255,0,0,50"** is red with 50% alpha)
> - color names. The valid names are: black, blue, cyan, darkgray, gray, lightgray, green, magenta, orange, pink, red, white, yellow, transparent.

Examples

Drawing a pie chart

In this example we're going to draw a pie chart with three slices of 45°, 90° and 225°. Keep in mind that for the arc function, 0° is the the rightmost point of the circle, and that from the starting point arcs are drawn in an anti-clockwise direction.



Let's describe the instructions we're going to use:

- **fill('230,230,250')** - Fill the background with the specified color
- **stroke(5)** - Set the line width for the pie's border to 5 pixels
- **arc(10,10,280,280,0,360,pie,black,yellow);** - draw a background yellow circle
- **stroke(0)** - Draw no border on the slices
- **arc(10,10,280,280,0,45,pie,black,red)** - draw the first slice from 0° to 45° in red
- **arc(10,10,280,280,45,90,pie,black,green);** - draw the second slice from 45° to 135° in green

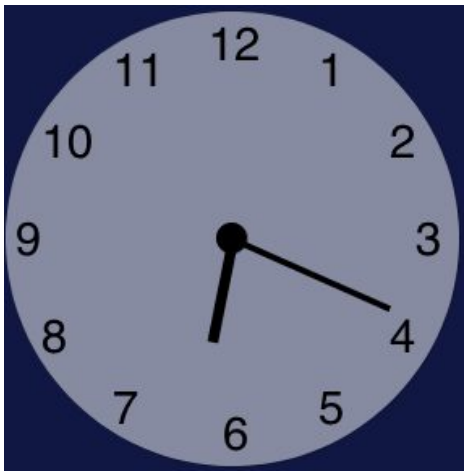
There's no need to draw the third slice, since we already have a yellow background circle.

The line of code in Java or JavaScript:

```
uiSet("myimage","draw","fill('230,230,250');stroke(5);arc(10,10,280,280,0,360,pie,black,yellow);
stroke(0);arc(10,10,280,280,0,45,pie,black,red);arc(10,10,280,280,45,90,pie,black,green);")
```

Drawing a clock

In this example we're going to draw a working clock in JavaScript from the events environment.



In the following code, we declare two functions in the init event that are used to draw the clock. On the time event, which occurs every minute, we call the drawClock function, which returns a string with the instructions needed to draw a clock, and then set it with a uiset.

To try the example, paste this code in **event.txt** and from the Project Editor create a project with an **image object** with **id "myimage"** and **size 300x300** pixels. Since the function that draws the clock is parametric, it's possible to modify the

size of the clock or even draw more than one clock.

```
init : {
        // get position from angle and radius
        function getPos(angle,radius) {
                var radians = (angle-90)*Math.PI/180;
                return {x: radius * Math.cos(radians), y: radius * Math.sin(radians)}
        }

        // given the position, size and font size in pixels and optional time zone (e.g. +1,-2,...)
        // returns a string with the instructions needed to draw a clock
        function drawClock(x,y,size,fontSize,timeZone) {
                var v,p,s;

                var d = new Date();
                var m = d.getMinutes()/60; // minutes 0..1
                var h = ((d.getHours()+(timeZone?timeZone:0))%12)/12; // hours 0..1

                // center point x and y
                var cx = x+Math.round(size/2);
                var cy = y+Math.round(size/2);

                // background
                v = "stroke(0);ellipse("+x+","+y+","+size+","+size+",black,'255,255,255,50');";

                // numbers
                if (fontSize) {
                        for (var i=1; i<=12; i++) {
                                p = getPos(i*30,size*.48-fontSize*.5);
                                v += "string('"+(i<7?' '+i:i)+"',";
                                v += Math.round(cx+p.x-fontSize*.5)+","+Math.round(cy+p.y+fontSize*.35);
                                v += ",helvetica,normal,"+fontSize+",black);";
                        }
                }

                // hours line
                p = getPos((h+m/10)*360,size*.22);
                s = (size/42);
                v += "stroke("+s+");line("+cx+","+cy+",";
                v += Math.round(cx + p.x)+",";
                v += Math.round(cy + p.y)+",";
                v += "black);";

                // minutes line
                p = getPos(m*360,size*.37);
                s = (size/75);
                v += "stroke("+s+");line("+cx+","+cy+",";
                v += Math.round(cx + p.x)+",";
                v += Math.round(cy + p.y)+",";
                v += "black);";

                // cap
                s = Math.round(size/15);
                var hs = Math.round(s/2);
                v += "stroke(0);ellipse("+(cx-hs)+","+(cy-hs)+","+s+","+s+",black,black);";

                return v;
        }
}

time : {
        // get the instructions to draw a clock
        var v = drawClock(0,0,280,29);

        // to add a clock:
```

```
        // v += drawClock(100,100,50,50,-1);

        // uiset of the draw attribute
        uiSet("myimage","draw",v);
}
```

Event Keywords

An event is defined as a keyword that identifies the event type, usually followed by one or more parameters and conditional operators. Every keyword uses a specific syntax that interprets parameters according to its own rules.

System Events

DAY

A stable condition that is true during day time, as computed based on the latitude and longitude data set in hsyco.ini and the SunriseOffsetMinutes and SunsetOffsetMinutes parameters.

Event	State	Description
DAY	stable	triggered at sunrise; remains true until sunset

Examples:

```
DAY : IO k.33 k.34 k.35 = 0
```

HAACTIVE

This event is triggered when HSYCO is a master or slave in a high availability configuration and the server is active.

Event	State	Description
HAACTIVE	transient	this HSYCO server is active

Examples:

Send an email when the slave HSYCO server becomes active:

```
HAACTIVE AND $HAMASTER$ = 0 : MAIL email@hsyco.com = hsyco@hsyco.com "Sent from HSYCO" "The slave server is
now active"
```

HSYCOSTART

This event is triggered only once when HSYCO starts, after the initialization and HTTP server start-up, but before the execution of the different field devices interface threads.

Event	State	Description
HSYCOSTART	transient	occurs only once at start-up

Examples:

```
HSYCOSTART : $count = 0
```

INIT

This event is triggered when HSYCO reloads the events.txt file.

Reloading occurs when HSYCO starts and every time the file is manually saved, even with no content changes.

Event	State	Description
INIT	transient	events.txt has been reloaded

Examples:

```
INIT : $count = 0
```

NIGHT

A stable condition that is true during night time, as computed based on the latitude and longitude data set in hsyco.ini and the SunriseOffsetMinutes and SunsetOffsetMinutes parameters.

Event	State	Description
NIGHT	stable	triggered at sunset; remains true until sunrise

Examples:

```
NIGHT: IO k.33 k.34 k.35 = 1
```

POWER

Triggered by a change of the current power load level, as set using the powerSet() Java method or JavaScript function, or the POWER action.

Event	State	Description
POWER	transient	triggered by any change to the power level
POWER > power POWER < power POWER >= power POWER <= power	stable	occurs when a change in the power level is detected. It is true if the comparison condition is satisfied

Parameters:

- power - the power in Watt.

Examples:

```
POWER > 2000 : IO k.70 = 0
POWER : LOG "NEW POWER: " $POWER$
```

PROGRAMTIMER

Triggered by a program timer.

Event	State	Description
PROGRAMTIMER name	transient	triggered after the program timer's set time-interval

Parameters:

- name - program timer id.

Examples:

Turns on light k.33 approximately 10 seconds after the click on a user button:

```
USER timer : PROGRAMTIMER test = SET 10
PROGRAMTIMER test : IO k.33 = 1
```

SUNAZIMUTH

This event is triggered every time there is a change in the integer part of Sun's azimuth angle (the angle between North and the Sun's projection on the horizon), as computed based on the latitude and longitude data set in hsyco.ini and the SunriseOffsetMinutes and SunsetOffsetMinutes parameters.

Event	State	Description
SUNAZIMUTH	transient	triggered when the Sun's azimuth changes
SUNAZIMUTH = value SUNAZIMUTH > value SUNAZIMUTH < value SUNAZIMUTH >= value SUNAZIMUTH <= value	stable	true if the condition matches. Angles are compared ignoring decimals

Parameters:

- value - angle in decimal degrees, as a positive integer number, between 0 and 359.

Examples:

```
SUNAZIMUTH > 180 AND SUNAZIMUTH < 220 : IO k.54 = 0
```

SUNELEVATION

This event is triggered every time there is a change in the integer part of Sun's elevation angle (the angle between the horizon and the Sun), as computed based on the latitude and longitude data set in hsyco.ini and the SunriseOffsetMinutes

and SunsetOffsetMinutes parameters.

Event	State	Description
SUNELEVATION	transient	triggered when the Sun's elevation changes
SUNELEVATION = value SUNELEVATION > value SUNELEVATION < value SUNELEVATION >= value SUNELEVATION <= value	stable	true if the condition matches. Angles are compared ignoring decimals

Parameters:

- value - angle in decimal degrees, as an integer number, between -90 and 90.

Examples:

```
SUNELEVATION > 45 : IO k.55 = 0
```

TIME

The current time event.

In its simple form, it is just a transient event that is triggered every minute.

It can also be used as a stable condition, comparing the current time with a constant time in the HHMM format.

Event	State	Description
TIME	transient	triggered at the beginning of each minute
TIME = hhmm TIME > hhmm TIME < hhmm TIME >= hhmm TIME <= hhmm	stable	compares the current time with the specified time in the HHMM format

Examples:

```
TIME : CAMERAREC entrance = 60
TIME = 0930 : IO k.33 = 0
```

Cameras

CAMERA

This event is true while HSYCO is recording video from a specific camera.

Recording could start automatically when a camera notifies a recording request with the recording HTTP API call:

```
http://192.168.0.50/x/camerarec?camera=name
http://192.168.0.50/x/camerarec?camera=name&zone=id
```

Recording can also be triggered by the CameraRecTrigger() Java call or the CAMERAREC or CAMERARECFULL actions in EVENTS.

Event	State	Description
CAMERA name	stable	the camera is recording
CAMERA name = zone	transient	transient event triggered by a camera recording request with zone information

Parameters:

- name - the camera id
- zone - the zone id.

Examples:

```
CAMERA entrance
```

CAMERACOMMAND

Triggered by a camera control input from the Web interface.

It can be used to execute arbitrary actions when the user clicks on the active areas of the camera view.

Event	State	Description
CAMERACOMMAND name = function	transient	if this event has a positive match in EVENTS, the standard PTZ command associated to the camera's PTZ driver is not executed

Parameters:

- name - the camera id
- function - see the table below.

icon	function	action
	focus	far
	focus	near
	zoom	tele

	zoom	wide
	move	left
	move	right
	move	up
	move	down
	move	stop

Examples:

```
CAMERACOMMAND entrance = focusfar
```

CAMERAVIEW

This event is used to check if HSYCO is currently displaying the real-time video from a camera.

The event will reset to off when no Web client has been loading images from the camera for several consecutive seconds.

Event	State	Description
CAMERAVIEW name = status	stable	the camera real-time view status

Parameters:

- name - the camera id
- status - set to “on” or “1” to match the event when HSYCO is displaying the live view, and to “off” or “0” to match the event when HSYCO is not displaying a live view of the camera on any client.

Examples:

```
CAMERAVIEW entrance = on
```

DMX

DMX

Events on the DMX bus channels.

Event	State	Description

DMX address	transient	any change to the channel value
DMX address = ON	stable	channel value > 0
DMX address = OFF	stable	channel value = 0
DMX address = value DMX address > value DMX address < value DMX address >= value DMX address <= value	stable	the channel value is equal, greater, greater or equal, less, less or equal to the given value. Valid values are numbers between 0 and 255

Parameters:

- address - when using one DMX gateway, address will be a standard DMX-512 address between 1 and 512. If there is more than one gateway, 1000 must be added to the DMX address for the second gateway, 2000 for the third one and so on. For example, 2100 stands for the address 100 on the DMX bus controlled by the third gateway.

Examples:

```
DMX 100 = ON
DMX 1200 = OFF
DMX 135 = OFF OR DMX 100 > 50
DMX 41 <= $LEVEL
```

DMXSTART

This event is triggered when starting the monitor threads for each DMX bus, once per bus at the start of the execution of HSYCO, and also after every restart of the monitor thread.

Event	State	Description
DMXSTART busid	transient	occurs when HSYCO connects to the DMX gateway. HSYCO establishes the connection at start-up. If the gateway is turned off and then on, or if communication errors occur, HSYCO automatically re-establishes the connection as soon as possible, calling this event in case of success

Parameters:

- busid - the id of the DMX gateway. The first gateway has address 0. The second has address 1, etc. busid can be omitted if 0.

Examples:

```
DMXSTART: DMX 100-200 = OFF
DMXSTART 1: DMX 1100-1200 = OFF
```

Infrared Control

IR

This event is triggered when an IRTrans receives and successfully decodes an infrared command.

💡 IRTrans can only receive and decode infrared commands that have been previously stored in its internal FLASH memory database.

Event	State	Description
IR name = command	transient	command received by an IRTrans device

Parameters:

- name - the IRTrans, as defined in the IRTrans parameter in hsyco.ini
- command - the command format is “remote.command”, that is the remote control database name followed by a dot and the command name.

Examples:

```
IR theater = dvd.play
```

I/O Servers

IO

Triggered by a status change of any data port of an I/O server.

Event	State	Description
IO name	transient	any change of an I/O variable
IO name = value IO name > value IO name < value IO name >= value IO name <= value	stable	the value is equal, greater, greater or equal, less, less or equal to the given value. **💡 The names and values of the I/O variables are specifically related to the type of I/O server. Values can be numeric or strings**

Parameters:

- name - the data point name of the input or output port. According to the type of server, the format changes but it generally appears as server name, as declared with the ioServers parameter in hsyco.ini, followed by a dot and the input/output port name. See the Application Notes for more information.

Examples:

```
IO tempmeter.indoor > 25.5
IO contacts.c1 = 1
```

IOSTART

This event indicates that the connection with an I/O server has been correctly established.

It should be safe to associate this event with actions that set the I/O server writable variables.

Event	State	Description
IOSTART id	transient	Triggered by the start of the communication thread of each I/O server - once for each server at the beginning of the execution of HSYCO, and also after every reboot of the communication thread, for example after communication errors

Parameters:

- id - the I/O server number, starting from 0, based on the listing order of the ioServers parameter in hsyco.ini; If only one I/O server is available, it can be omitted.

Examples:

```
IOSTART: IO contacts.c1 = 1
IOSTART 2: IO relays.door = 1
```

Leak Detector

LEAK

This stable event is triggered by a LEAK action when the Intelligent Leak Detector detects a leak condition, or the leak condition is reset.

See the LEAK action for additional information.

Event	State	Description
LEAK name	transient	any status change of the leak detector
LEAK name = ON	stable	leak condition detected
LEAK name = OFF	stable	leak condition reset

Parameters:

- name - the name of a leak detector. Using different names you can implement several independent leak detectors.

Examples:

```
HSYCOSTART : PROGRAMTIMER modbusread = repeat 4
PROGRAMTIMER modbusread : IO modbus.2.768 = readholdingregisters:uint, LEAK water = IO modbus.2.768
LEAK water = ON : MAIL john@example.com = hsyco@example.com “Urgent Message from HSYCO” “Leak Detected”
LEAK water = OFF : MAIL john@example.com = hsyco@example.com “Message from HSYCO” “Leak Reset”
```

Network Services

HTTP

This transient event is triggered when the HSYCO HTTP server receives a GET request with the following path:

```
/x/httpcall?query
```

💡 **The HTTP client's IP address must be in the trusted network range, or the httpcall request will be rejected.**

If a matching event is found, and the httpCallEvent JavaScript or Java callbacks are not defined, the HTTP server will respond with an "ack" text/plain answer to the HTTP client (with HTTP response code 200).

HSYCO will return a 404 Not Found response code if all of the following conditions are true:

- no HTTP event matches the query in events.txt
- the httpCallEvent JavaScript callback is not defined in events.txt, or doesn't return a value
- the httpCallEvent Java callback is not defined in user.java or in any plugins, or returns null.

Event	State	Description
HTTP query	transient	the HSYCO HTTP/HTTPS server received an HTTP GET request: /x/httpcall?query

Parameters:

- query - the string following the ? in the URL path.

Examples:

```
HTTP "test" : log = "test http call"
```

LOCATION

This event is triggered when a change in the association status of a client to the Wi-Fi access points is detected.

It is a stable condition that represents the current connection state an location of WiFi clients logged into the HSYCO Web interface.

Event	State	Description
LOCATION address = zone	stable	a device connects to the access point identified by the zone name, as declared in the LocationBases parameter in hsyco.ini
LOCATION address = ON	stable	a device connects to the WiFi network
LOCATION address = OFF	stable	a device disconnects from the WiFi network

Parameters:

- address - the WiFi device's MAC address. It is made of 6 byte pairs, written in hexadecimal format and separated by ":".

Examples:

```
LOCATION aa:01:33:e4:41:00 = entrance
LOCATION aa:01:33:e4:41:00 = ON
LOCATION aa:01:33:e4:41:00 = OFF
```

PING

This transient event is generated by a PING action, that tests the reachability of one or more IP hosts, identified by their hostnames or IP addresses, within the optional timeout defined in milliseconds (or using a default timeout of 200 ms).

Event	State	Description
PING hostname = ON	transient	the device has been reached after a PING action
PING hostname = OFF	transient	the device is not reachable after a PING action

Parameters:

- hostname - the IP device name or IP address, as called by the corresponding PING action.

Examples:

```
TIME: PING 192.168.1.1
PING 192.168.1.1 = OFF : LOG = "router is off-line"
```

URL

This transient event is generated by a URL action, that sends an HTTP or HTTPS request to a given url.

The response can be checked for a specific numeric status code or looking at a string in the response body.

Event	State	Description
URL “url” = status_code	transient	the response status code to the HTTP url request is equal to status_code
URL “url” *= “string”	transient	the response data to the HTTP url request contains the string

Parameters:

- url - a full url, including the mandatory http:// or https:// query scheme and optional query string
- status_code - the numeric status code returned by the server’s response; 200 is usually the code returned by a successful request
- string - the event matches any response containing the given string in the response body.

Examples:

```
TIME 0800: URL GET "http://10.0.0.5/cgi-bin/cmd.sh?id=relay1"
URL "http://10.0.0.5/cgi-bin/cmd.sh?id=relay1" : LOG = "relay opened"
```

PBX

PBX

Called when a call notification is sent to HSYCO by the PBX system.

The PBX should generate HTTP requests to the HSYCO Web Server, using the following formats:

```
http://192.168.0.50/x/vcall/<from>/<to>
```

or:

```
http://192.168.0.50/x/vcall?from=<from>&to=<to>
```

For example, the HTTP request:

```
http://192.168.0.50/x/vcall?from=21&to=25
```

is interpreted as a call notification from internal number 21 to number 25.

💡 **The HTTP client's IP address must be in the trusted network range, or the httpcall request will be rejected.**

Event	State	Description
PBX from = to	transient	phone call event, with originator and destination data

Parameters:

- from - the caller number
- to - the called number.

Examples:

A call from number 22 to number 44 turns on light k.33:

```
PBX 22 = 44 : IO k.33 = 1
```

Squeezebox

MUSIC

Triggered by a status change in a Squeezebox player. It is a stable condition that represents the current ON/OFF/PLAY/PAUSE status of the player.

Event	State	Description
MUSIC address = ON	stable	player on
MUSIC address = OFF	stable	player off
MUSIC address = PLAY	stable	player on - play

MUSIC address = PAUSE	stable	player on - pause

Parameters:

- address - player number, starting from 0 for the first player, based on the listing order of the slimPlayers parameter in hsyco.ini.

Examples:

```
MUSIC 0 = ON
MUSIC 1 = PLAY
```

Timers and Schedulers

TIMER

This is a stable event that matches the current status of a user timer or scheduler.

It is triggered by the timer's activation and deactivation.

Event	State	Description
TIMER name = ON	stable	true when the timer or scheduler rule is active (ON status)
TIMER name = OFF	stable	true when the timer or scheduler rule is not active (OFF status)

Parameters:

- name - the timer's id, as declared with the Timers parameter in hsyco.ini, or the scheduler's rule name.

Examples:

```
TIMER irrigation = ON : IO k.33 k.34 = 1
TIMER irrigation = OFF : IO k.33 k.34 = 0
```

User Interface

PAGE

Triggered when the specified page of a project is viewed by a client.

There is no guarantee that the event would be fired the exact moment the page is loaded.

Network delays or errors could cause delayed triggering of this event.

💡 **You can use the UISESSIONSET action with this event**

Event	State	Description

PAGE project = page	transient	a Web client is viewing a specific page of a project

Parameters:

- project - the name of the project
- page - the page id, with the #landscape or #portrait suffix if the page has distinct orientation versions in the project.

Examples:

Count how many times the portrait and landscape versions of a project's menu page are viewed:

```
PAGE "myproject" = "menu#landscape" : $landscape_counter + 1
PAGE "myproject" = "menu#portrait" : $portrait_counter + 1
```

USER

Triggered by user clicks on buttons created in the Web interface with the (user), (usermini), (usermicro), (userrgb) or (userimage) objects, by a USER action in EVENTS, or by the user() Java method or JavaScript function.

Event	State	Description
USER name = param	transient	this format requires a match with both name and parameter
USER name	transient	this format requires a match with name only

Parameters:

- name - the name field of the (user) object
- param - the id of the (user) object.

Examples:

Turns on light k.33 approximately 10 seconds after the click on a user button:

```
USER timer : PROGRAMTIMER test = SET 10
PROGRAMTIMER test : IO k.33 = 1
```

Events Programming

The EVENTS programming environment is based on a simple language that associates one or more actions to a field event or combination of conditions.

Programming EVENTS is as simple as editing the events.txt file, located in the main directory. Compared to Java, there is no compilation process. When you save the file, it is automatically reloaded and becomes immediately effective.

Introduction

An event is an expression that refers to the persistent status or transient event of devices, to conditions applied to local variables, and to various internal events. An action is a control command sent to a device, or several other internal functions, like setting variables and program timers.

Example:

```
IO k.33 = 1 : IO k.34 = 1
```

when the light actuator with data point name k.33 is turned on, HSYCO automatically turns on the actuator with data point name k.34.

Example:

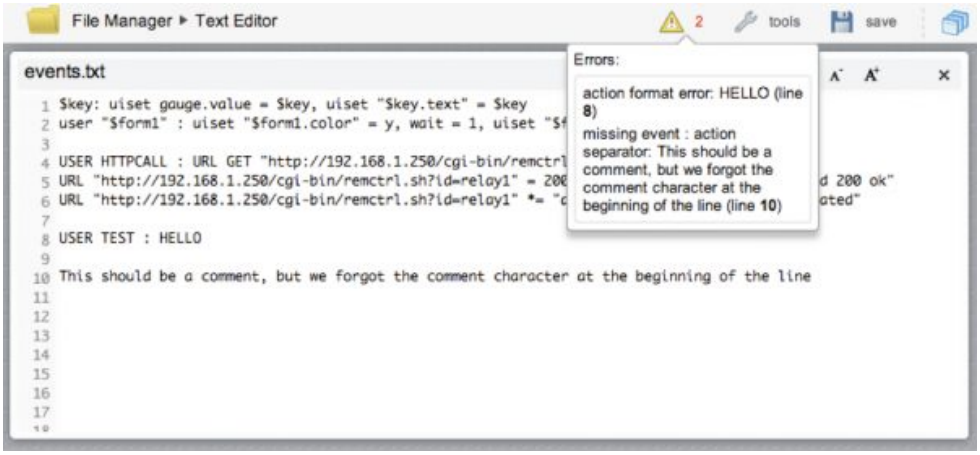
```
DAY : IO k.33 k.34 k.35 k.36 = 0
```

at sunrise, the actuators named k.33, k.34, k.35 and k.36 are turned off.

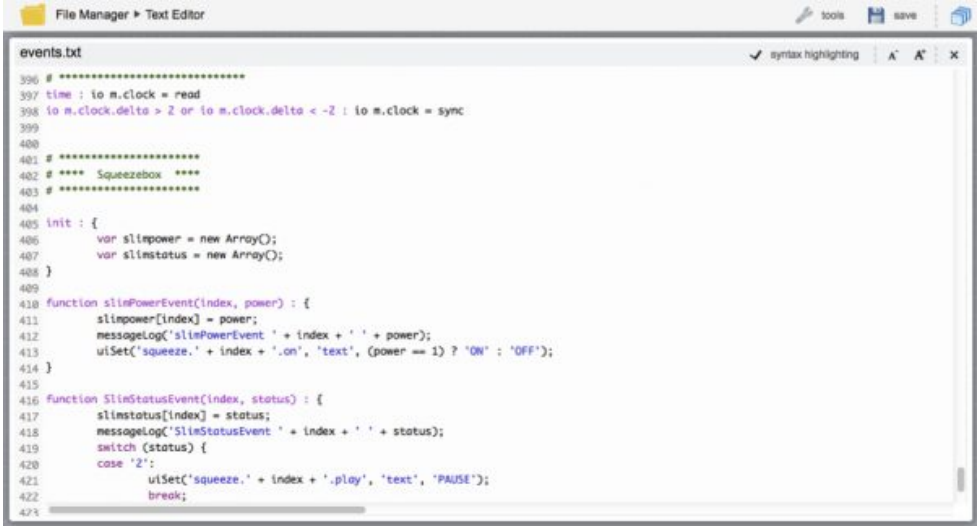
In this chapter we will describe the general format of events.txt, and all the built-in events and actions.

The events.txt file is usually modified with the text editor in HSYCO's File Manager. When you save the file, it is automatically checked and, if errors are found, a warning icon appears in the top bar.

Touch that icon to show the list of errors. Errors are also written in the log file. Lines containing errors are ignored.



You can also enable the editor's syntax highlighter for improved readability.



The EVENTS Language

EVENTS uses the simple format:

```
event : actions
```

Several events can be combined together to implement complex expressions which represent the combination of different conditions.

The logical operators AND, OR, NOT and the round brackets are available to define the operations precedence.

If the brackets are not used, the AND and OR operations are executed from left to right. The NOT operator has priority and is executed on the first expression after it, before AND or OR conditions.

Example:

```
IO k.33 OR IO K.34
```

this event occurs any time the actuators 33 or 34 change their status.

Example:

```
CAMERA "entrance" AND NOT IO K.34 = 0
```

is an event that occurs when the camera “entrance” starts recording, but only if the K.34 datapoint is not 0.

Example:

```
IO k.33 > 5 OR IO K.34 = 0 AND IO K.35 = 0
```

is equivalent to

```
(IO k.33 > 5 OR IO K.34 = 0) AND IO K.35 = 0
```

Example:

```
NOT IO k.33 = 1 AND IO k.34 = 1
```

is equivalent to

```
(NOT IO k.33 = 1) AND IO k.34 = 1
```

Comments

Comments are vital to describe the functions, even if the EVENTS language is very easy to understand.

A comment is identified by the # character.

All characters following # will be ignored until the end of line, unless the # character is enclosed between double quotes.

Example:

```
# actions executed at sunset
NIGHT:
        IO k.33 k.34 k.35 k.36 = 1,     # garden lights on
        IO k.43 k.44 = UP,              # rise the awnings
        LOG = "# and this is not a comment"
```

Other General Rules

Keywords are case insensitive. Devices ids are also case insensitive. The EVENTS interpreter converts all ids to lower case, so it is recommended to use only lower case ids in hsyco.ini if you use EVENTS.

The built-in constant values (ON, OFF, UP, DOWN, STOP, FLIP, RECON, RECOFF, MERGE, UNMERGE, PLAY) can be written in capital letters or in lower case. All other custom values will retain the case.

Example:

```
IO k.43 = UP and IO k.44 = DOWN : CAMERA cam1 = 60
```

is equivalent to:

```
io k.43 = up AND io k.44 = down : camera cam1 = 60
```

Example:

```
IO k.36 = 1 : UISET message.text = "Stairs light ON"
```

The text “Stairs light ON” remains unchanged.

Spaces and not-relevant tabulation characters are ignored.

In some cases separation spaces are not necessary:

```
IO k.36=1
```

is a valid expression, while

```
IOk.36 = 1
```

is not valid because the keywords must be separated by their attributes.

The EVENTS interpreter tries to understand strings even when not enclosed in double quotes, but we consider a good practice to always use double quotes to delimit text strings, and this may even become mandatory in future releases.

Example:

```
IO k.36 = 1 : LOG = "I turn on the light: 36"
```

Stable and transient events

```
IO k.33
```

is a transient event. It occurs only during the status change of the actuator with data point name k.33. If it is part of a complex expression, it will be true only when the expression evaluation is triggered by that specific change event, and will be false when the expression evaluation is triggered by any other event. For example:

```
IO k.33 AND IO k.33
```

is an event that will never occur, because two transient events can never be simultaneously true.

A stable event is value based condition. For example:

```
IO k.33 > 40
```

is a stable event. It occurs when the status value of actuator k.33 is greater than 40 (more than 40% of a dimmer level).

Unlike the transient events, this expression is true anyway, even after the transient event, so:

```
IO k.33 > 40 AND IO k.34 > 40
```

is an event that may occur when both the actuators k.33 and k.34 are on with a level greater than 40%.

In some cases it would be necessary to use a transient event associated to a device, also combined with status expressions of the device itself.

Example:

```
IO k.33 = 1 AND IO k.34 = 1 AND NOT IO k.33
```

The expression NOT IO k.33 excludes the events generated from any status change of device k.33. Meanwhile IO k.33 = 1 makes the expression true only when the device status is ON.

As a result, this event occurs when the devices k.33 and k.34 are simultaneously ON, but only on the status change of device k.34, and ignoring the status change of device k.33.

Example:

```
IO k.33 = 1 AND IO k.34 = 1 AND NOT IO k.33 :
        IO k.33 = 0, WAIT = 0.5, IO k.33 = 1
```

If light k.33 is already ON, as soon as light k.34 is turned on, light k.33 will flash once.

By omitting the expression NOT IO k.33 the light wouldn't stop blinking because the action would trigger the same event again and again.

One Event, More Rules

You can define several rules referencing the same event.

When HSYCO detects any status change in the system, it scans EVENTS for all event expressions that could be affected by that event. All relevant expression are then evaluated and, if true, the associated lists of actions are executed.

This process is normally performed top-down on the events.txt file, but the execution order is not guaranteed.

Example:

```
IO k.34 = 1 : IO k.36 = 0
IO k.34 = 1 AND DAY : IO k.37 = 0
```

When the actuator k.34 is turned on, these rules cause actuator k.36 to go off, while the actuator k.37 will be turned off only during the day.

Actions

You can associate multiple, comma separated actions to each event. For example:

```
NIGHT : IO k.33 = 1, IO k.34 = 1, IO k.88 = UP
```

In order to speed up the writing and make rules clearer, many commands let you specify a list of addresses as attributes of the command. The rule written above can be rewritten in a shorter form as:

```
NIGHT : IO k.33 k.34 = 1, IO k.88 = UP
```

Each event : action rule is normally written on a single line. However the readability can be improved by typing the list of actions on following lines, after adding a comma at the end of the preceding lines. It Is also possible to write only the event expression on the first line, followed by : and start the list of actions in the following line:

```
NIGHT :
        IO k.33 k.34 = 1, IO k.86 k.87 k.88 = UP
```

The event expression can't be written on more than one line.

Variables

Variables are identified by any text name starting with the character $. For example:

```
$level
```

Variable names are case insensitive, so $level and $LEVEL are the same variable. Only letters and numbers are allowed, no special characters.

A variable can be assigned any value and doesn't need to be declared before being used.

Variables and Actions

Variables can be assigned constant values or expressions results:

```
HSYCOSTART : $count = 0
```

assigns the value 0 to the variable $count as soon as HSYCO starts.

The basic arithmetic operations (+, -, * and /), the division's reminder (%) and rounding to an arbitrary number of digits (ROUND n) are supported.

💡 **You can round a number from 0 to 9 decimal digits. Rounding to 0 decimal digits rounds to the nearest integer.**

If the original value of the variable and the operator's value are both numbers ending with %, the arithmetic operation will retain the % in the result.

If only one of the vaules end with %, that number will be stripped of the trailing % and divided by 100.0 before executing the arithmetic operation.

A variable text can be appended to another text by simply using the + operator. This automatically performs a string append only if the variable original content or the operand are not numerical.

Example:

```
IO k.33 = 1 : $count + 1
```

increases by 1 the value of $count any time the actuator k.33 is turned on.

Example:

```
$voltage : $rv = $voltage, $rv ROUND 2
```

the $rv variable is set to the value of $voltage, rounded to 2 decimal digits.

Example:

```
IO k.33 = 1 : $count + 1, $count % 4
```

like the previous example, but it also computes the modulo of 4 of the variable, writing back the result in the variable itself. This way this variable will be increased up to 3, before turning into 0 again.

To delete a variable, assign an empty string to it:

```
IO k.33 = 0 : $status = ""
```

When appending strings, more than one string can be written after the equals sign or, in general, after the operator. In this case the strings will be concatenated.

This is particularly useful when assigning a complex text to a variable.

Example:

```
time : $body = "The daily power consumption " $date:y/m/d$ " at " $time:h:m:s$ " is " $power$ " Watt"
```

You can also apply a decimal number format pattern to variables, using the FORMAT "pattern" operator.

A pattern contains a positive and negative sub-pattern, for example, "#,##0.00;(#,##0.00)". Each sub-pattern has a prefix, numeric part, and suffix. The negative sub-pattern is optional; if absent, then the positive sub-pattern prefixed with the localized minus sign ('-' in most locales) is used as the negative sub-pattern. That is, "0.00" alone is equivalent to "0.00;-0.00".

The # symbol represents a digit that is not shown when that digit is zero. The 0 symbol represents a digit that is always shown.

For more information on the pattern format, see the java.text.DecimalFormat class documentation.

💡 **The pattern should always be prefixed with a two letter country or language id and a colon character, for example, “EN:#,##0.00”. In this case the pattern and the resulting format is localized based on the country or language id. If the localization id is not present, the format operator uses the localization rules of the Language parameter in hsyco.ini.**

Example:

```
$voltage : $rf = $voltage, $rf FORMAT "EN:#.00"
```

the $rf variable is set to the value of $voltage, always showing the two most significant decimals, even if the number is integer.

Example:

```
$voltage : $rf = $voltage, $rf FORMAT "IT:#,00"
```

the $rf variable is set to the value of $voltage, always showing the two most significant decimals, and using the Italian localization rules, with the comma character as decimal separator.

Variables and Events

Variables can be part of event expressions, so you can compare the content with constant values or another variable, or compare a device status with the value of a variable.

The comparison operators are:

```
=    >    <    >=    <=    *=
```

*= is a contains operator, matching any left-side string that contains the right-side string.

The comparison is based on numeric values if both operands are numerical, otherwise it is text based, ignoring the difference between upper and lower case letters. If one of the two operands is a number and the other is "off", the number is compared with 0. For example:

```
HSYCOSTART : $count = 0
IO k.33 = 1 : $count + 1
$count : LOG="Increase the count to value: "  $count
$count = 10 : LOG="Tenth turning on of light 33"
```

An event defined just with the variable name, thus omitting the comparison operator, is a transient event which will occur when the variable status changes, regardless of its value.

It is possible to verify if a variable is defined. The event condition:

```
$varname = ""
```

is true if the variable is not defined or if has been explicitly set to "".

Persistent and Volatile Variables

Every time HSYCO is restarted, all values previously set in the variables are normally lost. By simply adding the character ! at the end of the variable name, a variable becomes persistent:

```
$count!
```

is a persistent variable whose value is retained when HSYCO is restarted. Persistent variables are extremely useful, but a variable doesn't need to be always persistent. We recommend to avoid using persistent variable if not necessary. Also, because the ! is part of the name, $count and $count! are totally different variables.

Predefined Variables

The EVENTS built-in variables are:

```
$TIME$
$TIME:HMS$
$DATE:YMD$
$DATE:DOW$
$SUNAZIMUTH$
$SUNELEVATION$
$SUNSET$
$SUNRISE$
$HAMASTER$
$POWER$
```

Predefined variables names always end with the $ character.

> Predefined variables can be used in events rules, but they are not trigger keywords, so they don't directly trigger the rule evaluation when their value changes.

For example, the following line will never execute the LOG action because, while the $TIME:M$ > 30 rule will indeed become true during the second half of each hour, there will be no trigger to evaluate the rule:

```
$TIME:M$ > 30 : LOG = "Second half of the hour"
```

Starting from this example, in order to create a rule that actually matches every minute of the second half on the hour, you should add the TIME keyword to the rule, like this:

```
TIME and $TIME:M$ > 30 : LOG = "Second half of the hour"
```

$TIME$, $SUNSET$ and $SUNRISE$ are particularly useful for events related to the present time, and to sunrise and sunset time. The internal representation of these variables is a positive integer number representing the time in seconds since a reference time.

$TIME$ and $DATE$ also accept special modifiers, like:

```
$TIME:HMS$
$DATE:YMD$
```

They return the current hour, minute and second, as well as day, month and year. You can reference individual date/time fields or combine them, also defining custom separation characters.

```
$DATE:DOW$
```

returns the day of week as a number, with 1 for Monday and 7 for Sunday.

```
$SUNAZIMUTH$
```

is the Sun's current azimuth in integer degrees, between 0 and 359.

```
$SUNELEVATION$
```

is the Sun's current elevation over the horizon in integer degrees, between -90 and 90.

```
$HAMASTER$
```

is set to 1 when the HSYCO Server is defined as the master unit in a high availability configuration or when high availability is not used, or 0 when defined as the slave unit.

```
$POWER$
```

is translated to the current power load - an integer value in Watt. This is the same number that appears in the Web interface.

Time and date examples:

```
$TIME:H$
```

returns the hour, in two-digit format; $TIME:M$ the minutes; $TIME:S$ the seconds

```
$TIME:H:M:S$
```

returns the text 09:15:30

```
$DATE:Y$
```

returns the year; $DATE:M$ the month; $DATE:D$ the day

```
$DATE:D/M/Y$
```

returns the text 14/07/2011.

Sunset example:

```
DAY OR HSYCOSTART: $sunset1h=$sunset$, $sunset1h - 3600, IO k.33 k.34 = 0, $lights = off
```

The variable $sunset1h now contains the time in seconds of one hour before the next sunset, and the following event will cause the turning on of lights at that time:

```
TIME > $sunset1h AND NOT $lights = on:
        IO k.33 k.34 = 1, $lights = on
```

JavaScript Programming

HSYCO integrates a JavaScript interpreter, allowing you to embed JavaScript code in the events.txt file.

There are two ways to embed JavaScript in events: using JavaScript actions and with JavaScript callbacks.

JavaScript actions

Actions can be directly implemented in JavaScript, using curly brackets to enclose JavaScript code:

```
event : { /* JavaScript Code here */ }
```

When the event is triggered, HSYCO will execute the JavaScript code enclosed in the brackets.

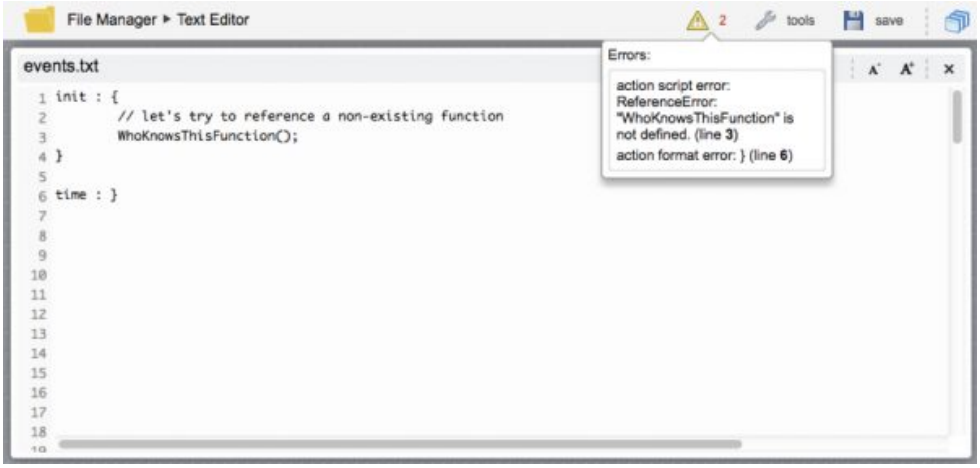
💡 **The basic action format and the JavaScript action format are exclusive and cannot be mixed after the colon. It is of course possible to define two distinct event:action rules, using the same event, one with standard actions and the other with embedded JavaScript.**

The code can be written on multiple lines, only the opening bracket needs to be in the same line of the event section, for example:

```
TIME : {
        //  JavaScript Code here
}
```

The syntax highlighter and integrated syntax error viewer are also available for JavaScript. When you save the events.txt file from the editor, the file is parsed and all JavaScript code is interpreted. Syntax errors, including calling functions that are not defined, are reported in the error viewer, and also logged in the log file.

In the following example, at line 3, we are referencing a JavaScript function that doesn't exist. We also have another error at line 6, where the open curly bracket is missing from the action.



Coding JavaScript actions

The code you can write in actions is standard JavaScript. Because the interpreter is fully embedded in the HSYCO Java core, you also have access to the standard Java classes and, most importantly, to the HSYCO's Java commands and utility methods, with only minor differences and a few limitations related to the differences in data types between JavaScript and Java languages.

In the following example, when the I/O data point k36 goes to 1, the I/O data point k.37, a dimmer in this case, will step from 10% to 100% in 10% increments every 400 milliseconds:

```
io k.36 = 1 : {
       for (i = 10; i <= 100; i += 10) {
               ioSet('k.37', i);
               sleep(400);
       }
}
```

As you can see we have combined standard JavaScript features, like the for loop, with HSYCO's command APIs, the ioSet() and sleep() methods, to create a simple JavaScript action and associate it to a specific event.

You can directly use Java classes too, for example:

```
time = 0000 : {
       (new java.io.File("myfile.txt")).renameTo(new java.io.File("myfile_backup.txt"));
}
```

Here we are using the File class, in the java.io package, to rename a file every day at midnight.

Comments in the JavaScript code

You can use the normal JavaScript comment formats. Single line comments start with // and multi line comments start with /* and end with */.

Because the EVENTS parser is also run on JavaScript code, you can also use the # character to mark text as comment until the end of the line is reached.

As with the standard EVENTS syntax, if you need to use the # character in JavaScript strings, you should use double quotes, not single quotes, to delimit the string containing # characters.

JavaScript code errors checking

The EVENTS parser is able to report syntax errors in the embedded JavaScript code. If errors are detected, the whole event:action rule is not loaded in the EVENTS engine, to avoid run-time errors.

Unfortunately, to perform these checks, the parser needs to perform a dummy run of all JavaScript code every time events.txt is loaded.

During the test run, all calls to HSYCO methods are automatically skipped, and JavaScript variables are segregated in a dedicate test namespace, but any other native Java or JavaScript command will execute as if the event was triggered at run-time.

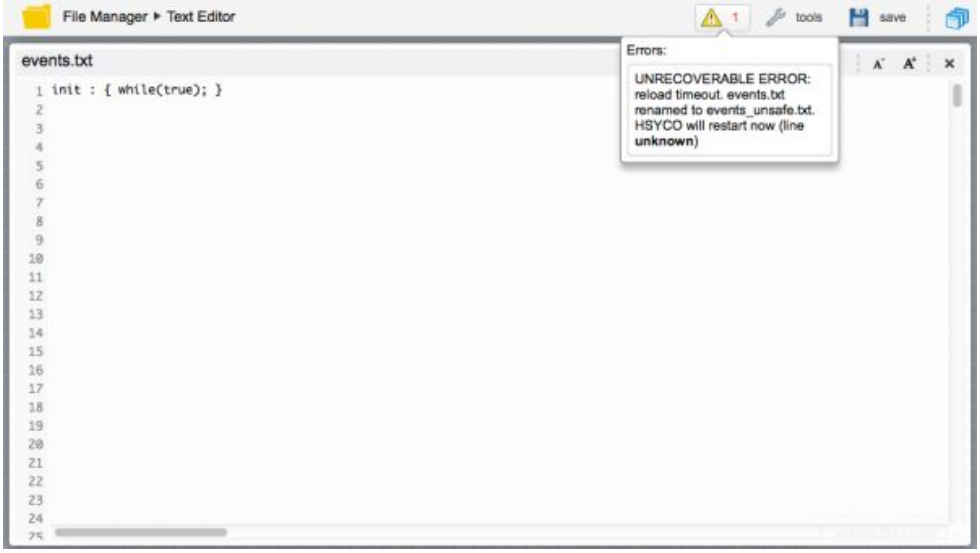
In the file rename example, the file would actually be renamed not only at midnight, but also every time events.txt is reloaded.

If you want to avoid unwanted execution of code, you can check the `__run` predefined JavaScript variable, that is set to 0 when code is test run, and to 1 when executed on proper events. A safe file rename example would be:

```
time = 0000 : {
       if (__run == 1) {
               (new java.io.File("myfile.txt")).renameTo(new java.io.File("myfile_backup.txt"));
       }
}
```

Infinite loops in code, or JavaScript code that runs for a very long time are handled in a special way in HSYCO. When events.txt is saved, If any code section takes more than 20 seconds to execute during the test run, HSYCO will abort the JavaScript engine, rename events.txt to events_unsafe.txt and, a few seconds later, will restart the whole HSYCO server engine to protect the integrity of the system.

An error message will be visible in the editor, before HSYCO restarts.



The EventsLoadTimeout parameter in the general system configuration parameters of Settings allows you to change the default 20 seconds timeout.

Predefined variables

Besides `__run`, there are a few other predefined variables that you can use at run-time: `triggerKey`, `triggerValue` and `sessionId`.

`triggerKey` and `triggerValue` contain the internal representation of the event's trigger keyword and value that caused the execution of the JavaScript action. These variables could be quite useful when you want to execute code that needs to know which specific event caused its execution, for example when you have a complex event expression, that could be triggered by different events.

We are not documenting the returned values for these variables, but you can easily find out with a single line of code:

```
... : { messageLog(triggerKey + "/" + triggerValue + ".") }
```

`sessionId` returns the internal representation of the Web client session on events like USER and PAGE. You can use the `sessionId` value directly to set session-specific user interface attributes.

The INIT JavaScript scope

JavaScript code that is associated to the INIT event keyword is different from code associated to any other event, as it is executed in a global scope that is shared with all other JavaScript segments. In more practical terms, if you need to define JavaScript functions or variables that will be used in other JavaScript actions, you should define them in one or more JavaScript actions directly associated to the INIT event, like in the following example:

```
INIT : {
       var counter = 0;

       function myLog(text) {
               messageLog(text);
       }
```

```
}

TIME : {
       if (__run == 1) {
               counter++;
               myLog("My Counter is: " + counter);
       }
}
```

Check the log file, and you will see something like:

```
2014.01.24 15:41:00.519 - My Counter is: 1
2014.01.24 15:42:00.497 - My Counter is: 2
2014.01.24 15:43:00.498 - My Counter is: 3
```

JavaScript Reserved Words

The following are keywords and may not be used as variables, functions, methods, or object identifiers, because ECMAScript specifies special behavior for them:

- break
- case
- catch
- continue
- debugger
- default
- delete
- do
- else
- finally
- for
- function
- if
- in
- instanceof
- new
- return
- switch
- this
- throw
- try
- typeof
- var
- void
- while
- with

Words reserved for possible future use

The following are reserved as future keywords by the ECMAScript specification. They have no special functionality at present, but they might at some future time, so they cannot be used as identifiers.

Although some of the following keywords may be used in the current release of the JavaScript engine embedded in HSYCO, we recommend to avoid using any of these to avoid issues with future releases of the HSYCO Server.

List of future keywords, defined in ECMAScript 1:

- const
- enum
- export
- extends
- import
- super

List of future keywords, defined in ECMAScript 2:

- abstract
- boolean
- byte
- char
- class
- const
- double
- enum
- export
- extends
- final
- float
- goto
- implements
- import
- int
- interface
- long
- native
- package
- private
- protected
- public
- short
- static
- super
- synchronized
- throw
- throws
- transient
- volatile

List of future keywords, defined in ECMAScript 3:

- abstract
- boolean
- byte
- char
- class
- const
- double
- enum

- export
- extends
- final
- float
- goto
- implements
- import
- int
- interface
- long
- native
- package
- private
- protected
- public
- short
- static
- super
- synchronized
- throws
- transient
- volatile

List of future keywords, defined in ECMAScript 5:

- class
- enum
- export
- extends
- implements
- import
- interface
- let
- package
- private
- protected
- public
- static
- super
- yield

Reserved word usage

In order to be able to call methods of Java classes having the same name of reserved keywords, you can use a notation that refers to the name as "IdentifierNames", avoiding conflict with reserved keywords. For example, to call the delete method of a Java File object, you should use the following notation:

```
var myfile = new java.io.File("logs/myfile.txt");
myfile['delete']();
```

JavaScript callbacks

You can also associate JavaScript actions directly to specific event callback functions. This method offers more control and flexibility over the basic event:action syntax. The JavaScript callbacks are equivalent to the Java callbacks, providing full access to all core events in pure JavaScript.

The JavaScript callback/action syntax in events.txt is:

```
function f(p) : { /* JavaScript Code here */ }
```

where f(p) is one of the callback functions with its parameters. The parameters' names in the callback declaration are available as variables in the JavaScript action code.

This syntax is quite similar to what a normal function declaration in JavaScript would be, just with the added colon character between the function's parameter's list and the function's code.

Just like ordinary functions, if the JavaScript action code is associated to a callback function and not to a basic event, you are allowed to use the return statement to return a value to the HSYCO event engine. Some callbacks will need return values to implement specific features. For example:

```
function WebRootRequestEvent(addr, secure, useragent) : {
       return "hsycoserver/manager";
}
```

This code will redirect root HTTP requests received by the HSYCO server to the manager page. In fact the WebRootRequestEvent() callback is called on every HTTP request received by the HTTP server, and takes an optional return value as the redirection URL.

Java Programming

You can write custom Java code to respond to any HSYCO event.

Java is precompiled, and offers the best performance and powerful programming features of a fully standard Java language.

The Java programming framework is very similar to the EVENTS language structure - a library of callback methods are asynchronously called on system and filed events. You are allowed to extend these methods with standard Java code and a comprehensive library of command and utility methods that already implement all common functions you may need to interact with field systems and with the user interface.

You are free to use any standard Java feature or external classes. This is a fully standard Java environment, not a sandboxed solution.

You are also going to use Java to extend the core HSYCO system with plug-in applications and custom I/O Servers, as well as to access other embedded HSYCO features, like the integrated SQL engine. These topics are discussed in the Advanced Programming chapter.

user.java

The basic Java extension of HSYCO is based on a class called user, declared in a user.java file, contained in the hsyco/com/hsyco directory and compiled in a java.class file.

💡 **The user.class file must exist for the HSYCO to work correctly, even if there is no customized Java code.**

HSYCO SERVER ships with an empty user.class file pre-installed.

The user class should extend the userBase class, a skeleton class that provides methods to execute actions on the field devices, as well as internal functions, and the callback framework methods, which are the methods called by HSYCO after any event is detected.

The dummy user.java file contains only the class declaration:

```
/*  HSYCO CONTROLLER, USER CODE
 *  (c) 2014 Home Systems Consulting SpA
 *  For information, see the Home Systems Consulting web site:
 *  http://www.homesystemsconsulting.com/
 */

package com.hsyco;
public class user extends userBase {
}
```

The HSYCO run-time environment should be restarted for user.class changes to become effective. The default setting of the AutoKillFiles parameter in hsyco.ini causes the automatic restart of HSYCO when user.class is changed.

The Java Callback Methods API and Java Command and Utility Methods API chapters describes the callback methods and all the command and utility methods provided by userBase.

These methods are the only official HSYCO Java API methods.

You should avoid using other public methods provided by other classes within the HSYCO package. These methods could change without notice in any new HSYCO release, causing compatibility issues between your code and different versions of HSYCO.

Predefined Constants

The table below lists the predefined constants that you should use as parameters or are returned by the HSYCO Java API.

Name	Description
MERGED	DMX merged channel
NOCHANGE	don’t change current status
OFF	off action and status
ON	on action and status
PAUSE	pause status (only Squeezebox music player)
PLAY	play status (only Squeezebox music player)
UNKNOWN	unknown status value

Java Callback Methods API

Callback methods are called by the HSYCO events management system.

By implementing these methods, you can associate your custom Java code to each event.

In some cases, the return values of these methods can influence the behavior of HSYCO.

Most of the methods, propagate exceptions to the top, that is to HSYCO internal code. In this case, HSYCO generates an error message in the log.

System Functions

DaylightEvent

```
static void DaylightEvent(boolean day)
```

Called at sunrise and sunset, according to the latitude and longitude values set in hsyco.ini, and the optional SunriseOffsetMinutes and SunsetOffsetMinutes parameters.

Parameters:

- day: boolean - true at sunrise, false at sunset.

getUserVersion

```
static String getUserVersion()
```

This method returns a string, used by the HSYCO run-time to generate the user code version information log line at startup.

It is recommended to return a meaningful string with an appropriate identification of your user.java code and its version.

Returns:

- String - user code version information.

haActiveEvent

```
static void haActiveEvent(boolean active)
```

Triggered by the change of state of an HSYCO server in a master/slave high availability configuration.

Parameters:

- active: boolean - true if the server is active, false if not active.

PowerEvent

```
static int PowerEvent(int power)
```

Triggered by a change of the current power load level, as set using the powerSet() Java API or the POWER action.

Thanks to this method, you are allowed to alter the power value shown in the Web interface, for example to aggregate power readings acquired from other sensors.

Parameters:

- power: int - the power level, in Watts.

Returns:

- int - if PowerEvent() returns -1 or doesn't return a value, HSYCO status is updated with the detected power value, the value returned by PowerEvent() is otherwise used.

programTimerEvent

```
static void programTimerEvent(String name)
```

Called when a program timer is activated.

Program timers are set using programTimerSet(), programTimerReset(), programTimerRepeat(), or using the corresponding actions in EVENTS.

Parameters:

- name: String - program timer name.

SchedulerEvent

```
static void SchedulerEvent(String groupname, String schedulename)
```

This callback blocking method allows to call user methods at configurable intervals.

You define schedules using a group name and a schedule name.

Schedules under the same group run in the same thread and are executed sequentially, based on their interval in milliseconds. Schedules in different groups run in parallel.

Parameters:

- groupname: String - the scheduler’s group name
- schedulename: String - the scheduler’s name.

StartupEvent

```
static void StartupEvent()
```

Called only once when HSYCO starts, after the initialization and HTTP server start-up, but before the execution of the different field interface threads. It is not executed in a thread, it is therefore a blocking method, that must complete its

execution before all other HSYCO services can be started.

SunPositionEvent

```
static void SunPositionEvent(int azimuth, int elevation)
```

Called when the Sun changes its height with respect to the horizon or its angle from true north.

Parameters:

- azimuth: int - the current Sun angle from true north, in decimal degrees
- elevation: int - the current Sun elevation in decimal degrees from the horizon. Elevation is negative at night.

TimeEvent

```
static void TimeEvent(long time)
```

Called every 60 seconds.

HSYCO tries to synchronize the execution at the beginning of the minute, executing this method at seconds 0 of every minute.

💡 **However, the execution is not guaranteed. In case of heavy load HSYCO might not be able to execute the call with accurate timing. In extreme cases, it could occasionally skip some calls.**

Parameters:

- time: long - the current time in milliseconds.

varEvent

```
static void varEvent(String name, String value)
```

Triggered by the change of value of a program variable.

Parameters:

- name: String - the variable name. Names are case-insensitive
- value: String - the value to be assigned to the program variable.

Cameras

CameraCommandEvent

```
static int cameraCommandEvent(String function, String action, String camera)
```

Triggered by a camera control input from the Web interface.

It can be used to execute arbitrary actions when the user clicks on the active areas of the camera view, or to replace some or all of the standard PTZ commands with custom actions.

Parameters:

- function: String - see the table below
- action: String - see the table below
- camera: String - the camera name.

Returns:

- int - if CameraCommandEvent() returns -1 or doesn't return a value, the standard PTZ command associated to the PTZ driver is skipped, otherwise it is executed normally.

icon	function	action
	focus	far
	focus	near
	zoom	tele
	zoom	wide
	move	left
	move	right
	move	up
	move	down
	move	stop

CameraMotionEvent

```
static void CameraMotionEvent(String eventName, long remoteTime)
```

Called when HSYCO starts recording video from a camera.

Recording starts when a camera notifies a recording request with the recording HTTP API call (the zone=id portion is optional):

```
http://192.168.0.50/x/camerarec?camera=name&zone=id
```

Recording can also be triggered by the CameraRecTrigger() Java method or JavaScript function, or the CAMERAREC or CAMERARECFULL actions in EVENTS.

Parameters:

- eventName: String - the event identification string, either the camera name only or name:id if recording is triggered by a camerarec HTTP call with the optional zone=id parameter
- remoteTime: long - the current time in milliseconds.

CameraViewEvent

```
static void CameraViewEvent(String camera, boolean active)
```

CameraViewEvent() is called on changes of the live view status of cameras.

When at least one web client is showing the live feed from a camera, the camera is marked as active.

When no live feed request is received for a several consecutive seconds, the camera is marked as not active.

Parameters:

- camera: String - the camera name
- active: boolean - true if active, false if not.

DMX

DmxEvent

```
static void DmxEvent(int channel, int state)
```

Called after changes to DMX-512 channels.

Parameters:

- channel: int - DMX channel address, from 1 to 512 which might be preceded by the DMX gateway number, starting from 0
- state: int - the new channel value, from 0 to 255.

DmxFilter

```
static int DmxFilter(int channel, int state, boolean reverse)
```

This method allows to filter the value of the status of each channel on the DMX-512 buses, sending to the gateway different values from those set in HSYCO.

This way, it is possible to apply chromatic corrections, or any on the fly translation of channels values.

💡 **This is a blocking method and is called when sending commands to the DMX gateway.**

It is also called after reading current channels values from the gateway. In this case the conversion function should be symmetric and return a complementary value.

Parameters:

- channel: int - the DMX channel address, from 1 to 512 which might be preceded by the DMX gateway number, starting from 0
- state: int - the unfiltered value of the channel, from 0 to 255
- reverse: boolean - false if called when writing to the gateway; true if called when reading from the gateway.

Returns:

- int - the channel value that will be sent to the DMX gateway.

💡 **The JavaScript DmxFilter function is called only when the Java DmxFilter callback is not defined or returned the original value.**

DmxStartupEvent

```
static void DmxStartupEvent(int serverIndex)
```

Called when starting the monitor threads for each DMX bus, once per bus at the start of the execution of HSYCO, and also after every restart of the monitor thread, for example after communication errors.

It is safe, inside this method, to execute command methods for the same DMX gateway.

💡 **This is a blocking method. The DMX driver will start after this function returns.**

Parameters:

- serverIndex: int - number of the DMX bus, starting from 0.

Infrared Control

IREvent

```
static void IREvent(boolean received, int irtransid, String event)
```

This method is called when an IRTrans receives or sends an IR command from its local IR database.

💡 **IRTrans can only receive and decode infrared commands that have been previously stored in its internal FLASH memory database.**

Parameters:

- received: boolean - true if the command has been received by the IRTrans, false if sent
- irtransid: int - IRTrans number, starting from 0 for the first device, based on the listing order of the IRTrans parameter in hsyco.ini
- event: String - the string of the command received or issued, formatted as: "remote_name,command".

I/O Servers

IOEvent

```
static void IOEvent(String id, String value)
```

Triggered by the value change of any data point of an I/O server.

Parameters:

- id: String - the server id and the name of the data point. According to the type of server, the format changes but it generally appears as server name, as declared the ioServers parameter in hsyco.ini, followed by a specific name representing the data point within the system
- value: String - the new value. For binary inputs or outputs, the returned values are “0” or “1”.

IOStartupEvent

```
static void IOStartupEvent(int serverIndex)
```

Triggered by the start of the communication thread of each I/O server - once for each server at the beginning of the execution of HSYCO, and also after every reboot of the communication thread, for example after communication errors.

This is a blocking method. The I/O server driver will start after this function returns.

Parameters:

- serverIndex: int - the I/O server number, starting from 0, based on the listing order of the ioServers parameter in hsyco.ini.

Modbus

For detailed information about the Modbus protocol, and the response PDU formats, you should refer to the MODBUS Application Protocol Specification V1.1b documentation at:
http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

ModbusEvent

```
static byte[] ModbusEvent(String name, InetAddress addr, int unitid, byte[] pdu)
```

Called by the Modbus Server I/O Server on every Modbus request received from a Modbus client.

The returned byte array is used as the Modbus response PDU (function code and data).

Parameters:

- name: String - the name of the Modbus Server I/O server
- addr: InetAddress - IP address of the Modbus client
- unitid: int - the unit id set in the received Modbus request
- pdu: byte[] - the request PDU data, in hexadecimal string format.

Returns:

- byte[] - the response PDU data.

Network Services

httpCallEvent

```
static String httpCallEvent(String address, boolean secure, String query)
```

Called when the HSYCO HTTP server receives a GET request with the following path:

```
/x/httpcall?query
```

💡 **The HTTP client's IP address must be in the trusted network range, or the httpcall request will be rejected.**

This method returns a string, that will be sent as the text/plain answer to the HTTP client (with HTTP response code 200).

HSYCO will return a 404 Not Found response code if all of the following conditions are true:

- no HTTP event matches the query in events.txt
- the httpCallEvent JavaScript callback is not defined in events.txt, or doesn't return a value
- the httpCallEvent Java callback is not defined in user.java or in any plugins, or returns null.

If the httpCallEvent is defined both in JavaScript and Java, the Java version return value (unless null) will be used in the HTTP response.

Parameters:

- address: String - the IP address of the HTTP client
- secure: boolean - true if the request is sent to the HTTPS server, false if the request is sent to the HTTP server
- query: String - the string following the ? in the URL path.

Returns:

- String - the string to send as the text/plain answer to the HTTP client.

LocationEvent

```
static void LocationEvent(long mac, InetAddress ip, int zoneId)
```

Called, if the location service is active, when a variation in the association of a client to the Wi-Fi Access Points is detected.

Parameters:

- mac: long - the MAC address of the client
- ip: InetAddress - the IP address of the client, if available, or null (the IP address could be null if the associated device is not connected to the HSYCO Web interface)
- zoneId: int - in case of association, it is the Access Point sequence number, according to the order defined in the LocationBases parameter in hsyco.ini; -1 if the client was de-associated from the Wi-Fi network.

PBX

PBXCallEvent

```
static boolean PBXCallEvent(String host, String caller, String called)
```

Called when a call notification is sent to HSYCO by the PBX system.

If the method returns true, the request is recorded on the log file with the [OK] status, with [ERROR] otherwise.

The PBX should generate HTTP requests to the HSYCO Web Server, using the following formats:

```
http://192.168.0.50/x/vcall/<from>/<to>
```

or:

```
http://192.168.0.50/x/vcall?from=<from>&to=<to>
```

For example, the HTTP request:

```
http://192.168.0.50/x/vcall?from=21&to=25
```

is interpreted as a call notification from internal number 21 to number 25.

💡 **The HTTP client's IP address must be in the trusted network range, or the httpcall request will be rejected.**

Parameters:

- host: String - the IP address of the PBX server
- caller: String - the caller number
- called: String - the called number.

Returns:

- boolean - return true to generate a log entry with [OK] status, false for [ERROR] status.

Squeezebox

SlimPowerEvent

```
static void SlimPowerEvent(int index, int power)
```

Called when a Squeezebox player is turned on or off.

Parameters:

- index: int - player number, starting from 0 for the first player, based on the listing order of the slimPlayers parameter in hsyco.ini
- power: int - 0: OFF, 1: ON.

SlimStatusEvent

```
static void SlimStatusEvent(int index, int status)
```

Called when a Squeezebox player changes its playback mode.

Parameters:

- index: int - player number, starting from 0 for the first player, based on the listing order of the slimPlayers parameter in hsyco.ini
- status: int - 0: OFF, 1: ON, 2: PAUSE, 3: PLAY, -1: UNKNOWN.

SlimVolumeEvent

```
static void SlimVolumeEvent(int index, int volume)
```

Called when a Squeezebox player changes its volume level.

Parameters:

- index: int - player number, starting from 0 for the first player, based on the listing order of the slimPlayers parameter in hsyco.ini
- volume: int - an integer number representing the volume change (not the absolute audio level) in a scale from 0 to 100; positive numbers correspond to a volume increase, negative numbers to a decrease.

Timers and Schedulers

UserTimerEvent

```
static boolean UserTimerEvent(String name, boolean active)
```

Called when a user timer or scheduler's rule should be activated, and before executing the associated action in EVENTS.

It is also executed at the timer or scheduler's rule deactivation.

This method must return a true value in order to make the timer or rule actually active and the associated actions executed.

Whenever the method returns false, the timer will not be activated, and the method will be called again once a minute for the scheduled duration of the timer.

In the same way, if the method returns false when the timer or rule is deactivated, the timer or rule will not be deactivated, extending the timer activation beyond the scheduled duration.

This is a blocking method.

Parameters:

- name: String - timer name or scheduler's rule name
- active: boolean - true when activating and false when deactivating the timer or scheduler's rule.

Returns:

- boolean - true to confirm the timer event, false to block the event.

💡 **The JavaScript UserTimerEvent function is alway called, but its return value is used only when the Java UserTimerEvent callback is not defined or returned true.**

User Interface

pageEvent

```
static void pageEvent(String address, String session, String userid, String project, String page)
```

Called when a Web client initially loads a project and when navigating between pages.

There is no guarantee that the event would be fired the exact moment the page is loaded.

Network delays or errors could cause delayed triggering of this event.

Parameters:

- address: String - the client's IP address string in textual presentation
- session: String - session id string that uniquely identifies the client session
- userid: String - the user id
- project: String - project's id
- page: String - the page id of the current page.

uiClearEvent

```
static void uiClearEvent(String session)
```

Called, following an explicit call of uiClear() or after the client session inactivity timeout expires, if the client session becomes active again (i.e. the client starts handshaking again with the server with the same session id).

Parameters:

- session: String - session id string that uniquely identifies the client session.

userCommand

```
static String userCommand(String name, String param)
```

Called by user clicks on buttons created in the Web interface with the (user), (usermini), (usermicro), (userrgb) or (userimage) objects, by a USER action in EVENTS, or by the user() Java method and JavaScript function.

This method returns a string. Any returned string causes the log of an [OK] status. If you want to prevent logging, for example to avoid sensitive information to be written in the log file, you should return the string "!".

If you want to force the Web interface to show a specific page after the button is pressed (it would be like pressing a (link) object), return a string starting with "page:" followed by the page name; in this case, userCommand() will be called again when that popup or page is closed, with "/close" appended to param.

Parameters:

- name: String - the name field of the (user) object
- param: String - the param of the (user) object.

Returns:

- String - log or go to page, see text above. If this function returns null, an [ERROR] status is logged in the log file, while any other string causes the log of an [OK] status.

userCommand - with session information

```
static String userCommand(String session, String userid, String name, String param)
```

Same as userCommand(String name, String param), but with additional session and user id information passed as parameters. You should only use one of the two userCommand() callback methods in user.java. If you declare both, only this four parameters version will be called.

Parameters:

- session: string - a session id string that uniquely identifies the client session
- userid: string - the user id
- name: String - the name field of the (user) object
- param: String - the param of the (user) object.

Returns:

- String - log or go to page, see text above. If this function doesn't return a value, an [ERROR] status is logged in the log file, while any other string causes the log of an [OK] status.

userSubmit

```
static String userSubmit(String session, String uid, String name, HashMap<String, String> fields)
```

This callback is similar to userCommand(String session, String uid, String name, String param) and should be used only when the four parameters version of userCommand() is also defined in user.java.

When userSubmit() is defined, it will be called when a (submit!id), (submitmini!id) or (submitmicro!id) button is pressed, while userCommand() will continue to be called on (user) buttons.

userSubmit() provides a more convenient access to the values of multiple input fields in a form.

This function can optionally return a string. Any returned string causes the log of an [OK] status. If you want to prevent logging, for example to avoid sensitive information to be written in the log file, you should return the string “!”.

If you want to force the Web interface to show a specific page after the button is pressed (it would be like pressing a (link) object), return a string starting with “page:” followed by the page name; in this case, userCommand() will be called when that popup or page is closed, with "/close" appended to param.

Parameters:

- session: String - a session id string that uniquely identifies the client session
- userid: String - the user id
- name: String - the id of the (submit!id) object

- fields: HashMap<String, String> - the hash map of all input fields in the container where the (submit) object is located.

Returns:

- String - log or go to page, see text above. If this function doesn't return a value, an [ERROR] status is logged in the log file, while any other string causes the log of an [OK] status.

WebRootRequestEvent

```
static String WebRootRequestEvent(InetAddress addr, boolean secure, String useragent)
```

This method is called when the Web server receives a root URL request. It can return a string to redirect the browser to a valid URL, or null to prevent redirection.

Parameters:

- addr: InetAddress - the HTTP client's address
- secure: boolean - true if this is an HTTPS request
- useragent: String - the Web browser's user agent information.

Returns:

- String - optional return string containing a valid absolute or relative URL

Java Command and Utility Methods API

These methods are used to control field devices, to acquire status information and to execute general utility functions.

If the userLog option is enabled in the system general settings, commands that execute relevant functions will be logged in the daily log file:

```
2011.07.30 11:39:16.133 – USERCODE: method – parameter
```

System Functions

dateSet

```
static boolean dateSet(int year, int month, int day, int hour, int minute, int second)
```

Sets the HSYCO Server operating system's clock and hardware clock, in the local time zone.

Parameters:

- year: int - the year, greater than 1900
- month: int - the month, between 1-12
- day: int - the day, between 1-31
- hour: int - the hour, between 0-23
- minute: int - the minute, between 0-59
- second: int - the second, between 0-59.

Returns:

- boolean - true if successful, false if not.

getNextSunrise

```
static long getNextSunrise(long now, boolean withoffset)
```

Computes the time of the next sunrise after the time set in the now parameter.

The sunrise time is returned in milliseconds since midnight, January 1, 1970 UTC.

Parameters:

- now: long - the reference time in milliseconds
- withoffset: boolean - true to take the sunrise offset into account, as defined with the SunriseOffsetMinutes in hsyco.ini, false to ignore the offset and return the actual sunrise time (Based on civil dawn, begins when the geometric center of the sun is 6° below the horizon).

Returns:

- long - the next sunrise time in milliseconds since midnight, January 1, 1970 UTC.

getNextSunset

```
static long getNextSunset(long now, boolean withoffset)
```

Computes the time of the next sunset after the time set in the now parameter.

The sunset time is returned in milliseconds since midnight, January 1, 1970 UTC.

Parameters:

- now: long - the reference time in milliseconds
- withoffset: boolean - true to take the sunset offset into account, as defined with the SunriseOffsetMinutes in hsyco.ini, false to ignore the offset and return the actual sunrise time (Based on civil dusk, ends when the geometric center of the sun reaches 6° below the horizon).

Returns:

- long - the next sunset time in milliseconds since midnight, January 1, 1970 UTC.

haActiveState

```
static boolean haActiveState()
```

When HSYCO is installed in a high availability configuration, this method returns true if the system (master or slave) is active, false if it is not active.

If high availability is not configured, the method returns true.

Returns:

- boolean - high availability current state.

haActive

```
static void haActive(boolean active)
```

Force a master HSYCO server to become inactive, or return to the active state.

This method should be called on the master only, and has no effect on the slave server.

Parameters:

- active: boolean - set to true to force a master to become inactive, or false to return it to the active state after the master was forced to become inactive.

isDaylight

```
static boolean isDaylight()
```

Returns the day or night status at present time.

The result is based on the latitude and longitude data set in hsyco.ini and the SunriseOffsetMinutes and SunsetOffsetMinutes parameters.

Returns:

- boolean - true during the day, false at night.

powerSet

```
static void powerSet(int power)
```

Sets the electric power state variable.

The power value is shown in the navigation or status bar of the Web interface.

Besides setting the power value, this method also triggers the execution of the powerEvent() callback and the POWER event in EVENTS.

Parameters:

- power: int - the power in Watt units.

programTimerClear

```
static void programTimerClear(String name)
```

Cancels a program timer.

Parameters:

- name: String - the program timer name.

programTimerRepeat

```
static void programTimerRepeat(String name, int seconds)
```

Sets a program timer that is executed repeatedly with an interval equal to the specified number of seconds.

If a program timer with the same name has already been set, the number of seconds to the timer's execution is reset to the new value.

Parameters:

- name: String - the program timer name
- seconds: int - the interval in seconds of the timer's execution.

programTimerReset

```
static void programTimerReset(String name, int seconds)
```

Sets a program timer to be executed after the specified number of seconds.

If a program timer with the same name has already been set, the number of seconds to the timer's execution is reset to the new value.

Parameters:

- name: String - the program timer name
- seconds: int - the number of seconds to the timer's execution.

programTimerSet

```
static void programTimerSet(String name, int seconds)
```

Sets a program timer to be executed after the specified number of seconds. If a program timer with the same name has already been set, the action will be ignored.

Parameters:

- name: String - the program timer name
- seconds: int - the number of seconds to the timer's execution.

schedulerRegister

```
static void schedulerRegister(String groupname, String schedulename, int interval)
```

This function registers a new scheduled callback.

Parameters:

- groupname: String - the group name
- schedulename: String - the schedule name
- interval: int - interval in milliseconds.

schedulerRemove

```
static void SchedulerRemove(String groupname, String schedulename)
```

Deletes a schedule.

Parameters:

- groupname: String - the group name
- schedulename: String - the schedule name.

sleep

```
static void sleep(long millis)
```

Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.

Parameters:

- millis: long - the length of time to sleep in milliseconds.

varGet

```
static String varGet(String name)
```

Returns the value of a program variable, or null if no variable with the given name has been defined.

Program variables can be set using varSet() in user.java or in the EVENTS programming environment.

To access a variable used in EVENTS, the name used in varGet() should start with the $ character.

Variables names ending with ! are considered persistent and their values are preserved when HSYCO restarts; normal variables are deleted when HSYCO starts.

Parameters:

- name: String - the variable name. Names are case-insensitive.

Returns:

- String - the current value of the program variable.

varSet

```
static void varSet(String name, String value)
```

Sets a program variable to the value parameter.

Variables set with varSet() can be read using varGet() and are also available in the EVENTS programming environment, if the variable name starts with $.

If you need to use variables in Java or JavaScript but don't want these variables to be available in EVENTS, just use names that don't start with the $ character.

Variables names ending with ! are considered persistent and their values are preserved when HSYCO restarts; normal variables are deleted when HSYCO starts.

Parameters:

- name: String - the variable name. Names are case-insensitive
- value: String - the value to be assigned to the program variable.

Cameras

cameraCommand

```
static int cameraCommand(String function, String action, String camera)
```

Sends PTZ and focus commands to cameras.

Parameters:

- function: String - see the table below
- action: String - see the table below
- camera: String - the camera name.

icon	function	action
	focus	far
	focus	near
	zoom	tele
	zoom	wide
	move	left
	move	right
	move	up
	move	down
	move	stop

Returns:

- int - the returned value is included in the log message, and has no other purpose.

cameraMode

```
static void cameraMode(String camera, boolean enabled)
```

Disables or enables a camera.

When the live view is disabled, recording is also disabled.

Parameters:

- camera: String - the camera name

- enabled: boolean - true to enable the camera, false to disable.

cameraRecMode

```
static void cameraRecMode(String camera, boolean enabled)
```

Disables or enables recording from a camera, not affecting live view.

Parameters:

- camera: String - the camera name
- enabled: boolean - true to enable the recording, false to disable.

cameraRecTrigger

```
static void cameraRecTrigger(String camera, String source, int seconds)
```

Records video from a camera.

Parameters:

- camera: String - the camera name
- source: String - a string used to generate a log message, to identify the source that triggered the recording
- seconds: int - recording time in seconds. If set on 0, stops the current recording.

cameraRecTriggerFull

```
static void cameraRecTriggerFull(String camera, String source, int seconds)
```

Records video from a camera, like cameraRecTrigger(), but ignores the DroppedFrames parameter in hsyco.ini, so that all frames captured from the camera during the recording period are recorded with no skips.

Parameters:

- camera: String - the camera name
- source: String - a string used to generate a log message, to identify the source that triggered the recording
- seconds: int - recording time in seconds. If set on 0, stops the current recording.

Data Logger

dataLoggerClear

```
static boolean dataLoggerClear(String name)
```

Permanently deletes all historical data of the specified data logger.

Parameters:

- name: String - data logger name.

Returns:

- boolean - true if successful, false if not.

dataLoggerOptions

```
static boolean dataLoggerOptions(String name, String param, String value)
```

Sets a specific option for the specified data logger.

So far the only available option is “origin” which sets the x-bar of all the charts linked to the data logger to the passed value.

Parameters:

- name: String - data logger name
- param: String - option to be set (“origin” is the only option you can currently set with this method)
- value: String - value for the specified option.

Returns:

- boolean - true if successful, false if not.

dataLoggerSave

```
static boolean dataLoggerSave(String type, String[] names, String path, boolean timestamp)
```

Creates or updates a CSV file containing the data processed by the specified data loggers.

The possible types are:

- log: append the last acquired value by the data logger to the specified file
- stat: updates the specified file with the currently gathered data by the specified data logger.

Parameters:

- type: String - log type (log or stat)
- names: String[] - data logger names
- path: String - path of the file to be created
- timestamp: boolean - if true and the type is log adds a timestamp column in the CSV file for the logged values.

Returns:

- boolean - true if successful, false if not.

dataLoggerUpdate

```
static boolean dataLoggerUpdate(String name, Double value)
```

Supplies the specified data logger with a new value to be processed.

Parameters:

- name: String - data logger name
- value: Double - value to be processed. Setting value to null, with counter data loggers only, resets the internal delta baseline.

Returns:

- boolean - true if successful, false if not.

DMX

dmxGet

```
static int dmxGet(int channel)
```

Returns the current status of a channel on the DMX-512 bus.

Parameters:

- channel: int - when using one DMX gateway, address will be a standard DMX-512 address between 1 and 512. If there is more than one gateway, 1000 must be added to the DMX address for the second gateway, 2000 for the third one and so on.

Returns:

- int - channel current status, between 0 and 255, or UNKNOWN (-1) if the status is unknown.

dmxOff

```
static int dmxOff(int channel)
```

Sets a channel on the DMX-512 bus to 0. It also saves the previous value, so it can be restored with dmxOn().

Parameters:

- channel: int - when using one DMX gateway, address will be a standard DMX-512 address between 1 and 512. If there is more than one gateway, 1000 must be added to the DMX address for the second gateway, 2000 for the third one and so on.

Returns:

- int - 1 if successful, 0 if not.

dmxOff - multi channel

```
static int dmxOff(int[] channels)
```

Sets a list of DMX-512 channels to 0, also saving the previous values, so they can be restored with dmxOn().

When changing status to multiple channels, this method is more efficient than the single channel command, and updates all the channels at the same time.

💡 **When this method is executed, the variation of even only one of the channels listed in the array causes the execution of the callback methods associated to the DMX events for all the channels listed in the array, even those which are not modified by this call.**

Parameters:

- channels: int[] - an array that contains the list of channels. When using one DMX gateway, the channel address will be a standard DMX-512 address between 1 and 512.

Returns:

- int - 1 if successful, 0 if not.

dmxOn

```
static int dmxOn(int channel)
```

Sets a DMX-512 channel to the value preceding the last call to dmxOff().

Parameters:

- channel: int - when using one DMX gateway, address will be a standard DMX-512 address between 1 and 512. If there is more than one gateway, 1000 must be added to the DMX address for the second gateway, 2000 for the third one and so on.

Returns:

- int - 1 if successful, 0 if not.

dmxOn - multi channel

```
static int dmxOn(int[] channels)
```

Sets a list of DMX-512 channels to the values preceding the last call to dmxOff() for each channel.

When changing status to multiple channels, this method is more efficient than the single channel command, and updates all the channels at the same time.

💡 **When this method is executed, the variation of even only one of the channels listed in the array causes the execution of the callback methods associated to the DMX events for all the channels listed in the array, even those which are not modified by this call.**

Parameters:

- channels: int[] - an array that contains the list of channels. When using one DMX gateway, the channel address will be a standard DMX-512 address between 1 and 512.

Returns:

- int - 1 if successful, 0 if not.

dmxMerge - range

```
static int dmxMerge(int from, int to, boolean merge)
```

Sets all DMX-512 channels between the parameters from and to to merge mode or normal mode. In merge mode, the channels on the DMX OUT bus of the gateway follow the values of the channels of the DMX IN bus.

Parameters:

- from: int - first address of the range. When using one DMX gateway, address will be a standard DMX-512 address between 1 and 512. If there is more than one gateway, 1000 must be added to the DMX address for the second gateway, 2000 for the third one and so on
- to: int - last address of the range
- merge: boolean - true to enable merge mode, false to return to normal mode.

Returns:

- int - 1 if successful, -1 if to < from, 0 for other errors.

dmxMerge - multi channel

```
static int dmxMerge(int[] channels, boolean merge)
```

Sets a list of DMX-512 channels to merge mode or normal mode. In merge mode, the channels on the DMX OUT bus of the gateway follow the values of the channels of the DMX IN bus.

Parameters:

- channels: int[] - an array that contains the list of channels. When using one DMX gateway, the channel address will be a standard DMX-512 address between 1 and 512.
- merge: boolean - true to enable merge mode, false to return to normal mode.

Returns:

- int - 1 if successful, -1 if to < from, 0 for other errors.

dmxSet

```
static int dmxSet(int channel, int state)
```

Sets the value of a single DMX-512 channel.

Parameters:

- channel: int - when using one DMX gateway, address will be a standard DMX-512 address between 1 and 512. If there is more than one gateway, 1000 must be added to the DMX address for the second gateway, 2000 for the third one and so on

state: int - the new value of the channel, from 0 to 255.

Returns:

- int - 1 if successful, 0 if not.

dmxSet - multi channel

```
static int dmxSet(int[] channels, int[] state)
```

Sets a list of DMX-512 channels.

When changing status to multiple channels, this method is more efficient than the single channel command, and updates all the channels at the same time.

💡 **When this method is executed, the variation of even only one of the channels listed in the array causes the execution of the callback methods associated to the DMX events for all the channels listed in the array, even those which are not modified by this call.**

Parameters:

- channels: int[] - an array that contains the list of channels. When using one DMX gateway, the channel address will be a standard DMX-512 address between 1 and 512.
- states: int[] - an array that contains the list of values for each channel, from 0 to 255.

Returns:

- int - 1 if successful, 0 if not.

Infrared Control

irtransCommand

```
static int irtransCommand(String irid, String command)
```

Sends a command to an IRTrans.

An infrared command is composed by the name that identifies the commands database, followed by a comma and the name of the command in the database.

More commands in a sequence can be separated by “;”.

It is possible to insert the special command @N, which introduces a wait time of N milliseconds before sending the next command.

This command:

```
denon1036,on;@4000;denon1036,tv
```

sends the “on” command of the “denon-1036” remote, then waits 4 seconds before sending the “tv” command.

Moreover, the special command @0 allows to send IR commands without wait times before sending the next command.

When the @N delay before an IR command is not specified, there will be a default 500ms pause between commands; with @0 this pause will be reduced to 1ms.

For example:

```
denon,on;@0;nad,on
```

sends the “on” command of the “denon” remote, immediately followed by the “on” command of the “nad” remote.

💡 **Every IRTrans can contain a database of several remote controls, and each database stores the list of the distinct IR commands. If the hsyco/ir directory contains a .ccfhex file corresponding to the database name, then the CCF string in the file is used instead of sending the command stored in the IRTrans internal database.**

Parameters:

- irid: String - identifies the IRTrans, as defined with the IRTrans parameter in hsyco.ini
- command: String - the sequence of IR commands to be sent.

Returns:

- int - 1 if successful, 0 if not.

I/O Servers

ioGet

```
static String ioGet(String id)
```

Retrieves the current status of an I/O data point.

Parameters:

- id: String - the full data point name, including the I/O server id prefix.

Returns:

- String - the data point value.

ioServersRestart

```
static int ioServersRestart(String name)
```

Restarts an I/O server that was previously stopped.

This command has no effect on a running I/O server.

💡 **Note that this command will return before the I/O server is actually restarted.**

Parameters:

- name: String - the I/O server name.

Returns:

- int - 1 if successful, -1 if the server name is not defined.

ioServersStop

```
static int ioServersStop(String name)
```

Stops the execution of an I/O server.

💡 **Note that this command will return before the I/O server is actually stopped.**

Parameters:

- name: String - the I/O server name.

Returns:

- int - 1 if successful, -1 if the server name is not defined.

ioSet

```
static void ioSet(String id, String value)
```

Writes to I/O servers variables or data points.

Parameters:

- id: String - the full data point name, including the I/O server id prefix
- value: String - the new value.

Leak Detector

The Intelligent Leak Detector is used to generate warning for potential water or other quantities leaks by analyzing any generic flow counter.

The detector should be called at regular intervals, possibly faster than the minimum time needed for the counter to increment its value by one.

It will generate a leak warning when the measured flow remains relatively constant over a certain amount of time.

💡 **Use this function at your own risk!**

The Intelligent Leak Detector uses a correlation algorithm to distinguish between a constant flow and an irregular flow. A

relatively constant flow over a certain amount of time is considered as a potential leak and generates a leak event.

In some conditions, also depending on the type of flow counter used, this algorithm could fail from properly and timely recognizing a real leak, or it could generate false leak warnings.

💡 **Use different names to implement multiple independent leak detectors.**

leakDetectorClear

```
static void leakDetectorClear(String name)
```

Clears the detector's internal state.

In normal applications, there is no need to call this method, as the detector automatically clears its state from old data.

Parameters:

- name: String - leak detector name.

leakDetectorOptions

```
static void leakDetectorOptions(String name, String param, String value)
```

The detector's logic uses two parameters to set its sensitivity and time base. You can change the defaults to adapt to your specific conditions.

The "period" parameter changes the time base, expressed in seconds. The default is 1200 seconds. The detector generates a warning if the flow deviation constantly remains below the deviation threshold for the time period.

The "deviation" parameter changes the deviation percentage, and should be a number between 1 and 100. The default is 50. Set the period or deviation to 0 to temporarily disable the detector.

💡 **Small deviation values cause the algorithm to activate leak warnings only when the flow is very constant in time, becoming less susceptible to false alarms, but a value that is too low could miss actual leaks. Larger values could easily generate false alarms.**

Both period and deviation parameters can be changed anytime, even when the detector is already processing data.

Parameters:

- name: String - leak detector name
- param: String - set to "period" or "deviation"
- value: String - the period or deviation value.

leakDetectorUpdate

```
static int leakDetectorUpdate(String name, double value)
```

Calls the leak detector passing the current liters counter value.

Parameters:

- name: String - leak detector name

- value: double - the flow counter value.

Returns:

- int - -1 if the detector status is not changed, 0 if the status has changed to a reset condition, and 1 if the status has changed to a leak condition.

Log

errorLog

```
static void errorLog(String message)
```

Generates an error message in the daily log file.

Parameters:

- message: String - the text of the error message.

fileLog

```
static void fileLog(String pathname, String message)
```

Appends a generic text message at the end of a file.

Parameters:

- pathname: String - the file name. You can specify any path, starting from HSYCO's base directory. If the file or parent directories don't exist, they will be created automatically
- message: String - the message that will be appended at the last line of the text file.

messageLog

```
static void messageLog(String message)
```

Generates an information message in the daily log file.

Parameters:

- message: String - the text of the error message.

Mail

sendMail - simple text message

```
static int sendMail(String to, String from, String subject, String body)
```

Sends a multipart email message, mixing text and multiple images from cameras, live or recorded.

HSYCO SERVER sends the mail either directly to the recipient's domain mail server if the SmtpName parameter is not defined in hsyco.ini, or using a specific email account with user authentication and traffic encryption if the SMTP server and account parameters are set.

If the email message is sent directly to the recipient's domain mail server, you need to ensure that the mail server accepts mail to the destination address being sent with the from address and the public IP of the HSYCO SERVER.

💡 **HSYCO does not automatically retry sending the message if the destination mail server is not available when the send mail function is called.**

Parameters:

- to: String - a valid email address of the intended recipient. You can optionally specify the destination SMTP server name or IP address by appending :<server name or address to the recipient's email address, for example: john@example.com:192.168.1.1
- from: String - a valid email address to be used as the sender address
- subject: String - the message subject
- body: String - the message text.

Returns:

- int - 1: message sent; 0: message not sent; -1: other errors, including address errors.

sendMail - multi-part message

```
static int sendMail(String to, String from, String subject, Vector<String> body)
```

Sends a multipart email message, mixing text and multiple images from cameras, live or recorded.

HSYCO SERVER sends the mail either directly to the recipient's domain mail server if the SmtpName parameter is not defined in hsyco.ini, or using a specific email account with user authentication and traffic encryption if the SMTP server and account parameters are set.

If the email message is sent directly to the recipient's domain mail server, you need to ensure that the mail server accepts mail to the destination address being sent with the from address and the public IP of the HSYCO SERVER.

💡 **HSYCO does not automatically retry sending the message if the destination mail server is not available when the send mail function is called.**

Parameters:

- to: String - a valid email address of the intended recipient. You can optionally specify the destination SMTP server name or IP address by appending :<server name or address to the recipient's email address, for example: john@example.com:192.168.1.1
- from: String - a valid email address to be used as the sender address
- subject: String - the message subject
- body: Vector<String> - the message body. To send an ordinary text, just fill the vector with the desired text strings. To send an image, add a string to the vector with the following format: "cam:<cameraname>[:<seconds_back>]". For example, "cam:door" sends a live frame from the camera called "door"; "cam:door:2" sends a frame that was recorded two seconds before the last recorded frame; "cam:door:0" sends the last recorded frame. To send files as attachments, use the following format: "file:<file name>". The file path is relative to the HSYCO root directory.

Returns:

- int - 1: message sent; 0: message not sent; -1: other errors, including address errors.

Modbus

For detailed information about the Modbus protocol, and the response PDU formats, you should refer to the MODBUS Application Protocol Specification V1.1b documentation at:
http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

modbusReadCoils

```
static byte[] modbusReadCoils(String name, int unit, int address, int quantity)
```

Reads the content of a contiguous block of coils in a MODBUS device, using function code **0x01**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x01, the second to the byte count of the data part, and the data starts from the third byte.

The coils in the response message are packed as one coil per bit of the data field. Status is indicated as 1= ON and 0= OFF. The LSB of the first data byte contains the output addressed in the query. The other coils follow toward the high order end of this byte, and from low order to high order in subsequent bytes. If the returned output quantity is not a multiple of eight, the remaining bits in the final data byte will be padded with zeros (toward the high order end of the byte). The byte count field in the response PDU specifies the quantity of complete bytes of data.

If there is an error, the first byte is set to 0x81 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: String - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: int - the MODBUS slave unit identifier (1 to 247)
- address: int - the starting address of the block of coils
- quantity: int - the quantity of coils you are reading (1 to 2000).

Returns:

- byte[] - response PDU.

modbusReadDiscreteInputs

```
static byte[] modbusReadDiscreteInputs(String name, int unit, int address, int quantity)
```

Reads the content of a contiguous block of discrete inputs in a MODBUS device, using function code **0x02**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x02, the second to the byte count of the data part, and the data starts from the third byte.

💡 **The discrete inputs in the response message are packed as one input per bit of the data field. Status is indicated as 1= ON; 0= OFF. The LSB of the first data byte contains the input addressed in the query. The other inputs follow toward the high order end of this byte, and from low order to high order in subsequent bytes. If the returned input quantity is not a multiple of eight, the remaining bits in the final data byte will be padded with zeros (toward the high order end of the byte). The byte count field of the response PDU specifies the quantity of complete bytes of data.**

If there is an error, the first byte is set to 0x82 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: String - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: int - the MODBUS slave unit identifier (1 to 247)
- address: int - the starting address of the block of discrete inputs
- quantity: int - the quantity of discrete inputs you are reading (1 to 2000).

Returns:

- byte[] - response PDU.

modbusReadHoldingRegisters

```
static byte[] modbusReadHoldingRegisters(String name, int unit, int address, int quantity)
```

Reads the content of a contiguous block of holding registers in a MODBUS device, using function code **0x03**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x03, the second to the byte count of the data part, and the data starts from the third byte.

If there is an error, the first byte is set to 0x83 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: String - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: int - the MODBUS slave unit identifier (1 to 247)
- address: int - the starting address of the block of registers
- quantity: int - the quantity of registers you are reading (1 to 125).

Returns:

- byte[] - response PDU.

modbusReadInputRegisters

```
static byte[] modbusReadInputRegisters(String name, int unit, int address, int quantity)
```

Reads the content of a contiguous block of input registers in a MODBUS device, using function code **0x04**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x04, the second to the byte count of the data part, and the data starts from the third byte.

If there is an error, the first byte is set to 0x84 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: String - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: int - the MODBUS slave unit identifier (1 to 247)
- address: int - the starting address of the block of registers
- quantity: int - the quantity of registers you are reading (1 to 125).

Returns:

- byte[] - response PDU.

modbusReadWriteMultipleRegisters

```
static byte[] modbusReadWriteMultipleRegisters(String name, int unit, int readaddress, int readquantity, int
writeaddress, byte[] bytes)
```

Writes a block of contiguous registers in a MODBUS device, then reads the content of a contiguous block of holding registers, using function code **0x17**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x17, the second to the byte count of the data part, and the data starts from the third byte.

If there is an error, the first byte is set to 0x97 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: String - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: int - the MODBUS slave unit identifier (1 to 247)
- readaddress: int - the starting address of the block of registers to read
- readquantity: int - the quantity of registers you are reading (1 to 125)
- writeaddress: int - the starting address of the block of registers to write
- bytes: byte[] - data to be written in the registers, using big-endian encoding (the most significant byte is sent first).

Returns:

- numeric array - response PDU.

modbusWriteMaskRegister

```
static byte[] modbusWriteMaskRegister(String name, int unit, int address, byte[] mask, byte[] bytes)
```

Writes a register with a bit mask in a MODBUS device, using function code **0x16**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x16, the next two bytes to the mask and the last two to register value.

If there is an error, the first byte is set to 0x96 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: String - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: int - the MODBUS slave unit identifier (1 to 247)
- mask: byte[] - the AND mask to be applied to the register
- bytes: byte[] - data to be written in the register, using big-endian encoding (the most significant byte is sent first).

Returns:

- byte[] - response PDU.

modbusWriteMultipleCoils

```
static byte[] modbusWriteMultipleCoils(String name, int unit, int address, int coils, byte[] data)
```

Writes multiple coils in a MODBUS device, using function code **0x0F**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x0F, the next two to the data address of the first coil, and the last two to the number of coils written.

If there is an error, the first byte is set to 0x8F and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: String - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: int - the MODBUS slave unit identifier (1 to 247)
- address: int - the coil's address
- coils: int - the number of coils to be written
- bytes: byte[] - The requested ON/OFF states are specified by contents of the request data field, following the same data format defined in the Modbus specification for function 0x0F. A logical '1' in a bit position of the field requests the corresponding output to be ON. A logical '0' requests it to be OFF. The first byte represents coils with the lowest address, with the least significant bit addressing the lowest address coil.

Returns:

- byte[] - response PDU.

modbusWriteMultipleRegisters

```
static byte[] modbusWriteMultipleRegisters(String name, int unit, int address, byte[] bytes)
```

Writes a block of contiguous registers in a MODBUS device, using function code **0x10**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x10, the next two bytes to the starting address and the last two to the quantity of registers.

If there is an error, the first byte is set to 0x90 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: String - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: int - the MODBUS slave unit identifier (1 to 247)
- address: int - the starting address of the block of registers
- bytes: byte[] - data to be written in the registers, using big-endian encoding (the most significant byte is sent first).

Returns:

- byte[] - response PDU.

modbusWriteSingleCoil

```
static byte[] modbusWriteSingleCoil(String name, int unit, int address, boolean value)
```

Writes a single coil in a MODBUS device, using function code **0x05**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x05 and the last two to 0xFF00 if the coil is on, and 0x0000 if the coil is off.

If there is an error, the first byte is set to 0x85 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: String - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: int - the MODBUS slave unit identifier (1 to 247)
- address: int - the coil's address
- value: boolean - true to turn the coil on, false to turn it off.

Returns:

- byte[] - response PDU.

modbusWriteSingleRegister

```
static byte[] modbusWriteSingleRegister(String name, int unit, int address, byte[] bytes)
```

Writes a single register in a MODBUS device, using function code **0x06**.

Returns a standard response PDU.

If there is a correct response from the device, the first byte is set to 0x06 and the last two to register value.

If there is an error, the first byte is set to 0x86 and the second byte to the error code.

This function can be invoked passing a name parameter that represents either a predefined I/O Server, or the host name or IP address of the MODBUS gateway.

When using the gateways's IP address or name, you can call this function to access gateways that are not defined as I/O servers.

Parameters:

- name: String - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit: int - the MODBUS slave unit identifier (1 to 247)
- address: int - the register's address
- bytes: byte[] - data to be written in the register, using big-endian encoding (the most significant byte is sent first).

Returns:

- byte[] - response PDU.

Network Services

ping

```
static boolean ping(String host, int timeout)
```

Sends an IP network ping command to the host specified.

Parameters:

- host: String - the hostname or IP address
- timeout: int - timeout in milliseconds.

Returns:

- boolean - true if the host is reachable, false otherwise.

urlGet

```
static String urlGet(String url, String user, String password)
```

Sends a GET HTTP or HTTPS request.

The HTTP basic or digest access authentication methods are supported.

Parameters:

- url: String - a full url, including the mandatory http:// or https:// query scheme and optional query string
- user: String - the user for HTTP authentication (set to null if authentication is not needed)
- password: String - the password for HTTP authentication (set to null if authentication is not needed).

Returns:

- String - a string with the HTTP return code, followed by ":" and the response content, or null if a generic error occurred (including malformed URLs).

urlPost

```
static String urlPost(String url, String contenttype, String data, String user, String password)
```

Sends a POST HTTP or HTTPS request.

The HTTP basic or digest access authentication methods are supported.

Parameters:

- url: String - a full url, including the mandatory http:// or https:// query scheme and optional query string
- contenttype: String - the MIME content type of the data being sent (e.g. "text/plain")
- data: String - the data being sent with the POST request
- user: String - the user for HTTP authentication (set to null if authentication is not needed)
- password: String - the password for HTTP authentication (set to null if authentication is not needed).

Returns:

- String - a string with the HTTP return code, followed by ":" and the response content, or null if a generic error occurred (including malformed URLs).

wakeOnLan

```
static int wakeOnLan(String broadcast, String address)
```

Sends a Wake on LAN command to the LAN specified in the broadcast.

Parameters:

- broadcast: String - the LAN broadcast address. For example “192.168.0.255”
- address: String - the IP or MAC address of the network interface. For example, IP address “192.168.0.1” or MAC address “45-ab-44-00-00-ba”.

💡 **If the IP address is used to identify the destination interface, HSYCO will try to automatically determine the corresponding MAC address, permanently saving this association27. So, when using the IP address, and only the first time this method is called, the device with have to be on to allow HSYCO to determine the MAC address.**

Returns:

- int - 1 if successful, -1 if not.

Public Announcement

The audioPlay() functions are used to play recorded audio files or text-to-speech messages.

Audio can be sent to the Web browser, the server’s audio line out connector, the internal speaker or audio out line of Axis cameras, and the speaker of SNOM phones or PA devices.

See the Audio and Public Announcement section for additional information.

Audio Destination	Description
speaker	the server’s audio line out connector
web	the Web browser’s audio output 💡 **this feature is supported only on Firefox and Chrome browsers**
axis@camera id	audio sent to an Axis camera, using the camera id defined in the Cameras parameter in hsyco.ini

snom@ip:port	audio sent to SNOM phones and public announcement devices, to the multicast IP address and port specified (the IP address and port should be configured as multicast addresses on each phone)

audioPlay - file

```
static int audioPlay(String where, File file)
```

Plays a pre-recorded audio file.

Parameters:

- where: String - the audio destination; see the table above
- File: String - the audio file, relative to HSYCO's root directory.

Returns:

- int - the index of this message in the play back queue if successful, -1 in case of errors.

audioPlay - text to speech

```
static int audioPlay(String where, String voice, String text)
```

Converts a text message to audio using the installed text-to-speech engine, and plays the audio.

Parameters:

- where: String - the audio destination; see the table above
- voice: String - the voice name for the text-to-speech engine; see the Audio and Public Announcement section for additional information
- text: String - the text message for text-to-speech conversion.

Returns:

- int - the index of this message in the play back queue if successful, -1 in case of errors.

Serial Communication Ports

closeComm

```
static void closeComm(String portname)
```

Closes a serial port connection when the serial port is defined ad a “server” type, and is ignored otherwise.

💡 **When used with serial servers having a fail-over configuration, this method forces a reconnect at the next call of readComm() or writeComm(), so that a switch between the fail-over IPs would happen if the current IP is not responding.**

Parameters:

- portname: String - the name of the communications port, as defined in the CommPorts parameter in hsyco.ini.

readComm

```
static String readComm(String portname, int len)
```

Reads a sequence of bytes from a communication port.

The number of bytes read depends on the availability of data in the buffer of the communication port, until the maximum number specified in the len parameter.

💡 **There is a 2 seconds default time-out on reads. If the buffer is empty, this function will wait until the time-out expires before returning.**

If len is set to 0, the function doesn't return data, but it empties the communication port's buffer from older data.

💡 **If both verboseLog and userLog are true, the full trace of received and sent bytes is written to the log file.**

Parameters:

- portname: String - the name of the communications port, as defined in the CommPorts parameter in hsyco.ini
- len: int - maximum number of bytes to be read from the port, or 0 to empty the buffer without reading data.

Returns:

- String - null in case of errors, a string with the hexadecimal representation of the received byte sequence, or an empty string if the incoming data buffer is empty. For example "45AB4400BA".

writeComm

```
static int writeComm(String portname, String data)
```

Sends a sequence of bytes to a communication port.

💡 **If both verboseLog and userLog are true, the full trace of received and sent bytes is written to the log file.**

Parameters:

- portname: String - the name of the communications port, as defined in the CommPorts parameter in hsyco.ini
- data: String - a string with the hexadecimal representation of the sequence of bytes to be sent. For example "45AB4400BA".

Returns:

- int - the number of bytes sent if successful, or 0 in case of errors.

Squeezebox

slimButton

```
static int slimButton(String player, String button)
```

Controls the Squeezebox players, simulating a remote control button.

Parameters:

- player: String - the Squeezebox name, as listed in the slimPlayers parameter in hsyco.ini
- button: String - the Squeezebox command.

Returns:

- int - 1 if successful, 0 otherwise.

slimCommand

```
static String slimCommand(String player, String command)
```

Sends commands to the Squeezebox players.

Parameters:

- player: String - the Squeezebox name, as listed in the slimPlayers parameter in hsyco.ini
- command: String - the Squeezebox command, as described in the documentation of the Command Line Interface (CLI) of SqueezeCenter.

Returns:

- String - the protocol answer to the command if successful, or null in case of errors.

User Interface

All identified UI objects support dynamic attributes that can be changed at run-time.

The primary commands used to set dynamic attributes are the UISET action in EVENTS, and the uiSet() Java and JavaScript function.

Set levels

You can set dynamic attributes at different levels:

- global, using UISET and not specifying any session information - will set the attribute to the desired value for all clients
- client session, using UISESSIONSET on USER and PAGE events, or uiSessionSet(session, id, attr, value) in JavaScript and uiSet(String session, String id, String attr, String value) in Java - will set the attribute for a single specific client session
- application scope, using uiSessionSet(session, id, attr, value) in JavaScript and uiSet(String session, String id, String attr, String value) in Java - will set the attribute only for projects and pages that are explicitly bound (subscribed) to the same scope. You can use any string literal to identify a scope, and simply use that string in the session parameter of uiSet or uiSessionSet, instead of the client session id.

Priority rules

Overlapping global, session or scope UI attributes set on the same object is possible. In this case, a session set has the highest priority, followed by global set and finally the scope set. So, if you have a scope set on an object, and a global set is called on the same attribute of the same object, this global set will override the scope set. A session set will override both the scope and global set.

Session lifetime

While a global set will last until HSYCO is restarted, the client sessions have a limited lifetime.

Basically a client session is created whenever you load a URL pointing to an HSYCO project, and is automatically cleared after one hour of inactivity, where session activity is defined as the client being actively connected to the HSYCO server.

Scopes also have the same lifetime of client sessions, but activity in this case is defined as at least one client that is showing a page or project that is bound to the scope.

You can also create persistent scopes, simply using scope literals ending with the ! character. Persistent scopes will not follow be cleared, even after prolonged inactivity, and will last until HSYCO is restarted.

uiClear

```
static void uiClear(String session)
```

Removes all session specific UI object attributes set with uiSet().

uiClear() is also implicitly called after the client session inactivity timeout expires.

The uiClearEvent() callback is called following the uiClear() execution if the client session becomes active again (i.e. the client starts handshaking again with the server with the same session id).

Parameters:

- session: String - session id string that uniquely identifies the client session.

uiGet - global

```
static String uiGet(String id, String attr)
```

Gets the current value of an attribute of an identified UI object, set with the uiSet() function or the UISET action in EVENTS.

Parameters:

- id: String - object id
- attr: String - attribute name.

Returns:

- String - the current value of the attribute of an identified UI object, or null if the object attribute has not been set.

uiGet - session

```
static String uiGet(String session, String id, String attr)
```

Gets the session-specific value of an attribute of an identified UI object, set with the uiSessionSet() function or the UISESSIONSET action in EVENTS

Parameters:

- session: String - session id string that uniquely identifies the client session.
- id: String - object id
- attr: String - attribute name.

Returns:

- String - the session value of the attribute of an identified UI object, or null if the object attribute has not been set.

uiSet - session

```
static void uiSet(String session, String id, String attr, String value)
```

Changes the dynamic attributes of identified user interface (UI) objects for a specific client session.

Session specific attributes values will override values set globally.

id is the unique name that identifies the object, specified in the index.hsm file with the extension !id after the object type.

Parameters:

- session: String - session id string that uniquely identifies the client session.
- id: String - object id
- attr: String - attribute name
- value: String - attribute value.

uiSet - global

```
static void uiSet(String id, String attr, String value)
```

Changes the dynamic attributes of the identified user interface (UI) objects.

id is the unique name that identifies the object, specified in the index.hsm file with the extension !id after the object type.

Parameters:

- id: String - object id
- attr: String - attribute name
- value: String - attribute value.

user

```
static void user(String name, String param)
```

Triggers a USER event, the execution of the userCommand() JavaScript function and of the userCommand() Java method.

It can be used as a calling mechanism between Java, JavaScript and EVENTS.

Parameters:

- name: String - the name parameter, passed to userCommand() and the USER event
- param: String -the param parameter, passed to userCommand() and the USER event.

JavaScript Callback Functions API

System Functions

DaylightEvent

```
DaylightEvent(day)
```

Called at sunrise and sunset, according to the latitude and longitude values set in hsyco.ini, and the optional SunriseOffsetMinutes and SunsetOffsetMinutes parameters.

Parameters:

- day: boolean - true at sunrise, false at sunset.

haActiveEvent

```
haActiveEvent(active)
```

Triggered by the change of state of an HSYCO server in a master/slave high availability configuration.

Parameters:

- active: boolean - true if the server is active, false if not active.

PowerEvent

```
PowerEvent(power)
```

Triggered by a change of the current power load level, as set using the powerSet() Java API or the POWER action.

Thanks to this function, you can alter the power value shown in the Web interface, for example to aggregate power readings acquired from other sensors.

Parameters:

- power: numeric - the power level, in Watts.

Returns:

- numeric - if PowerEvent() returns -1 or doesn't return a value, HSYCO status is updated with the detected power value, the value returned by PowerEvent() is otherwise used.

💡 **The JavaScript PowerEvent function is always called, but its return value is used only when the Java PowerEvent callback is not defined or returned -1.**

programTimerEvent

```
programTimerEvent(name)
```

Called when a program timer is activated.

Program timers are set using programTimerSet(), programTimerReset(), programTimerRepeat(), or using the corresponding actions in EVENTS.

Parameters:

- name: string - program timer name.

SchedulerEvent

```
SchedulerEvent(groupname, schedulename)
```

This callback blocking function allows to call user methods at configurable intervals.

You define schedules using a group name and a schedule name.

Schedules under the same group run in the same thread and are executed sequentially, based on their interval in milliseconds. Schedules in different groups run in parallel.

Parameters:

- groupname: string - the scheduler's group name
- schedulename: string - the scheduler's name.

StartupEvent

```
StartupEvent()
```

Called only once when HSYCO starts, after the initialization and HTTP server start-up, but before the execution of the different field interface threads. It is not executed in a thread, it is therefore a blocking method, that must complete its execution before all other HSYCO services can be started.

SunPositionEvent

```
SunPositionEvent(azimuth, elevation)
```

Called when the Sun changes its height with respect to the horizon or its angle from true north.

Parameters:

- azimuth: numeric - the current Sun angle from true north, in decimal degrees
- elevation: numeric - the current Sun elevation in decimal degrees from the horizon. Elevation is negative at night.

TimeEvent

```
TimeEvent(time)
```

Called every 60 seconds.

HSYCO tries to synchronize the execution at the beginning of the minute, executing this method at seconds 0 of every minute.

💡 **However, the execution is not guaranteed. In case of heavy load HSYCO might not be able to execute the call with accurate timing. In extreme cases, it could occasionally skip some calls.**

Parameters:

- time: numeric - the current time in milliseconds.

varEvent

```
varEvent(name, value)
```

Triggered by the change of value of a program variable.

Parameters:

- name: string - the variable name. Names are case-insensitive
- value: string - the value to be assigned to the program variable.

Cameras

CameraCommandEvent

```
CameraCommandEvent(function, action, camera)
```

Triggered by a camera control input from the Web interface.

It can be used to execute arbitrary actions when the user clicks on the active areas of the camera view, or to replace some or all of the standard PTZ commands with custom actions.

Parameters:

- function: string - see the table below
- action: string - see the table below
- camera: string - the camera name.

Returns:

- numeric - if CameraCommandEvent() returns -1 or doesn't return a value, the standard PTZ command associated to the PTZ driver is skipped, otherwise it is executed normally.

💡 **The JavaScript CameraCommandEvent function is always called, but its return value is used only when the Java CameraCommandEvent callback is not defined or returned -1.**

icon	function	action
	focus	far
	focus	near
	zoom	tele
	zoom	wide
	move	left
	move	right
	move	up
	move	down
	move	stop

CameraMotionEvent

```
CameraMotionEvent(event, time)
```

Called when HSYCO starts recording video from a camera.

Recording starts when a camera notifies a recording request with the recording HTTP API call (the zone=id portion is optional):

```
http://192.168.0.50/x/camerarec?camera=name&zone=id
```

Recording can also be triggered by the CameraRecTrigger() Java method or JavaScript function, or the CAMERAREC or CAMERARECFULL actions in EVENTS.

Parameters:

- event: string - the event identification string, either the camera name only or name:id if recording is triggered by a camerarec HTTP call with the optional zone=id parameter
- time: numeric - the current time in milliseconds.

CameraViewEvent

```
CameraViewEvent(camera, active)
```

CameraViewEvent() is called on changes of the live view status of cameras.

When at least one web client is showing the live feed from a camera, the camera is marked as active.

When no live feed request is received for a several consecutive seconds, the camera is marked as not active.

Parameters:

- camera: string - the camera name
- active: boolean - true if active, false if not.

DMX

DmxEvent

```
DmxEvent(channel, state)
```

Called after changes to DMX-512 channels.

Parameters:

- channel: numeric - DMX channel address, from 1 to 512 which might be preceded by the DMX gateway number, starting from 0
- state: numeric - the new channel value, from 0 to 255.

DmxFilter

```
DmxFilter(channel, state, reverse)
```

This function allows to filter the value of the status of each channel on the DMX-512 buses, sending to the gateway different values from those set in HSYCO.

This way, it is possible to apply chromatic corrections, or any on the fly translation of channels values.

💡 **This is a blocking function and is called when sending commands to the DMX gateway.**

It is also called after reading current channels values from the gateway. In this case the conversion function should be symmetric and return a complementary value.

Parameters:

- channel: numeric - the DMX channel address, from 1 to 512 which might be preceded by the DMX gateway number, starting from 0
- state: numeric - the unfiltered value of the channel, from 0 to 255
- reverse: boolean - false if called when writing to the gateway; true if called when reading from the gateway.

Returns:

- numeric - the channel value that will be sent to the DMX gateway.

💡 **The JavaScript DmxFilter function is called only when the Java DmxFilter callback is not defined or returned the original value.**

DmxStartupEvent

```
DmxStartupEvent(index)
```

Called when starting the monitor threads for each DMX bus, once per bus at the start of the execution of HSYCO, and also after every restart of the monitor thread, for example after communication errors.

It is safe, inside this function, to execute commands for the same DMX gateway.

💡 **This is a blocking function. The DMX driver will start after this function returns.**

Parameters:

- Index: numeric - number of the DMX bus, starting from 0.

Infrared Control

IREvent

```
IREvent(received, irtransid, event)
```

This function is called when an IRTrans receives or sends an IR command from its local IR database.

💡 **IRTrans can only receive and decode infrared commands that have been previously stored in its internal FLASH memory database.**

Parameters:

- received: boolean - true if the command has been received by the IRTrans, false if sent
- irtransid: numeric - IRTrans number, starting from 0 for the first device, based on the listing order of the IRTrans parameter in hsyco.ini
- event: string - the string of the command received or issued, formatted as: "remote_name,command".

I/O Servers

IOEvent

```
IOEvent(name, value)
```

Triggered by the value change of any data point of an I/O server.

Parameters:

- id: string - the server id and the name of the data point. According to the type of server, the format changes but it generally appears as server name, as declared the ioServers parameter in hsyco.ini, followed by a specific name representing the data point within the system

- value: string - the new value. For binary inputs or outputs, the returned values are “0” or “1”.

IOStartupEvent

```
IOStartupEvent(index)
```

Triggered by the start of the communication thread of each I/O server - once for each server at the beginning of the execution of HSYCO, and also after every reboot of the communication thread, for example after communication errors.

This is a blocking method. The I/O server driver will start after this function returns.

Parameters:

- index: numeric - the I/O server number, starting from 0, based on the listing order of the ioServers parameter in hsyco.ini.

Modbus

For detailed information about the Modbus protocol, and the response PDU formats, you should refer to the MODBUS Application Protocol Specification V1.1b documentation at:
http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

ModbusEvent

```
ModbusEvent(name, addr, unitid, pdu)
```

Called by the Modbus Server I/O Server on every Modbus request received from a Modbus client.

The returned byte array is used as the Modbus response PDU (function code and data).

Parameters:

- name: string - the name of the Modbus Server I/O server
- addr: string - IP address of the Modbus client
- unitid: numeric - the unit id set in the received Modbus request
- pdu: string - the request PDU data, in hexadecimal string format.

Returns:

- string - the response PDU data, in hexadecimal string format.

The JavaScript ModbusEvent function is called only when the Java ModbusEvent callback is not defined or returned a null value.

Network Services

httpCallEvent

```
httpCallEvent(address, secure, query)
```

Called when the HSYCO HTTP server receives a GET request with the following path:

```
/x/httpcall?query
```

💡 **The HTTP client's IP address must be in the trusted network range, or the httpcall request will be rejected.**

This function shoud return a string, that will be sent as the text/plain answer to the HTTP client (with HTTP response code 200).

HSYCO will return a 404 Not Found response code if all of the following conditions are true:

- no HTTP event matches the query in events.txt
- the httpCallEvent JavaScript callback is not defined in events.txt, or doesn't return a value
- the httpCallEvent Java callback is not defined in user.java or in any plugins, or returns null.

If the httpCallEvent is defined both in JavaScript and Java, the Java version return value (unless null) will be used in the HTTP response.

Parameters:

- address: string - the IP address of the HTTP client
- secure: boolean - true if the request is sent to the HTTPS server, false if the request is sent to the HTTP server
- query: string - the string following the ? in the URL path.

Returns:

- string - the string to send as the text/plain answer to the HTTP client.

LocationEvent

```
LocationEvent(mac, ip, zoneId)
```

Called, if the location service is active, when a variation in the association of a client to the Wi-Fi Access Points is detected.

Parameters:

- mac: string - the MAC address of the client
- ip: string - the IP address of the client, if available, or null (the IP address could be null if the associated device is not connected to the HSYCO Web interface)
- zoneId: numeric - in case of association, it is the Access Point sequence number, according to the order defined in the LocationBases parameter in hsyco.ini; -1 if the client was de-associated from the Wi-Fi network.

PBX

PBXCallEvent

```
PBXCallEvent(host, caller, called)
```

Called when a call notification is sent to HSYCO by the PBX system.

If the method returns true, the request is recorded on the log file with the [OK] status, with [ERROR] otherwise.

The PBX should generate HTTP requests to the HSYCO Web Server, using the following formats:

```
http://192.168.0.50/x/vcall/<from>/<to>
```

or:

```
http://192.168.0.50/x/vcall?from=<from>&to=<to>
```

For example, the HTTP request:

```
http://192.168.0.50/x/vcall?from=21&to=25
```

is interpreted as a call notification from internal number 21 to number 25.

💡 **The HTTP client's IP address must be in the trusted network range, or the httpcall request will be rejected.**

Parameters:

- host: string - the IP address of the PBX server
- caller: string - the caller number
- called: string - the called number.

Returns:

- boolean - return true to generate a log entry with [OK] status, false for [ERROR] status.

Squeezebox

SlimPowerEvent

```
SlimPowerEvent(index, power)
```

Called when a Squeezebox player is turned on or off.

Parameters:

- index: numeric - player number, starting from 0 for the first player, based on the listing order of the slimPlayers parameter in hsyco.ini
- power: numeric - 0: OFF, 1: ON.

SlimStatusEvent

```
SlimStatusEvent()
```

Called when a Squeezebox player changes its playback mode.

Parameters:

- index: numeric - player number, starting from 0 for the first player, based on the listing order of the slimPlayers parameter in hsyco.ini
- status: numeric - 0: OFF, 1: ON, 2: PAUSE, 3: PLAY, -1: UNKNOWN.

SlimVolumeEvent

```
SlimVolumeEvent(index, volume)
```

Called when a Squeezebox player changes its volume level.

Parameters:

- index: numeric - player number, starting from 0 for the first player, based on the listing order of the slimPlayers parameter in hsyco.ini
- volume: numeric - an integer number representing the volume change (not the absolute audio level) in a scale from 0 to 100; positive numbers correspond to a volume increase, negative numbers to a decrease.

Timers and Schedulers

UserTimerEvent

```
UserTimerEvent(name, active)
```

Called when a user timer or scheduler's rule should be activated, and before executing the associated action in EVENTS.

It is also executed at the timer or scheduler's rule deactivation.

This method must return a true value in order to make the timer or rule actually active and the associated actions executed.

Whenever the method returns false, the timer will not be activated, and the method will be called again once a minute for the scheduled duration of the timer.

In the same way, if the method returns false when the timer or rule is deactivated, the timer or rule will not be deactivated, extending the timer activation beyond the scheduled duration.

This is a blocking method.

Parameters:

- name: string - timer name or scheduler's rule name
- active: boolean - true when activating and false when deactivating the timer or scheduler's rule.

Returns:

- boolean - true to confirm the timer event, false to block the event.

💡 **The JavaScript UserTimerEvent function is alway called, but its return value is used only when the Java UserTimerEvent callback is not defined or returned true.**

User Interface

pageEvent

```
pageEvent(address, session, userid, project, page)
```

Called when a Web client initially loads a project and when navigating between pages.

There is no guarantee that the event would be fired the exact moment the page is loaded.

Network delays or errors could cause delayed triggering of this event.

Parameters:

- address: string - the client's IP address
- session: string - session id string that uniquely identifies the client session
- userid: string - the user id
- project: string - project's id
- page: string - the page id, with the #landscape or #portrait suffix if the page has distinct orientation versions in the project.

uiClearEvent

```
uiClearEvent(session)
```

Called, following an explicit call of uiClear() or after the client session inactivity timeout expires, if the client session becomes active again (i.e. the client starts handshaking again with the server with the same session id).

Parameters:

- session: string - session id string that uniquely identifies the client session.

userCommand

```
userCommand(session, userid, name, param)
```

Called by user clicks on buttons created in the Web interface with the user objects, by a USER action in EVENTS, or by the user() Java method and JavaScript function.

This function can optionally return a string. Any returned string causes the log of an [OK] status. If you want to prevent logging, for example to avoid sensitive information to be written in the log file, you should return the string "!".

If you want to force the Web interface to show a specific page after the button is pressed (it would be like pressing a link object, return a string starting with "page:" followed by the page name; in this case, userCommand() will be called again when that popup or page is closed, with "/close" appended to param.

Parameters:

- session: string - a session id string that uniquely identifies the client session
- userid: string - the user id
- name: string - the name field of the user object
- param: string - the param of the user object

Returns:

- string - log or go to page, see text above. If this function doesn't return a value, an [ERROR] status is logged in the log file, while any other string causes the log of an [OK] status.

💡 **The JavaScript userCommand function is alway called, but its return value is used only when the Java userCommand callback is not defined or returned null or an empty string.**

userSubmit

```
userSubmit(session, userid, name, fields)
```

This function is similar to userCommand(session, userid, name, param).

When userSubmit() is defined, it will be called when a submit button is pressed, while userCommand() will continue to be called on user buttons.

userSubmit() provides a more convenient access to the values of multiple input fields in a form.

This function can optionally return a string. Any returned string causes the log of an [OK] status. If you want to prevent logging, for example to avoid sensitive information to be written in the log file, you should return the string “!”.

If you want to force the Web interface to show a specific page after the button is pressed (it would be like pressing a link object), return a string starting with “page:” followed by the page name; in this case, userCommand() will be called when that popup or page is closed, with "/close" appended to param.

Parameters:

- session: string - a session id string that uniquely identifies the client session
- userid: string - the user id
- name: string - the id of the (submit!id) object
- fields: array - the array of all input fields in the container where the (submit) object is located.

Returns:

- string - log or go to page, see text above. If this function doesn't return a value, an [ERROR] status is logged in the log file, while any other string causes the log of an [OK] status.

💡 **The JavaScript userSubmit function is alway called, but its return value is used only when the Java userCommand or userSubmit callback are not defined or returned null or an empty string.**

Examples:

The following code shows how to retrieve all field names and values of a submitted form.

```
function userSubmit(session, user, name, values) : {
        if (session.length > 0) {
                messageLog(user + ', ' + name + '=' + values);
                var keys = values.keySet().toArray();
                for (var i = 0; i < keys.length; i++) {
                        messageLog(user + ', ' + keys[i] +  ' = ' + values.get(keys[i]));
                }
                return "";
        }
}
```

Create a project with a form and a few input fields in a container. Pressing the submit button should return something like this in the log file:

```
2014.01.31 13:54:32.034 - staff, submit1={keypad1=344, date1=20140131, input1=this is a text field}
2014.01.31 13:54:32.038 - staff, keypad1 = 344
2014.01.31 13:54:32.039 - staff, date1 = 20140131
2014.01.31 13:54:32.042 - staff, input1 = this is a text field
2014.01.31 13:54:32.043 - WEB USER COMMAND [submit1]: input1@this is a text field@date1@20140131@keypad1@344
[OK]
```

WebRootRequestEvent

```
WebRootRequestEvent(addr, secure, useragent)
```

This method is called when the Web server receives a root URL request. It can return a string to redirect the browser to a valid URL, or null to prevent redirection.

Parameters:

- addr: string - the HTTP client's address
- secure: boolean - true if this is an HTTPS request
- useragent: string - the Web browser's user agent information.

Returns:

- string - optional return string containing a valid absolute or relative URL

Examples:

The following code redirects any root HTTP request received by the HSYCO server to the manager page.

```
function WebRootRequestEvent(addr, secure, useragent) : {
        return "hsycoserver/manager";
}
```

Working with Forms

Introduction to Forms

There are several objects that allow the creation of a form to send data to the server. These objects are of three types: fields, submit buttons and containers.

A date **field**

The corresponding date **field panel**

A submit button

- Fields:
  - **Input**: for text values
  - **Time**: for time values
  - **Date**: for date values
  - **Keypad**: for numeric values
  - **Checkbox**: for true/false values
  - **RadioButton**: for a value from a list
- Field panels:
  - **TimePanel**: similar to time

- **DatePanel**: similar to date
- **KeypadPanel**: similar to keypad

- Submit buttons:
  - **Submit**: default size
  - **SubmitMini**: "mini" size
  - **SubmitMicro**: "micro" size
  - **SubmitImage**: custom image

- **Container**: to group multiple objects

Fields of type **time**, **date** and **keypad** have a specific panel used to input the value, that will be visible in a popup when the field is clicked. These panels can also be added directly to the page, using the field panel objects: **timepanel**, **datepanel** and **keypadpanel** objects.

Containers are used to group fields and submit buttons.

Submit buttons are used to commit changes and send all the fields' values to the server. Only the fields inside the same container and nested containers will be considered.

Fields can also have an **autosubmit** mode, that sends the value when there's a change, instead of waiting for the user to click on a **submit** button. This mode is automatically activated if there's no **submit** button on the page or in one of the parent containers. This autosubmit mode can also be toggled using a specific UISet (for example see the autosubmit UI Attribute on the input object).

Field panels have an OK button that works as a **submit** button. If there's another **submit** button, the OK buttons on the panels won't be visible.

Creating a Form

Client-side

A form can be created either using the Project Editor to edit the project, or by manually writing the code in the project's index file (index.hsm).

Using the Project Editor, the desired fields have to be placed on the page (or inside a container). Each field requires a specific id so the code on the Server-side can identify it properly. Placing a **submit** button will change the behavior of the fields accordingly.

Server-side

To receive the values that the client is sending, code needs to be written on the Server-side. In a form, when one of these things happen:

- a change is detected on a field in autosend mode
- the OK button is pressed on a field panel
- a **submit** button is pressed

a request will be sent that can be dealt with from Java or JavaScript in the events environment.

If the field's or submit button's id starts with **$**, then the server will also automatically set a variable named **$<id>** to the appropriate value.

Example: single field, no submit button

If a single field is sending its value (the above first two cases), HSYCO will call the method userCommand in events.txt.

```
userCommand(session, userid, name, param)
```

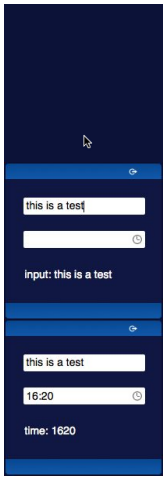
The method will receive:

- **name** - the id of the field object
- **param** - the value of the field object

The USER event **USER <field id>** will be generated as well.

The same event can also be written in java (see userCommand or userCommand with session information)

In the following example we have: an input field with id "myinput", a time field with id "mytime", a text object with id "mytext" and no submit buttons. The request will be sent for each field whenever there's a change in the field's value and the text object's text will be set as a result.

Final results:	**index.hsm:**
	(#skin blue) (#language en) (#size 200x200) (header Test Form) (menu) (input!myinput x20y17; width:150px) (time!mytime x20y58; width:150px) (text!mytext x20y112;) (endofmenu)

The javascript code in events.txt:

```
function userCommand(session, userid, name, value) : {
        // if we're receiving the input's value...
        if (name == "myinput") {
                uiSet("mytext","text","input: "+value);
        }
        // if we're receiving the time's value...
        else if (name == "mytime") {
                uiSet("mytext","text","time: "+value);
        }
        return ""; // any returned string causes the log of an [OK] status
}
```

Example: mutiple fields with a submit button

If a **submit** button has been pressed, HSYCO will call the method userSubmit in events.txt.

```
userSubmit(session, userid, name, fields)
```

The method will receive:

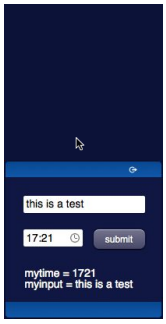
- **name** - the id of the submit button object
- **fields** - the hash map of all input fields in the container where the (submit) object is located

The USER event **USER <submit id>** will be generated as well.

The same event can also be written in java (see userSubmit)

In the following example we have: an input field with id "myinput", a date field with id "mydate", a text object with id "mytext" and a submit button with id "mysubmit". The request will be sent when the submit button is pressed and the text object's text will be set as a result.

Final results:	**index.hsm:**

	(#skin blue)
	(#language en)
	(#size 200x200)
	
	(header Test Form)
	
	(menu)
	  (input!myinput x20y17; width:150px)
	  (date!mydate x20y58; width:150px)
	  (text!mytext x20y112;)
	  (submitmicro!mysubmit x141y106; gr; send)
	(endofmenu)

The javascript code in events.txt:

```
function userSubmit(session, user, name, values) : {
        if (values) { // values is null when the code is compiled for test purposes
                var text = "";
                var keys = values.keySet().toArray();
                for (var i = 0; i < keys.length; i++) {
                        text += keys[i] + ' = ' + values.get(keys[i]) + "<br>";
                }

                uiSet("mytext","text",text);
        }
        return ""; // any returned string causes the log of an [OK] status
}
```

Working with User Objects

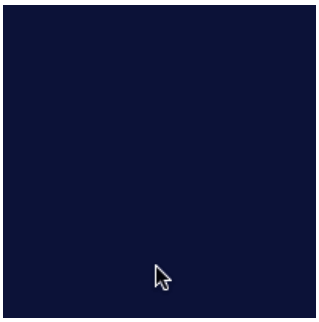
User objects are used to send commands to the server. There are few different user objects:

- User and its variants in different sizes: UserMini, UserMicro
- UserImage, with a custom image
- UserList, with a list of user objects

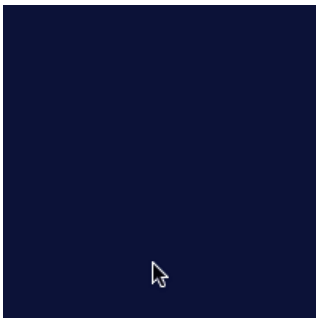
Commands can be dealt with from Java or JavaScript in the events environment.

Every user object has a name and a parameter, that are sent to the server, and a repeat mode. If this mode is enabled, the behaviour will change:

- **No repeat mode** - the command is sent on the *mouse up* or *touch up* events



- **Repeat mode enabled** - three types of commands are sent:
  - down: on *mouse down* or *touch down* events
  - repeat: if the button is still pressed, a repeat command is sent every second
  - up: on *mouse up* or *touch up* events, for a short press (less than a second, so no repeat commands have been sent yet)
  - stop: on *mouse up* or *touch up* events, for a long press (more than a second, so at least a repeat command has been sent)



A UserList object has a list of items, each behaving like a user object.

> While programming keep in mind that due to connection errors or user inputs (e.g. changing page) communication could be interrupted, so in the case of a user object with repeat mode enabled, the final command (up or stop) could be lost

Example

In this example we're going to use three user buttons to control a dimmer:

- on/off: no repeat, switches the dimmer on and off
- +: repeat mode, while pressed will increase the dimmer's value
- -: repeat mode, while pressed will decrease the dimmer's value

Final results:	**index.hsm:**
	(#skin blue) (#language en) (#size 200x200) (header Test User) (menu) (usermini onoff;; r1c1; gr; on/off) (usermini plus;; r2c1; gr; +; enabled) (usermini less;; r3c1; gr; -; enabled) (button dummy.dimmer.1; x93y34; label) (text!mytext x0y126;) (endofmenu)

The javascript code in events.txt:

```
init : {
        // dimmer value, initially off
        var dimmerValue = 50;
        var dimmerOn = false;
        // initialize dimmer
        ioSet("dummy.dimmer.1",dimmerValue);
        ioSet("dummy.dimmer.1","off");
}

function userCommand(session, userid, name, value) : {
        // split value to get command type and repetitions (e.g. value/repeat/1)
        var v = value?value.split("/"):[];
        // onoff user button
        if (name == "onoff") {
                dimmerOn = !dimmerOn; // toggle on/off
                if (dimmerOn) {
                        ioSet("dummy.dimmer.1","on");
                } else {
                        ioSet("dummy.dimmer.1","off");
                }
                // output text
                uiSet("mytext","text",name+" "+value+"<br>dimmer on: "+dimmerOn);
        }
        // plus and less user buttons, with command down or repeat
        else if (v[1] == "down" || v[1] == "repeat") {
                // delta: increases on every repeat (1,5,10,15...)
                var d = 1;
                if (v[1] == "repeat")
                        d = parseInt(v[2])*5;
                // plus or less
                if (name == "plus")
                        dimmerValue += d;
                else
                        dimmerValue -= d;
                // check limits
                if (dimmerValue > 100)
                        dimmerValue = 100;
                else if (dimmerValue < 0)
```

```
                        dimmerValue = 0;
                // still on?
                dimmerOn = dimmerValue > 0;
                // set new value
                ioSet("dummy.dimmer.1",dimmerValue);
                // output text
                uiSet("mytext","text",name+" "+value+"<br>dimmer value: "+dimmerValue+" ("+
(d>0?"+":"")+d+")");
        }
        return "";
}
```

Advanced Programming

This chapter goes beyond the application programming techniques you may normally use to develop customized applications in HSYCO.

HSYCO is a very open development environment. Any feature offered by the Java Runtime Environment is available to the developer to write custom Java code without restrictions.

You can also extend the core services of HSYCO, writing plug-in applications and your own I/O servers.

Use the links below to explore HSYCO advanced programming.

Plugins Applications

Plugins Applications Development

Custom I/O Drivers

Custom I/O Drivers Development

Custom Drivers Callback Methods API

Custom Drivers Command and Utility Methods API

Client-side Programming

Client JavaScript Extension API

The SQL Engine

Using the Embedded SQL Database Engine

Custom Drivers Callback Methods API

A few callback methods are called by the HSYCO core engine during initialization, exit, and for normal operations.

You should write your custom interface code inside these methods, and also set the default values of some static properties.

Static Properties

DEFAULTSOCKETPORT

```
DEFAULTSOCKETPORT
```

Used to set the default IP socket port, to be used when the port field is not set in the configuration. For example, if your driver is interfacing an HTTP-based service, you may want to set DEFAULTSOCKETPORT to 80.

COMMANDSQUEUESIZE

```
COMMANDSQUEUESIZE
```

Defines the size of the command queue.

The command queue is used to accumulate commands that the driver should execute, and are generated by ioSet() JavaScript or Java calls, the IOSET event or users pressing (button) objects in the Web user interface.

The command queue should be large enough to let the driver have time to consume incoming commands and not lose any. At the same time a queue that is too large could let a large number of messages to accumulate in case the driver has temporary issues talking to the external system.

HSYCO's built-in drivers usually have queue sizes between 64 and 1024. 256 is normally a good value.

SHUTDOWNWHENSLAVE

```
SHUTDOWNWHENSLAVE
```

This is the default value of the "shutdown when inactive" configuration parameter. If true and the "shutdown when inactive" is not defined, the driver will be automatically shut down when the HSYCO server is in a high availability configuration and the server is not the active server.

Callback Methods

init

```
boolean init(String name, HashMap<String, String> config)
```

Called once every time the driver is started.

Returning false will fail the initialization procedure, and the driver will not start.

Parameters:

- name: String - this driver's identifier
- config: HashMap<String, String> - a hash table of all basic configuration parameters, see the table below, and all options.

Parameter	Description
comm	communication port identifier
host	the host IP name address
port	the IP port
user	user name
password	password
shutdowninactive	the shutdown when inactive parameter

Returns:

- boolean - true if the initialization process is successful (HSYCO will consider the driver instance initialized and start calling the loop() callback); false to fail the initialization procedure (HSYCO will retry the initialization process later).

loop

```
boolean loop()
```

Called repeatedly while the driver should be alive.

Returning false will not update the keep-alive timer, causing the driver to quit after a while.

This call should always return within about 30 seconds, to avoid the keep-alive watch-dog intervention.

Returns:

- boolean - return true if the loop processing is successful, false on errors.

end

```
boolean end()
```

Called once when the driver quits after it has started correctly.

Should always return true.

Returns:

- boolean - always return true.

user

```
String user(String session, String userid, String id, HashMap<String, String> fields)
```

Called on user keys, gui input objects, submit and USER actions in Events or Java, when the user interface object identifier has a prefix that matches the driver's identifier followed by a "." character.

This method is optional, and doesn't need to be implemented if not required.

This method returns a string. Any returned string causes the log of an [OK] status. If you want to prevent logging, for example to avoid sensitive information to be written in the log file, you should return the string “!”.

If you want to force the Web interface to show a specific page after the button is pressed (it would be like pressing a (link) object), return a string starting with “page:” followed by the page name; in this case, userCommand() will be called again when that popup or page is closed, with "/close" appended to param.

Parameters:

- session: string - a session id string that uniquely identifies the client session
- userid: string - the user id
- id: String - see the table below
- fields: HashMap<String, String> - see the table below.

Type of input	id	fields
single (user) type object	<name>.<params> attributes of the (user) object. <name> is stripped of the driver's identifier prefix	null
single (input) type object	<id>.<value> where <id> is the (input) object identifier, without the driver's identifier prefix, and value is the (input) field value	null
multiple (input) type objects in (container) with (submit)	the ideintifier of the (submit) object, without the driver's identifier prefix	the identifiers of all objects in the container, and their values

Returns:

- String - log or go to page, see text above. If this function returns null, an [ERROR] status is logged in the log file, while any other string causes the log of an [OK] status.

command

```
void command(String name, String value)
```

Called for each command that should be processed by the driver.

This method is optional, and doesn't need to be implemented if not required.

Parameters:

- name: String - the I/O data point name, stripped of the driver's identifier prefix
- value: String - the data point value.

Custom Drivers Command and Utility Methods API

Constants

LIGHT

```
LIGHT
```

Used for the function field of deviceSet().

AUTOM

```
AUTOM
```

Used for the function field of deviceSet().

DIMMER

```
DIMMER
```

Used for the function field of deviceSet().

BINARY

```
BINARY
```

Used for the function field of deviceSet().

System Functions

dateSet

```
boolean dateSet(int year, int month, int day, int hour, int minute, int second)
```

Sets the HSYCO Server operating system's clock and hardware clock, in the local time zone.

Parameters:

- year: int - the year, greater than 1900
- month: int - the month, between 1-12
- day: int - the day, between 1-31
- hour: int - the hour, between 0-23
- minute: int - the minute, between 0-59
- second: int - the second, between 0-59.

Returns:

- boolean - true if successful, false if not.

getNextSunrise

```
long getNextSunrise(long now, boolean withoffset)
```

Computes the time of the next sunrise after the time set in the now parameter.

The sunrise time is returned in milliseconds since midnight, January 1, 1970 UTC.

Parameters:

- now: long - the reference time in milliseconds
- withoffset: boolean - true to take the sunrise offset into account, as defined with the SunriseOffsetMinutes in hsyco.ini, false to ignore the offset and return the actual sunrise time (Based on civil dawn, begins when the geometric center of the sun is 6° below the horizon).

Returns:

- long - the next sunrise time in milliseconds since midnight, January 1, 1970 UTC.

getNextSunset

```
long getNextSunset(long now, boolean withoffset)
```

Computes the time of the next sunset after the time set in the now parameter.

The sunset time is returned in milliseconds since midnight, January 1, 1970 UTC.

Parameters:

- now: long - the reference time in milliseconds
- withoffset: boolean - true to take the sunset offset into account, as defined with the SunriseOffsetMinutes in hsyco.ini, false to ignore the offset and return the actual sunrise time (Based on civil dusk, ends when the geometric center of the sun reaches 6° below the horizon).

Returns:

- long - the next sunset time in milliseconds since midnight, January 1, 1970 UTC.

haActiveState

```
boolean haActiveState()
```

When HSYCO is installed in a high availability configuration, this method returns true if the system (master or slave) is active, false if it is not active.

If high availability is not configured, the method returns true.

Returns:

- boolean - high availability current state.

haActive

```
static void haActive(boolean active)
```

Force a master HSYCO server to become inactive, or return to the active state.

This method should be called on the master only, and has no effect on the slave server.

Parameters:

- active: boolean - set to true to force a master to become inactive, or false to return it to the active state after the master was forced to become inactive.

sleep

```
void sleep(long millis)
```

Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.

Parameters:

- millis: long - the length of time to sleep in milliseconds.

varGet

```
String varGet(String name)
```

Returns the value of a program variable, or null if no variable with the given name has been defined.

Program variables can be set using varSet() in user.java or in the EVENTS programming environment.

To access a variable used in EVENTS, the name used in varGet() should start with the $ character.

Variables names ending with ! are considered persistent and their values are preserved when HSYCO restarts; normal variables are deleted when HSYCO starts.

Parameters:

- name: String - the variable name. Names are case-insensitive.

Returns:

- String - the current value of the program variable.

varSet

```
void varSet(String name, String value)
```

Sets a program variable to the value parameter.

Variables set with varSet() can be read using varGet() and are also available in the EVENTS programming environment, if the variable name starts with $.

If you need to use variables in Java or JavaScript but don't want these variables to be available in EVENTS, just use names that don't start with the $ character.

Variables names ending with ! are considered persistent and their values are preserved when HSYCO restarts; normal variables are deleted when HSYCO starts.

Parameters:

- name: String - the variable name. Names are case-insensitive
- value: String - the value to be assigned to the program variable.

Driver Control

init

```
boolean init(String name)
```

Must be called once inside the init() callback.

```
public class Driver extends driverBase {

        public boolean init(String name, HashMap<String, String> config) {

                super.init(name); // do not remove
                return true;
        }
}
```

Parameters:

- name: String - the driver's identifier.

Driver I/O

ioSet

```
void ioSet(String name, String value)
```

Writes back a command to this driver.

This method should only be used to push back a command to the command queue, to be processed again after all currently pending commands.

Parameters:

- name: String - the data point name, without the I/O server id prefix
- value: String - the new value.

ioWrite

```
void ioWrite(String name, String value)
```

Writes a new value to a data point.

Events will be triggered only the new value is different from the current one.

Parameters:

- name: String - the data point name, without the I/O server id prefix
- value: String - the new value.

ioWriteNoLog

```
void ioWriteNoLog(String name, String value)
```

Writes a new value to a data point.

Events will be triggered only the new value is different from the current one.

The change is not logged.

Parameters:

- name: String - the data point name, without the I/O server id prefix
- value: String - the new value.

ioWriteNoEvents

```
void ioWriteNoEvents(String name, String value)
```

Writes a new value to a data point.

Events are not triggered.

Parameters:

- name: String - the data point name, without the I/O server id prefix
- value: String - the new value.

ioWriteNoEventsNoLog

```
void ioWriteNoEventsNoLog(String name, String value)
```

Writes a new value to a data point.

Events are not triggered.

The change is not logged.

Parameters:

- name: String - the data point name, without the I/O server id prefix
- value: String - the new value.

ioWriteForced

```
void ioWriteForced(String name, String value)
```

Writes a new value to a data point.

Events are always triggered, even if there is no change to the value.

Parameters:

- name: String - the data point name, without the I/O server id prefix
- value: String - the new value.

ioWriteForcedNoLog

```
void ioWriteForcedNoLog(String name, String value)
```

Writes a new value to a data point.

Events are always triggered, even if there is no change to the value.

The change is not logged.

Parameters:

- name: String - the data point name, without the I/O server id prefix
- value: String - the new value.

Log

errorLog

```
void errorLog(String message)
```

Generates an error message in the daily log file.

Parameters:

- message: String - the text of the error message.

fileLog

```
void fileLog(String pathname, String message)
```

Appends a generic text message at the end of a file.

Parameters:

- pathname: String - the file name. You can specify any path, starting from HSYCO's base directory. If the file or parent directories don't exist, they will be created automatically
- message: String - the message that will be appended at the last line of the text file.

messageLog

```
void messageLog(String message)
```

Generates an information message in the daily log file.

Parameters:

- message: String - the text of the error message.

Mail

sendMail - simple text message

```
int sendMail(String to, String from, String subject, String body)
```

Sends a multipart email message, mixing text and multiple images from cameras, live or recorded.

HSYCO SERVER sends the mail either directly to the recipient's domain mail server if the SmtpName parameter is not defined in hsyco.ini, or using a specific email account with user authentication and traffic encryption if the SMTP server and account parameters are set.

If the email message is sent directly to the recipient's domain mail server, you need to ensure that the mail server accepts mail to the destination address being sent with the from address and the public IP of the HSYCO SERVER.

💡 **HSYCO does not automatically retry sending the message if the destination mail server is not available when the send mail function is called.**

Parameters:

- to: String - a valid email address of the intended recipient. You can optionally specify the destination SMTP server name or IP address by appending :<server name or address to the recipient's email address, for example: john@example.com:192.168.1.1
- from: String - a valid email address to be used as the sender address
- subject: String - the message subject
- body: String - the message text.

Returns:

- int - 1: message sent; 0: message not sent; -1: other errors, including address errors.

sendMail - multi-part message

```
int sendMail(String to, String from, String subject, Vector<String> body)
```

Sends a multipart email message, mixing text and multiple images from cameras, live or recorded.

HSYCO SERVER sends the mail either directly to the recipient's domain mail server if the SmtpName parameter is not defined in hsyco.ini, or using a specific email account with user authentication and traffic encryption if the SMTP server and account parameters are set.

If the email message is sent directly to the recipient's domain mail server, you need to ensure that the mail server accepts mail to the destination address being sent with the from address and the public IP of the HSYCO SERVER.

HSYCO does not automatically retry sending the message if the destination mail server is not available when the send mail function is called.

Parameters:

- to: String - a valid email address of the intended recipient. You can optionally specify the destination SMTP server name or IP address by appending :<server name or address to the recipient's email address, for example: john@example.com:192.168.1.1
- from: String - a valid email address to be used as the sender address
- subject: String - the message subject
- body: Vector<String> - the message body. To send an ordinary text, just fill the vector with the desired text strings. To send an image, add a string to the vector with the following format: "cam:<cameraname>[:<seconds_back>]". For example, "cam:door" sends a live frame from the camera called "door"; "cam:door:2" sends a frame that was recorded two seconds before the last recorded frame; "cam:door:0" sends the last recorded frame. To send files as attachments, use the following format: "file:<file name>". The file path is relative to the HSYCO root directory.

Returns:

- int - 1: message sent; 0: message not sent; -1: other errors, including address errors.

Public Announcement

The audioPlay() functions are used to play recorded audio files or text-to-speech messages.

Audio can be sent to the Web browser, the server's audio line out connector, the internal speaker or audio out line of Axis cameras, and the speaker of SNOM phones or PA devices.

See the Audio and Public Announcement section for additional information.

Audio Destination	Description
speaker	the server's audio line out connector
web	the Web browser's audio output **this feature is supported only on Firefox and Chrome browsers**
axis@camera id	audio sent to an Axis camera, using the camera id defined in the Cameras parameter in hsyco.ini
snom@ip:port	audio sent to SNOM phones and public announcement devices, to the multicast IP address and port specified (the IP address and port should be configured as multicast addresses on each phone)

audioPlay - file

```
int audioPlay(String where, File file)
```

Plays a pre-recorded audio file.

Parameters:

- where: String - the audio destination; see the table above
- File: String - the audio file, relative to HSYCO's root directory.

Returns:

- int - the index of this message in the play back queue if successful, -1 in case of errors.

audioPlay - text to speech

```
int audioPlay(String where, String voice, String text)
```

Converts a text message to audio using the installed text-to-speech engine, and plays the audio.

Parameters:

- where: String - the audio destination; see the table above
- voice: String - the voice name for the text-to-speech engine; see the Audio and Public Announcement section for additional information
- text: String - the text message for text-to-speech conversion.

Returns:

- int - the index of this message in the play back queue if successful, -1 in case of errors.

Serial Communication Ports

closeComm

```
void closeComm(String portname)
```

Closes a serial port connection when the serial port is defined ad a “server” type, and is ignored otherwise.

💡 **When used with serial servers having a fail-over configuration, this method forces a reconnect at the next call of readComm() or writeComm(), so that a switch between the fail-over IPs would happen if the current IP is not responding.**

Parameters:

- portname: String - the name of the communications port, as defined in the CommPorts parameter in hsyco.ini.

readComm

```
String readComm(String portname, int len)
```

Reads a sequence of bytes from a communication port.

The number of bytes read depends on the availability of data in the buffer of the communication port, until the maximum number specified in the len parameter.

💡 **There is a 2 seconds default time-out on reads. If the buffer is empty, this function will wait until the time-out expires before returning.**

If len is set to 0, the function doesn't return data, but it empties the communication port's buffer from older data.

💡 **If both verboseLog and userLog are true, the full trace of received and sent bytes is written to the log file.**

Parameters:

- portname: String - the name of the communications port, as defined in the CommPorts parameter in hsyco.ini
- len: int - maximum number of bytes to be read from the port, or 0 to empty the buffer without reading data.

Returns:

- String - null in case of errors, a string with the hexadecimal representation of the received byte sequence, or an empty string if the buffer of the incoming data is empty. For example "45AB4400BA".

readCommBytes

```
byte[] readCommBytes(String portname, int len)
```

Reads a sequence of bytes from a communication port.

The number of bytes read depends on the availability of data in the buffer of the communication port, until the maximum number specified in the len parameter.

💡 **There is a 2 seconds default time-out on reads. If the buffer is empty, this function will wait until the time-out expires before returning.**

If len is set to 0, the function doesn't return data, but it empties the communication port's buffer from older data.

💡 **If both verboseLog and userLog are true, the full trace of received and sent bytes is written to the log file.**

Parameters:

- portname: String - the name of the communications port, as defined in the CommPorts parameter in hsyco.ini
- len: int - maximum number of bytes to be read from the port, or 0 to empty the buffer without reading data.

Returns:

- byte[] - null in case of errors, a byte array of the received byte sequence, or an empty byte array if the incoming data buffer is empty.

writeComm

```
int writeComm(String portname, String data)
```

Sends a sequence of bytes to a communication port.

💡 **If both verboseLog and userLog are true, the full trace of received and sent bytes is written to the log file.**

Parameters:

- portname: String - the name of the communications port, as defined in the CommPorts parameter in hsyco.ini
- data: String - a string with the hexadecimal representation of the sequence of bytes to be sent. For example "45AB4400BA".

Returns:

- int - the number of bytes sent if successful, or 0 in case of errors.

writeCommBytes

```
int writeCommBytes(String portname, byte[] data)
```

Sends a sequence of bytes to a communication port.

💡 **If both verboseLog and userLog are true, the full trace of received and sent bytes is written to the log file.**

Parameters:

- portname: String - the name of the communications port, as defined in the CommPorts parameter in hsyco.ini
- data: byte[] - the bytes to be sent.

Returns:

- int - the number of bytes sent if successful, or 0 in case of errors.

User Interface

deviceSet

```
void deviceSet(int function, String address, String value, String comment)
```

Used to set the current state of the following user interface device objects: (button), (3button), (buttonicon), (buttonimage) and (dimmer).

💡 **When implementing a driver that directly supports device objects, you should normally call both deviceSet() and ioWrite() to set the user interface objects and also a corresponding I/O data point.**

Parameters:

- function: int - the function name. Use LIGHT, AUTOM, DIMMER or BINARY
- address: String - the address (or identifier) of the user interface device object, without the I/O server id prefix
- value: String - the data point value
- comment: String - a description text that will be written to systemtopo.txt at the end of the initialization process, when still inside the init() callback, or null for normal operation (after the initialization phase).

enableSystemtopoDiscovery

```
void enableSystemtopoDiscovery()
```

Enables the systemtopo.txt update process.

💡 **This method should only be called from the init() callback code, and should be followed by deviceSet() calls before leaving the init() callback.**

uiClear

```
void uiClear(String session)
```

Removes all session specific UI object attributes set with uiSet().

uiClear() is also implicitly called after the client session inactivity timeout expires.

Parameters:

- session: String - session id string that uniquely identifies the client session.

uiSet - session

```
void uiSet(String session, String id, String attr, String value)
```

Changes the dynamic attributes of identified user interface (UI) objects for a specific client session.

💡 **Session specific attributes values will override values set globally.**

id is the unique name that identifies the object, specified in the index.hsm file with the extension !id after the object type.

Parameters:

- session: String - session id string that uniquely identifies the client session.
- id: String - object id
- attr: String - attribute name
- value: String - attribute value.

uiSet - global

```
void uiSet(String id, String attr, String value)
```

Changes the dynamic attributes of the identified user interface (UI) objects.

id is the unique name that identifies the object, specified in the index.hsm file with the extension !id after the object type.

Parameters:

- id: String - object id
- attr: String - attribute name
- value: String - attribute value.

user

```
void user(String name, String param)
```

Triggers a USER event, the execution of the userCommand() JavaScript function and of the userCommand() Java method.

It can be used as a calling mechanism between Java, JavaScript and EVENTS.

Parameters:

- name: String - the name parameter, passed to userCommand() and the USER event
- param: String -the param parameter, passed to userCommand() and the USER event.

Custom I/O Drivers

HSYCO offers a development framework to easily build your own I/O server, written in Java, so you can interface with external systems that are not part of the built-in library of supported systems.

A user-defined I/O server, called “driver” from now on, behaves in the exactly the same way as the standard servers.

The Directory Structure and Naming Convention

Each driver has a dedicated sub-directory under the “drivers” directory in HSYCO’s root directory. The name of that directory is the driver’s name, and should be all lowercase letters [a-z] and numbers [0-9] only.

You should define names carefully, so to avoid naming conflicts with other drivers you may wish to install on the same HSYCO server.

All built-in I/O server names are reserved and should not be used as names for custom drivers.

The driver’s directory must contain at least two files: the Driver.class containing the Java code, and settings.json, used to let the Manager’s Settings application know about your custom driver and its configuration options.

You can have other files as well, for other custom Java classes and additional resources.

Driver.java

A driver is created as a Driver class in package drivers.<driver name>.

It should be a public class that extends driverBase.

The following code is the skeleton for a driver called mydriver.

```
package drivers.mydriver;

import java.util.HashMap;

import com.hsyco.driverBase;

public class Driver extends driverBase {
        public static final int DEFAULTSOCKETPORT = 0;
        public static final int COMMANDSQUEUESIZE = 256;
        public static final boolean SHUTDOWNWHENSLAVE = true;

        private String name = null;

        // called once every time the driver is started.
        // Returning false will fail the initialization procedure, and the driver will not start
        public boolean init(String name, HashMap<String, String> config) {

                super.init(name); // do not remove
                this.name = name;
                return true;
        }

        // called repeatedly while the driver should be alive.
        // Returning false will not update the keep-alive timer, causing the driver to quit after a while.
```

```
        // This call should always return within about 30 seconds, to avoid the keep-alive watch-dog
intervention
        public boolean loop() {

                return true;
        }

        // called once when the driver quits after it has started correctly.
        // Should always return true
        public boolean end() {

                return true;
        }

        // Optional - called on user keys, gui input objects, submit and user action in Events or Java
        public String user(String session, String user, String id, HashMap<String, String> fields) {

                return "";
        }

        // Optional - called for each command that should be processed by the driver
        public void command(String name, String value) {

        }
}
```

As you can see, the code structure is pretty simple. A handful of callback methods are called by the HSYCO core engine during initialization, exit, and for normal operations.

These methods, described in the Custom Drivers Callback Methods API documentation, are where you will write your custom interface code, also using a number of utility methods that are described in the Custom Drivers Command and Utility Methods API documentation.

settings.json

settings.json is a simple text file in JSON format, used to let the Manager's Settings application know about your custom driver and its configuration options.

settings.json is not strictly needed for the driver to function, but without it you would not be able to use the Manager's Settings application for HSYCO configuration.

This is an example of a simple settings.json file.

```
{
        "MYDRIVER": {
                "connection": "ip",
                "auth": "yes",
                "options": {
                        "decimals": {
                                "type": "select",
                                "format": "0|1|2"
                        },
                        "startupevents": {
                                "type": "select",
                                "format": "true|false"
                        },
                        "code": {
                                "type": "input",
                                "format": "\\d+",
                                "key_format": "\\d"
```

```
                        }
                }
        }
}
```

The structure of this file is a JSON object with a single field with the same name of the driver in uppercase letters.

The driver field's value is an object with the following fields:

"connection"

can have the following values:

- "com": the driver communicates through a serial port, thus the I/O Server settings panel will enable the "Comm" field
- "ip": the driver communicates through an IP connection, thus the I/O Server settings panel will enable the "IP" field
- "com,ip": the driver may communicate both via serial port or IP connection. Both fields will be enabled
- "": the driver does not require communication parameters to be set. The "Comm" and "IP" fields will be disabled
- "com?" or "ip?": adding a '?' after "com" or "ip" indicates that the communication parameters are optional.

"comm_type"

this field is required if the "connection" field specifies the use of a serial port. The allowed values are the following:

- "comm": the used port will be selected among one of the entries listed in the Comm Ports section of the Manager's Settings
- "port": the used port will be manually entered. For instance one has to specify the name of the actual communication port (e.g. "tty0")
- "io:<driver_name>": the serial communication occurs through another I/O Server of type <driver_name>, e.g. "io:MODBUSTCP". Thus one will select among a list of defined I/O Servers.

"auth"

can have the following values:

- "no": the authentication paramenters are not required by the driver
- "yes": the driver requires the user and password parameters
- "password": the driver requires only the password parameter.

"is_commport"

this field is optional and defaults to "no". It can have the following values:

- "no": this driver cannot be used as a comm port
- "yes": this driver can be used as a comm port.

"options"

the value of this filed is an object whose fields represent additional parameters for the driver. For each option add a new field properly named and set the value to an object with the following fields and values:

"type"

can have the following values:

- "select": the Settings application will show a select field for the user to chose the value of the option from a list
- "input": the Settings application will show an input field for the user to type the value of the option

"format"

a regex string describing the format of allowed values for this option (e.g. a value of "[-|+]?\\d+" indicates that the value must be a relative integer number). If the type of this option is "select", then the value of this field must be set to the list of possible values, using the format "<val_1>|<val_2>| ... |<val_n>" (e.g. "1|2|3")

"key_format"

to be used for options of type "input". It defines the regex expression used to validate each character entered in the input field. For instance, if an option has a numeric value, the value shall be set to "\\d".

Creating a Driver Bundle

The most effective way to create a single file bundle for a driver is to simply zip the driver's directory.

You will distribute the zip file, and simply install it using the file manager to upload it to the main drivers directory.

The file manager will automatically expand the file in place.

Please note that HSYCO will have to be manually restarted when you upload a new driver or update an existing one, unless the AutoKillFiles configuration parameter is set to explicitly list all relevant files of the driver.

Client JavaScript Extension API

> **WARNING! The client-side JavaScript extensions API is experimental and subject to changes.**

A project can have specific javascript code that runs on the client-side and manages various events. This code has to be written in an index.js file located inside the project's directory.

Events

These are functions executed on specific events.

StartupEvent

```
function StartupEvent()
```

Executed when the interface is first loaded. It's a good place to set up the interface and set variables, for example.

userCommand

```
function userCommand(name, param)
```

Called by user clicks on user buttons: User, UserMini, UserMicro, UserRGB or UserImage objects.

If you want to navigate to a specific page when the button is pressed (it would be like pressing a Link object), return a string starting with "page:" followed by the page name; in this case, userCommand() will be called again when that popup or page is closed, with "/close" appended to param.

Parameters:

- name: string - the name field of the user object
- param: string - the param of the user object

Returns:

- null: proceed to send the request to server as it is
- "": assume the event was resolved. Don't send the request to the server
- "page:page name": navigate to the specified page
- "page:back": navigate to the previous page
- "page:forward": navigate to the next page
- "page:close": if the user button that generated the call has an open popup linked to it, close it
- "error": error. Don't send the request to the server
- an object, to specify new name and param values to be sent to the server

```
{name:"new name", param:"new param"}
```

userSubmit

```
function userSubmit(name, param)
```

Called on every submit request (from Submit objects). Parameters and return values are the same as the userCommand function.

uiEvent

```
function uiEvent(id, attr, value)
```

Executed on every UISet received from the server, intercepts a UISet and allows to change it, prevent its execution or execute any other custom code (e.g. set variables).

Parameters:

- id: string - id of the ui object
- attr: string - attribute name
- value: string - new value being set

Returns:

- null: discard the UISet
- <string>: set a new value for the UISet. To keep the UISet as it is, return the initial value

Functions

varSet

```
varSet(name,value)
```

Sets the value of a variable. If the name begins with **!**, the variable will be stored in the browser's cache, so that it'll available even after reloading the page or closing and reopening the browser.

varGet

```
varGet(name)
```

Gets the value of a variable previously set with varSet.

uiSet

```
uiSet(id,attr,value)
```

Sets a UI Attribute. See Project, Page or UI Objects list of attributes.

webLog

```
webLog(string)
```

Adds a line to the daily log file, viewable from the Manager's Log Viewer.

Examples

Using varget and varset

In this example we declare a persistent variable (stored in the browser's local storage) the first time any user button is pressed, we increment it each time up to 10 and then we reset it to 0. When the variable equals 3, the page "page1" is shown. When the variable equals 7, the request isn't forwarded to the server.

```
function userCommand(name, param) {
        var a = varGet("!counter"); // get the variable value

        if (!a) // first time and when a is 0
                a = 1;
        else if (a < 10)
                a++;
        else // reset
                a = 0;

        varSet("!counter",a); // set the variable value

        webLog("usercommand:"+name+","+param+". Counter:"+a)
        if (a == 3)
                return "page:page1";
        else if (a == 7)
                return "";
        else
                return null;
}
```

User Object

One or more User objects can be declared to send requests from the client to the server.

```
var user = new User();

function StartupEvent() {
        user.setOnLoadedEvent(onUserLoaded);
        user.setOnErrorEvent(onUserError);
        // send reques
        user.send("myname","myparam");

        varSet("myvar1","myvalue");
        varSet("myvar2",5.3);
        varSet("myvar3",[1,2,3]);
        varSet("myvar4",{"a":1,"b":2});
}
```

Methods

.send(name, param)

sends a virtualremote (name and param will be url encoded)

.setMaxWaitingTime(msec)

set max waiting time in msec. If a request exceeds this time, it will be aborted (and fire an onError event, if set).

.setMaxRetries(n)

set max number of retry attempts. If a a request fails, it will keep on retrying until it succeeds or the max number of retries is reached. Before retrying it will fire an onRetry event, if set. If n is 0, and retryOnErrorDelay is not 0, it will keep on retrying forever (it will never fire an onError event, if set).

.setRetryOnErrorDelay(msec)

set msec of delay before retrying after a request has failed. If msec is 0 it won't retry (and will fire an onError event, if set).

.setOnLoadedEvent(f)

f is a function that is called when the request is successful. To get the response text or xml, use .getResponseText() and .getResponseXML()

.setOnErrorEvent(f)

f is a function called when the request fails. f is called with an error code parameter: function(errCode) error codes are enumerated as .ERROR_GENERAL, .ERROR_CONNECTION ...

.setOnRetryEvent(f)

f is a function that is called when the last request failed, before retrying. To get the response text or xml, use .getResponseText() and .getResponseXML()

.getResponseText()

get the last response text

.getResponseXML()

get the last response XML

.free()

free the memory and reset the behaviour. It will be reinitialized if any method is called.

Error codes

- **.ERROR_GENERAL** : general error
- **.ERROR_CONNECTION** : no connection
- **.ERROR_MAXWAITTIME** : max waiting time exceeded
- **.ERROR_LOGOUT** : client is logged out
- **.ERROR_LOCK** : client is locked out
- **.ERROR_NOACCESS** : request returned noaccess

Examples

Try one time, then fire onLoaded or onError

```
var user = new User();

function StartupEvent() {
user.setOnLoadedEvent(onUserLoaded);
user.setOnError(onUserError);
  // send request
  user.send("myname","myparam");
}

function onUserLoaded() {
  webLog("loaded "+user.getResponseText());
  // do something with the response
  user.free(); // won't be using it again
}
function onUserError(errCode) {
  webLog("error");
  user.free(); // won't be using it again
}
```

Retry forever, until it succeeds

```
var user = new User();

function StartupEvent() {
  user.setOnLoadedEvent(onUserLoaded);
  user.setRetryOnErrorDelay(1000); // wait one second before retrying on error
  // send request
  user.send("myname","myparam");
}

function onUserLoaded() {
  webLog("loaded "+user.getResponseText());
  // do something with the response
  user.free(); // won't be using it again
}
```

Retry 3 times, then fire onError

```
var user = new User();

function StartupEvent() {
  // init user object
  user.setRetryOnErrorDelay(100); // retry almost immediately on error
  user.setMaxRetries(3); // retry 3 times, then fire error
  user.setOnLoadedEvent(onUserLoaded);
  user.setOnError(onUserError);
  // send request
  user.send("myname","myparam");
}

function onUserLoaded() {
  webLog("loaded "+user.getResponseText());
  // do something with the response
  user.free(); // won't be using it again
}

function onUserError(errCode) {
  webLog("error, already tried 3 times");
  if (errCode == user.ERROR_MAXWAITTIME)
  webLog("the last request failed because it exceeded the waiting time");
  user.free(); // won't be using it again
}
```

Plugins Applications

In the User Interface and Programming chapters we have discussed how to create user interface projects to customize the Web-based GUI and how to develop custom logic using EVENTS, JavaScript and Java. This is the correct approach to the development of one-off customized solutions.

But in some cases you may want to develop general-purpose applications with the intent of reusing them on multiple installations. In this case you would need to bundle both the user interface design and the application code in one package that is easier to distribute and install.

This is exactly what the HSYCO's plugins framework has been designed for.

There is not much new to learn on order to create a plugin.

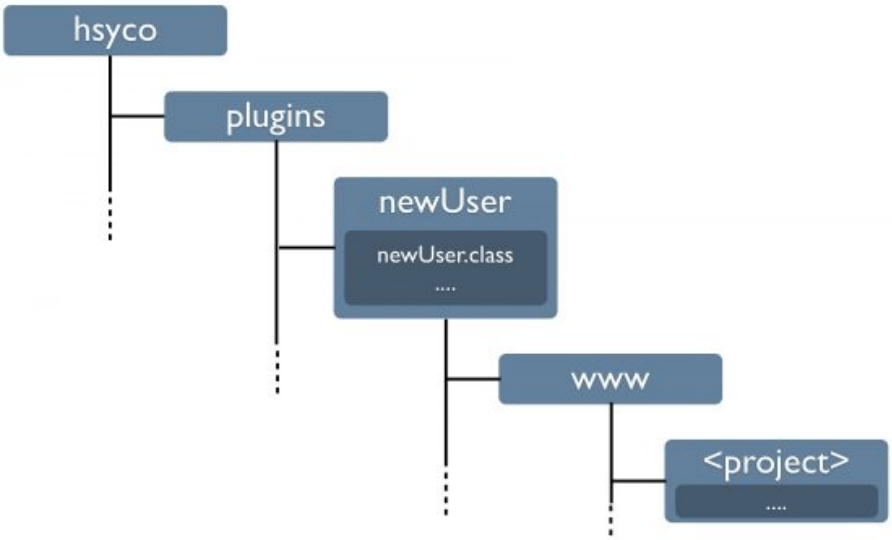
Plugins are in fact based on normal index.hsm projects and the standard Java API you use in user.java. Only difference is they are bundled in a directory structure that simplifies isolation and deployment of the plugin package. You must also adhere to a specific naming convention for files and sub-directories within your plugin package.

The Directory Structure and Naming Convention

Each plugin has a dedicated sub-directory under the "plugins" directory in HSYCO's root directory.

The name of that directory is the plugin name.

You should define names carefully, so to avoid naming conflicts with other plugins you may wish to install on the same HSYCO server.



Besides being the directory name, the plugin name is not relevant for the end user, so even using a complex, long name will not negatively impact the user experience.

A plugin is usually a combination of program logic and user interface, but you can also have plugins that have just a user interface and no server-side logic, or plugins with no user interface, just Java code reacting to system or field events.

Java

You will write your Java code directly in your plugin directory.

This means that your Java code will have to be in a package named plugins.<directory name>.

The class where you declare your callback code, just like code in user.java, must be public and extend the userBase class.

For example, let's create a Java class for a plugin named newUser. The callback class file must be declared to be part of the plugins.newUser package, like this:

```
package plugins.newUser;

import com.hsyco.*;

public class newUser extends userBase {

        public static void StartupEvent() throws Exception {
        }

        public static void TimeEvent(long time) throws Exception {
        }
}
```

The public class name in this example has been named newUser, just like the plugin's directory name.

Using the same name is considered a good practice, but is not strictly enforced.

The HSYCO engine looks at all .class files inside each plugin directory, searching one public class with the correct package name and extending userBase.

It will then look in that class only for callback methods.

You can freely add other class files in your plugin directory, as long as they don't extend userBase.

Besides the package and naming conventions, you can use the same callback and command methods used in user.java.

To check that your plugin's callbacks are properly defined and discovered by HSYCO, enable the verboseLog option in the system's general settings, then check the log file. You should see a line for each callback that was correctly registered. Using the example above, the log will be:

```
2014.02.18 13:19:05.074 - Plugin method StartupEvent() in class plugins.newUser.newUser registered
2014.02.18 13:19:05.077 - Plugin method TimeEvent() in class plugins.newUser.newUser registered
```

User Interface

The plugin's user interface consists or one or more regular projects, just like the projects under the main www directory.

In fact, for your convenience, you should use the project editor to create your user interface, then simply move the project directory under the main www to a www directory inside your plugin's directory.

The project directory will have the index.hsm file and optionally other required files and directories.

The project's name or names will be used in the URL request to access the user interface, exactly like regular projects.

Precedence

Plugins will coexist with other plugins and with regular projects and application logic in EVENTS, JavaScript and Java.

If a project name exists in the main www directory with the same name of a plugin's project, it will have priority over the plugin's project.

The user will see the main project in www, and not the plugin's user interface.

On the other hand, HSYCO will always execute all defined Java callbacks, in all plugins and in user.java.

The execution order is not guaranteed and your application should be designed to execute properly independently of the execution order.

Anyways, there are a few callback methods that return values, like userCommand(), userSubmit() and a few others.

In these cases HSYCO will always execute the user.java call first, then all plugins' callbacks, but will use their return value only if the user.java method returns a null or an empty string.

The general rule for plugin callbacks that return values is that they should always return null if they don't need to intercept the specific event that triggered the call.

This way multiple plugins will be able to coexist without blocking calls to others.

Scope

Because plugins are often designed as self-contained applications, it could be a good design practice to use application scope sessions for the user interface interaction. Read the User Interface section of the Java Command and Utility Methods API chapter for additional information on scope sessions.

When applicable, this will result in more efficient communication between the browser client and the HSYCO server.

Also, user interface objects identifiers should follow a naming convention using a prefix that uniquely identifies the plugin, to avoid potential unwanted conflicts with identifiers used by other applications.

Creating a Plugin Bundle

The most efficient way to create a single file bundle for a plugin is to simply zip the plugin's directory.

You will distribute the zip file, and simply install it using the file manager to upload it to the main plugins directory.

The file manager will automatically expand the file in place.

💡 **Please note that HSYCO will have to be manually restarted when you upload a new plugin or update an existing one, unless the AutoKillFiles configuration parameter is set to explicitly list all relevant files of the plugin.**

Using the Embedded SQL Database Engine

HSYCO embeds HSQLDB (HyperSQL Database), a powerful SQL database engine you can use in your applications via the standard JDBC API.

HSQLDB (HyperSQL DataBase) is a leading SQL relational database engine written in Java. It has a JDBC driver and supports nearly full ANSI-92 SQL plus many SQL:2008 enhancements. The HSQLDB documentation is available at: http://hsqldb.org.

The JDBC API is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases. The JDBC API provides a call-level API for SQL-based database access. The JDBC documentation is available at: http://java.sun.com/products/jdbc/overview.html.

HSQLDB offers a small, fast, multithreaded and transactional database engine which offers in-memory and disk-based tables and supports embedded and server modes.

HSQLDB is directly integrated in the core HSYCO package, so there is no need to add any additional jar file to the HSYCO classpath.

The database that HSYCO uses is called “hsyco” and its files are saved in the data/ directory under the base directory, where hsyco.ini and all other configuration files are also located.

We consider the data/ directory a reserved HSYCO path and discourage the use of it for storing user’s databases or in general for any customized purpose.

We suggest using userdata/ as the general base path for all customized file storage needs, including the location of your own HSQLDB databases.

HSYCO uses HSQLDB to store persistent variables.

It is very likely that we will use it for other purposes in the future, so it is important that you follow a few simple guidelines to avoid potential conflicts in future releases.

Moreover, the resources load of the SQL engine has a direct impact on the performance and stability of HSYCO SERVER.

Connect

First of all, define the imports:

```
import java.sql.*;
```

This connects to a DB called mydb in the userdata/ directory, with a write delay of 100 milliseconds. You can of course use different connection parameters:

```
static Connection dbConnection =
DriverManager.getConnection("jdbc:hsqldb:file:userdata/mydb;hsqldb.write_delay_millis=100", "user",
"password");
dbConnection.setAutoCommit(true);
```

You should normally set the auto-commit mode if you don’t need transactions.

Disconnect

```
try {
        dbConnection.createStatement().execute("shutdown");
        variablesDatabaseConnection.close();
} catch (Exception e) {}
```

This is the normal shutdown command, followed by a call to close the database connection. Use this sequence if you are not going to access again the database for some time.

You should also consider using a special form of closing the database, using the SHUTDOWN COMPACT command. This should be done periodically, to clean up and minimize the size of data files.

Tables

HyperSQL defines three types if database tables, according to the way the data is stored. These are memory tables, cached tables and text tables.

Memory tables are the default type when the CREATE TABLE command is used.

Memory tables are persistent, but saving data to the files and reading back when the database is opened becomes very time consuming if the database is large.

Cached tables are created with the CREATE CACHED TABLE command. Only part of their data or indexes is held in memory, which is good for handling large tables. The database engine takes less time to start up. The disadvantage of cached tables is a reduction in speed. Do not use cached tables if your data set is relatively small.

The following example shows how to create a table:

```
try {
        dbConnection.createStatement().execute("create table mytable (id varchar(255) not null unique, value
varchar(65536))");
} catch (Exception e) {}
```

Statement

Tables can be managed to execute query, insert and update operations. For this purpose, use the Statement and PreparedStatement classes.

A java.sql.Statement object is used to execute queries and data change statements. A java.sql.Statement can be reused to execute a different statement each time.

A java.sql.PreparedStatement object is used to execute a single statement repeatedly. The SQL statement usually contains parameters, which can be set to new values before each reuse.

When a PreparedStatement object is created, the engine keeps the compiled SQL statement for reuse, until the PreparedStatement object is closed. As a result, repeated use of a PreparedStatement is much faster than using a Statement object.

Query

The following examples show how to make a query, an insert and an update operation in a database table.

```
try {
        PreparedStatement stmt = dbConnection.prepareStatement("select value from mytable where id = ?");
        stmt.setString(1, id);
        ResultSet rs = stmt.executeQuery();
        if (rs.next()) {
                String value = rs.getString(1);
                rs.close();
                return value;
        } else {
                return null;
        }
} catch (Exception e) {}
```

Introduction to I/O Servers

An I/O Server is the logical representation of a field system integrated in HSYCO. A field system can be, for instance, a security system, a BUS network or even something more abstract as weather information gathered from the Internet.

All I/O Servers provides for **datapoints** through which they receive commands and/or trigger events. That is, you can use these datapoints in your programming logic or user interface to send commands to the field system or to act upon its changes.

The first step when integrating a new system is to configure the corresponding I/O Server. This is accomplished using the I/O Servers section of the Settings application accessible through the Manager.

From here you can define a new I/O Server, set its ID and specify the communication parameters and its options. Details of this settings are provided in the I/O Servers application notes for each integrated system.

The ID of the I/O Server is used to reference it in the HSYCO environment.

Using Datapoints

A datapoint is univocally identified by the combination of the I/O Server ID and the ID of the datapoint, using the format <io_server_id>.<datapoint_id>.

All the available datapoints for an I/O Server are detailed in its application note. A datapoint can be *readable* (R) and/or *writable* (W).

Let's introduce these concepts with an example. Assume an I/O Server has the following datapoints table:

ID	Value	R/W	Description
foo	1	R	...
	2	R	...
	reset	W	...
bar	a	R	...
		W	...
	b	R	...
		W	...
	unknown	R	...

This table specifies that this I/O Server has 2 datapoints: "foo" and "bar". If we read the "foo" datapoint, we can see 2 possible values: "1" or "2", further we can write only the value "reset" to it. On the other hand, the "bar" datapoint can be read in three different states ("a", "b" or "unknown") and we can set it to the values "a" and "b".

Assuming we assigned the ID "ios" to this I/O Server, we will reference these datapoints as `ios.foo` and `ios.bar`.

> If a datapoint is both readable and writable, writing a value to it doesn't mean that a subsequent reading will yield the same value, it simply means that we are issuing a command to the I/O Server. For instance, after setting a datapoint

> representing a light to "on", we will continue reading its value as "off" until the field system notifies HSYCO that the light is actually on.

Readable Datapoints

The main use for readable datapoints is to execute actions when they trigger an event. This can be done in Events Programming using the "IO" event keyword or in JavaScript or Java Programming using the callback method `IOEvent`.

Events example:

```
IO ios.foo = 1 : ...
```

JavaScript example:

```
function IOEvent(name, value) : {
        if (name == 'ios.foo' && value == '1') {
                ...
        }
}
```

Java example:

```
static void IOEvent(String name, String value) {
        if (name.equals("ios.foo") && value.equals("1")) {
                ...
        }
}
```

Events are generally triggered whenever the readable datapoint's value changes, but some datapoints may trigger events even if the value is unchanged. This is done to better fit the behavior of some field systems and is specified in the I/O Server application note.

The value of a readable datapoint can also be read at any time.

In Events, you can assign the value to variables in an action:

```
... : $var = IO ios.bar, $var * 10, LOG = $var
```

or directly pass it as value to other actions:

```
... : UISET object.value = IO ios.bar
... : DATALOGGER mydat = IO ios.foo
```

In JavaScript and Java you can recall the value of a readable datapoint using the utility method `ioGet`:

```
...
x = ioGet("ios.foo");
myFunction(x);
...
```

Writable Datapoints

Writable datapoint are used to send commands to the field system.

Using Events Programming you can write a datapoint using the "IO" action keyword, e.g.:

```
... : IO ios.bar = a
```

Using JavaScript or Java Programming, you can call the method `ioSet`, e.g.:

```
...
ioSet("ios.bar", "a");
...
```

The commands are queued and executed as soon as the I/O Server is ready to process them.

User Interface

Some I/O Servers provide for the possibility to send commands directly from the user interface and to have UI Objects showing the field system status without writing any programming logic. In an I/O Server application note the section "USER Commands" lists all the possible commands it is able to process from the user interface, while the section "UISET Actions" shows the actions it performs on UI Objects to show the status of the field system.

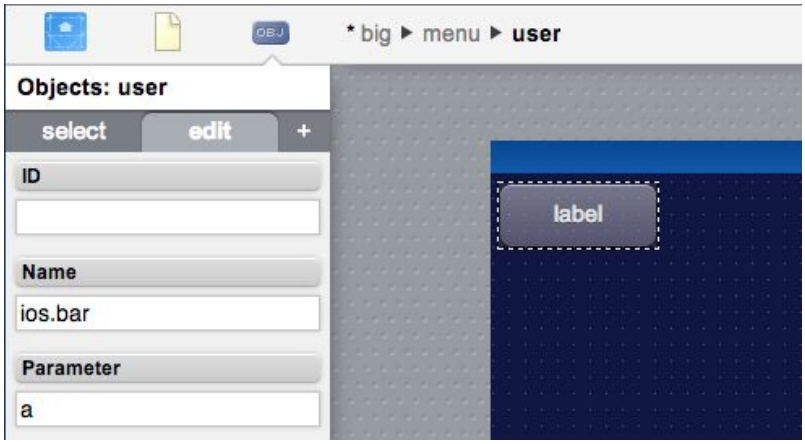
USER Commands

To send a command to an I/O Server from the interface use any object able to send USER commands (e.g. User buttons, UserImages, ...) and set its parameters according to the table in the I/O Server application note.

Consider this example table:

Name	Param	Action
foo	reset	...
bar	a	...
	b	...

We can, for instance, send the command "bar = a" with a User button:



All we have to do is set the "Name" field of the User button to the name specified in the table (as usual, prefixed by the I/O Server ID, i.e. "ios.bar") and the "Parameter" field to the value we want to send. No further programming is needed.

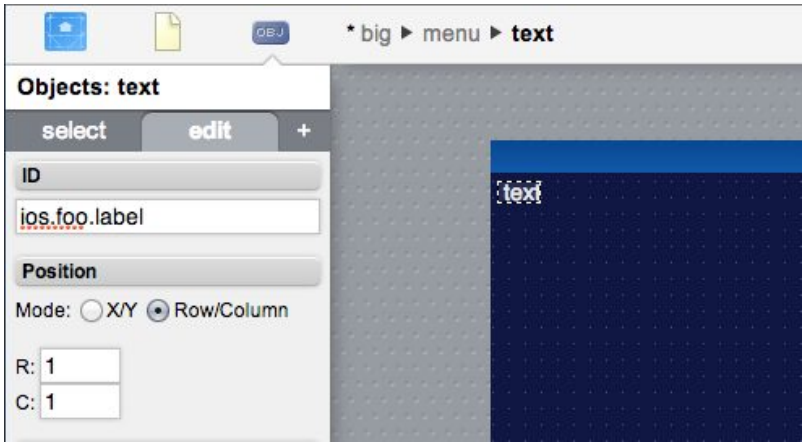
UISET Actions

To show the status of the field system, many I/O Servers perform UISET actions to change attributes of specific UI objects.

Consider this example table:

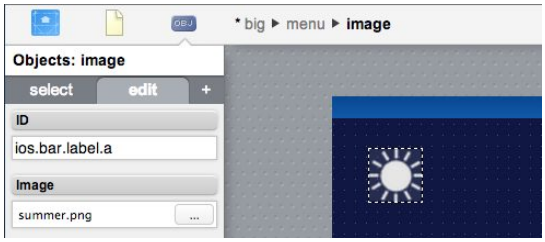
ID	Attribute	Set to	
foo.label	value	the value of the datapoint foo	
bar.label.a	visible	true	when bar's value is "a"
		false	when bar's value is not "a"
bar.label.b	visible	true	when bar's value is "b"
		false	when bar's value is not "b"

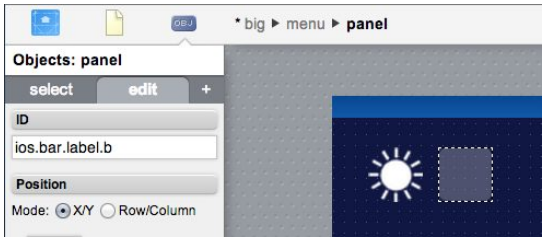
According to this table, we can use any object that has the attribute "value" to see the value of the datapoint foo, for instance we can use a Text object:



All we have to do is setting the ID of the object as specified in the table (prefixed with the I/O Server ID), e.g. "ios.foo.label". Without any other programming we have a label in our interface that show "1" or "2" according to the value of the "foo" datapoint.

Further, always according to the above table, we can have objects appear and disappear (i.e. attribute "visible" set to true or false) from our interface depending on the state of datapoint "bar":





The Image object will be visible when datapoint bar has value "a" and invisible otherwise, while the Panel object will be visible when datapoint bar has value "b" and invisible otherwise.

I/O Servers

This page lists all the available I/O Servers provided by HSYCO and links to their application note.

For an introduction on I/O Servers and how to use the application notes refer to Introduction to I/O Servers.

	Airzone	Airzone HVAC control system
	Arduino	Arduino Uno, Arduino Ethernet, Arduino Due and other compatible boards
	Arteco	Arteco Everywhere general-purpose digital input-output Ethernet controller
	Aton	Aton AH66T multi-room audio system
	Ats	Aritech Master ATS security panel
	AxisDecoder	Control and streaming to the AXIS Video Decoder
	BACnet	BACnet/IP client
	Bentel	Bentel KYO 320 security panel
	CSx75	Aritech Comfort CSx75 security panel
	Contatto	Duemmegi Contatto proprietary bus building automation system
	Daikin	Daikin's VRF systems with Intelligent Controller
	Domino	Duemmegi Domino proprietary bus building automation system
	Dummy	a special I/O Server that simulates lighting and automation devices and can be used for demonstrations, prototyping and as a virtual I/O Server to implement complex control logic
	Elmo	EL.MO. ETR security panel
	Elsner	Elsner's Modbus version of the P03/3 weather station
	Enttec	Enttec DMX USB PRO adapter that allows to control any DMX-512 standard bus.
	ESC/VP	Epson's Networked projectors that support the ESC/VP control protocol
	Fp2000	GE UTC Interlogix fire panels based on the FP2000 protocol

	Giacomini	GKBUS GiakoClima HVAC system
	GSM	standard GSM modems for SMS messages and voice call events
	Galaxy	Honeywell Galaxy Dimension security panel
	Guardall	Guardall QX32i, PX80 and PX500 security panels
	Hephais	Hephais 1600 fire panel with Modbus interface
	Hidedgesolo	HID's Edge Solo stand-alone networked EdgeReader access control solution
	HSYCO	Access remote HSYCO servers data-points
	HwgIO	HW group I/O Controller
	HwgMonitor	HW group monitoring devices
	Inim	Inim SmartLiving 515, 1050 and 10100 security panels
	KNX	KNX/EIB protocol via standard KNXnet/IP gateways
	Mitsubishi	Mitsubishi's VRV systems that support the M-NET proprietary protocol (G-50, GB-50, AG-150)
	Modbus	HSYCO acts as a Modbus TCP client or Modbus RTU master
	Modbus Server	a Modbus TCP/IP server, allowing external Modbus clients to connect to HSYCO and perform any standard Modbus function
	MyHome	BTicino/Legrand My Home proprietary bus home automation system
	Notifier	AM-2000 AM-4000 AM-6000 fire panels with Modbus interface
	Nuvo	NuVo multi-room audio
	Paradox	Paradox EVO 48/192 security panels
	PowerOne	Aurora's PowerOne inverters with support for Aurora proprietary communication protocol
	RayCONTROL	HVAC control system by FCC Planterm
	SamsungTV	Samsung Smart TV
	Sip	SIP VoIP Client

	SSH	SSH Client for remote shell and secure file copy
	System	HSYCO Server performance and system monitor
	Tecnoalarm	Tecnoalarm Tecno-Out security panel
	Tridonic	Dali protocol via the Tridonic SCI2 interface module
	WXonline	Internet-based weather conditions and forecast data acquisition using Yahoo!'s data
	Wyrestorm	Wyrestorm MX-0808-PP, MX-0606-PP and MX-0404-PP HDMI matrix
	Yamaha RX-V	Yamaha's RX-Vx067 family of networked A/V receivers

Airzone

Airzone is a control framework for ducted air conditioning systems which allows for the control of several zones independently.

This I/O server provides support for this Airzone systems: INNOBUS, FLEXA, ANTREE and CEN.

Communication

Airzone communicates with HSYCO through a modbus connection employing a RS-485 gateway.

RS-485 parameters:

Baud rate	19200
Data bits	8
Stop bit	1
Parity	even
Flow control	none

HSYCO Configuration

Add an AIRZONE I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: the modbus gateway the device is connected to and its modbus address (separated by "/"). You can both specify the IP address of the gateway or the name of the corresponding CommPort.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system

pollinterval	20	n > 0	the data acquisition interval, in seconds
version	1	1	use protocol version 1
		2	use protocol version 2

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
z<n>.name[Note 2]	<text>	R	the name of zone <n> is set to <text>
power	0	R	the system is off
		W	turn off the system
	1	R	the system is on
		W	turn on the system
setpoint.temp	<val>	W	set the setpoint temperature of all the zones to <val>
temp.unit	c	R	the system unit for the temperature is set to °C/10
	f	R	the system unit for the temperature is set to °F
mode	stop	RW	read/write the working mode of the system
	cold		
	heatpump		
	ventilation		
	heatingfloor		
	combheating		
	dehumid[Note 2]		
mode.eco[Note 1]	0	W	set the Eco-Sleep mode off on all the zones
	1	W	set the Eco-Sleep mode on on all the zones

mode.control	manual	W	set the control mode of all the zones to manual
	automatic	W	set the control mode of all the zones to automatic
mode.hold	0	W	disable the touch hold of the system
	1	W	enable the touch hold of the system
mode.fanspeed	auto	R	the fancoil fan speed of the system is set to automatic
		W	set the fancoil fan speed of the system to automatic
	1	R	the fancoil fan speed of the system is set to speed 1
		W	set the fancoil fan speed of the system to speed 1
	2	R	the fancoil fan speed of the system is set to speed 2
		W	set the fancoil fan speed of the system to speed 2
	3	R	the fancoil fan speed of the system is set to speed 3
		W	set the fancoil fan speed of the system to speed 3
	flip	W	cycle on the fancoil fan speed status following the order: auto - 1 - 2 - 3 - auto ...
mode.sleep[Note 2]	off	W	set sleep mode to off on all zones
	30	W	set sleep mode to 30 minutes on all zone
	60	W	set sleep mode to 60 minutes on all zones
	90	W	set sleep mode to 90 minutes on all zones
	flip	W	cycle on the sleep mode status following the order: off - 30 - 60 - 90 - off ...
config.motregulationtype[Note 1]	on/off	W	set the motorization regulation type of all the zones to "on/off"

	proportional	W	set the motorization regulation type of all the zones to "proportional"
config.motcontrolled[Note 1]	0	W	disable the control of the motorization on all the zones
	1	W	enable the control of the motorization on all the zones
config.ac[Note 2]	0	W	disable air conditioning on all zones
	1	W	enable air conditioning on all zones
config.heatingfloorctrl	0	W	disable the heating floor controller on all zones
	1	W	enable the heating floor controller on all zones
config.angle.heating	90	W	write the intelligent motorized grille opening angle on heating mode (expressed in °) on all the zones
	50		
	45		
	40		
config.angle.cooling	90	W	write the intelligent motorized grille opening angle on cooling mode (expressed in °) on all the zones
	50		
	45		
	40		
config.minairdamper	0	W	disabled the minimum air damper control on all the zones
	1	W	enabled the minimum air damper control on all the zones
config.probe	off	W	set the local probe to off on all zones
	remote	W	set the remote probe as input on all zones
	floor	W	set the probe as electrical heating floor protection on all zones
config.ecoinput	disabled	W	disable the Eco-Sleep activation input on all zones
	open	W	enable the Eco-Sleep activation input in “normally open” mode on all zones

	closed	W	enable the Eco-Sleep activation input in “normally closed” mode on all zones
config.forcedoffinput[Note 1]	disabled	W	disable the local forced off input on all zones
	open	W	enable the local forced off input in “normally open” mode on all zones
	closed	W	enable the local forced off input in “normally closed” mode on all zones
z<n>.power	0	R	the zone is off
		W	turn off the zone
	1	R	the zone is on
		W	turn on the zone
	flip	W	switch the status of the zone
z<n>.setpoint.temp	<val>	R	the setpoint temperature of this zone is set to <val>
		W	set the setpoint temperature of the zone to <val>
	up	W	increase the setpoint temperature of 0.5 °C or 1°F depending on the system settings
	down	W	decrease the setpoint temperature of 0.5 °C or 1°F depending on the system settings
z<n>.localtemp	<val>	R	the local temperature is <val>
z<n>.remotetemp	<val>	R	the temperature measured by the remote probe is <val>
z<n>.temp	<val>	R	the temperature measured by the input probe of this zone is <val>. For protocol version 1, if the 'z<n>.config.probe' datapoint has value 'remote', then <val> corresponds to the value of the 'z<n>.remotetemp' datapoint, otherwise to the value of the 'z<n>.localtemp' datapoint. For protocol version 2, if the 'z<n>.config.probe' datapoint has value '1', then <val> corresponds to the value of the 'z<n>.remotetemp' datapoint,

			otherwise to the value of the 'z<n>.localtemp' datapoint.
z<n>.mode.eco[Note 1]	0	R	the Eco-Sleep mode is disabled for zone <n>
		W	set the Eco-Sleep mode off for zone <n>
	1	R	the Eco-Sleep mode is active for zone <n>
		W	set the Eco-Sleep mode on for zone <n>
	flip	W	switch the Eco-Sleep mode status of zone <n>
z<n>.mode.control	manual	R	the control mode for zone <n> is set to manual
		W	set the control mode of zone <n> to manual
	automatic	R	the control mode for zone <n> is set to automatic
		W	set the control mode of zone <n> to automatic
	flip	W	switch the control mode of zone <n>
z<n>.mode.hold	0	R	the touch hold is disabled on zone <n>
		W	disable the touch hold of zone <n>
	1	R	the touch hold is enabled on zone <n>
		W	enable the touch hold of zone <n>
z<n>.mode.fanspeed	auto	R	the fancoil fan speed of zone <n> is set to automatic
		W	set the fancoil fan speed of zone <n> to automatic
	1	R	the fancoil fan speed of zone <n> is set to speed 1
		W	set the fancoil fan speed of zone <n> to speed 1
	2	R	the fancoil fan speed of zone <n> is set to speed 2

		W	set the fancoil fan speed of zone <n> to speed 2
	3	R	the fancoil fan speed of zone <n> is set to speed 3
		W	set the fancoil fan speed of zone <n> to speed 3
	flip	W	cycle on the fancoil fan speed status following the order: auto - 1 - 2 - 3 - auto ...
z<n>.mode.fancoilpower[Note 1]	0	R	the activation status of the fancoil of this zone if set to off
	1	R	the activation status of the fancoil of this zone if set to on
z<n>.mode.sleep[Note 2]	off	R	sleep mode is off on zone <n>
		W	set sleep mode to off on zone <n>
	30	R	sleep mode is set to 30 minutes on zone <n>
		W	set sleep mode to 30 minutes on zone <n>
	60	R	sleep mode is set to 60 minutes on zone <n>
		W	set sleep mode to 60 minutes on zone <n>
	90	R	sleep mode is set to 90 minutes on zone <n>
		W	set sleep mode to 90 minutes on zone <n>
	flip	W	cycle on the sleep mode status following the order: off - 30 - 60 - 90 - off ...
z<n>.mode.prog[Note 2]	0	R	time programming is disabled on this zone
	1	R	time programming is enabled on this zone
z<n>.mode	stop	RW	read/write the working mode of zone <n>. It is possible to write the working mode of a zone only if it is configured as master (setting the working mode of a master zone will also change the mode of all the zones connected to the same controller)
	cold		
	heatpump		
	ventilation		

	heatingfloor		
	combheating		
	dehumid[Note 2]		
z<n>.config.master	0	R	zone <n> is not configured as master
		W	set zone <n> as not master
	1	R	zone <n> is configured as master
		W	set zone <n> as master
z<n>.config.motregulationtype[Note 1]	on/off	R	the motorization regulation type of zone <n> is set to "on/off"
		W	set the motorization regulation type of zone <n> to "on/off"
	proportional	R	the motorization regulation type of zone <n> is set to "proportional"
		W	set the motorization regulation type of zone <n> to "proportional"
z<n>.config.motcontrolled[Note 1]	0	R	the control of the motorization is disabled on zone <n>
		W	disable the control of the motorization on zone <n>
	1	R	the control of the motorization is enabled on zone <n>
		W	enable the control of the motorization on zone <n>
z<n>.config.ac[Note 2]	0	R	air conditioning is disabled on zone <n>
		W	disable air conditioning on zone <n>
	1	R	air conditioning is enabled on zone <n>
		W	enable air conditioning on zone <n>
z<n>.config.heatingfloorctrl	0	R	the heating floor controller is disabled on zone <n>
		W	disable the heating floor controller on zone <n>

	1	R	the heating floor controller is enabled on zone <n>
		W	enable the heating floor controller on zone <n>
z<n>.config.antifreeze[Note 2]	0	R	antifreeze function not configured on zone <n>
	1	R	antifreeze function configured on zone <n>
z<n>.config.angle.heating	90	RW	read/write the intelligent motorized grille opening angle on heating mode (expressed in °)
	50		
	45		
	40		
z<n>.config.angle.cooling	90	RW	read/write the intelligent motorized grille opening angle on cooling mode (expressed in °)
	50		
	45		
	40		
z<n>.config.minairdamper	0	R	the minimum air damper control on zone <n> is disabled
		W	disabled the minimum air damper control on zone <n>
	1	R	the minimum air damper control on zone <n> is enabled
		W	enabled the minimum air damper control on zone <n>
z<n>.config.probe	off[Note 1]	R	the local probe is off on zone <n>
		W	set the local probe to off on zone <n>
	remote[Note 1]	R	the remote probe temperature is set to be the input temperature on zone <n>
		W	set the remote probe as input on zone <n>
	floor[Note 1]	R	the probe is used as electrical heating floor protection probe on zone <n>
		W	set the probe as electrical heating floor protection on zone <n>

	0[Note 2]	R	remote probe not connected on zone <n>
	1[Note 2]	R	remote probe connected on zone <n>
z<n>.config.ecoinput	disabled	R	the Eco-Sleep activation input is disabled on zone <n>
		W	disable the Eco-Sleep activation input on zone <n>
	open	R	the Eco-Sleep activation input is enabled and set to “normally open” mode on zone <n>
		W	enable the Eco-Sleep activation input in “normally open” mode on zone <n>
	closed	R	the Eco-Sleep activation input is enabled and set to “normally closed” mode on zone <n>
		W	enable the Eco-Sleep activation input in “normally closed” mode on zone <n>
z<n>.config.forcedoffinput[Note 1]	disabled	R	the local forced off input is disabled on zone <n>
		W	disable the local forced off input on zone <n>
	open	R	the local forced off input is enabled and set to “normally open” mode on zone <n>
		W	enable the local forced off input in “normally open” mode on zone <n>
	closed	R	the local forced off input is enabled and set to “normally closed” mode on zone <n>
		W	enable the local forced off input in “normally closed” mode on zone <n>
z<n>.status.outlet	0	R	the motorized outlet is closed on zone <n>
	1	R	the motorized outlet is open on zone <n>
z<n>.status.motor	0	R	the outlet motor is inactive on zone <n>

	1	R	the outlet motor is active on zone <n>
z<n>.status.heatingfloor	0	R	the heating floor is inactive on zone <n>
	1	R	the heating floor is active on zone <n>
z<n>.status.moduletype	outlet	R	the module type of zone <n> is motorized outlet
	fancoil	R	the module type of zone <n> is fancoil module
z<n>.status.ecoinput[Note 1]	0	R	the module Eco-Sleep input is inactive on zone <n>
	1	R	the module Eco-Sleep input is active on zone <n>
z<n>.status.onoffinput[Note 1]	0	R	the module on/off input is inactive on zone <n>
	1	R	the module on/off input is active on zone <n>
z<n>.status.fanspeed	0	R	the fancoil ventilation is off on zone <n>
	1	R	the fancoil ventilation speed is set to 1 on zone <n>
	2	R	the fancoil ventilation speed is set to 2 on zone <n>
	3	R	the fancoil ventilation speed is set to 3 on zone <n>
z<n>.status.propopening	1	R	the proportional opening status is set to step 1 on zone <n>
	2	R	the proportional opening status is set to step 2 on zone <n>
	3	R	the proportional opening status is set to step 3 on zone <n>
	4	R	the proportional opening status is set to step 4 on zone <n>
z<n>.status.thermconnection	0	R	the thermostat is not connected to zone <n>
	1	R	the thermostat is connected to zone <n>
z<n>.status.antifreeze[Note 2]	0	R	antifreeze function not active on zone <n>

	1	R	antifreeze function active on zone <n>
z<n>.error.motconnection	0	R	the motor is properly connected to the module of zone <n>
	1	R	the motor is not connected to the module of zone <n>
z<n>.error.motblock	0	R	the motor is not blocked zone <n>
	1	R	the motor is blocked on zone <n>
z<n>.error.probeopen	0	R	the probe input is not in open circuit on zone <n>
	1	R	the probe input is in open circuit on zone <n>
z<n>.error.probeshort	0	R	the probe input is not in short circuit on zone <n>
	1	R	the probe input is in short circuit on zone <n>
z<n>.error.config	0	R	the configuration is OK on zone <n>
	1	R	the configuration is not compatible with zone <n>
z<n>.error.comm	0	R	communication with remote probe OK on zone <n>
	1	R	communication with remote probe lost on zone <n>
z<n>.error	0	R	none of the above errors is active on zone <n>
	1	R	at least one of the above errors is active on zone <n>
z<n>.error	0	W	unset the system time
	<d>:<h>:<m>	W	set the system time to the specified value. <d> indicates the day (1 = Monday, ..., 7 = Sunday) <h> specifies the hour (from 0 to 23) and <m> the minutes (from 1 to 59). For instance a value of “2:13:10” represents the time 1:10 pm of Tuesday.

User Interface

Airzone Object

The user interface for the Airzone control framework:

File:UI Object airzone.png

The (airzone) object is listed in the Project Editor's new object list only when at least one Airzone I/O Server is defined.

Parameters

- **server id**: the server ID
- **zone**: the zone
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(airzone <server id>; <zone>; <position>)
```

E.g.

```
(airzone serverid; zone; x10y20)
```

UISET Actions

ID	Attribute	Set to	
connection.label.online	visible	true	when datapoint connection = online
		false	when datapoint connection = offline
connection.label.offline	visible	true	when datapoint connection = offline
		false	when datapoint connection = online
z<n>.name[Note 2]	value	name of zone <n>	
power.label.on	visible	true	when the system is on
		false	when the system is off
power.label.off	visible	true	when the system is off
		false	when the system is on
mode.label.stop	visible	true	when the working mode of the system is set to 'stop'
		false	when the working mode of the system is not set to 'stop'
mode.label.cold	visible	true	when the working mode of the system is set to 'cold'

		false	when the working mode of the system is not set to 'cold'
mode.label.heatpump	visible	true	when the working mode of the system is set to 'heatpump'
		false	when the working mode of the system is not set to 'heatpump'
mode.label.ventilation	visible	true	when the working mode of the system is set to 'ventilation'
		false	when the working mode of the system is not set to 'ventilation'
mode.label.heatingfloor	visible	true	when the working mode of the system is set to 'heatingfloor'
		false	when the working mode of the system is not set to 'heatingfloor'
mode.label.combheating	visible	true	when the working mode of the system is set to 'combheating'
		false	when the working mode of the system is not set to 'combheating'
mode.label.dehumid[Note 2]	visible	true	when the working mode of the system is set to 'dehumid'
		false	when the working mode of the system is not set to 'dehumid'
mode.fanspeed.label.auto	visible	true	when the fancoil fan speed of the system is set to 'auto'
		false	when the fancoil fan speed of the system is not set to 'auto'
mode.fanspeed.label.1	visible	true	when the fancoil fan speed of the system is set to 1
		false	when the fancoil fan speed of the system is not set to 1
mode.fanspeed.label.2	visible	true	when the fancoil fan speed of the system is set to 2
		false	when the fancoil fan speed of the system is not set to 2
mode.fanspeed.label.3	visible	true	when the fancoil fan speed of the system is set to 3
		false	when the fancoil fan speed of the system is not set to 3
z<n>.power.label.on	visible	true	when zone <n> is on

		false	when zone <n> is off
z<n>.power.label.off	visible	true	when zone <n> is off
		false	when zone <n> is on
z<n>.localtemp	value	the local temperature of zone <n> in °C or °F depending on the system settings	
z<n>.remotetemp	value	the remote probe temperature of zone <n> in °C or °F depending on the system settings	
z<n>.temp	value	the value of datapoint 'z<n>.temp' depending on the protocol version	
z<n>.setpoint.temp	value	the setpoint temperature of zone <n> in °C or °F depending on the system settings	
	visible	true	when zone <n> is on and set to manual control mode
		false	when zone <n> is off or not manual mode
z<n>.mode.eco.label.on[Note 1]	visible	true	when zone <n> is set to Eco-Sleep mode
		false	when zone <n> is not set to Eco-Sleep mode
z<n>.mode.eco.label.off[Note 1]	visible	true	when zone <n> is not set to Eco-Sleep mode
		false	when zone <n> is set to Eco-Sleep mode
z<n>.mode.sleep[Note 2]	value	the current time set for the sleep mode or “OFF” if disabled	
z<n>.mode.label.stop	visible	true	when the working mode of zone <n> is set to 'stop'
		false	when the working mode of zone <n> is not set to 'stop'
z<n>.mode.label.cold	visible	true	when the working mode of zone <n> is set to 'cold'
		false	when the working mode of zone <n> is not set to 'cold'
z<n>.mode.label.heatpump	visible	true	when the working mode of zone <n> is set to 'heatpump'
		false	when the working mode of zone <n> is not set to 'heatpump'
z<n>.mode.label.ventilation	visible	true	when the working mode of zone <n> is set to 'ventilation'

		false	when the working mode of zone <n> is not set to 'ventilation'
z<n>.mode.label.heatingfloor	visible	true	when the working mode of zone <n> is set to 'heatingfloor'
		false	when the working mode of zone <n> is not set to 'heatingfloor'
z<n>.mode.label.combheating	visible	true	when the working mode of zone <n> is set to 'combheating'
		false	when the working mode of zone <n> is not set to 'combheating'
z<n>.mode.label.dehumid[Note 2]	visible	true	when the working mode of zone <n> is set to 'dehumid'
		false	when the working mode of zone <n> is not set to 'dehumid'
z<n>.mode.control.manual	visible	true	when the control mode of zone <n> is set to 'manual'
		false	when the control mode of zone <n> is set to 'automatic'
z<n>.mode.control.automatic	visible	true	when the control mode of zone <n> is set to 'automatic'
		false	when the control mode of zone <n> is set to 'manual'
z<n>.mode.fanspeed.label.auto	visible	true	when the fancoil fan speed of zone <n> is set to 'auto'
		false	when the fancoil fan speed of zone <n> is not set to 'auto'
z<n>.mode.fanspeed.label.1	visible	true	when the fancoil fan speed of zone <n> is set to 1
		false	when the fancoil fan speed of zone <n> is not set to 1
z<n>.mode.fanspeed.label.2	visible	true	when the fancoil fan speed of zone <n> is set to 2
		false	when the fancoil fan speed of zone <n> is not set to 2
z<n>.mode.fanspeed.label.3	visible	true	when the fancoil fan speed of zone <n> is set to 3
		false	when the fancoil fan speed of zone <n> is not set to 3
z<n>.status.motor.label.on	visible	true	when the the outlet motor of zone <n> is active

		false	when the the outlet motor of zone <n> is not active
z<n>.status.motor.label.off	visible	true	when the the outlet motor of zone <n> is not active
		false	when the the outlet motor of zone <n> is active
z<n>.error.label.0	visible	true	when zone <n> is correctly working
		false	when zone <n> is in an error state
z<n>.error.label.1	visible	true	when zone <n> is in an error state
		false	when zone <n> is correctly working

Note 1
: Only for protocol version 1

Note 2
: Only for protocol version 2

USER Commands

To send a user command to the I/O Server just set the name to the ID of a writable (W) datapoint and the parameter to one of the possible values.

For instance, setting the name of a user button to '<ioserver_id>.z1.power' and its param field to 'flip', will result in having a switch for zone 1. Naming it '<ioserver_id>.z1.mode' with a valid working mode as param (e.g. 'stop' or 'cold') will result in a button for setting the mode of the zone.

Release Notes

3.1.0

- Added support for protocol version 2

3.0.0

- Initial release

Airzone, Innobus, Flexa, and Antree are registered trademarks of Airzone.

Arduino

Arduino is a microcontroller platform and development tools combined into an easy-to-use, standardized and inexpensive architecture that can greatly enhance the physical input/output capabilities of HSYCO.

Arduino UNO

Arduino Ethernet

HSYCO supports Arduino through an open source library installed in the Arduino IDE. This library implements a bidirectional communication protocol with HSYCO. A sketch in Arduino that allows HSYCO to directly read and control any of the digital and analog pins on any Arduino board is only a few lines long, and is provided as a built-in example in the HSYCO library. HSYCO supports Arduino Uno, Arduino Ethernet, Arduino Due and other compatible boards. The connection is serial, usually over the USB port, or TCP over the Ethernet. The Arduino I/O Server provides for datapoints to be used in the EVENTS language and Java to read and control Arduino pins, but also to exchange custom messages between your sketch in Arduino and HSYCO applications.

Communication

Arduino can be connected to HSYCO through a USB cable (serial communication) or via Ethernet depending on the model.

Serial (USB)

The comm port ID depends on your HSYCO server configuration. It should be "ttyACM0" if there are no other devices connected to the USB ports.

These are the default serial communication parameters used by the HSYCO library:

Baud rate	115200
Data bits	8
Stop bit	1

Parity	none
Flow control	none

Ethernet

The IP address of the board is specified in its sketch (see Arduino Programming) and the default port is 23.

Arduino Configuration

IDE Configuration

Before you start writing sketches using the Arduino IDE, you should unzip and copy the HSYCO or HSYCONET libraries in the libraries directory.

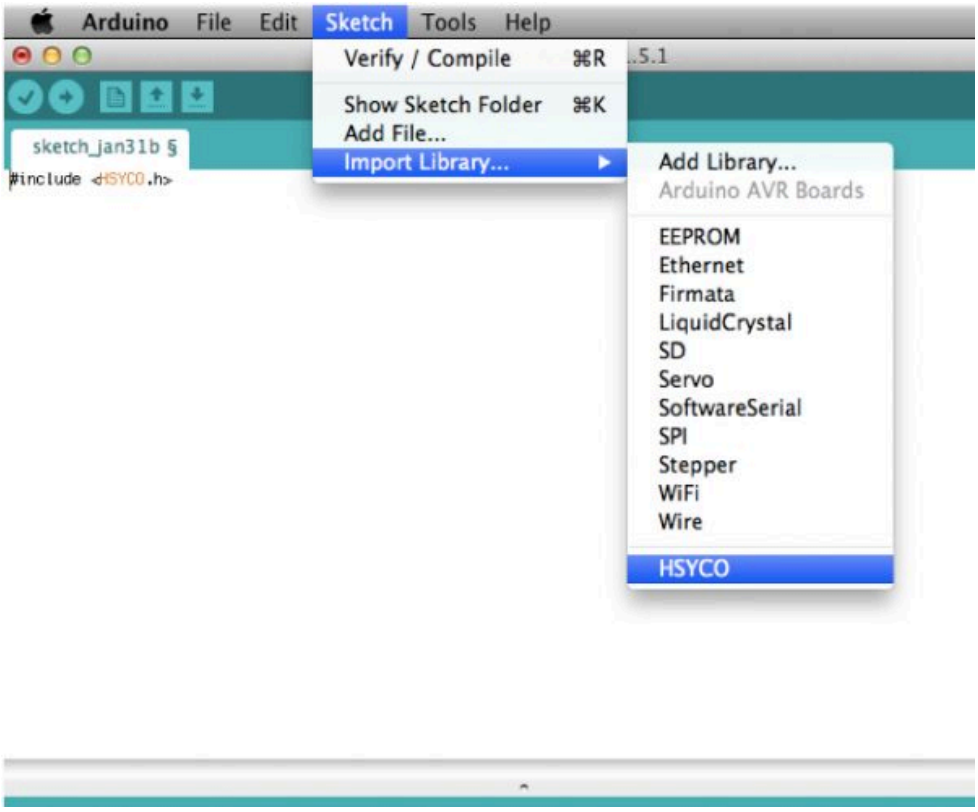
On **Mac**, the libraries directory is in your home directory, Documents/Arduino/libraries.

On **Linux** it is in the shetchbook directory of your home directory.

On **Windows** the folder is in My Documents\Arduino.

Use the HSYCO library when you are connecting Arduino and HSYCO via a serial/USB cable, or HSYCONET if using TCP over an Ethernet network.

Quit and reopen the Arduino IDE to make it aware of the new library. After restarting the Arduino IDE, you can create a new custom sketch with File → New, then Sketch → Import Library → HSYCO.

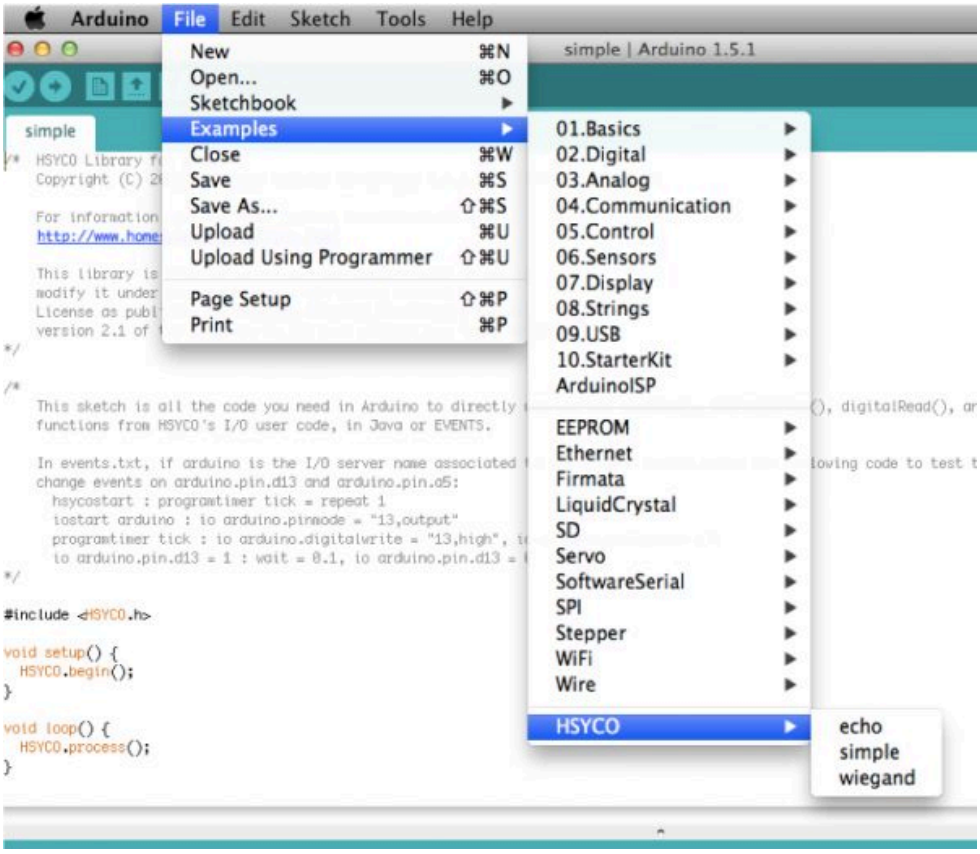


Arduino Programming

The HSYCO or HSYCONET library in Arduino will map all I/O pins, and the most relevant I/O functions in Arduino, to datapoints that you can use to set output pins and read input pins of the Arduino board.

If you only need to directly control Arduino's pins from HSYCO EVENTS or Java, without any custom sketch code in Arduino, select File → Examples → HSYCO → simple.

Upload the simple sketch to your board, and you are ready to use Arduino from HSYCO.



This sketch is all the code you need in Arduino to directly call the `pinMode()`, `digitalWrite()`, `digitalRead()`, `analogRead()` and `analogWrite()` functions from HSYCO.

Simple sketch using the HSYCO library (serial communication):

```
#include <HSYCO.h>

void setup() {
    HSYCO.begin();
}

void loop() {
    HSYCO.process();
}
```

Simple sketch using the HSYCONET library (Ethernet communication):

```
#include <HSYCONET.h>
#include <Ethernet.h>
#include <SPI.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192, 168, 1, 177);
```

```
IPAddress subnet(255, 255, 255, 0);
IPAddress gateway(192, 168, 1, 1);

void setup() {
    Ethernet.begin(mac, ip, gateway, subnet);
    HSYCONET.begin();
}

void loop() {
    HSYCONET.process();
}
```

HSYCO Library

The HSYCO class implements a bidirectional serial communication protocol with the Arduino I/O Server in HSYCO. Following are the available methods:

`void begin()`

Initializes the HSYCO library and serial communication settings, using the default serial port (Serial) and 115200 bps speed.

`void begin(long speed)`

Initializes the HSYCO library and serial communication settings, using the default serial port (Serial) and the specified speed.

Parameters:

> `speed` - communication speed in bits per second (baud).

`void begin(Stream &stream)`

Initializes the HSYCO library, using the specified stream. The stream should be properly configured.

Parameters:

> `stream` - the Stream object used to communicate with HSYCO.

`void process()`

Process incoming commands from the Arduino I/O Server. This method should always be present in Arduino's main loop, even when you are only using Arduino for sending data to HSYCO, or the Arduino I/O Server will fail to connect to the board.

`void ioSet(char* name, char* value)`

Set the datapoint of the I/O Server with ID `name` to the specified `value`. The change event in HSYCO is forced, so that the ioSet() call in Arduino will always fire an event, even when the value is not changed.

Parameters:

> `name` - the datapoint ID; it cannot contain the 0x0a, 0x0d, 0xff, or '=' characters
> `value` - the value to set the datapoint to; it cannot contain the 0x0a, 0x0d or 0xff special characters.

`void ioSet(char* name, int value)`

Set the datapoint of the I/O Server with ID `name` to the specified numerical `value`. The change event in HSYCO is forced, so that the ioSet() call in Arduino will always fire an event, even when the value is not changed.

Parameters:

`name` - the datapoint ID; it cannot contain the 0x0a, 0x0d, 0xff, or '=' characters
`value` - the numerical value to set the datapoint to; it cannot contain the 0x0a, 0x0d or 0xff special characters.

`void attach(callbackFunction myFunction)`

Specifies a function to call when an IO write is executed on HSYCO, using the IO action in EVENTS or the ioSet() Java method, if the datapoint ID is not one of the reserved ones.

Parameters:

`myFunction` - the function to call. This function must take two char* parameters, corresponding to the datapoint ID and value.

`void detach()`

Turns off the callback to a previously attached function.

HSYCONET Library

The HSYCONET class implements a bidirectional TCP/IP communication protocol with the Arduino I/O Server in HSYCO. Following are the available methods:

`void begin()`

Initializes the HSYCONET library and the internal socket server.

`void process()`

Process incoming commands from the Arduino I/O Server. This method should always be present in Arduino's main loop, even when you are only using Arduino for sending data to HSYCO, or the Arduino I/O Server will fail to connect to the board.

`void ioSet(char* name, char* value)`

Set the datapoint of the I/O Server with ID `name` to the specified `value`. The change event in HSYCO is forced, so that the ioSet() call in Arduino will always fire an event, even when the value is not changed.

Parameters:

`name` - the datapoint ID; it cannot contain the 0x0a, 0x0d, 0xff, or '=' characters
`value` - the value to set the datapoint to; it cannot contain the 0x0a, 0x0d or 0xff special characters.

`void ioSet(char* name, int value)`

Set the datapoint of the I/O Server with ID `name` to the specified numerical `value`. The change event in HSYCO is forced, so that the ioSet() call in Arduino will always fire an event, even when the value is not changed.

Parameters:

`name` - the datapoint ID; it cannot contain the 0x0a, 0x0d, 0xff, or '=' characters
`value` - the numerical value to set the datapoint to; it cannot contain the 0x0a, 0x0d or 0xff special characters.

`void attach(callbackFunction myFunction)`

Specifies a function to call when an IO write is executed on HSYCO, using the IO action in EVENTS or the ioSet() Java method, if the datapoint ID is not one of the reserved ones.

Parameters:

> `myFunction` - the function to call. This function must take two char* parameters, corresponding to the datapoint ID and value.

```
void detach()
```

Turns off the callback to a previously attached function.

HSYCO Configuration

Add an Arduino I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

Serial (USB)

Select the "Comm" option and select the ID of the comm port connected to the board.

Ethernet

Select the "IP" option and enter:

- **Address**: IP address of the Arduino board as specified in its sketch
- **Port**: TCP/IP port used by the board. Leave blank if the default port (23) is used.

High Availability

- **Shutdown when inactive**: defaults to false.

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established with the board
	offline	R	HSYCO can't connect to the board
pinmode	<pin>,input	W	execute the `pinMode()` function, configuring the pin <pin> as INPUT, OUTPUT or INPUT_PULLUP
	<pin>,output		
	<pin>,input_pullup		
digitalwrite	<pin>,high	W	execute the `digitalWrite()` function, setting the pin <pin> to high (5V or 3.3V) or low (0V)
	<pin>,low		
digitalread	<pin>	W	execute the `digitalRead()` function, reading the value of pin <pin>

analogwrite	<pin>,<value>	W	execute the `analogWrite()` function, setting the pin <pin> to a PWM output with duty cycle equal to <value>; the value must be between 0 (always off) and 255 (always on)
analogread	<pin>	W	execute the `analogRead()` function, reading the value of pin <pin>
pin.<n>	0	R	pin <n> is low
		W	execute the `digitalWrite()` function, setting the digital pin <n> to low (0V) equal to digitalwrite = <n>,low
	1	R	pin <n> is high
		W	execute the `digitalWrite()` function, setting the digital pin <n> to high (5V or 3.3V) equal to digitalwrite = <n>,high
pin.a<n>	<value>	R	analog pin <n> is set to duty cycle equal to <value>
		W	execute the `analogWrite()` function, setting the analog pin <n> to a PWM output with duty cycle equal to <value>; the value must be between 0 (always off) and 255 (always on) equal to analogwrite = a<n>, <value>
pin.dac<n>	<value>	R	DAC pin <n> is set to <value>
		W	execute the `analogWrite()` function, setting the DAC pin <n> to <value> equal to analogwrite = dac<n>, <value>
<custom_name>	<value>	R	the ioSet(<custom_name>, <value>) method was called from the Arduino sketch
		W	if a callback function is defined in the Arduino sketch, this function is called passing <custom_name> and <value> as parameters. The parameters cannot contain the 0x0a, 0xff, or '=' characters

Pins syntax

The naming convention HSYCO uses for the pins corresponds to the one used by Arduino.

Digital pins are numbered from 0 up, with no prefix.

Analog pins start with the "a" prefix, like "a0", "a1" etc. (all lowercase).

The DAC analog out pins in Arduino DUE are "dac0" and "dac1" (all lowercase).

Release Notes

3.3.0

- initial release

Arduino is a registered trademark of Arduino team.

Arteco

Arteco Everywhere is a general purpose digital input-output Ethernet controller.

It is available in two versions, a 4 inputs, 4 outputs module, and an 8 inputs 8 outputs module.

The ARTECOIO I/O Server in HSYCO creates a data point representation that you can use in the EVENTS language and Java to read the input pins and control the outputs. It also automatically allows you to create control objects for the Web user interface.

HSYCO Configuration

Add an ARTECOIO I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: host name or IP address of the Arteco module (the default address is 192.168.10.96)
- **IP Port**: TCP/IP port to use, leave blank to use default port 80
- **User**: username, leave blank to use default username "admin"
- **Password**: password, leave blank to use default password "arteco".

High Availability

- **Shutdown when inactive**: defaults to false.

Options

ID	Default	Values	Description
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
inputdiscovery	false	true	automatically creates the list of all inputs in the systemtopo.txt file. Should be enabled to allow the automatic update of (button) objects' states

		false	auto-detect for input devices is disabled
outputdiscovery	false	true	automatically creates the list of all outputs in the systemtopo.txt file. Should be enabled to allow the automatic update of (button) objects' states
		false	auto-detect for output devices is disabled
inputs	8	0 ... 8	use input pins from 0 to the specified value
outputs	8	0 ... 8	use output pins from 0 to the specified value

Datapoints

The data points associated to the input pins are named "i0" to "i7". The data points associated to the output pins are named "o0" to "o7".

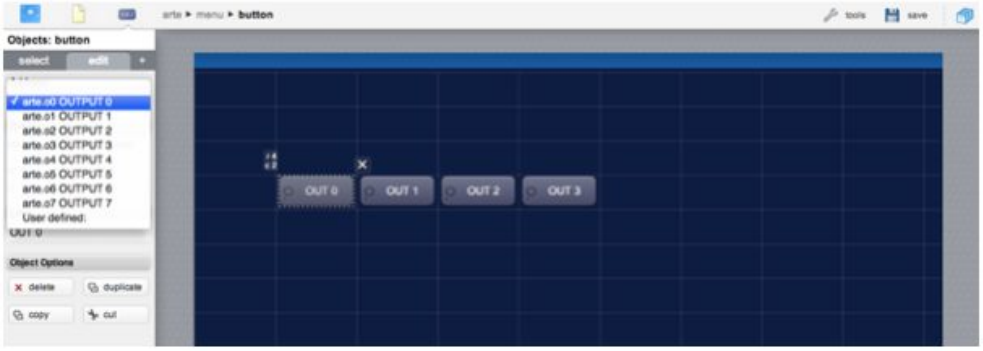
The I/O Server generates asynchronous events based on the status of the Arteco module's input and output pins. The output data points can be written to control the output pins.

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
i<n>	0	R	input <n> is low
	1	R	input <n> is high
o<n>	0	R	output <n> is low
		W	set output <n> to off
	1	R	output <n> is high
		W	set output <n> to on
	flip	W	invert the status of output <n>
	blink	W	continuously blink output <n>
	pulse	W	set output <n> to on, then off after 1 sec.

User Interface

When using the default settings of the Arteco I/O Server, all output pins are available in the Project Editor and can be easily associated to any control button of the Web interface.

You can also set the *inputdiscovery* option in order to associate control buttons to the module's inputs: the control buttons will show the input status, and have no effect when pressed.



Release Notes

3.3.0

- initial release

ARTECO is a registered trademark of Arteco Srl.

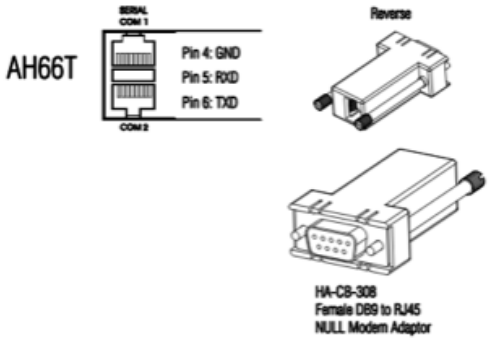
Aton

Aton AH66T is a multi-zone audio system. The integration with HSYCO can be accomplished via direct control through the RS232 port on the module.

The Aton I/O Server in HSYCO supports the integration of more than one unit within the same HSYCO interface.

Communication

Employ a DE-9 (often called DB-9) female-male RS-232 straight cable to connect the AH66T COM1 port to the serial port on HSYCO SERVER (female DB9 to RJ45 adaptor is included in AH66T packaging).



RS-232 parameters:

Baud rate	same as set on the module (115200/57600/38400/19200/9600)
Data bits	8
Stop bit	1
Parity	none
Flow control	none

ATON Configuration

The ATON module must be configured to enable the communication with HSYCO.

Use the ATON SystemWorx tool to modify the configuration:

💡 **Make sure to specify "ExtControl" as Driver Type of COM Port 1, set the serial communication parameters, and transfer the configuration to the module.**

HSYCO Configuration

Add an ATON I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select the comm port connected to the panel.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
pollinterval	1	n > 0	the data acquisition interval, in seconds

modules	1	1 or 2	number of modules included in the connected system

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
power	1	R	the system is on
	0	R	the system is off
		W	turn the system off
z<n>.power	1	R	zone <n> is on
		W	turn zone <n> on
	0	R	zone <n> is off
		W	turn zone <n> off
	flip	W	toggle the power status of zone N
z<n>.source	0	R	zone <n> is off
	1...6	R	the audio source of zone <n> is set to the reported source number
		W	set the audio source of zone <n> to the specified source number
	r1	R	the audio source of zone <n> is set to the radio of module 1
		W	set the audio source of zone <n> to the radio of module 1
	r2	R	the audio source of zone <n> is set to the radio of module 2
		W	set the audio source of zone <n> to the radio of module 2
z<n>.volume	off	R	zone <n> is muted
		W	mute zone <n>
	on	W	un-mute zone <n>

	0...100%	R	the volume level of zone <n> is set to the specified value
		W	set the volume level of zone <n> to the specified value
	up	W	increase the volume level of zone <n>
	down	W	decrease the volume level of zone <n>
z<n>.mute	1	R	zone <n> is muted
		W	mute zone <n>
	0	R	zone <n> is un-muted
		W	un-mute zone <n>
	flip	W	toggle the mute status of zone <n>
z<n>.bass	<val>	R	the bass level of zone <n> is set to the specified value (between “-12” and “+12”)
		W	set the bass level of zone <n> to the specified value (between “-12” and “+12”)
	up	W	increases the bass level of zone <n>
	down	W	decreases the bass level of zone <n>
z<n>.treble	<val>	R	the treble level of zone <n> is set to the specified value (between “-12” and “+12”)
		W	set the treble level of zone <n> to the specified value (between “-12” and “+12”)
	up	W	increases the treble level of zone <n>
	down	W	decreases the treble level of zone <n>
z<n>.dnd	1	R	“do not disturb” mode is enabled on zone <n>
		W	enable “do not disturb” mode on zone <n>
	0	R	“do not disturb” mode is disabled on zone <n>

		W	disable “do not disturb” mode on zone <n>
	flip	W	toggles the “do not disturb” mode on zone <n>
z<n>.loud	1	R	the loudness EQ setting is enabled on zone <n>
		W	enable the loudness EQ setting on zone <n>
	0	R	the loudness EQ setting is disabled on zone <n>
		W	disable the loudness EQ setting on zone <n>
	flip	W	toggle the loudness EQ setting on zone <n>
z<n>.preamp	fix	R	the pre-amp output of zone <n> is set to fixed
		W	set the pre-amp output of zone <n> to fixed
	var	R	the pre-amp output of zone <n> is set to variable
		W	set the pre-amp output of zone <n> to variable
	flip	W	toggle the status of the pre-amp output of zone <n>
z<n>.mono	1	R	the audio mode of zone <n> is set to mono
		W	set the audio mode of zone <n> to mono
	0	R	the audio mode of zone <n> is set to stereo
		W	set the audio mode of zone <n> to stereo
	flip	W	toggle the audio mode of zone <n>
z<n>.whm	1	R	“whole house music” mode is active on zone <n>
		W	activate “whole house music” mode on zone <n>
	0	R	“whole house music” mode is not active on zone <n>

		W	deactivate “whole house music” mode on zone <n>
	flip	W	toggle the “whole house music” mode on zone <n>
m<n>.sti<m>	1	R	sense trigger input <m> on module <n> is present
	0	R	sense trigger input <m> on module <n> is absent
m<n>.asi<m>	1	R	audio source input <m> on module <n> is present
	0	R	audio source input <m> on module <n> is absent
m<n>.tuner.band	am	R	the tuner of module <n> is set on the AM radio band
		W	set the tuner of module <n> on the AM radio band
	fm	R	the tuner of module <n> is set on the FM radio band
		W	set the tuner of module <n> on the FM radio band
m<n>.tuner.tune	<xxxx>	R	the tuner of module <n> is tuned on the reported frequency
		W	tune the frequency of the tuner of module <n> on the specified value
	up	W	increase the frequency of the tuner of module <n>
	down	W	decrease the frequency of the tuner of module <n>
m<n>.tuner.seek	up	W	tune module <n> on the next valid station with higher frequency
	down	W	tune module <n> on the next valid station with lower frequency
m<n>.tuner.group	<group>	R	the selected group of preset radio stations for the tuner of module <n> is set to the reported value (A,B,C, or D)
		W	set the selected group of preset radio stations for the tuner of module <n> to the specified value (A,B,C, or D)

m<n>.tuner.mode	preset	R	the tuning mode for the tuner of module <n> is set to preset
		W	set the tuning mode for the tuner of module <n> to preset
	direct	R	the tuning mode for the tuner of module <n> is set to direct
		W	set the tuning mode for the tuner of module <n> to direct
m<n>.tuner.preset	<xx>	R	the current preset for the tuner of module <n> is the reported value
		W	tunes module <n> to the specified preset value
	up	W	set the tuner of module <n> to the next higher preset
	down	W	set the tuner of module <n> to the next lower preset
	save	W	saves the currently selected station on module <n> as a preset
	del	W	delete the current preset on module <n>
m<n>.tuner.scan	up	W	scan through presets upwards. The station is automatically changed every 10 seconds until another command is sent
	down	W	scan through presets downwards. The station is automatically changed every 10 seconds until another command is sent
m<n>.tuner.signal	<val>	R	the signal strength of the currently tuned station corresponds to the reported value
m<n>.tuner.mono	1	R	the audio mode of the tuner of module <n> is set to mono
		W	set the audio mode of the tuner of module <n> to mono
	0	R	the audio mode of the tuner of module <n> is set to stereo
		W	set the audio mode of the tuner of module <n> to stereo

	flip	W	toggle the audio mode of the tuner of module <n>
m<n>.tuner.mdf.name	<text>	R	the name metadata of the current station of the tuner of module <n> is set to the reported text
m<n>.tuner.mdf.text	<text>	R	the text metadata of the current station of the tuner of module <n> is set to the reported text
m<n>.tuner.mdf.gen	<text>	R	the genre metadata of the current station of the tuner of module <n> is set to the reported text
m<n>.tuner.mdf.cl	<text>	R	the name call letters of the current station of the tuner of module <n> is set to the reported text
m<n>.tuner.digit	<0...9>	W	send the specified digit for direct tuning to module <n>
	canc	W	clears any digit previously entered on module <n>
	ok	W	sends the enter command to module <n>

User Interface

Aton Object

The user interface for the Aton multi-room system:

File:UI Object aton.png

The Aton object is listed in the Project Editor's new object list only when at least one Aton I/O Server is defined.

Parameters

- **server id**: the server ID
- **module**:
- **zone**: zone number of the Aton multi-room system (max 12 zones)
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(aton <server id>; <module>; <zone>; <position>)
```

E.g.

```
(aton serverid; module; zone; x10y20)
```

UISET Actions

ID	Attribute	Set to
z<n>.source	value	the source number set for zone <n>
z<n>.volume	value	the volume level set for zone <n>
z<n>.bass	value	the bass level set for zone <n>
z<n>.treble	value	the treble level set for zone <n>
m<n>.tuner.band	value	the band set for the tuner of module <n> (i.e. “AM” or “FM”)
m<n>.tuner.group	value	the selected group of preset radio stations for the tuner of module <n> (i.e. A,B,C, or D)
m<n>.tuner.mdf.cl	value	the call letters metadata of the current station of the tuner of module <n>
m<n>.tuner.mdf.gen	value	the genre metadata of the current station of the tuner of module <n>
m<n>.tuner.mdf.text	value	the radio text metadata of the current station of the tuner of module <n>
m<n>.tuner.mdf.name	value	the name metadata of the current station of the tuner of module <n>
m<n>.tuner.mode	value	the tuning mode set for the tuner of module <n> (i.e. “DIRECT” or “PRESET”)
m<n>.tuner.preset	value	the current preset for the tuner of module <n>
m<n>.tuner.band	value	the signal strength level of the current station for the tuner of module <n>
connection.label	visible	true when the system is not connected to HSYCO, false otherwise
power.label.1	visible	true when the system is on, false otherwise
power.label.0	visible	true when the system is off, false otherwise
z<n>.power.label.1	visible	true when zone <n> is on, false otherwise

z<n>.power.label.0	visible	true when zone <n> is off, false otherwise
z<n>.dnd.label.1	visible	true when “do not disturb” mode is active on zone <n>, false otherwise
z<n>.dnd.label.0	visible	true when “do not disturb” mode is not active on zone <n>, false otherwise
z<n>.loud.label.1	visible	true when the loudness EQ setting is enabled on zone <n>, false otherwise
z<n>.loud.label.0	visible	true when the loudness EQ setting is disabled on zone <n>, false otherwise
z<n>.mute.label.1	visible	true when zone <n> is muted, false otherwise
z<n>.mute.label.0	visible	true when zone <n> is not muted, false otherwise
z<n>.mono.label.1	visible	true when the audio mode of zone <n> is set to mono, false otherwise
z<n>.mono.label.0	visible	true when the audio mode of zone <n> is set to stereo, false otherwise
z<n>.whm.label.1	visible	true when “whole house music” mode is active on zone <n>, false otherwise
z<n>.whm.label.0	visible	true when “whole house music” mode is not active on zone <n>, false otherwise
m<n>.tuner.mono.label.1	visible	true when the audio mode of the tuner of module <n> is set to mono, false otherwise
m<n>.tuner.mono.label.0	visible	true when the audio mode of the tuner of module <n> is set to stereo, false otherwise

USER Commands

To send a user command to the I/O Server just set the name to the ID of a writable (W) datapoint and the parameter to one of the possible values.

For instance, setting the name of a user button to “aton.z1.power” and its param field to “flip”, will result in having a switch for zone 1. Naming it “aton.z1.source” with a valid source as param (e.g. “1”, “2”, or “R1”) will result in a button for setting the audio source of the zone.

The volume control is both a display field for the current volume status, and a control to set the volume level of a zone. You should use the slider or sliderbutton object for this control and set both its ID and Address fields to “serverID.zN.volume”.

Release Notes

3.1.0

- initial version release

ATON and AH66T are registered trademarks of The AVC Group, LLC.

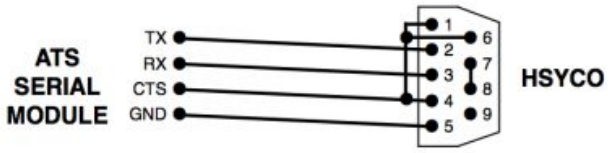
Ats

Aritech ATS panels are multi-area intrusion detection systems. The integration with HSYCO can be accomplished via direct control through the RS-232 port on the serial module. The ATS I/O Server supports the integration of more than one unit within the same HSYCO interface.

Communication

Employ an RS-232 cable to connect the panel directly to HSYCO or to an Ethernet/RS-232 gateway.

The cable must follow this pinout diagram:



RS-232 parameters:

Baud rate	same as panel settings (default 4800)
Data bits	8
Stop bit	1
Parity	none
Flow control	RTS/CTS

HSYCO Configuration

Add an ATS I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select the comm port connected to the panel.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands

		false	disable UI support
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
pollinterval	5	n > 0	the data acquisition interval, in seconds
logsize	20	n ≥ 0	the number of log lines to display in the UI object
maxzone	256	0 ... 256	highest zone index set on the panel. This option is recommended to improve the I/O Server's performances
maxoutput	256	0 ... 256	highest output index set on the panel. This option is recommended to improve the I/O Server's performances
maxras	16	0 ... 16	highest RAS (Remote Arming Station) index set on the panel. This option is recommended to improve the I/O Server's performances

ats.ini

The ats.ini file is a specific configuration file located in the root directory. Here you can define the names of zones and areas so that they will be automatically added to the user interface.

Add a new line for each name using this format:

```
<io_server_id>.zone.<n> = <name>
<io_server_id>.area.<n> = <name>
```

For instance:

```
ats.zone.1 = Kitchen
ats.zone.2 = Corridor
ats.area.1 = Floor 1
```

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel

armed	1	R	at least one area is armed
		W	arm all active areas
	0	R	no area is armed
		W	disarm all active areas
alarm	1	R	at least one area is in alarm
	0	R	no area is in alarm
	reset	W	reset system alarms
clock	read	W	request the reading of the panel's clock value
	<date_time>	R	the panel's clock is set to <date_time>. The value has the format "yyyy-mm-dd hh:mm:ss" (e.g. "2012-12-30 16:59:23"). This event is only triggered after a request.
a<n>.name	<text>	R	area <n> has been assigned the name <text>
z<n>.name	<text>	R	zone <n> has been assigned the name <text>
a<n>.armed	1	R	area <n> is armed
		W	arm area <n>
	0	R	area <n> is disarmed
		W	disarm area <n>
	force	W	force arm area <n>
a<n>.alarm	1	R	area <n> is in alarm
	0	R	area <n> is not in alarm
a<n>.open	1	R	area <n> has open zones
	0	R	area <n> has no open zones
a<n>.isolated	1	R	area <n> has isolated zones
	0	R	area <n> has no isolated zones
a<n>.exit	1	R	exit time is active on area <n>
	0	R	exit time is not active on area <n>
a<n>.entry	1	R	entry time is active on area <n>

	0	R	entry time is not active on area <n>
a<n>.alarm.local	1	R	local alarms are active on area <n>
	0	R	no local alarm is active on area <n>
a<n>.warning	1	R	console warning is active on area <n>
	0	R	console warning is not active on area <n>
z<n>.state1[Note]	tamper	R	tamper active on zone <n>
	alarm	R	alarm active on zone <n>
	closed	R	zone <n> closed
	night	R	night holdup active on zone <n>
	local	R	local alarm active on zone <n>
	isolated	R	zone <n> isolated
z<n>.state2[Note]	short	R	zone <n> in short circuit
	open	R	zone <n> open
	closed	R	zone <n> closed
	active	R	zone <n> active
	masked	R	zone <n> masked
z<n>.isolated	1	W	isolate zone <n>
	0	W	de-isolate zone <n>
z<n>.alarm	reset	W	reset alarms on zone <n>
o<n>.active	1	R	output <n> is active
		W	activate output <n>
	0	R	output <n> is not active
		W	deactivate output <n>
o<n>.inverted	1	R	output <n> is inverted
	0	R	output <n> is not inverted
ras<n>.offline	1	R	RAS <n> is offline
	0	R	RAS <n> is online
ras<n>.tamper	1	R	RAS <n> is tampered

	0	R	RAS <n> is not tampered
ras<n>.isolated	1	R	RAS <n> is isolated
	0	R	RAS <n> is not isolated
ras<n>.inhibited	1	R	RAS <n> is inhibited
	0	R	RAS <n> is not inhibited
dgp<n>.offline	1	R	DGP <n> is offline
	0	R	DGP <n> is online
dgp<n>.isolated	1	R	DGP <n> is isolated
	0	R	DGP <n> is not isolated
dgp<n>.inhibited	1	R	DGP <n> is inhibited
	0	R	DGP <n> is not inhibited
dgp<n>.battery.missing	1	R	battery missing on DGP <n>
	0	R	battery ok on DGP <n>
dgp<n>.battery.low	1	R	battery low on DGP <n>
	0	R	battery ok on DGP <n>
dgp<n>.battery.test	1	R	battery test active on DGP <n>
	0	R	no battery test active on DGP <n>
dgp<n>.battery.test.fail	1	R	battery test failed on DGP <n>
	0	R	battery test ok on DGP <n>
dgp<n>.mains.fail	1	R	mains failure on DGP <n>
	0	R	mains ok on DGP <n>
dgp<n>.fuse.fail	1	R	fuse failure on DGP <n>
	0	R	fuses ok on DGP <n>
dgp<n>.siren.fail	1	R	siren failure on DGP <n>
	0	R	siren ok on DGP <n>
dgp<n>.tamper	1	R	DGP <n> is tampered
	0	R	DGP <n> is not tampered

Note

The value of a zone status (datapoints z<n>.state1 and z<n>.state2) depends on the configuration of the zone typology.

User Interface

Ats Object

The user interface for the Aritech ATS multi-area intrusion detection system:

The Ats object is listed in the Project Editor's new object list only when at least one Aritech ATS I/O Server is defined.

Parameters

- **server id**: the server ID
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(ats <server id>; <position>)
```

E.g.

```
(ats serverid; x10y20)
```

UISET Actions

ID	Attribute	Set to
log<n>	value	the line <n> of the log list (<n> from 1 to 20, <n> = 1 : latest entry)
log0	value	the latest line of the log for a short period
a<n>.name	value	the name assigned to area <n>
z<n>.name	value	the name assigned to zone <n>
connection.label	visible	true when the system is not connected to HSYCO, false otherwise
armed.label.1	visible	true when at least one area is armed, not visible otherwise
armed.label.0	visible	true when no area is armed, not visible otherwise
alarm.label.1	visible	true when at least one area is in alarm, not visible otherwise
alarm.label.0	visible	true when no area is in alarm, not visible otherwise

a<n>.armed.label.1	visible	true when area <n> is armed, not visible otherwise
a<n>.armed.label.0	visible	true when area <n> is not armed, not visible otherwise
a<n>.alarm.label.1	visible	true when area <n> is in alarm, not visible otherwise
a<n>.alarm.label.0	visible	true when area <n> is not in alarm, not visible otherwise
a<n>.open.label.1	visible	true when area <n> has open zones, not visible otherwise
a<n>.open.label.0	visible	true when area <n> has no open zone, not visible otherwise
a<n>.isolated.label.1	visible	true when area <n> has isolated zones, not visible otherwise
a<n>.isolated.label.0	visible	true when area <n> has no isolated zone, not visible otherwise
a<n>.exit.label.1	visible	true when the exit time is active on area <n>, not visible otherwise
a<n>.exit.label.0	visible	true when the exit time is not active on area <n>, not visible otherwise
a<n>.entry.label.1	visible	true when the entry time is active on area <n>, not visible otherwise
a<n>.entry.label.0	visible	true when the entry time is not active on area <n>, not visible otherwise
a<n>.alarm.local.label.1	visible	true when local alarms are active on area <n>, not visible otherwise
a<n>.alarm.local.label.0	visible	true when no local alarm is active on area <n>, not visible otherwise
a<n>.warning.label.1	visible	true when the console warning is active on area <n>, not visible otherwise
a<n>.warning.label.0	visible	true when the console warning is not active on area <n>, not visible otherwise
o<n>.active.label.1	visible	true when output <n> is active, not visible otherwise

o<n>.active.label.0	visible	true when output <n> is inactive, not visible otherwise
o<n>.inverted.label.1	visible	true if output <n> is inverted, not visible otherwise
o<n>.inverted.label.0	visible	true if output <n> is not inverted, not visible otherwise
ras<n>.offline.label.1	visible	true if RAS <n> is offline, not visible otherwise
ras<n>.offline.label.0	visible	true if RAS <n> is online, not visible otherwise
ras<n>.tamper.label.1	visible	true when RAS <n> is tampered, not visible otherwise
ras<n>.tamper.label.0	visible	true when RAS <n> is not tampered, not visible otherwise
ras<n>.isolated.label.1	visible	true if RAS <n> is isolated, not visible otherwise
ras<n>.isolated.label.0	visible	true if RAS <n> is not isolated, not visible otherwise
ras<n>.inhibited.label.1	visible	true if RAS <n> is inhibited, not visible otherwise
ras<n>.inhibited.label.0	visible	true if RAS <n> is not inhibited, not visible otherwise
dgp<n>.offline.label.1	visible	true if DGP <n> is offline, not visible otherwise
dgp<n>.offline.label.0	visible	true if DGP <n> is online, not visible otherwise
dgp<n>.isolated.label.1	visible	true if DGP <n> is isolated, not visible otherwise
dgp<n>.isolated.label.0	visible	true if DGP <n> is not isolated, not visible otherwise
dgp<n>.inhibited.label.1	visible	true if DGP <n> is inhibited, not visible otherwise
dgp<n>.inhibited.label.0	visible	true if DGP <n> is not inhibited, not visible otherwise
dgp<n>.battery.missing.label.1	visible	true if the battery of DGP <n> is missing, not visible otherwise
dgp<n>.battery.missing.label.0	visible	true if the battery of DGP <n> is ok, not visible otherwise
dgp<n>.battery.test.label.1	visible	true if the battery test is active on DGP <n>, not visible otherwise

dgp<n>.battery.test.label.0	visible	true if the battery test is not active on DGP <n>, not visible otherwise
dgp<n>.battery.low.label.1	visible	true if the battery of DGP <n> is low, not visible otherwise
dgp<n>.battery.low.label.0	visible	true if the battery of DGP <n> is ok, not visible otherwise
dgp<n>.mains.fail.label.1	visible	true if a mains failure is active on DGP <n>, not visible otherwise
dgp<n>.mains.fail.label.0	visible	true if no mains failure is active on DGP <n>, not visible otherwise
dgp<n>.battery.test.fail.label.1	visible	true if the battery test failed on DGP <n>, not visible otherwise
dgp<n>.battery.test.fail.label.0	visible	true if the battery test succeeded on DGP <n>, not visible otherwise
dgp<n>.fuse.fail.label.1	visible	true if a fuse failed on DGP <n>, not visible otherwise
dgp<n>.fuse.fail.label.0	visible	true if no fuse failed on DGP <n>, not visible otherwise
dgp<n>.siren.fail.label.1	visible	true if a siren failed on DGP <n>, not visible otherwise
dgp<n>.siren.fail.label.0	visible	true if no siren failed on DGP <n>, not visible otherwise
dgp<n>.tamper.label.1	visible	true when DGP <n> is tampered, not visible otherwise
dgp<n>.tamper.label.0	visible	true when DGP <n> is not tampered, not visible otherwise

USER Commands

Name	Param	Action
armed	1	arm all active areas
	0	disarm all active areas
alarm	reset	reset system alarms
a<n>.armed	1	arm area <n>
	0	disarm area <n>
	force	force arm area <n>

z<n>.isolated	1	isolate zone <n>
	0	de-isolate zone <n>
z<n>.alarm	reset	reset alarms on zone <n>
o<n>.active	1	activate output <n>
	0	deactivate output <n>

Release Notes

3.2.0

- added event “ID.clock”
- removed I/O Server option “maxdgp”

3.1.0

- initial release

ATS and Aritech are registered trademarks of UTC Fire & Security.

AxisDecoder

The AXIS P7701 Video Decoder provides a simple monitoring solution by enabling analog or digital monitors/projectors to connect to and display live video from Axis network cameras and video encoders.

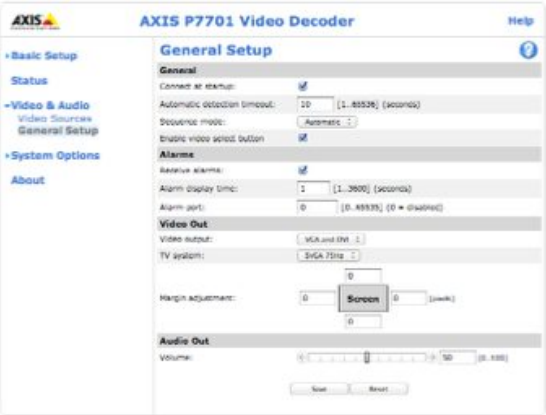
The HSYCO's AXISDEC I/O Server not only controls the decoder's sources selection, but is also able to generate dynamic MJPEG streams that are compatible with this decoder, extending it to display camera grids, non AXIS cameras, and even generic text or images with overlays.

Thanks to these features, you could easily use this decoder as a cost-effective, small, low-power and extremely reliable display driver for basic, customized digital signage applications.

Axis P7701 Configuration

Source Selection Control

To enable the decoder's source selection from HSYCO, you have to simply enable the "receive alarm" function:



Configure HSYCO as a video source

In order to be able to stream video from HSYCO to the P7701 decoder, HSYCO will present itself just like a standard AXIS camera. Just configure the HSYCO server as an ordinary video source:

Enter the HSYCO Server's IP address and HTTP port number, select HTTP as the "Transport method", and enter "hsyco" in the user name field. The password field is not used; HSYCO will only accept connections from encoders that are registered as I/O servers in hsyco.ini, and having an IP address in the trustedNet range.

HSYCO Configuration

Add an AXISDEC I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: host name or IP address of the decoder
- **IP Port**: TCP/IP port to use, leave blank to use default port 80.

Authentication

If you want to send source selection commands or the reboot command to the decoder, you need to add the authentication parameters:

- **User**: username set on the decoder
- **Password**: password set on the decoder.

High Availability

- **Shutdown when inactive**: defaults to false.

Options

ID	Default	Values	Description
streaming	true	true	video streaming is enabled

		false	video streaming is disabled, you will only be allowed to control the decoder's source selection
resolution	640x480	<width>x<height>	the display resolution in pixels, up to 1280x720
quality	0.7	0.0 ... 1.0	JPEG stream compression quality. Use 1.0 for maximum quality (requires more bandwidth)

Datapoints

ID	Value	R/W	Description
command	restart	W	send the reboot command to the video decoder
source	<source name>	R	source command sent successfully
		W	send the source selection command to the decoder. The source name has to be defined in the decoder's video sources
	<source name>:<message>	R	source command sent successfully
		W	send the source selection command to the decoder and a text message that the decoder will display when switching to the new source
stream	reset	W	see the following section on streaming content to the video decoder
	on	W	see the following section on streaming content to the video decoder
	off	W	see the following section on streaming content to the video decoder
	file: <parameters>	W	see the following section on streaming content to the video decoder
	cam: <parameters>	W	see the following section on streaming content to the video decoder
	text: <parameters>	W	see the following section on streaming content to the video

			decoder

Streaming to the Video Decoder

You can stream HSYCO's cameras video, still images and text to the decoder.

To do this, you simply write to the ID.stream data point. You can dynamically draw multiple graphic objects on the screen. Each new object is drawn on top of those that are already on screen. You can also assign a name to each object; naming objects allows you to replace a single object on screen maintaining the current drawing order.

You also have special commands to reset a screen, removing all objects and drawing a black screen, and to blank an existing screen without removing the objects.

Writing something without assigning a name to the object is equivalent to sending a reset command and then drawing that object.

If you don't have an Axis Decoder available for testing, you can use a Firefox browser to simulate the decoder. Simply open the Firefox browser on a device having the same IP address as set with the ioServersIP parameter in hsyco.ini, and use the following URL:

```
http://<HSYCO server's IP address>:<HTTP port>//axis-cgi/mjpg/video.cgi
```

Reset

When HSYCO starts and the AXIS decoder connects to HSYCO, it will show a black stream. After you have drawn objects to the decoder's stream, you can return to a black screen, deleting all objects, using the reset command:

```
IO video.stream = reset
```

Off and On

The off command is used to temporarily blank a screen, without removing the objects. It works just like the blank command in video projectors: the screen becomes black, but you can quickly turn it back on with the on command:

```
IO video.stream = off
IO video.stream = on
```

Drawing Files

You can draw any image file that you have locally under HSYCO's root directory. You can set the image's position on screen and its drawing size, or draw the image in its original size.

Note that resizing a file is a CPU intensive task, so it is advisable to save the files in the actual size that you need for drawing, rather than forcing the I/O server to do the resizing work on the fly.

The file drawing command's syntax in EVENTS is:

```
IO <id>.stream = "file[<obj name>]:<file pathname> width:<W> height:<H> top:<Y> left:<X>"
```

or:

```
IO <id>.stream = "file[<obj name>]:<on | off>
```

The on/off command is used to make an existing image visible or invisible.

The only required parameter is the file's pathname, relative to HSYCO's root directory so, assuming we have a directory named "images" in the root directory, and the file's name is "mountains.jpg", a valid command could be:

IO video.stream = file:images/mountains.jpg

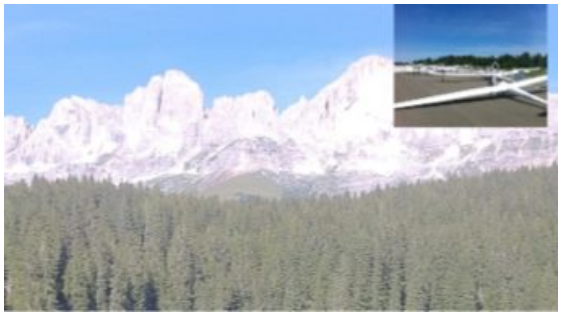
This command simply draws the image full-screen, eventually resizing it to fit the screen's size, based on the resolution defined in hsyco.ini. For more complex drawings, you will define the object's names as well as size and position. Size and position are expressed in pixels. The top-left corner has position coordinates 0,0.

To draw a background image and a smaller image in the top-right portion of the screen, assuming a resolution of 1280x720, use these two commands:

IO video.stream = file[background]:images/mountains.jpg IO video.stream = "file[pip]:images/file1.png width:352 height:288 top:0 left:900"

To change the "pip" image's visibility:

```
IO video.stream = file[pip]:off
IO video.stream = file[pip]:on
```

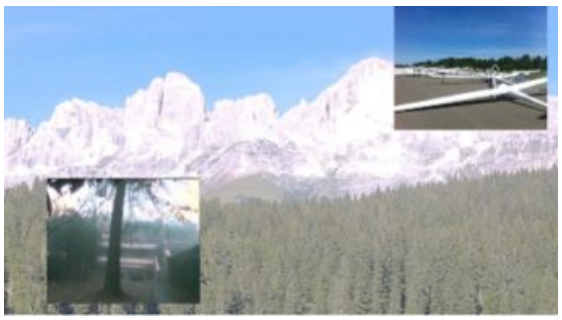


Note that the second command has its value enclosed in double quotes. This is because it has spaces in it, and without quotes it would not be considered as a single value by the EVENTS interpreter.

Drawing Live Images from Cameras and Grids

Just like you can draw static images, you can also draw HSYCO's live cameras and grids. The camera drawing syntax in EVENTS is: IO <id>.stream = "cam[<obj name>]:<camera name> width:<W> height:<H> top:<Y> left:<X>" or: IO <id>.stream = "cam[<obj name>]:<on | off>

The on/off command is used to make an existing camera visible or invisible. The <camera name> parameter is mandatory, and should be either a camera name, or gridN, where N is a positive integer, to identify a grid view of multiple cameras, as defined in hsyco.ini. Drawing cameras using a size that is different from the original size will be more resource intensive. To add a camera feed to the stream, use the command: IO video.stream = "cam[pip2]:openspace width:352 height:288 top:400 left:100"



Drawing Text

You can overlay text strings to the stream, choosing the font, style, color, and size of the text.

The text drawing syntax in EVENTS is:

```
IO <id>.stream = “text[<obj name>]:<text>;; [top:<Y>] [left:<X>] [font:<F>] [style:<[bold],[italic]>] [size:
<S>] [color:<C>]”
```

or:

```
IO <id>.stream = “text[<obj name>]:<on | off>
```

The on/off command is used to make an existing text visible or invisible.

Note that the text you are drawing must terminate with two semicolons. After this terminator you can define the optional drawing attributes. If there are no parameters, the two semicolons are not required.

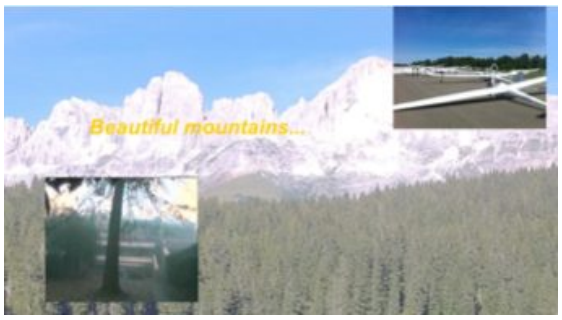
The font attribute is defined as a font name, like “arial”, “helvetica”, “courier” etc. The style can be “bold”, “italic”, or “bold,italic”.

The size is in pixels.

The color can be set using an RGB comma separated numeric notation, with primary color numbers from 0 to 255 (for example, “color:255,0,0” is red), or using color names. The valid names are: black, blue, cyan, darkgray, gray, lightgray, green, magenta, orange, pink, red, white, yellow.

To add text to the stream, use the command:

```
IO video.stream = "text[text]: Beautiful mountains... ;; top:300 left:200 font:Arial size:48
style:bold,italic color:orange"
```



Release Notes

3.0.0

- initial release

AXIS is a registered trademark of AXIS Communications.

BACnet

BACnet is a protocol developed by ASHRAE for building automation and control networking, specifically designed to meet the communication needs of building automation and control systems for applications such as heating, ventilating, air-conditioning control, lighting control, access control and fire detection systems. HSYCO conforms to a functional subset of the BACnet protocol, as defined in ANSI/ASHRAE Standard 135-2010. HSYCO can only be configured as a BACnet/IP client device. BACnet support is implemented as an I/O Server with a data point representation that you can use in the EVENTS language and Java to read and write BACnet's objects and properties, and to manage asynchronous notification services.

> The BACnet Vendor Identification Number that has been issued by ASHRAE to Home Systems Consulting SpA is 590.

Communication

HSYCO supports BACnet/IP, where the BACnet messaging protocol is transported over standard IP networks, over local networks. Routing of BACnet traffic is currently not supported by the HSYCO BACnet/IP I/O Server. You should configure only one I/O Server for each LAN used for BACnet traffic. Support for multiple BACnet/IP local networks is achieved by configuring one I/O Server for each network.

The BACnet I/O Server doesn't support segmented messages.

HSYCO Configuration

Add a BACNET I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: broadcast IP address of the LAN used for BACnet traffic (e.g. 192.168.1.255)
- **IP Port**: TCP/IP port to use, leave blank to use default port 47808.

High Availability

- **Shutdown when inactive**: defaults to false.

Options

ID	Default	Values	Description
deviceid	590001	0 ... 4194303	BACnet device identifier assigned to the HSYCO BACnet client
disableverbosedump	false	true	when verbose log is enabled, don't write IP packets dump to the log files
		false	when verbose log is enabled, all BACnet IP packets are written to the log files
eventslog	false	true	if the general eventsLog option is also true in System Settings,

			BACnet events for this gateway are written in the log files
		false	BACnet events for this gateway are not written in the log files, even if the eventsLog option is true in System Settings
gui	false	true	enables support of the (bacnetutility) Web GUI object
		false	disables support for the Web GUI

Datapoints

The following services are supported:

- Alarm and Event Services:
- ConfirmedCOVNotification
- UnconfirmedCOVNotification
- SubscribeCOV
- Object Access Services:
- ReadProperty
- WriteProperty
- Remote Device Management Services:
- Who-Is
- I-Am

The BACnet I/O Server supports all standard object types and properties for the ReadProperty service, while object's properties that could be written using the WriteProperty service are listed in the supported object types table.

Following HSYCO's naming convention for BACnet, all object types and properties are written in lowercase letters with no spaces or underscore characters. For example, Present_Value is written as "presentvalue", and the binary input object type is written as "binaryinput".

The BACnet datapoint names always begin with the same format:

```
<device_id>.<object_type>.<object_id>
```

where <device_id> is the BACnet device numeric identifier, <object_type> is the object type name with no spaces and all lowercase and <object_id> is the object instance numeric identifier.

Data point names returning properties values also have the object property identifier at the end with no spaces and all lowercase characters:

```
<device_id>.<object_type>.<object_id>.<property_id>
```

For example this datapoint

```
123.analogoutput.1.presentvalue
```

represents the present value of the analog output BACnet object "1" of the BACnet device "123".

The BACnet Utility provides a graphical interface to browse all BACnet/IP devices available on the network, to show all objects of a device, and all properties and values of any object instance.

Datapoints Table

ID	Value	R/W	Description
<device_id>.<object_type>.<object_id>	readproperty:<property_id>	W	executes the ReadProperty Service request, reading the <property_id> property of the <device_id>.<object_type>.<object_id> object
<device_id>.<object_type>.<object_id>.<property_id>	<value>	R	the <property_id> property <value> read from <device_id>.<object_type>.<object_id> object, formatted according to the application datatype
<device_id>.<object_type>.<object_id>[<N>]	readproperty:<property_id>	W	executes the ReadProperty Service request, reading the <property_id> property of the <device_id>.<object_type>.<object_id> object at index [<N>]
<device_id>.<object_type>.<object_id>.<property_id>[<N>]	<value>	R	the <property_id> property <value> read from <device_id>.<object_type>.<object_id> object, formatted according to the application datatype. (if this property is an array of values)
<device_id>.<object_type>.<object_id>	writeproperty:<property_id>:<value>	W	executes the WriteProperty Service request, writing <value> in the <property_id> property of <device_id>.<object_type>.<object_id> object , without specifying the priority
<device_id>.<object_type>.<object_id>	writeproperty/<P>:<property_id>:<value>	W	executes the WriteProperty Service request, writing <value> in the <property_id> property of <device_id>.<object_type>.<object_id> object , specifying the priority

			<P>, as a number from 1 to 16
<device_id>.<object_type>.<object_id>	writeproperty/<P>:<property_id>	W	cancel the last value set at priority <P> in the <property_id> property of <device_id>.<object_type>.<object_id>
<device_id>.<object_type>.<object_id>[<N>]	writeproperty:<property_id>:<value>	W	executes the WriteProperty Service request, writing <value> in the <property_id> property array index <N> of <device_id>.<object_type>.<object_id> object , without specifying the priority
<device_id>.<object_type>.<object_id>[<N>]	writeproperty/<P>:<property_id>:<value>	W	executes the WriteProperty Service request, writing <value> in the <property_id> property array index <N> of <device_id>.<object_type>.<object_id> object , specifying the priority <P>, as a number from 1 to 16
<device_id>.<object_type>.<object_id>[<N>]	writeproperty/<P>:<property_id>	W	cancel the last value set at priority <P> in the <property_id> property of <device_id>.<object_type>.<object_id> object
<device_id>.<object_type>.<object_id>	subscribecov	W	sends a SubscribeCOV request with an indefinite lifetime for <device_id>.<object_type>.<object_id> object
<device_id>.<object_type>.<object_id>	subscribecov:confirmed	W	sends a SubscribeCOV request with an indefinite lifetime and the confirmed notifications option enabled for <device_id>.<object_type>.<object_id> object
<device_id>.<object_type>.<object_id>	subscribecov:unconfirmed	W	sends a SubscribeCOV request with an indefinite lifetime and the confirmed notifications option

			disabled for <device_id>.<object_type>.<object_id> object
<device_id>.<object_type>.<object_id>	subscribecov:cancel	W	sends a SubscribeCOV cancellation request for <device_id>.<object_type>.<object_id> object

Supported Object Types

Object Type	Writable Properties
accumulator	• description • devicetype • presentvalue • pulserate • valuebeforechange • valueset
analoginput	• description • devicetype • covincrement • presentvalue
analogoutput	• description • devicetype • covincrement • presentvalue • relinquishdefault
analogvalue	• description • covincrement • presentvalue • relinquishdefault
averaging	• attemptedsamples • description • windowinterval • windowsamples
binaryinput	• activetext • description • devicetype • inactivetext • presentvalue

binaryoutput	• activetext • description • devicetype • inactivetext • presentvalue • relinquishdefault
binaryvalue	• activetext • description • inactivetext • presentvalue • relinquishdefault
command	• description • presentvalue
lifesafetypoint	• description • devicetype • mode • reliability • trackingvalue
lifesafetyzone	• description • devicetype • mode • reliability • trackingvalue
loop	• covincrement • description
multistateinput	• description • devicetype • presentvalue
multistateoutput	• description • devicetype • presentvalue • relinquishdefault
multistatevalue	• description • presentvalue
program	• description • programchange
pulseconverter	• adjustvalue • covincrement • description • presentvalue
accessdoor	• description

	• dooralarmstate • doorstatus • lockstatus • presentvalue • relinquishdefault
loadcontrol	• description • enable • requestedshedlevel (as Real number)

Supported Application Datatypes

Tag#	Description	HSYCO Data Type
0	Null	null
1	Boolean	0: false 1: true
2	Unsigned Integer	0 ... (2^{32} - 1)
3	Signed Integer	-2^{31} ... (2^{31} - 1)
4	Real (IEEE-754 floating point)	floating point number (with decimal point separator)
5	Double (IEEE-754 double precision floating point)	double precision floating point number (with decimal point separator)
6	Octet String	the hexadecimal representation of byte values, for example: “3b0ac2ff”
7	Character String	a string. The default character set is UTF-8. Additional supported character sets are: ISO-8859-1, UCS-2, EUC-JP
8	Bit String	a string representing a sequence of bit values, for example: “111010001”
9	Enumerated	0 ... (2^{32} - 1)
10	Date	<yyyy>/<mm>/<dd>[/<dow>] <yyyy> is the year number; a * character is a wildcard matching any year, or an unspecified value. <mm> is the month number, with January as 1; a * character is a wildcard matching any month, or an unspecified value; “odd” represent all odd months; “even” represent all even months. <dd> is the day number, starting from 1 for the first day of the month; a * character is a wildcard matching any day, or an unspecified value; “odd” represent all odd days; “even” represent all even days; “last” represent the last day of the month.

		<dow> is the week day, and is only present if set to a day of the week, and not to the wildcard/unspecified value; days are represented as “mon”, “tue”, “wed”, “thu”, “fri”, “sat”, “sun”
11	Time	<hh>:<mm>:<ss>.<ff> <hh> is the hour, using a 24-hour clock from 0 to 23; a * character is a wildcard matching any hour, or an unspecified value. <mm> is the minute, from 0 through 59; a * character is a wildcard matching any minute, or an unspecified value. <ss> is the seconds, from 0 through 59; a * character is a wildcard matching any second, or an unspecified value. <ff> is the hundreds of seconds number, from 0 to 99; a * character is a wildcard matching any fraction of second, or an unspecified value.
12	BACnetObjectIdentifier	<object_type>.<object_id> <object_type> is represented with the type name for supported types, and with the BACnet object type number for unsupported objects. <object_id> is the object instance number

Examples

Reading the present value of an object

```
TIME : IO bac.123.analogoutput.1 = readproperty:presentvalue
IO bac.123.analogoutput.1.presentvalue : uiset mydata.text = IO bac.123.analogoutput.1.presentvalue
```

The first line executes the ReadProperty service to poll device 123 for the value of the Present Value property of object type Analog Input, instance 1. The second line sets a text object of the Web interface with the value read from the BACnet device.

Reading several values at once

```
HSYCOSTART : PROGRAMTIMER readbacnet = repeat 10
PROGRAMTIMER readbacnet : IO bac.123.analogoutput.1 bac.123.analogoutput.2 = readproperty:presentvalue
IO bac.123.analogoutput.1.presentvalue : uiset mydata1.text = IO bac.123.analogoutput.1.presentvalue
IO bac.123.analogoutput.2.presentvalue : uiset mydata2.text = IO bac.123.analogoutput.2.presentvalue
```

The second line is a compact form for issuing the ReadProperty service for multiple objects. The first line shows how to create a programtimer instance that is executed every 10 seconds.

Writing the present value of an object

```
USER bac = 5 : IO bac.21.pulseconverter.0 = writeproperty:adjustvalue:-3
TIME : IO bac.21.analogvalue.0 = writeproperty/14:presentvalue: $myvalue
```

The first line sets the Adjust Value property of object type Pulse Converter, object instance 0 in device 21 to the value -3. The second line is similar, but this time the priority is set to 14. Also note that the value after "presentvalue:" is from a variable. To ensure that the variable name is translated to its value, leave a space before the variable name. This way the EVENTS interpreter will automatically concatenate the strings, also replacing the variable name with its value.

Subscribe for change of value notifications

```
TIME : IO bac.21.analoginput.0 bac.21.binaryinput.2 = subscribecov
TIME : IO bac.21.binaryvalue.0 = subscribecov:unconfirmed
```

The first line sends a SubscribeCOV request, with confirmed notifications, for two distinct objects. The second line also sends a SubscribeCOV request, but with unconfirmed notifications. This is a less reliable notification mechanism, but more efficient in terms of network usage and devices workload.

Cancel change of value notifications

```
TIME : IO bac.21.analoginput.0 bac.21.binaryinput.2 = subscribecov:cancel
```

This command cancels the previous change of value notification request.

Release Notes

3.3.0

- added support for the Relinquish_Default property
- the BACnet Utility is now fully integrated in the Manager tools

3.1.0

- initial release

BACnet is a registered trademark of the American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc.

Bentel

The Bentel KYO320 is an advanced, multi-area security system integrated HSYCO.

The KYO320 driver in HSYCO supports the integration of more than one KYO320 unit on the same HSYCO SERVER.

The integration with HSYCO can be accomplished through the standard RS-232 port on the KYO320's motherboard.

Communication

Employ a DE-9 (often called DB-9) female-female RS-232 crossed cable to connect the RS-232 KYO port to the serial port on HSYCO SERVER. The cable must follow this pinout diagram:

```
2 ↔ 3
3 ↔ 2
5 ↔ 5
```

RS-232 parameters:

Baud rate	same as defined on the KYO320 (usually 9600, 38400 or 57600)
Data bits	8
Stop bit	1
Parity	even
Flow control	none

HSYCO Configuration

Add a BENTEL I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select the comm port connected to the panel.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support
language	system language	en	language of log and display messages. Supported values are

		it	it and en
zoneactivitylog	false	true	logs the open/close status of zones in the log page and security.log files
		false	disable zone status log
userlogs	false	true	generates logs that can be viewed with the userlist object, and can be selected to generate user events
		false	logs for userlist objects disabled. Normal text logs are generated even when this option is false
code			the user code to be used when sending IO commands to the KYO, if code is not explicitly set in the command
logsize	50	n ≥ 0	the number of log lines to display in the UI object
polltime	1000	n ≥ 0	the data polling interval, in milliseconds. Shorter time improve the response of the system, but increase the load on the CPU of both HSYCO and the KYO system

bentel.ini

The bentel.ini file is an optional configuration file located in the main directory (same directory as hsyco.ini or hsyco.jar) and used to define zones and partitions names. If defined, these names will be used instead of the original numbers in the log files and the bentel GUI object.

The bentel.ini is automatically detected at start-up. Changes to this file become effective when the I/O Server is restarted.

For each zone or partition, enter a line formatted as:

```
<io_server_id>.zone.<n> = <name>
<io_server_id>.area.<n> = <name>
```

For instance:

```
kyo.zone.1 = kitchen
kyo.zone.2 = garage
kyo.area.1 = basement
kyo.area.2 = second floor
```

Datapoints

ID	Value	R/W	Description

connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
system.fault	1	R	general system fault
	0	R	reset
system.batteryfault	1	R	general battery fault
	0	R	reset
system.lowbattery	1	R	low battery
	0	R	reset
system.nobattery	1	R	low battery
	0	R	reset
system.nopower	1	R	general no AC power fault
	0	R	reset
system.tamper	1	R	general tamper condition (central unit, zones, modules)
	0	R	reset
system.radiofault	1	R	general radio modules fault
	0	R	reset
system.radiolowbattery	1	R	radio modules low battery
	0	R	reset
system.duress	1	R	duress status
	0	R	reset
system.maintenance	1	R	maintenance mode
	0	R	reset
zone.<n>.disabled	1	R	zone <n> disabled
	0	R	zone <n> enabled
zone.<n>.alarm	1	R	zone <n> alarm
	0	R	reset
zone.<n>.tamper	1	R	zone <n> tamper alarm
	0	R	reset

zone.<n>	enable	W	enable zone <n> using default code
	disable	W	disable zone <n> using default code
	enable.<c>	W	enable zone <n> using code <c>
	disable.<c>	W	disable zone <n> using code <c>
partition.<n>.armed	1	R	area <n> is armed (any mode)
	0	R	reset
partition.<n>.disarmed	1	R	area <n> is not armed
	0	R	reset
partition.<n>.away	1	R	area <n> away mode
	0	R	reset
partition.<n>.stay	1	R	area <n> stay mode
	0	R	reset
partition.<n>.staynodelay	1	R	area <n> stay no delay mode
	0	R	reset
partition.<n>.alarm	1	R	area <n> alarm memory
	0	R	reset
partition.<n>	away	W	arm area <n> using default code
	disarm	W	disarm area <n> using default code
	stay	W	stay arm area <n> using default code
	staynodelay	W	instant stay arm area <n> using default code
	away.<c>	W	arm area <n> using code <c>
	disarm.<c>	W	disarm area <n> using code <c>
	stay.<c>	W	stay arm area <n> using code <c>
	staynodelay.<c>	W	instant stay arm area <n> using code <c>
out.<n>	off	W	turn off output <n> using default code

	on	W	turn on output <n> using default code
	off.<c>	W	turn off output <n> using code <c>
	on.<c>	W	turn on output <n> using code <c>

User Interface

Bentel Object

The user interface for the Bentel KYO320 multi-area intrusion detection system:

The Bentel object is listed in the Project Editor's new object list only when at least one Bentel I/O Server is defined.

Parameters

- **server id**: the server ID
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(bentel <server id>; <position>)
```

E.g.

```
(bentel serverid; x10y20)
```

UISET Actions

ID	Attribute	Set to
areas	value	the status of all partitions
logo	value	the latest entry of the security log for a short period
logs	value	the latest entries of the security log (as many as specified in hsyco.ini)
zones	value	the status of all known zones (zones are added to this list as soon as an event is detected on a zone)

connection.label.online	visible	true when the panel is connected to HSYCO, false otherwise
connection.label.offline	visible	true when the panel is not connected to HSYCO, false otherwise
partition.<n>.label.alarm	visible	true when partition <n> alarm is active, false otherwise
partition.<n>.label.armed	visible	true when partition <n> is armed, false otherwise
partition.<n>.label.disarmed	visible	true when partition <n> is disarmed, false otherwise
partition.<n>.label.away	visible	true when partition <n> is armed in away mode, false otherwise
partition.<n>.label.stay	visible	true when partition <n> is armed in stay mode, false otherwise
partition.<n>.label.staynodelay	visible	true when partition <n> is armed in stay no delay mode, false otherwise
zone.<n>.label.alarm	visible	true when zone <n> alarm is active
zone.<n>.label.tamper	visible	true when zone <n> tamper alarm is active
zone.<n>.label.disabled	visible	true when zone <n> is disabled

Log Files

A permanent record of all the events shown in the log page is saved in a file called security.log in the logs/YYYY directory for the current year.

This file is never overwritten or deleted by HSYCO.

```
2014.07.30 20:09:21.760 – SECOND FLOOR – ARMED
2014.07.30 20:09:21.763 – SECOND FLOOR – AWAY
2014.07.30 20:10:02.364 – [2] GARAGE – OPEN
2014.07.30 20:10:04.779 – [2] GARAGE – OK
2014.07.30 20:10:06.167 – BASEMENT – ALARM
2014.07.30 20:10:48.477 – BASEMENT – DISARMED
2014.07.30 20:10:48.481 – SECOND FLOOR – DISARMED
2014.07.30 20:39:31.326 – ZONE 281 – OPEN
2014.07.30 20:39:36.708 – ZONE 281 – OK
2014.07.30 20:40:41.571 – AC POWER – FAULT
2014.07.30 20:49:58.945 – ZONE 281 – OPEN
2014.07.30 20:50:04.181 – ZONE 281 – OK
2014.07.30 20:50:11.559 – AC POWER – OK
```

Other information related to the Bentel KYO driver are saved in the daily message.log files.

Release Notes

3.3.0

- Bentel KYO driver converted to I/O Server
- Java API write command methods are now obsolete and deprecated, replaced by IO write commands
- new (bentel) object
- new option userlogs. Used to create (userlist) clickable logs

2.0.0

- full integration within HSYCO 2.9.0

1.1.0

- change to the status polling thread to improve recovery time after transient communication problems

---

Bentel, KYO and KYO320 are registered trademarks of BENTEL SECURITY SRL.

CSx75

Aritech CSx75 is a multi-area security system. The integration with HSYCO can be accomplished via direct control through the RS-232 port on its panel.

Communication

Use the Aritech CS590 RS232 cable to connect the panel directly to HSYCO or to an Ethernet/RS-232 gateway.

RS-232 parameters:

Baud rate	same as configured on panel (default 9600)
Data bits	8
Stop bit	1
Parity	none
Flow control	none

CSx75 Configuration

The panel must be previously configured to have the serial port enabled and to use the binary control protocol. Refer to its configuration manual for more details.

HSYCO Configuration

Add a CSX75 I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select the comm port connected to the panel.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
pin		<code>	4 or 6-digits code. Use the specified PIN code to perform operations that require user authentication. If this option is not specified it will be required to manually insert the pin every time such operations are performed

gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
pollinterval	3	n > 0	the data acquisition interval, in seconds
logsize	20	n ≥ 0	the number of log lines to display in the UI object

csx75.ini

The csx75.ini file is a specific configuration file located in the root directory. Here you can define the names of the users so that they will be automatically added to the user interface. Add a new line for each user using this format:

```
<io_server_id>.user.<n> = <name>
```

For instance:

```
cs.user.1 = Bob
cs.user.2 = Alice
```

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
alarm	0	R	no partition is alarmed
	1	R	at least one partition is alarmed
armed	0	R	all partitions are disarmed
		W	disarm all partitions
	1	R	the system is (partially) armed
	away	W	arm all partitions in away mode
	stay	W	arm all partitions in stay mode

	auto	W	initiate auto-arm for all partitions
p<n>.armed	0	R	partition <n> is disarmed
		W	disarm partition <n>
	away	R	partition <n> is armed in away mode
		W	arm partition <n> in away mode
	stay	R	partition <n> is armed in stay mode
		W	arm partition <n> in stay mode
	auto	W	initiate auto-arm on partition <n>
p<n>.alarm	0	R	partition <n> is not in an alarm state
	1	R	partition <n> is in an alarm state
p<n>.tamper	0	R	no tamper condition for partition <n>
	1	R	partition <n> has been tampered
p<n>.last.user	<num>	R	last user number entered was <num>
p<n>.fire	0	R	no fire on partition <n>
	1	R	fire on partition <n>
p<n>.fire.trouble	0	R	no fire trouble on partition <n>
	1	R	fire trouble on partition <n>
p<n>.buzzer	0	R	buzzer off on partition <n>
	1	R	buzzer on on partition <n>
		W	start buzzer on partition <n>
p<n>.tlm.fault	0	R	TLM ok
	1	R	TLM memory fault
p<n>.instant	0	R	instant flag not set on partition <n>
	1	R	instant flag set on partition <n>
p<n>.siren	0	R	siren off on partition <n>

	1	R	siren on on partition <n>
p<n>.steady.siren	0	R	steady siren off on partition <n>
	1	R	steady siren on on partition <n>
p<n>.alarm.mem	0	R	alarm memory flag not set on partition <n>
	1	R	alarm memory flag set on partition <n>
p<n>.cancel	0	R	no cancel command entered on partition <n>
	1	R	cancel command entered on partition <n>
		W	issue cancel command
p<n>.exit.silent	0	R	silent exit disabled on partition <n>
	1	R	silent exit enabled on partition <n>
		W	enable silent exit for this arm cycle on partition <n>
p<n>.chime.mode	0	R	chime mode off on partition <n>
	1	R	chime mode on on partition <n>
	toggle	W	toggle chime mode on partition <n>
p<n>.entry	0	R	entry inactive on partition <n>
	1	R	entry active on partition <n>
p<n>.delay.exp	0	R	delay expiration warning off
	1	R	delay expiration warning on
p<n>.exit.1	0	R	exit 1 inactive on partition <n>
	1	R	exit 1 active on partition <n>
p<n>.exit.2	0	R	exit 2 inactive on partition <n>
	1	R	exit 2 active on partition <n>
p<n>.exit.error	0	R	exit error not triggered on partition <n>
	1	R	exit error triggered on partition <n>

p<n>.night.mode	0	R	night mode inactive on partition <n>
	1	R	night mode active on partition <n>
p<n>.reexit	0	R	re-exit inactive on partition <n>
	1	R	re-exit active on partition <n>
p<n>.force.arm	0	R	no force arm triggered on partition <n>
	1	R	force arm triggered by auto arm on partition <n>
p<n>.arm.ready	0	R	partition <n> not ready to arm
	1	R	partition <n> ready to arm
p<n>.force.arm.ready	0	R	partition <n> not ready to force arm
	1	R	partition <n> ready to force arm
p<n>.chime	0	R	chime off on partition <n>
	1	R	chime on on partition <n>
p<n>.error.beep	0	R	error beep off on partition <n>
	1	R	error beep on on partition <n>
p<n>.tone	0	R	activation tone off on partition <n>
	1	R	activation tone on on partition <n>
p<n>.entry.1	0	R	entry 1 inactive on partition <n>
	1	R	entry 1 active on partition <n>
p<n>.phone.1.alarm	0	R	no alarm sent using phone number 1 on partition <n>
	1	R	alarm sent using phone number 1 on partition <n>
p<n>.phone.2.alarm	0	R	no alarm sent using phone number 2 on partition <n>
	1	R	alarm sent using phone number 2 on partition <n>
p<n>.phone.3.alarm	0	R	no alarm sent using phone number 3 on partition <n>
	1	R	alarm sent using phone number 3 on partition <n>

p<n>.keyswitch.armed	0	R	keyswitch not armed on partition <n>
	1	R	keyswitch armed on partition <n>
z<n>.name	<text>	R	name of zone <n> is set to <text>
z<n>.open	0	R	zone <n> is closed
	1	R	zone <n> is open
z<n>.tamper	0	R	zone <n> is not tampered
	1	R	zone <n> is tampered
z<n>.trouble	0	R	zone <n> OK
	1	R	troubles on zone <n>
z<n>.bypassed	0	R	zone <n> not bypassed
	1	R	zone <n> bypassed
	toggle	W	toggle the bypass status on zone <n>
z<n>.inhibited	0	R	zone <n> not inhibited
	1	R	zone <n> inhibited
z<n>.battery.low	0	R	zone <n> battery OK
	1	R	zone <n> battery low
z<n>.supervision.loss	0	R	supervision OK on zone <n>
	1	R	loss of supervision on zone <n>
z<n>.alarm.mem	0	R	alarm memory flag not set on zone <n>
	1	R	alarm memory flag set on zone <n>
z<n>.bypass.mem	0	R	bypass memory flag not set on zone <n>
	1	R	bypass memory flag set on zone <n>
user<n>.name	<text>	R	name of user <n> is set to <text>
comm.fail	0	R	communication OK
	1	R	communication failure
fuse.fault	0	R	fuse OK

	1	R	fuse fault
panel.tamper	0	R	panel box not tampered
	1	R	panel box tampered
siren.tamper	0	R	siren OK
	1	R	siren tampered
battery.low	0	R	battery OK
	1	R	battery low
ac.fail	0	R	AC OK
	1	R	AC failure
expander.tamper	0	R	expander box not tampered
	1	R	expander box tampered
expander.ac.fail	0	R	expander's AC OK
	1	R	expander's AC failure
expander.battery.low	0	R	expander's battery OK
	1	R	expander's battery low
expander.fault	0	R	expander OK
	1	R	general fault for the expander
buzzer	0	R	global buzzer off
	1	R	global buzzer on
siren	0	R	global siren off
	1	R	global siren on
steady.siren	0	R	global steady siren off
	1	R	global steady siren on
bus.fault	0	R	bus line OK
	1	R	bus device had line seized
ac.power	0	R	AC power off
	1	R	AC power on
battery.low.mem	0	R	low battery memory flag not set
	1	R	low battery memory flag set
powerup.delay	0	R	no power up delay in progress

	1	R	power up delay in progress
walk.test	0	R	walk-test mode off
		W	stop walk-test mode
	1	R	walk-test mode active
		W	start walk-test mode
time.loss	0	R	system time ok
	1	R	loss of system time
aux1 p<n>.aux1	1	W	activate auxiliary function 1
aux2 p<n>.aux2	1	W	activate auxiliary function 2
panic p<n>.panic	fire	W	issue the fire panic command
	medical	W	issue the medical panic command
	police	W	issue the police panic command
smoke p<n>.smoke	reset	W	smoke detector reset
test p<n>.test	1	W	perform test
command	restart	W	restart the driver
clock	read	W	request the reading of the panel's clock value
	<val>	R	the panel's clock is set to <val>. The used format is "yyyy-mm-dd hh:mm:ss" (e.g. 2012-08-30 16:30:00). This datapoint is only updated after a request with the command "clock = read"
		W	set the panel's clock to <val>. One of the following formats must be used: • "yyyy-mm-dd hh:mm", e.g. 2012-08-30 16:30 • "yy/mm/dd-wd-hh:mm", where "wd" represents the day of the week (1 = Sunday, 2 = Monday, ..., 7 = Saturday), e.g. 12/08/30-5-16:30

CS507 Configuration

The I/O Server also allows to configure and to check the saved configuration of a CS507 add-on module connected to the panel.

To read and write the configuration saved on the module use the following datapoints:

ID	Value	R/W	Description
cs507.<addr>.out<n>	read	W	request the reading of the configuration of output <n> on the module with address <addr>
	<out_value>	R	output <n> on the module with address <addr> is configured as described by <out_value>, updated only after a read request
		W	write <out_value> to output <n> on the module with address <addr>
cs507.<addr>.prog<n>	read	W	request the reading of the configuration of time-program <n> on the module with address <addr>
	<prog_value>	R	the time-program <n> on the module with address <addr> is configured as described by <prog_value>, updated only after a read request
		W	write <prog_value> to the time-program <n> on the module with address <addr>
cs507.<addr>.holidays.<n>	read	W	request the reading of the days set as holidays for month <n> (1 = January, 12 = December) on the module with address <addr>
	<holidays_value>	R	the holidays for month <n> on the module with address <addr> are set to <holidays_value>, updated only after a read request
		W	set <holidays_value> as holidays for month <n> on the module with address <addr>
cs507.<addr>.user<n>	read	W	request the reading of the configuration of user <n> on the module with address <addr>
	<user_value>	R	user <n> on the module with address <addr> is configured as

			described by <user_value>, updated only after a read request
		W	write <user_value> to user <n> on the module with address <addr>

Outputs Configuration Values

The output configuration value <out_value> has the following format:

```
<ev>,<par>,<d>,<f1>,<f2>,<f3>,<f4>,<f5>,<f6>,<f7>,<f8>,<a1>,<a2>,<a3>,<a4>,<a5>,<a6>,<a7>,<a8>,<p1>,<p2>,
<p3>,<p4>,<p5>,<p6>,<p7>,<p8>,<X10_addr>
```

it is a comma-separated list of the following parameters:

- ev = number of the event associated to this output
- par = value of the parameter associated to the specified event (e.g. user or zone number)
- d = duration of the event
- f1 ... f8 = '1' if the flag is set, '0' otherwise. f1 sets the “Timing in minutes” flag; f2 corresponds to “Memorized”; f3 to “Interrupt with code”; f4 to “Outside time slot”; f5 to “Inside time slot”; f6 to “Inverted”; f7 to “Save in events memory”; f8 to “Disable new auto-arm”
- a1 .. a8 = '1' if the area flag is set, '0' otherwise. a1 corresponds to area 1 and so on
- p1 .. p8 = '1' if the time-program flag is set, '0' otherwise. p1 corresponds to program 1 and so on
- X10_addr = X10 address associated with the output (e.g. "A3")

Time-programs Configuration Values

The time-program configuration value <prog_value> has the following format:

```
<start>,<end>,<sun>,<mon>,<tue>,<wed>,<thu>,<fri>,<sat>,<holidays>
```

it is a comma-separated list of the following parameters:

- start = starting time of the program. It must be formatted as HH:MM (e.g. 14:30)
- end = ending time of the program. It must be formatted as HH:MM (e.g. 14:30)
- sun ... sat = '1' if the program is active on the corresponding day, '0' otherwise
- holidays = '1' if this program is disabled during the configured holidays, '0' otherwise

Holidays Configuration Values

The holidays configuration value <holidays_value> has the following format:

```
<d1>,<d2>,<d3>,<d4>,<d5>,<d6>,<d7>,<d8>
```

it is a comma-separated list of the 8 days set as holidays for the specified month. If less than 8 days are set, then the remaining days are set to 0.

Users Configuration Values

The user configuration value <user_value> has the following format:

```
<o1>,<o2>,<o3>,<o4>,<o5>,<o6>,<o7>,<o8>
```

where <01> ... <08> are set to '1' if the corresponding output is enabled for the user, '0' otherwise.

User Interface

CSx75 Object

The user interface for the Aritech CSx75 multi-area security system.

The CSx75 object is listed in the Project Editor's new object list only when at least one Aritech CSx75 I/O Server is defined.

Parameters

- **server id**: the server ID
- **partition**: partition number of the Aritech CSx75 multi-area security system
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(csx75 <server id>; <partition>; <position>)
```

E.g.

```
(csx75 serverid; partition; x10y20)
```

UISET Actions

ID	Attribute	Set to	
log	value	latest security log lines (as many as specified by the 'logsize' option)	
logo	value	latest line of the security log when a new entry is added, set back to blank after a short period	
z<n>.name	value	name assigned to zone <n> (base or double) or its number if no name is defined	
p<n>.last.user	value	ID of the last user who logged in on partition <n>	
user<n>.name	value	name assigned to user <n>	
connection.label	visible	true	when datapoint connection = offline
		false	when datapoint connection = online
p<n>.armed.0.label	visible	true	when partition <n> is disarmed
		false	when partition <n> is armed
p<n>.armed.away.label	visible	true	when partition <n> is armed in "away" mode

		false	when partition <n> is not armed in "away" mode
p<n>.armed.stay.label	visible	true	when partition <n> is armed in "stay" mode
		false	when partition <n> is not armed in "stay" mode
<datapoint_id>.1.label	visible	true	when datapoint with ID <datapoint_id> has value '1'
		false	when datapoint with ID <datapoint_id> has value '0'
<datapoint_id>.0.label	visible	true	when datapoint with ID <datapoint_id> has value '0'
		false	when datapoint with ID <datapoint_id> has value '1'

USER Commands

Name	Param	Action
armed[Note]	0	disarm all partitions
	stay	arm all partitions in "stay" mode
	away	arm all partitions in "away" mode
	auto	initiate auto-arm on all partitions
p<n>.armed[Note]	0	disarm partition <n>
	stay	arm partition <n> in "stay" mode
	away	arm partition <n> in "away" mode
	auto	initiate auto-arm on partition <n>
p<n>.cancel[Note]	1	issue cancel command on partition <n>
p<n>.buzzer	1	start buzzer on partition <n>
p<n>.exit.silent	1	enable silent exit for this arm cycle on partition <n>
Z<n>.bypassed	toggle	toggle the bypass status of zone <n>
p<n>.chime.mode	toggle	toggle chime mode on partition <n>
walk.test[Note]	0	stop walk-test mode
	1	start walk-test mode
aux1 p<n>.aux1	1	activate auxiliary function 1

aux2 p<n>.aux2	1	activate auxiliary function 2
panic p<n>.panic	fire	issue the fire panic command
	medical	issue the medical panic command
	police	issue the police panic command
smoke p<n>.smoke	reset	smoke detector reset
test p<n>.test	1	perform test
command	restart	restarts the driver

Note

To perform this operation it is necessary to enter the user code first. To enter the user code through the user interface use user buttons to create a virtual keypad and assign them name and param values according to this table:

Name	Param	Action
digit	{0 ... 9}	append the specified digit to the entered user code
	c	delete the last entered digit

To visualize the number of entered digits use 4 or 6 text objects (depending on the number of digits used for user codes) and set their IDs to:

ID	Description
pin.<n> (0 ≤ <n> ≤ 5)	shows the character '*' when the corresponding digit has been entered, shows '_' otherwise

The entered code will be cleared after a short period of inactivity.

Log Files

A permanent record of all the CSX75 log messages, also shown in the log page, is saved in a file called security.log in the logs/YYYY directory for the current year.

This file is never overwritten or deleted by HSYCO.

```
2014.03.21 14:50:40.010 - cs - Disarmed - Partition 1
2014.03.21 14:52:01.322 - cs - Walk test mode on
2014.03.21 14:53:17.687 - cs - Walk test mode off
2014.03.21 14:53:28.852 - cs - Bypass - [2] Zona 2 n
2014.03.21 14:53:29.998 - cs - Armed (stay) - Partition 1
2014.03.21 14:53:32.557 - cs - Bypass restore - [2] Zona 2 n
2014.03.21 14:53:33.698 - cs - Disarmed - Partition 1
2014.03.21 17:17:03.364 - cs - Online
2014.03.21 17:40:42.664 - cs - Bypass - [2] Zona 2 n
2014.03.21 17:40:43.859 - cs - Armed (stay) - Partition 1
2014.03.21 17:40:48.772 - cs - Bypass restore - [2] Zona 2 n
2014.03.21 17:40:50.067 - cs - Disarmed - Partition 1
```

```
2014.03.21 17:48:25.277 - cs - Bypass - [1] Zona 1 n
2014.03.21 17:48:26.729 - cs - Bypass restore - [1] Zona 1 n
```

Other information related to the Aritech CSx75 driver are saved in the daily message.log files

Release Notes

3.2.1

- error log messages changed
- fixed bug that prevented to delete the first entered pin digit
- bug fix: user.IOStartupEvent() was not called at startup

3.2.0

- modified event “ID.clock”

3.1.0

- added the CS507 add-on module configuration

3.0.3

- initial version release

CSx75 and Aritech are registered trademarks of GE Security.

Daikin

Daikin air-conditioning systems in combination with the intelligent Touch Controller (iTC) allow for remote control and supervision, via Ethernet communication, of the whole framework.

This driver is compatible with the intelligent Touch Controller DCS601C51 and complies with the DCS007A51 http open protocol.

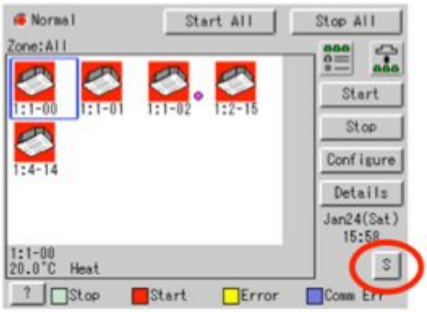
The HTTP license provide by Daikin is required.

Communication

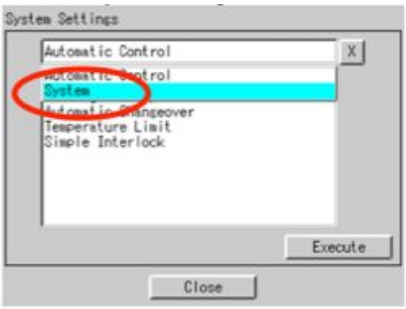
Use an Ethernet straight cable to connect the iTC to the network.

In order to be able to control the system with HSYCO some initial configuration is needed:

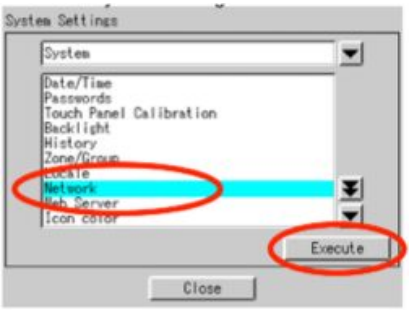
- From the main screen, press the S button:



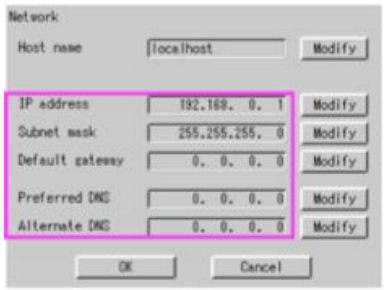
- Select System:



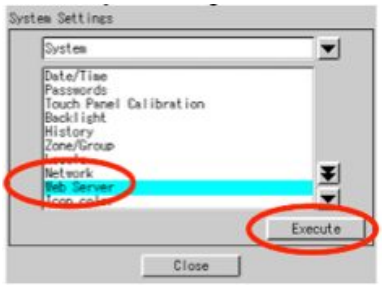
- Select Network and press Execute:



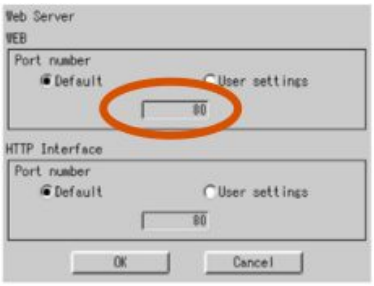
- Configure the network settings:



- Go back to the System Settings page, select Web Server and press Execute:



- Set the port number of the Web server:



System Architecture

The system can handle up to 64 devices of the following typologies: HRV, VRV, D3Dio, and D3Di. The driver allows to address each device individually, or grouped by type, or as a whole system. The following image shows a simple system configuration:

Depending on the type of the device, different settings can be monitored and/or set.

The following table shows the items that can be monitored (M) and set (S) for each type:

Item	VRV	HRV	D3Dio	D3Di
Power status	M/S	M/S	M/S	M
Operation mode	M/S	--	--	--
Ventilation modes and level	--	M/S	--	--

Suction temperature	M	--	--	--
Set-point temperature	M/S	--	--	--
Fan speed and direction	M/S	--	--	--
Filter signal	M/S	--	--	--
Malfunction status and code	M	M	M	M

HSYCO Configuration

Add a DAIKIN I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: IP address set on the iTC
- **IP Port**: TCP/IP port set on the iTC.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
pollinterval	20	n > 0	the data acquisition interval, in seconds

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established to the device
	offline	R	HSYCO can't connect to the device
power	0	R	the whole system is off

		W	turn all devices off
	1	R	at least one device is on
		W	turn all devices off
<type>.power	0	R	all devices of the specified type are off
		W	turn all devices of the specified type off
	1	R	at least one device of the specified type is on
		W	turn all devices of the specified type on
<n>.power	0	R	the addressed device is off
		W	turn the device off
	1	R	the addressed device is on
		W	turn the device on
	unknown	R	the power state of the device is not known
opmode <n>.opmode <type>.opmode	heat cool fan dependent dry	R	the operation mode of the target is set to the specified value [Note 1]
		W	set the operation mode of the target to the specified value
	auto	R	---
		W	set the operation mode of the target to the specified value
	ventilation auto-heat auto-cool	W	---
	unknown	R	the operation mode of the device is not known
vent.mode <n>.vent.mode <type>.vent.mode	auto exchange bypass	R	the ventilation mode of the target is set to the specified value[Note 1]
		W	set the ventilation mode of the target to the specified value
	unknown	R	the ventilation mode of the device is not known
		W	---

vent.level <n>.vent.level <type>.vent.level	auto low high	R	the ventilation of the target is set to the specified value [Note 1]
		W	set the ventilation level of the target to the specified value
	switch	R	---
		W	switch the ventilation level cycling among the values low, high, and auto
	unknown	R	the ventilation level of the device is not known
		W	---
vent.freshup <n>.vent.freshup <type>.vent.freshup	0	R	the ventilation fresh up option is disabled [Note 1]
		W	disable the ventilation fresh up option
	1	R	the ventilation fresh up option is enabled [Note 1]
		W	enable the ventilation fresh up option
	unknown	R	the ventilation fresh up setting of the device is not known
setpoint <n>.setpoint <type>.setpoint	<val>	R	the set-point temperature of the target is <val> (expressed in °C/10)[Note 1]
		W	set the set-point temperature to <val> (in °C/10)
	up	W	increase the set-point temperature of 0.5 °C
	down	W	decrease the set-point temperature of 0.5 °C
<n>.temp	<val>	R	the measured temperature from the target is <val> (expressed in °C/10)
fan.speed <n>.fan.speed <type>.fan.speed	low mid high	R	the fan speed of the target is set to the specified value[Note 1]
		W	set the fan speed to the specified value
	switch	W	switch the fan speed cycling among the values low, mid, and high

	unknown	R	the fan speed is not known
fan.direction <n>.fan.direction <type>.fan.direction	0 ... 4	R	the fan direction is set to the specified level[Note 1]
		W	set the fan direction to the specified level
	swing	R	the fan is in swing mode[Note 1]
		W	set the fan in swing mode
	switch	W	switch the fan mode cycling among 0, 1, 2, 3, 4, and swing
	unknown	R	the fan direction is not known
<n>.filter.sign	0	R	the filter signal is not active on the device
	1	R	the filter signal is active on the device
<n>.setpoint.enabled	0	R	the set point is disabled
	1	R	the set point is enabled
<n>.temp.enabled	0	R	the temperature measurement is disabled
	1	R	the temperature measurement is enabled
error <n>.error	0	R	the target is working properly
	1	R	an error has been detected
<n>.error.code	<xx>	R	the specified error code is reported (if <ioserver_id>.<n>.error = 1)

Note 1

Events concerning a group of devices (i.e. broadcast and type-related events) assume the value of the last updated device in the group. That is, for instance, if all VRVs are in operation mode “cool” and only one of them is changed to “heat” then the event <ioserver_id>.vrv.opmode = heat will trigger.

User Interface

UISET Actions

ID	Attribute	Set to
connection.label	visible	true when the device appears offline, false otherwise

power	value	shows the status of the system: ON: at least one AC is on; OFF: all devices are off
<type>.power	value	shows the status of the specified type-group: ON: at least one AC of the specified type is on; OFF: all devices of the specified type are off
<n>.power	value	shows the status of the addressed device: ON, OFF
power.label.on <n>.power.label.on <type>.power.label.on	visible	true when the addressed target is on, false otherwise
power.label.off <n>.power.label.off <type>.power.label.off"	visible	true when the addressed target is off, false otherwise
setpoint <n>.setpoint <type>.setpoint	value	shows the set-point temperature in °C
temp <n>.temp <type>.temp	value	shows the measured temperature in °C
opmode.label.<mode> <n>.opmode.label.MODE <type>.opmode.label.<mode>	visible	true when the operation mode of the target is set to the specified mode, false otherwise. The possible values for <mode> are: fan, heat, cool, dependent, dry, auto-heat, and auto-cool
vent.mode.label.MODE <n>.vent.mode.label.MODE <type>.vent.mode.label. <mode>	visible	true when the ventilation mode of the target is set to the specified mode, false otherwise. The possible values for <mode> are: auto, exchange, and bypass
vent.level.label.<val> <n>.vent.level.label.VAL <type>.vent.level.label.<val>	visible	true when the ventilation level of the target is set to the specified value, false otherwise. The possible values for <val> are: low, high, and auto
vent.freshup.label.on <n>.vent.freshup.label.on <type>.vent.freshup.label.on	visible	true when the ventilation fresh up option is enabled, false otherwise
vent.freshup.label.off <n>.vent.freshup.label.off <type>.vent.freshup.label.off	visible	true when the ventilation fresh up option is disabled, false otherwise.

fan.speed.label.<val> <n>.fan.speed.label.<val> <type>.fan.speed.label.<val>	visible	true when the fan speed of the target is set to the specified value, false otherwise. The possible values for <val> are: low, mid, and high
fan.direction.label.<val> <n>.fan.direction.label.VAL <type>.fan.direction.label.<val>	visible	true when the fan direction of the target is set to the specified value, false otherwise. The possible values for <val> are: 0, 1, 2, 3, 4 and swing
error.label <n>.error.label	visible	true when the system/device is in an error state, false otherwise
<n>.filter.sign.label	visible	true when the filter signal of the device is active, false otherwise

USER Commands

The Daikin I/O Server supports a few system commands to be inserted within the Web interface. The controls are directly supported as ordinary user objects, by setting the *name* and *param* fields according to the corresponding datapoints table.

For instance, setting the name of a user object to “dkn.1.power” and its param field to “on”, will result in having a switch for the device with address 1. Naming it “dkn.1.opmode” with a valid working mode as param (e.g. “heat” or “cold”) will result in a button for setting the mode of the device.

Release Notes

3.0.0

- initial release

Daikin and intelligent Touch Controller are registered trademarks of Daikin Corp.

ESC/VP

This I/O Server control SEIKO EPSON projectors which include a wide range of multi-source projectors remotely controllable via Ethernet, serial port, and USB connection. HSYCO SERVER integrates these devices through Ethernet connection.

The used protocol is compliant with the ESC/VP21 protocol specifications.

Projector Configuration

Use an Ethernet straight cable to connect the device to the HSYCO SERVER network.

In order to be able to control the device with HSYCO some initial configuration of the SEIKO EPSON device is needed.

In the “Network” section of the main menu, assign a suitable fixed IP address to the device and set the DHCP option to off.

Further, in the “Extended” section, make sure the option “Standby mode” is set to “communication on”.

HSYCO Configuration

Add an ESC/VP I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: IP address of the device
- **IP Port**: TCP/IP port to use, leave blank to use default port 3629.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for UISET actions and USER commands
		false	disable UI support
pollinterval	3	n > 0	the data acquisition interval, in seconds

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the device

power	1	R	the projector is on and ready
		W	turn on the projector
	0	R	the projector is off or not ready
		W	turn off the projector
status	on	R	the projector is on and ready
	off		the projector is off
	warming		the projector is warming up
	cooling		the projector is cooling down
	error		the projector is in a standby error state
lamp	0 ... 65535	R	the lamp hours
luminance	low	R	the luminance is set to low
		W	set luminance to low
	high	R	the luminance is set to high
		W	set luminance to high
mute	1	R	the A/V mute is on
		W	set A/V mute to on
	0	R	the A/V mute is off
		W	set A/V mute to off
msel	black	R	the mute screen is set to black
		W	set the mute screen to black
	blue	R	the mute screen is set to blue
		W	set the mute screen to blue
	logo	R	the mute screen is set to show the user logo
		W	set the mute screen to show the user logo
source	<src>	R	the current source is <src>
		W	set the source to <src>
command	<cmd>	W	send <cmd> as command to the projector. The command must be compliant with the ESC/VP21 protocol

Source values

The source value <src> consists of a two-characters code which variates depending on the model of the projector. For instance, for the model G5000, the code “21” corresponds to the RGB input and the code “45” to the BNC video source. Refer to the ESC/VP21 Command User’s Guide for a complete list of source codes for the different models.

User Interface

UISET Actions

ID	Attribute	Set to	
connection	visible	true	when datapoint connection = offline
		false	when datapoint connection = online
power	value	ON	when the projector is on and ready
		OFF	when the projector is off or not ready
lamp	value	number of hours of activity of the lamp.	
	visible	true	when the projector is on
		false	when the projector is off
luminance	value	HIGH	when the luminance is set to high
		LOW	when the luminance is set to low
	visible	true	when the projector is on
		false	when the projector is off
mute	value	ON	when the A/V mute function is on
		OFF	when the A/V mute function is off
msel	value	BLACK	when the mute screen is set to black
		BLUE	when the mute screen is set to blue
		LOGO	when the mute screen is set to show the user logo

USER Commands

Name	Param	Action

power	on	turn on the projector
	off	turn off the projector
luminance	low	set the luminance to low
	high	set the luminance to high
mute	on	turn on the A/V mute
	off	turn off the A/V mute
source	<src>	set the source to <src>
msel	black	set the A/V mute screen to black
	blue	set the A/V mute screen to blue
	logo	set the A/V mute screen to show the user logo

Release Notes

3.2.0

- minor fixes to the I/O Server

1.0.0

- initial release

SEIKO EPSON is a registered trademark of SEIKO EPSON Corporation.

Elmo

EL.MO. ETR panels are multi-area intrusion detection systems. The integration with HSYCO can be accomplished via direct control through the RS232 port on the panel.

Communication

Employ the RS-232 cable provided with the EL.MO. panel to connect it directly to HSYCO or to an Ethernet/RS-232 gateway.

RS-232 parameters:

Baud rate	9600
Data bits	8
Stop bit	1
Parity	none
Flow control	none

EL.MO. Configuration

For the newer "G2" series of EL.MO. ETR panels it is necessary to enable the compatibility mode for direct connection in the panel configuration. Refer to its manual for further details.

HSYCO Configuration

Add an ELMO I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select the comm port connected to the panel.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support
startupevents	false	true	generate IO events also during the driver's start-up phase

		false	start generating events only after HSYCO is aligned with the current status of the system
offlinelogs	true	false	at startup request the log of all the events occurred while HSYCO was disconnected from the panel
		false	log only events occurred while HSYCO was connected to the panel
pollinterval	1	n > 0	the data acquisition interval, in seconds
logsize	20	n ≥ 0	the number of log lines to display in the UI object
maxinput		n ≥ 0	highest input number among the inputs used on the panel. If not specified all the available inputs will be included. This option is recommended to improve the I/O Server's performances
maxoutput		n ≥ 0	highest output number among the outputs used on the panel. If not specified all the available outputs will be included. This option is recommended to improve the I/O Server's performances
maxgroup		n ≥ 0	highest group number among the groups used on the panel. If not specified all the available groups will be included. This option is recommended to improve the I/O Server's performances
maxuser		n ≥ 0	highest user number among the users used on the panel. If not specified all the available users will be included. This option is recommended to improve the I/O Server's performances
userlogs	false	true	generates logs that can be viewed with the userlist object, and can be selected to generate user events
		false	logs for userlist objects disabled. Normal text logs are generated even when this option is false

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
armed	1	R	at least one partition is armed
		W	arm all active areas
	0	R	no partition is armed
		W	disarm all active areas
armed.max	1	R	at least one partition is armed in "max security" mode
		W	arm all active areas in "max security" mode
	0	R	no partition is armed in "max security" mode
		W	disarm all active areas in "max security" mode
alarm	1	R	alarm state active
	0	R	no active alarm
tamper	1	R	tamper alarm active
	0	R	no active tamper alarm
i<n>.name	<text>	R	the name of input <n> is set to <text>
i<n>.open	1	R	input <n> open
	0	R	input <n> closed
i<n>.alarm	1	R	alarm on input <n>
	0	R	no alarm on input <n>
i<n>.tamper	1	R	tamper alarm on input <n>
	0	R	no tamper alarm on input <n>
i<n>.bypassed	1	R	input <n> bypassed
		W	bypass input <n>
	0	R	input <n> enabled
		W	enable input <n>

o<n>	1	R	output <n> active
		W	activate output <n>
	0	R	output <n> inactive
		W	deactivate output <n>
a<n>.<m>.name	<text>	R	the name of partition <m> of area <n> is set to <text>
a<n>.armed	1	W	arm all partitions of area <n>
	0	W	disarm all partitions of area <n>
a<n>.armed.max	1	W	arm all partitions of area <n> in "max security" mode
	0	W	disarm all partitions of area <n> in "max security" mode
a<n>.<m>.armed	1	R	partition <m> of area <n> is armed
		W	arm partition <m> of area <n>
	0	R	partition <m> of area <n> is disarmed
		W	disarm partition <m> of area <n>
a<n>.<m>.armed.max	1	R	partition <m> of area <n> is armed in "max security" mode
		W	arm partition <m> of area <n> in "max security" mode
	0	R	partition <m> of area <n> is not armed in "max security" mode
		W	disarm partition <m> of area <n> in "max security" mode
usr<n>.name	<text>	R	the name of user <n> is set to <text>
kp<n>.tamper	1	R	keypad <n> tampered
	0	R	keypad <n> not tampered
fault.battery	1	R	battery fault active
	0	R	no battery fault active
panel.tamper	1	R	panel tampered
	0	R	panel not tampered
rel.tamper	1	R	tamper relay is on

	0	R	tamper relay is off
rel.alarm	1	R	alarm relay is on
	0	R	alarm relay is off

User Interface

Elmo Object

The user interface for the EL.MO. multi-area intrusion detection system:

The (elmo) object is listed in the Project Editor's new object list only when at least one EL.MO. I/O Server is defined

Parameters

- **server id**: the server ID
- **area**:
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(elmo <server id>; <area>; <position>)
```

E.g.

```
(elmo elm; 1; x10y20)
```

UISET Actions

ID	Attribute	Set to	
log	value	latest security log lines (as many as specified by the 'logsize' option)	
log0	value	latest line of the security log when a new entry is added, set back to blank after a short period	
i<n>.name	value	name assigned to input <n>	
a<n>.<m>.name	value	name assigned to partition <m> of area <n>	
usr<n>.name	value	name assigned to user <n>	
connection.label	visible	true	when datapoint connection = offline
		false	when datapoint connection = online
userlogs (only if option userlogs=true)	labels	comma-separated list of log entries	

	parameters	comma-separated list of log entries with format "yyyyMMddHHmmss message" (e.g. "20141230154513 ALARM ZONE 1")	
<datapoint_id>.label.1	visible	true	when datapoint with ID <datapoint_id> has value '1'
		false	when datapoint with ID <datapoint_id> has value '0'
<datapoint_id>.label.0	visible	true	when datapoint with ID <datapoint_id> has value '0'
		false	when datapoint with ID <datapoint_id> has value '1'

USER Commands

Name	Param	Action
armed	1	arm all active areas
	0	disarm all active areas
armed.max	1	arm all active areas in "max security" mode
	0	disarm all active areas in "max security" mode
a<n>.armed	1	arm all partitions of area <n>
	0	disarm all partitions of area <n>
a<n>.armed.max	1	arm all partitions of area <n> in "max security" mode
	0	disarm all partitions of area <n> in "max security" mode
a<n>.<m>.armed	1	arm partition <m> of area <n>
	0	disarm partition <m> of area <n>
a<n>.<m>.armed.max	1	arm partition <m> of area <n> in "max security" mode
	0	disarm partition <m> of area <n> in "max security" mode
i<n>.bypassed	1	bypass input <n>
	0	enable input <n>
o<n>	1	activate output <n>
	0	deactivate output <n>

Log Files

A permanent detailed record of all the EL.MO. log messages is saved in a file called security.log in the logs/YYYY directory for the current year. This file is never overwritten or deleted by HSYCO.

Other information related to the ELMO driver are saved in the daily message.log files.

Release Notes

3.3.0

- added “offlinelogs” option
- fixed bug on startup events

3.2.3

- fixes to the (elmo) object

3.1.0

- initial version release

EL.MO. and ETR100 are registered trademarks of EL.MO. SpA.

Elsner

The Elsner P03/3-MODBUS is a solid-state weather station with a standard Modbus RTU slave interface. It measures temperature, wind speed and brightness (eastern, southern and western sun) and recognizes precipitation.

Communication

The P03/3 has a Modbus interface. The default connection parameters are:

- Modbus address: 1
- RS-232 parameters:

Baud rate	19200
Data bits	8
Stop bit	1
Parity	even
Flow control	none

Refer to the P03/3 data sheet for installation, wiring and power supply requirements.

HSYCO Configuration

Add an ELSNERP03M I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select a MODBUSTCP I/O server.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description

gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
pollinterval	30	> 1	the data acquisition interval, in seconds
address	1	> 1	the Modbus address of the P03/3 device

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established to the P03/3
	offline	R	HSYCO can't connect to the P03/3
temp	signed integer -40C .. +80C	R	signed integer representing the measured temperature, in 10th of Celsius degrees (e.g. 210 corresponds to 21.0C)
light	positive integer 0 .. 99	R	ambient light in kilolux, equal to the maximum value detected by the three side sensors
light.east	positive integer 0 .. 99	R	ambient light in kilolux, detected from East-side sensor
light.south	positive integer 0 .. 99	R	ambient light in kilolux, detected from South-side sensor
light.west	positive integer 0 .. 99	R	ambient light in kilolux, detected from West-side sensor
wind	positive integer	R	wind in 10th of m/s (e.g. 32 corresponds to 3.2 m/s)
rain	0	R	no rain
	1	R	rain

User Interface

UISET Actions

ID	Attribute	Set to
temp	value	the current temperature in Celsius degrees with one decimal digit, followed by " °C"
light	value	the current ambient light in Klux, followed by " Klux"
wind	value	the current wind in m/s, followed by " m/s"

Release Notes

2.10

- full integration in HSYCO 2.10.0

Elsner is a registered trademark of Elsner Elektronik GmbH.

Fp2000

Aritech Interlogix FP2000 is a multi-area fire panel. The integration with HSYCO can be accomplished via direct control through the RS232 port on the panel. The FP2000 I/O Server in HSYCO supports the integration of more than one unit within the same HSYCO interface.

Communication

Employ an RS-232 cable to connect the panel directly to HSYCO.

RS-232 parameters:

Baud rate	same as panel settings (default 9600)
Data bits	8
Stop bit	1
Parity	none
Flow control	RTS/CTS

HSYCO Configuration

Add a FP2000 I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select the comm port connected to the panel.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
accesscode	0	0	do not request for authentication (the I/O Server

			will only be usable as panel emulator)
		1 ... 9999	use the specified value as access code for requesting authentication
keypaddisable	false	true	disable the emulator keypad buttons
		false	the emulator keypad buttons are enabled
controldisable	false	true	disable the emulator control buttons
		false	the emulator control buttons are enabled
displayhlcolor	339933	000000 ... FFFFFF	use the specified hexadecimal value as the highlight color for the display of the emulator

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
emulation	1	R	emulation is active
		W	activate emulation
	0	R	emulation is disabled
		W	disable emulation
led.general.fire.1 led.general.fire.2 led.general.processor led.general.supply.fault led.general.disabled led.general.fault led.general.supply led.general.system.fault led.brigade.delay.off led.brigade.delay.on led.brigade.stop led.brigade.fault led.brigade.signal led.sounder.delay.on led.sounder.silence led.sounder.fault led.sounder.sound led.sounder.delay.off led.control.test	on	R	the corresponding LED is on
	off	R	the corresponding LED is off
	blink	W	the corresponding LED is on, blinking

led.control.disabled led.control.silence.buzzer led.all led.panel			
status.alarm.count	<num>	R	count of fires
status.fault.count	<num>	R	count of faults
status.condition.count	<num>	R	count of conditions
status.event.count	<num>	R	count of all the events other than fire, fault or condition
clock	<date_time>	R	the panel's clock is set to <date_time>. The value has the format "yyyy-mm-dd hh:mm:ss" (e.g. "2012-12-30 16:59:23")
		W	set the data and time of the panel to <date_time>. The value has the format "yyyy-mm-dd hh:mm:ss" (e.g. "2012-12-30 16:59:23")
	read	W	request for reading date and time set on the panel
l<n>.s<m>.name	<name>	R	reports the name of sensor <m> on loop <n>
l<n>.s<m>.alarm.fire	1	R	fire alarm active for sensor <m> on loop <n>
	0	R	no fire alarm active for sensor <m> on loop <n>
l<n>.s<m>.alarm.communication	1	R	fault alarm active for sensor <m> on loop <n>
	0	R	no fault alarm active for sensor <m> on loop <n>
l<n>.s<m>.alarm.disabled	1	R	disabled alarm active for sensor <m> on loop <n>
	0	R	no disabled alarm active for sensor <m> on loop <n>
l<n>.s<m>.alarm.type	1	R	wrong type alarm active for sensor <m> on loop <n>
	0	R	no wrong type alarm active for sensor <m> on loop <n>
l<n>.s<m>.alarm.address	1	R	double address alarm active for sensor <m> on loop <n>
	0	R	no double address alarm active for sensor <m> on loop <n>

l<n>.s<m>.alarm.condition	1	R	pre condition alarm active for sensor <m> on loop <n>
	0	R	no pre condition alarm active for sensor <m> on loop <n>
l<n>.s<m>.alarm.maintenance	1	R	maintenance alarm active for sensor <m> on loop <n>
	0	R	no maintenance alarm active for sensor <m> on loop <n>
l<n>.s<m>.alarm.notype	1	R	no type alarm active for sensor <m> on loop <n>
	0	R	type alarm active for sensor <m> on loop <n>
l<n>.s<m>.alarm.enabled	1	R	enabled alarm active for sensor <m> on loop <n>
	0	R	no enabled alarm active for sensor <m> on loop <n>
l<n>.s<m>.alarm.twoset	1	R	two set alarm active for sensor <m> on loop <n>
	0	R	no two set alarm active for sensor <m> on loop <n>
l<n>.s<m>.alarm.segment	1	R	7 segment alarm active for sensor <m> on loop <n>
	0	R	no 7 segment alarm active for sensor <m> on loop <n>
l<n>.s<m>.alarm.led	1	R	LED alarm active for sensor <m> on loop <n>
	0	R	no LED alarm active for sensor <m> on loop <n>
l<n>.s<m>.alarm.segment.cont	1	R	7 segment continuous alarm active for sensor <m> on loop <n>
	0	R	no 7 segment continuous alarm active for sensor <m> on loop <n>
l<n>.s<m>.alarm.flag	1	R	flag alarm active for sensor <m> on loop <n>
	0	R	no flag alarm active for sensor <m> on loop <n>
l<n>.s<m>.alarm.pending	1	R	alarm pending for sensor <m> on loop <n>

	0	R	no alarm pending for sensor <m> on loop <n>
l<n>.s<m>.alarm	1	R	one of the above alarms is active for sensor <m> on loop <n>
	0	R	none of the above alarms is active for sensor <m> on loop <n>
l<n>.s<m>.alarm.count	<num>	R	reports the number of alarm states been active for sensor <m> on loop <n>
z<n>.name	<name>	R	reports the name of zone <n>
z<n>.alarm.fire	1	R	MCP fire alarm active on zone <n>
	0	R	no MCP fire alarm active on zone <n>
z<n>.alarm.fault	1	R	fault alarm active on zone <n>
	0	R	no fault alarm active on zone <n>
z<n>.alarm.coincidence	1	R	coincidence alarm active on zone <n>
	0	R	no coincidence fire alarm active on zone <n>
z<n>.alarm.isolated	1	R	isolated alarm active on zone <n>
	0	R	no isolated alarm active on zone <n>
z<n>.alarm.test	1	R	test alarm active on zone <n>
	0	R	no test alarm active on zone <n>
z<n>.alarm.condition	1	R	condition alarm active on zone <n>
	0	R	no condition alarm active on zone <n>
z<n>.alarm.enabled	1	R	enabled alarm active on zone <n>
	0	R	no enabled alarm active on zone <n>
z<n>.alarm.autofire	1	R	auto fire alarm active on zone <n>

	0	R	no auto fire alarm active on zone <n>
z<n>.alarm.action	1	R	action zone alarm active on zone <n>
	0	R	no action zone alarm active on zone <n>
z<n>.alarm.warning	1	R	pre warning alarm active on zone <n>
	0	R	no pre warning alarm active on zone <n>
z<n>.zone.alarm	1	R	one of the above alarms is active on zone <n>
	0	R	none of the above alarms is active on zone <n>
z<n>.alarm.count	<num>	R	reports the number of alarm states active on sensors of zone <n>
z<n>.fault.count	<num>	R	reports the number of faults states active on sensors of zone <n>
z<n>.condition.count	<num>	R	reports the number of conditions active on sensors of zone <n>
z<n>.coincidence.count	<num>	R	reports the number of coincidence alarms active on sensors of zone <n>
z<n>.isolate.count	<num>	R	reports the number of isolations in zone <n>
a<n>.name	<name>	R	reports the name of area <n>
a<n>.alarm.fire	1	R	MCP fire alarm active on area <n>
	0	R	no MCP fire alarm active on area <n>
a<n>.alarm.fault	1	R	fault alarm active on area <n>
	0	R	no fault alarm active on area <n>
a<n>.alarm.coincidence	1	R	coincidence alarm active on area <n>
	0	R	no coincidence alarm active on area <n>

a<n>.alarm.isolated	1	R	isolated alarm active on area <n>
	0	R	no isolated alarm active on area <n>
a<n>.alarm.condition	1	R	condition alarm active on area <n>
	0	R	no condition alarm active on area <n>
a<n>.alarm.enabled	1	R	enabled alarm active on area <n>
	0	R	no enabled alarm active on area <n>
a<n>.alarm.action	1	R	action alarm active on area <n>
	0	R	no action alarm active on area <n>
a<n>.area.alarm	1	R	one of the above alarms is active on area <n>
	0	R	none of the above alarms is active on area <n>
a<n>.alarm.count	<num>	R	reports the number of alarm states active on sensors of area <n>
a<n>.fault.count	<num>	R	reports the number of faults states active on sensors of area <n>
a<n>.condition.count	<num>	R	reports the number of conditions active on sensors of area <n>
a<n>.coincidence.count	<num>	R	reports the number of coincidence alarms active on sensors of area <n>
a<n>.isolate.count	<num>	R	reports the number of isolations in area <n>
o<n>.name	<name>	R	reports the number of output <n>
o<n>.status[Note 1]	1	R	output <n> is active
	0	R	output <n> is inactive
	read	W	request for reading the state of output <n>
i<n>.name	<name>	R	reports the number of input <n>

i<n>.status	1	R	input <n> is active
	0	R	input <n> is inactive
	read	W	request for reading the state of input <n>
event.action.text event.fire.text event.fault.text event.condition.text	<text>	R	reports the text of the most recently logged event accordingly to its category
key[Note 2]	<key_value>	W	send the specified emulation key command to the panel (see following table for the list of key values)

Note 1

the status of outputs and inputs is based on the reported events on the event log, thus the panel must be set so that outputs and inputs state variations are logged. Otherwise the reading of their status must be requested via the "o<n>.status = read" and "i<n>.status = read" commands.

Note 2

The following table shows the association between key values and relative function:

<key_value>	Function
1	Scroll
2	Display Alarm
4	Print Screen
5	Alpha Numeric
6	Right Arrow
8	Left Arrow
9	Silence Buzzer
10	Down Arrow
11	Reset
12	Disable
13	Enter
14	Test
15	Sound Sounder
16	Sounder Delay
17	Sounder Disable
18	Silence Sounder

19	Fire Brigade Disable
20	Fire Brigade Delay
22	Fire Brigade Stop
26	Up Arrow
27	Exit
29	Panel
30	All
31	Fire Brigade Signal
48	Key “0”
49	Key “1”
50	Key “2”
51	Key “3”
52	Key “4”
53	Key “5”
54	Key “6”
55	Key “7”
56	Key “8”
57	Key “9”
211	Fire Protection on
212	Fire Protection Silenced
213	Fire Protection Silenced
214	Fire Protection Enabled
215	Fire Protection Disabled
216	Fire Protection Disable
217	Fire Protection Delay off
218	Fire Protection Delay on
219	Fire Protection Delay
220	Fire Protection Delay
221	Fire Protection Delay
222	Fault Routing Test

223	Fault Routing Enabled
224	Fault Routing Disabled
225	Fault Routing Disable
226	Fault Routing Delay off
227	Fault Routing Delay on
228	Fault Routing Delay
229	Fire Brigade Test
230	Fire Brigade Enabled
231	Fire Brigade Disabled
232	Fire Brigade Delay off
233	Fire Brigade Delay on
234	Sounder Test
235	Sounder Enable
236	Sounder Disable
237	Sounder Delay off
238	Sounder Delay on
239	External Reset (from Repeater)

User Interface

FP2000 Object

The user interface for the Aritech Interlogix FP2000 multi-area fire panel:

File:UI Object fp2000.png

The Fp2000 object is listed in the Project Editor's new object list only when at least one Aritech Interlogix FP2000 I/O Server is defined.

Parameters

- **server id**: the server ID
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(fp2000 <server id>; <position>)
```

E.g.

```
(fp2000 serverid; x10y20)
```

UISET Actions

ID	Attribute	Set to	
log<n>	value	latest security log lines (as many as specified by the 'logsize' option, with <n> = 1 ... 20)	
log0	value	latest line of the security log when a new entry is added, set back to blank after a short period	
a<n>.name	value	name assigned to area <n>	
z<n>.name	value	name assigned to zone <n>	
l<n>.s<m>.name	value	name assigned to sensor <m> on loop <n>	
i<n>.name	value	name assigned to input <n>	
o<n>.name	value	name assigned to output <n>	
display	value	text displayed by the panel in real-time	
connection.label	visible	true	when datapoint connection = online
		false	when datapoint connection = offline
l<n>.sM.alarm.label.1	visible	true	when an alarm is active for sensor <m> on loop <n>
		false	when an alarm is not active for sensor <m> on loop <n>
l<n>.sM.alarm.label.0	visible	true	when no alarm is active for sensor <m> on loop <n>
		false	when an alarm is active for sensor <m> on loop <n>
o<n>.status.label.1	visible	true	when output <n> is active
		false	when output <n> is not active
o<n>.status.label.0	visible	true	when output <n> is not active
		false	when output <n> is active
i<n>.status.label.1	visible	true	when input <n> is active
		false	when input <n> is not active
i<n>.status.label.0	visible	true	when input <n> is not active
		false	when input <n> is active

led.<name>.label.off	visible	true	when the corresponding LED is off
		false	when the corresponding LED is on
led.<name>.label.on[Note 3]	visible	true	when the corresponding LED is on
		false	when the corresponding LED is off

Note 3

Replace <name> with one of the following strings:

```
general.fire.1
general.fire.2
general.fault
general.disabled
general.supply.fault
general.system.fault
general.processor
general.supply
control.silence.buzzer
control.disabled
control.test
sounder.sound
sounder.delay.on
sounder.delay.off
sounder.fault
sounder.silence
brigade.signal
brigade.delay.on
brigade.delay.off
brigade.fault
brigade.stop
panel
all
```

USER Commands

The FP2000 driver supports system commands to be inserted within the Web interface that replicates the functionalities of the buttons on the panel and more. Use any ordinary user button, setting the *name* and *param* fields according to the write function of the Datapoints table.

For instance, setting the name of a user button to “fp.key” and its param field to “11”, will result in having the reset button of the panel. Naming it “01.status” and setting “read” as param will result in a button for forcing the reading of the status of output 1.

Log Files

A permanent detailed record of all the FP2000 log messages is saved in a file called security.log in the logs/YYYY directory for the current year. This file is never overwritten or deleted by HSYCO.

Other information related to the FP2000 driver are saved in the daily message.log files.

Release Notes

3.1.0

- initial version release

Aritech Interlogix and FP2000 are registered trademarks of UTC Fire & Security.

GSM

This I/O Server provides services for sending and receiving SMSs and events generation for incoming calls via serial GSM modem.

The employed protocol complies with the AT commands set specified by the GSM standard set. It has been tested on the GenPro 16e and GenPro 18e modems by Erco&Gener and on the AirLink FXT009 modem by Sierra Wireless.

Communication

Employ a RS-232 straight cable to connect the modem directly to HSYCO or to an Ethernet/RS-232 gateway.

RS-232 parameters:

Baud rate	9600 (GenPro modems)
	115200 (default on AirLink FXT009 modems)
Data bits	8
Stop bit	1
Parity	none
Flow control	RTS/CTS

HSYCO Configuration

Add a GSM I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select the comm port the device is connected to.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
pin		<code>	4-digits PIN code of the SIM card (if used)
gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support
inboxsize	10	n ≥ 0	the number of incoming messages to display in the UI

			object
countrycode		<code>	the default country code (e.g. “39”, for Italy) to be used when not otherwise specified. If omitted all provided numbers shall include the country code
srvc		<number>	phone number of the SMS service center (use the international format, e.g. +393492000200). If not specified, the default number saved on the SIM card will be used. If not available, the driver will not be able to sent SMS messages
resend	4	n ≥ 0	number of sending attempts on an outgoing SMS in case of error

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the modem
sms.<num>	<text>	R	an SMS message has been received form the number <num>[Note 2] with <text> as body
		W	send an SMS message to the number <num>[Note 1] with <text> as body
call	<num>	R	incoming call from number <num>[Note 2] in progress. An event is generated for each “ring” during the call
	unknown	R	incoming call from an unknown number in progress
sms.<num>.sent	<mr>	R	a message has been successfully sent to the number <num> and the message reference number is <mr>
sms.<num>.errors	<val>	R	an error occurred while sending a message to number <num>. The value <val> is an increasing counter of the occurred errors since the I/O Server started

signal	{0 ... 31}	R	the GSM signal level has the reported value (ranging from 0 to 31)
	unknown	R	the GSM signal level is unknown

Note 1

If the option 'countrycode' is used, then the country code will be automatically added to the specified numbers, unless the latter begins with a '+' character. For instance, assuming the option 'countrycode' is set to “39”, the addressee number “3480077000” will be automatically changed to “+393480077000”, while the number “+46765007700” will remain unchanged. If, on the other hand, no default country code has been specified, then the addressee numbers must always be provided prepending the country code.

Note 2

The numbers are reported as sent from the modem and the format may vary depending on the number itself and the modem settings. If the reported number is an international number, then a '+' will be prepended to it, unless the country code is equal to the value of the 'countrycode' option. In the last case the country code will be omitted. In some cases, the sender number field might also be replaced by an alphanumeric string.

User Interface

SMS Object

The user interface for sending SMS messages:

File:UI Object sms.png

The SMS object is listed in the Project Editor’s new object list only when at least one GSM I/O Server is defined.

Parameters

- **server id**: the server ID
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(sms <server id>; <position>)
```

E.g.

```
(sms serverid; x10y20)
```

UISET Actions

ID	Attribute	Set to	
inbox	value	latest incoming SMS messages (as many as specified by the 'inboxsize' option)	
connection.label.online	visible	true	when datapoint connection = online
		false	when datapoint connection = offline

connection.label.offline	visible	true	when datapoint connection = offline
		false	when datapoint connection = online
signal.label.0	visible	true	when the GSM signal level is equal to 0
		false	when the GSM signal level is not 0
signal.label.1	visible	true	when the GSM signal level is between 1 and 8
		false	when the GSM signal level is not between 1 and 8
signal.label.2	visible	true	when the GSM signal level is between 9 and 16
		false	when the GSM signal level is not between 9 and 16
signal.label.3	visible	true	when the GSM signal level is between 17 and 24
		false	when the GSM signal level is not between 17 and 24
signal.label.4	visible	true	when the GSM signal level is above 24
		false	when the GSM signal level is below 25
signal.label.unknown	visible	true	when the GSM signal level is unknown
		false	when the GSM signal level is known
sending.label	visible	true	when the modem is sending a message
		false	when the modem is not sending a message
error.label	visible blink	true	for 5 seconds after there's been an error sending a message
		false	default value

USER Commands

UI Object	ID
input	<ioserver_id>.sms.number

input	<ioserver_id>.sms.text
submit	<ioserver_id>.sms.send

From the resulting interface, digit the addressee number (or comma-separated list of numbers) in the input field with ID “<ioserver_id>.sms.number” and the body text in the input field “<ioserver_id>.sms.text”, then press the submit button.

N.B. If the option 'countrycode' is used, then the country code will be automatically added to the numbers, unless the latter begins with a '+'. For instance, assuming the value of the option 'countrycode' is set to “39”, the addressee number “3480077000” will be automatically changed to “+393480077000”, while the number “+46765007700” will remain unchanged. If, on the other hand, no default country code has been specified, then the addressee numbers must always be provided prepending the country code.

PUK-locked SIM

In case the SIM card used for the GSM modem is PUK-locked, it is required to provide the PUK code and a new PIN to set on the SIM card. To this end, create in the root folder of HSYCO a file named “<ioserver_id>.puk”, where <ioserver_id> is the ID of the corresponding I/O Server (e.g. “gsm.puk”). In this file add a single line containing the PUK code and the new PIN code separated by a comma, for instance:

```
12345678,1234
```

The file will be automatically deleted after it has been used by the I/O Server.

Examples

Sending an SMS

Assuming that we are sending an SMS at 12 o' clock to number 777123456 located in Italy (International prefix code: +39) and the option "countrycode" of the GSM IO server is not specified, this is the corresponding EVENTS programming declaration:

```
time = 1200 : io gsm.sms.+39777123456 = "I'm sending and sms to the world"
```

You can send multiple SMSs to different numbers using this syntax:

```
time = 1200 : io gsm.sms.+39777123456 gsm.sms.+39777654321 = "I'm sending an sms to the world"
```

Receiving an SMS

Hsyco can receive SMSs through the GSM IO server and perform actions according to the text contained in the SMS.

```
io gsm.sms.777123456 = "Help" : io gsm.sms.777123456 = "I accept the following commands: Open, Status"
io gsm.sms.777123456 = "Open" : io knx.1/1/2 = 1, wait = 1, io knx.1/1/2 = 0
```

Release Notes

3.3.0

- added “resend” server option
- improved AT commands compatibility
- added PUK-unlock functionality

3.2.1

- added “srvc” server option
- bug fix: user.IOStartupEvent() was not called at startup

3.1.0

- initial release

GenPro 16e, GenPro 18e, and Erco&Gener are registered trademarks of Erco&Gener.

Galaxy

The Honeywell Galaxy Dimension panels are multi-area intrusion detection systems.

The integration with HSYCO can be accomplished via RS-232 serial connection and has been tested with firmware version 6.80.

Communication

Employ a RS-232 straight cable to connect the panel directly to HSYCO or to an Ethernet/RS-232 gateway.

RS-232 parameters:

Baud rate	9600
Data bits	8
Stop bit	1
Parity	none
Flow control	none

Galaxy Configuration

It is required to configure the panel for HSYCO to be able to communicate with it. Follow these instructions:

- Enable the engineer access
  Menu 48.1.1 - set to 1 (Enabled)
- Enable engineer mode
  Enter engineer PIN, followed by ENT
- Enable remote access and events reporting
  Menu 56.6.2 - set to 1 (SIA)
  Set SIA level to 4
  Enable (set to ON) all the events to be sent to HSYCO
  Menu 56.6.3 - set an account number (e.g. “0001”)

HSYCO Configuration

Add a Galaxy I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select the comm port connected to the panel.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
pin		<code>	PIN code assigned to the “remote” user. This option is mandatory
maxzone	4158	0 ... 4158	highest zone number among the zones used on the panel. This option is recommended to improve the I/O Server's performances
maxgroup	32	0 ... 32	highest index among the groups used on the panel. This option is recommended to improve the I/O Server's performances
maxoutput	256	0 ... 256	highest index among the outputs used on the panel. This option is recommended to improve the I/O Server's performances
guiobject	true	true	enable support for the galaxy UI object
		false	disable support for the galaxy UI object
guizones	false	true	enable the zones-related UISET actions
		false	disable the zones-related UISET actions
guioutputs	false	true	enable the outputs-related UISET actions
		false	disable the outputs-related UISET actions
startupevents	false	true	generate IO events also during the driver’s start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
pollinterval	5	n > 0	the data acquisition interval, in seconds
logsize	20	n ≥ 0	the number of log lines to display in the UI object
showall	false	true	in the zones list of the galaxy UI object show all the zones up to the number specified in the 'maxzone' option

		false	in the zones list of the galaxy UI object show only the zones for which a name has been defined in galaxy.ini

galaxy.ini

The galaxy.ini file is a specific configuration file located in the root directory. Here you can define the names of the zones so that they will be automatically added to the user interface. Add a new line for each zone using this format:

```
<io_server_id>.z<n> = <name>
```

For instance:

```
gxy.z1001 = Entrance
gxy.z1002 = Corridor
gxy.z2001 = Room 1
```

Datapoints

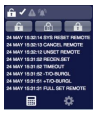
ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
g<n>.armed	0	R	group <n> is disarmed
		W	disarm group <n>
	tot	R	group <n> is armed in “away” mode
		W	arm group <n> in “away” mode
	part	R	group <n> is armed in “stay” mode
		W	arm group <n> in “stay” mode
	abort	W	abort group <n> arming
	force	W	force group <n> arming
g<n>.alarm	0	R	no active alarm on group <n>
	1	R	alarm on group <n>
	reset	R	alarms reset required on group <n>
g<n>.ready	0	R	group <n> is not ready to be armed

	1	R	group <n> is ready to be armed
g<n>.reset	1	W	reset alarms on group <n>
z<n>.open	0	R	zone <n> is closed
	1	R	zone <n> is open
z<n>.alarm	0	R	zone <n> is not in alarm
	1	R	zone <n> is in alarm
z<n>.bypassed	0	R	zone <n> is not bypassed
		W	unbypass zone <n>
	1	R	zone <n> is bypassed
		W	bypass zone <n>
z<n>.tamper	0	R	zone <n> is not tampered
	1	R	zone <n> is tampered
z<n>.fault	0	R	zone <n> is not in fault
	1	R	zone <n> is in fault
z<n>.masked	0	R	zone <n> is not masked
	1	R	zone <n> is masked
z<n>.resistance	0	R	zone <n> has normal resistance status
	1	R	zone <n> has high or low resistance
warning	0	R	all zones are working properly
	1	R	at least one zone is in fault or is masked or has high or low resistance
o<n>.active	0	R	output <n> is not active
		W	deactivate output <n>
	1	R	output <n> is active
		W	activate output <n>

User Interface

Galaxy Object

The user interfacce for the Galaxy multi-area intrusion detection system:



The Galaxy object is listed in the Project Editor's new object list only when at least one Galaxy I/O Server is defined.

Parameters

- **server id**: the server ID
- **group number**:
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(galaxy <server id>; <group number>; <position>)
```

E.g.

```
(galaxy serverid; group number; x10y20)
```

UISET Actions

ID	Attribute	Set to	
log	value	latest security log lines (as many as specified by the 'logsize' option)	
logo	value	latest line of the security log when a new entry is added, set back to blank after a short period	
z<n>.name	value	name assigned to zone <n> or its number if no name is defined	
connection.label.online	visible	true	when datapoint connection = online
		false	when datapoint connection = offline
connection.label.offline	visible	true	when datapoint connection = offline
		false	when datapoint connection = online
<datapoint_id>.label.1	visible	true	when datapoint with ID <datapoint_id> has value '1'
		false	when datapoint with ID <datapoint_id> has value '0'
<datapoint_id>.label.0	visible	true	when datapoint with ID <datapoint_id> has value '0'
		false	when datapoint with ID <datapoint_id> has value '1'

USER Commands

Name	Param	Action
g<n>.armed	0	disarm group <n>
	tot	arm group <n> in “away” mode
	part	arm group <n> in “stay” mode
	abort	abort group <n> arming
	force	force group <n> arming
g<n>.reset	1	reset alarms on group <n>
z<n>.bypassed	0	unbypass zone <n>
	1	bypass zone <n>
o<n>.active	0	deactivate output <n>
	1	activate output <n>

Log Files

A permanent detailed record of all the Galaxy log messages is saved in a file called security.log in the logs/YYYY directory for the current year. This file is never overwritten or deleted by HSYCO.

Other information related to the Galaxy I/O Server are saved in the daily message.log files.

Release Notes

3.3.0

- initial version release

Honeywell and Galaxy Dimension are registered trademarks of Honeywell International Inc.

Giacomini

The Giacoklima bus system equipped with the KM203 network controller allows for remote control and supervision, via Serial communication, of the whole heating/cooling framework.

Communication

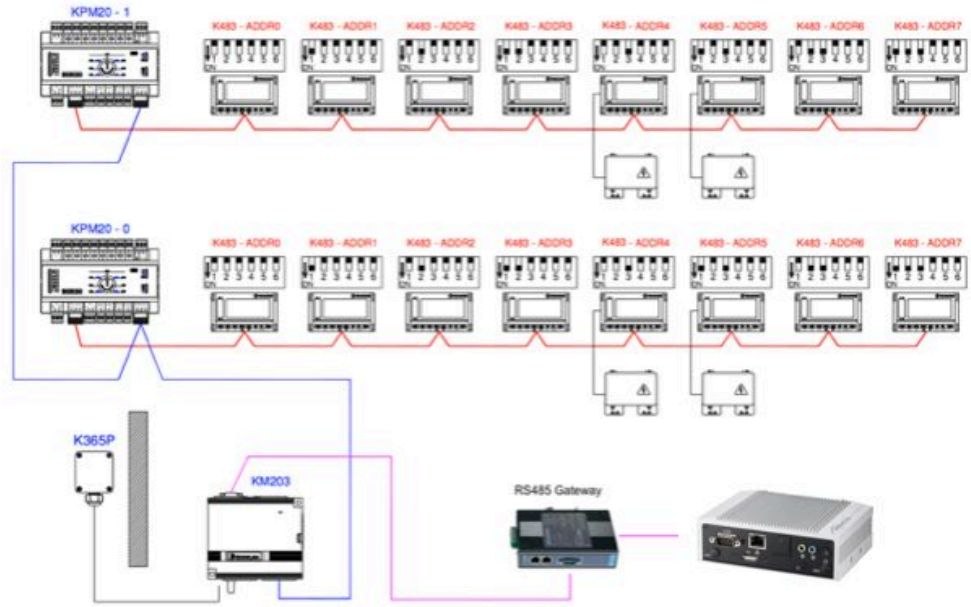
Connect HSYCO to the KM203 controller employing an RS485 gateway.

RS-232 parameters:

Baud rate	19200
Data bits	8
Stop bit	1
Parity	none
Flow control	none

The Giacoklima bus architecture

This image illustrates an example of the system architecture:



The RS-485 port is on the top left corner of the KM203 interface as shown in the picture above.

HSYCO Configuration

Add a GKBUS I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select the comm port connected to the device.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
pollinterval	1	n > 0	the data acquisition interval, in seconds

gkbus.ini

The gkbus.ini file is a specific configuration file located in the main directory (same directory as hsyco.ini or hsyco.jar).

Here you must add a line for each thermostat belonging to the framework and optionally specify a descriptive name of the corresponding area.

Use the format "<ioserver_id>.k<n>.<addr> = name", where <ioserver_id> is the id of the referred I/O server, <n> is the number of the KPM20 module the thermostat is connected to and <addr> is the secondary address of the thermostat.

For instance:

```
gk.k0.0 = Kitchen
gk.k0.1 = Bedroom
gk.k1.0 = Entrance
```

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
k<n>.<addr>.name	<name>	R	the name of the addressed thermostat is set to the reported string

k<n>.<addr>.temp	<val>	R	the room temperature measured by the addressed thermostat is equal to <val> in °C/10
k<n>.<addr>.setpoint.temp	<val>	R	the set-point temperature measured by the addressed thermostat is equal to <val> in °C/10
		W	set the set-point temperature of the addressed thermostat to <val> in °C/10
k<n>.<addr>.humidity	<val>	R	the relative humidity measured by the addressed thermostat is equal to <val>
season	summer	R	the current season is set to summer
		W	set the current season to summer
	winter	R	the current season is set to winter
		W	set the current season to winter
k<n>.<addr>.mode	off	R	the addressed thermostat is set to mode “OFF”
		W	set the addressed thermostat to mode “OFF”
	com	R	the addressed thermostat is set to mode “Comfort”
		W	set the addressed thermostat to mode “Comfort”
	stb	R	the addressed thermostat is set to mode “Stand-by”
		W	set the addressed thermostat to mode “Stand-by”
	ngt	R	the addressed thermostat is set to mode “Night"
		W	set the addressed thermostat to mode “Night”
mode	off	W	set all the thermostats to mode “OFF”
	com	W	set all the thermostats to mode “Comfort”
	stb	W	set all the thermostats to mode “Stand-by”

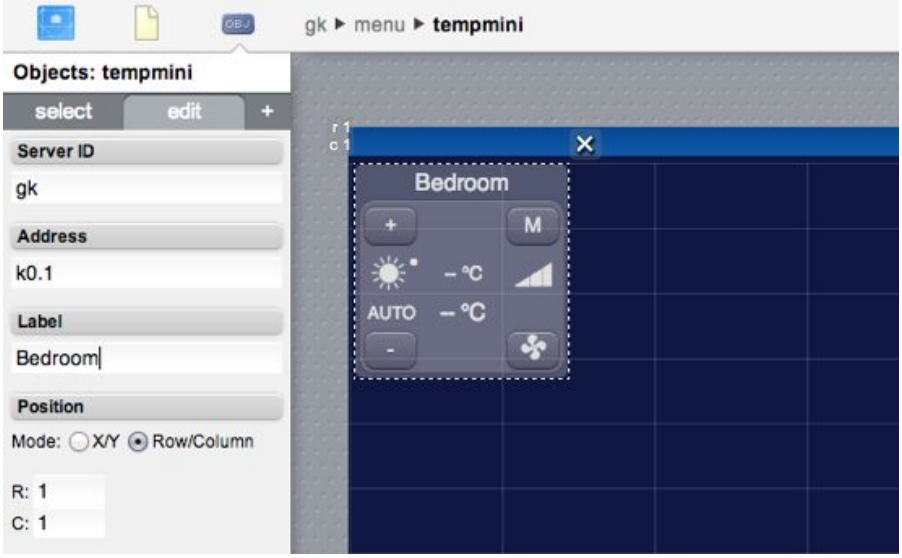
	ngt	W	set all the thermostats to mode “Night”
k<n>.pump	0	R	the pump of the addressed KPM20 module is not active
	1	R	the pump of the addressed KPM20 module is active
k<n>.in.temp	<val>	R	the inlet temperature of the addressed KPM20 module is equal to <val>
k<n>.out.temp	<val>	R	the return temperature of the addressed KPM20 module is equal to <val> in °C/10
k<n>.setpoint.temp	<val>	R	the working set-point temperature of the addressed KPM20 module is equal to <val> in °C/10
k<n>.valve	<val>	R	the opening percentage value of the mixing valve calculated by the addressed KPM20 module is equal to val
ext.temp	<val>	R	the measured external temperature is equal to <val> in °C/10
t1.winter	<val>	R	the T1 temperature parameter’s value for winter climatic compensation is equal to <val> in °C/10
		W	set the T1 temperature parameter’s value for winter climatic compensation to <val> in °C/10
t2.winter	<val>	R	the T2 temperature parameter’s value for winter climatic compensation is equal to <val> in °C/10
		W	set the T2 temperature parameter’s value for winter climatic compensation to <val> in °C/10
lim.winter	<val>	R	the T-Lim temperature parameter’s value for winter climatic compensation is equal to <val> in °C/10
		W	set the T-Lim temperature parameter’s value for winter climatic compensation to <val> in °C/10

setpoint.winter	<val>	R	the minimum set-point temperature for winter is equal to <val> in °C/10
		W	set the minimum set-point temperature for winter to <val> in °C/10
setpoint.summer	<val>	R	the minimum set-point temperature for summer is equal to <val> in °C/10
		W	set the minimum set-point temperature for summer to <val> in °C/10

User Interface

You can use a tempmini object to create a thermostat-equivalent interface.

Set the Server ID field to the I/O Server's name and specify the address of the thermostat (K483 module) in the Address field using the format "k<n>.<addr>" where <n> is the number of the KPM20 module the thermostat is connected to and <addr> is its address (e.g. k0.1 represents the second thermostat connected to the first KPM20):



UISET Actions

ID	Attribute	Set to
k<n>.<addr>.name	value	shows the name assigned to the addressed thermostat
k<n>.<addr>.temp	value	shows the measured room temperature in °C by the addressed thermostat
k<n>.<addr>.setpoint.temp	value	shows the set-point temperature in °C on the addressed thermostat

k<n>.<addr>.humidity	value	shows the measured relative humidity by the addressed thermostat
k<n>.in.temp	value	shows the inlet temperature of the addressed KPM20 module
k<n>.out.temp	value	shows the return temperature of the addressed KPM20 module
k<n>.setpoint.temp	value	shows the working set-point temperature in °C on the addressed KPM20 module
k<n>.valve	value	shows the opening percentage value of the mixing valve calculated by the addressed KPM20 module
ext.temp	value	shows the measured external temperature in °C
t1.winter	value	shows the T1 temperature parameter's value for winter climatic compensation in °C
t2.winter	value	shows the T2 temperature parameter's value for winter climatic compensation in °C
lim.winter	value	shows the T-Lim temperature parameter's value for winter climatic compensation in °C
setpoint.winter	value	shows the minimum set-point temperature for winter
setpoint.summer	value	shows the minimum set-point temperature for summer
connection.label.online	visible	true when the system is connected to HSYCO, false otherwise
connection.label.offline	visible	true when the system is not connected to HSYCO, false otherwise
season.label.winter	visible	true when the current season is set to winter, false otherwise
season.label.summer	visible	true when the current season is set to summer, false otherwise
k<n>.<addr>.mode.label.off	visible	true when the addressed thermostat is OFF, false otherwise
k<n>.<addr>.mode.label.com	visible	true when the working mode of the addressed thermostat is set

		to “comfort”, false otherwise
k<n>.<addr>.mode.label.stb	visible	true when the working mode of the addressed thermostat is set to “stand-by”, false otherwise
k<n>.<addr>.mode.label.ngt	visible	true when the working mode of the addressed thermostat is set to “night”, false otherwise
k<n>.<addr>.relay.label.0	visible	true when the relay of the addressed thermostat is not active, false otherwise
k<n>.<addr>.relay.label.1	visible	true when the relay of the addressed thermostat is active, false otherwise
k<n>.pump.label.0	visible	true when the pump of the addressed KPM20 module is not active, false otherwise
k<n>.pump.label.1	visible	true when the pump of the addressed KPM20 module is active, false otherwise

USER Commands

Name	Param	Action
k<n>.<addr>.setpoint.temp	<val>	sets the set-point temperature of the addressed thermostat to the specified value (expressed in °C/10)
	up	increases the set-point temperature of the addressed thermostat by 0.5°C
	down	decreases the set-point temperature of the addressed thermostat by 0.5°C
k<n>.<addr>.mode	off	sets the addressed thermostat on mode “OFF”
	com	sets the addressed thermostat on mode “Comfort”
	stb	sets the addressed thermostat on mode “Stand-by”
	ngt	sets the addressed thermostat on mode “Night”
season	summer	sets the working mode season to summer
	winter	sets the working mode season to winter
t1.winter	<val>	sets the T1 temperature parameter’s value for winter climatic compensation to the specified value (in °C/10)

t2.winter	<val>	sets the T2 temperature parameter's value for winter climatic compensation to the specified value (in °C/10)
lim.winter	<val>	sets the T-Lim temperature parameter's value for winter climatic compensation to the specified value (in °C/10)
setpoint.winter	<val>	sets the minimum set-point temperature for winter to the specified value (in °C/10)
setpoint.summer	<val>	sets the minimum set-point temperature for summer to the specified value (in °C/10)

Release Notes

3.3.0

- initial version release

Giacomini, GKBus and Giacoklima are registered trademarks of Giacomini SpA.

Release Notes

Guardall

Guardall QX32i, PX80 and PX500 are multi-area security systems. The integration with HSYCO can be accomplished via direct control through the RS-232 port on the panel. Th I/O Server supports QX32i, PX80 and PX500 Guardall units with software version 4.33.

Communication

Employ a female-female RS-232 crossed cable to connect the panel directly to HSYCO or to an Ethernet/RS-232 gateway. The cable must follow this pinout diagram:

```
2 ↔ 3
3 ↔ 2
5 ↔ 5
7 ↔ 8
8 ↔ 7
```

RS-232 parameters:

Baud rate	9600
Data bits	8
Stop bit	1
Parity	none
Flow control	none

Guardall Configuration

You must configure the unit to enable serial communication even if the system is armed. Digit the technician code on the keypad and select menu entry "99" (Total Configuration), then select entry “10” (Communication) and make sure that the serial connection is enabled when the system is armed (on).

Moreover, it is necessary to enable the required features to be available remotely. To this end, from the technician’s menu, select again menu entry “99”, then “12”. Here enter the number 1 and modify the settings accordingly to your requirements.

HSYCO Configuration

Add an Elmo I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select the comm port connected to the panel.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
logsize	20	n ≥ 0	the number of log lines to display in the UI object
arm	true	true	allow groups arming
		false	deny groups arming
disarm	true	true	allow groups disarming
		false	deny groups disarming

guardall.ini

The guardall.ini file is a specific configuration file located in the root directory. Here you can define the names of the groups so that they will be automatically added to the user interface. Add a new line for each group using this format:

```
<io_server_id>.group.<n> = <name>
```

For instance:

```
guard.group.1 = Kitchen
guard.group.2 = Living room
```

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
armed	0	R	all groups are disarmed
		W	disarm all groups
	1	R	the system is (partially) armed
		W	arm all groups

alarm	0	R	the system is not in a alarm state
	1	R	the system is in a alarm state
tamper	0	R	the system is not in a tamper state
	1	R	the system is in a tamper state
panel.tamper	0	R	the panel is not tampered
	1	R	the panel is tampered
bell.tamper	0	R	the bell is not tampered
	1	R	the bell is tampered
low.volts	0	R	voltage ok
	1	R	low voltage detected
sounder	0	R	the sounder is inactive
	1	R	the sounder is active
audio	0	R	the audio signal of the panel is inactive
	1	R	the audio signal of the panel is active
strobe	0	R	the strobe is inactive
	1	R	the strobe is active
mains.fail	0	R	no mains fail
	1	R	mains fail
power.fail	0	R	no power fail
	1	R	power fail
battery.fail	0	R	no battery fail
	1	R	battery fail
fuse.1.fault	0	R	fuse 1 is working
	1	R	fuse 1 in fault
fuse.2.fault	0	R	fuse 2 is working
	1	R	fuse 2 in fault
fuse.3.fault	0	R	fuse 3 is working
	1	R	fuse 3 in fault

line.fault	0	R	communication line is working
	1	R	communication line in fault
c<n>.name	<text>	R	the name of circuit <n> is set to <text>
g<n>.name	<text>	R	the name of group <n> is set to <text>
g<n>.armed	0	R	group <n> is disarmed
		W	disarm group <n>
	1	R	group <n> is armed
		W	arm group <n>
c<n>.alarm	0	R	circuit <n> is not in alarm state
	1	R	circuit <n> is in alarm state
c<n>.tamper.alarm	0	R	circuit <n> is not in tamper alarm
	1	R	circuit <n> is in tamper alarm
c<n>.open	0	R	circuit <n> is closed
	1	R	circuit <n> is open
c<n>.tamper	0	R	circuit <n> is not tampered
	1	R	circuit <n> is tampered
c<n>.fault	0	R	circuit <n> is working properly
	1	R	circuit <n> is faulty
c<n>.bypassed	0	R	circuit <n> is not bypassed
		W	unbypass circuit <n>
	1	R	circuit <n> is bypassed
		W	bypass circuit <n>
c<n>.isolated	0	R	circuit <n> is not isolated
		W	un-isolate circuit <n>
	1	R	circuit <n> is isolated
		W	isolate circuit <n>
c<n>.soaked	0	R	circuit <n> is not soaked
		W	un-soak circuit <n>

	1	R	circuit <n> is soaked
		W	soak circuit <n>
c<n>.status	<status_values>	R	the status of circuit <n> is set to the reported values
c<n>.attributes	<attributes_values>	R	the attributes of circuit <n> are set to the reported values
c<n>.resistance	<value>	R	the measured resistance on circuit <n> is <value> ohm
	>20k	R	the measured resistance on circuit <n> is greater than 20 K-ohm
k<n>.tamper	0	R	keypad <n> is not tampered
	1	R	keypad <n> is tampered
k<n>.fault	0	R	keypad <n> is working properly
	1	R	keypad <n> is faulty
logon.<n>	<user_num>	R	user with number <user_num> logged in from keypad <n>
	error	R	wrong pin entered in keypad <n>
system	reset	W	reset alarms

Circuit status values	
Value	**Description**
A	alarm/open
B	alarm/open and tamper
E	latched alarm and tamper
M	marginal clear
m	marginal alarm
N	normal
T	tamper
L	latched alarm
F	fault/no answer
X	Auxiliary alarm

Circuit attributes values

Value	Description
B	bypassed circuit
I	isolated circuit
S	soak failed
T	soaked circuit
Y	temporarily bypassed circuit
Q	temporarily isolated circuit
H	circuit in shunt
A	existing circuit
M	multi-alarm
F	bank reset failed/door forced
L	locked
O	open

Attributes can be combined together, for instance if a circuit is both isolated and bypassed, the attribute value will be “BI”.

User Interface

Guardall Object

The user interface for the Guardall multi-area intrusion detection system:

The Guardall object is listed in the Project Editor’s new object list only when at least one Guardall I/O Server is defined.

Parameters

- **server id**: the server ID
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(guardall <server id>; <position>)
```

E.g.

```
(guardall serverid; x10y20)
```

UISET Actions

ID	Attribute	Set to	
log	value	latest security log lines (as many as specified by the 'logsize' option)	
log0	value	latest line of the security log when a new entry is added, set back to blank after a short period	
c<n>.name	value	name assigned to circuit <n> or its number if no name is defined	
connection.label	visible	true	when datapoint connection = offline
		false	when datapoint connection = online
<datapoint_id>.label.1	visible	true	when datapoint with ID <datapoint_id> has value '1'
		false	when datapoint with ID <datapoint_id> has value '0'
<datapoint_id>.label.0	visible	true	when datapoint with ID <datapoint_id> has value '0'
		false	when datapoint with ID <datapoint_id> has value '1'

USER Commands

Name	Param	Action
armed	0	disarm all groups
	1	arm all groups
g<n>.armed	0	disarm group <n>
	1	arm group <n>
c<n>.bypassed	0	unbypass zone <n>
	1	bypass zone <n>
c<n>.isolated	0	un-isolate zone <n>
	1	isolate zone <n>
c<n>.soaked	0	un-soak zone <n>
	1	soak zone <n>
system	reset	reset alarms
input.bypassed	0	unbypass circuit
	1	bypass circuit
input.isolated	0	un-isolate circuit

	1	isolate circuit
input.soaked	0	un-soak circuit
	1	soak circuit

Log Files

If a Guardall anti-intrusion unit is connected to HSYCO, the security.log file will be written in the same directories as the daily log files.

This file is never overwritten or deleted from HSYCO SERVER.

```
2014.02.22 11:23:08.118 - guard - 2116 2000.1.4 22:50:16  Area disinserita - Utente: 0, Area: 1
2014.02.22 11:23:08.120 - guard - 2117 2000.1.4 22:50:16  Area disinserita - Utente: 0, Area: 2
2014.02.22 11:23:13.204 - guard - 2118 2000.1.4 22:50:22 Gruppo inserito - Utente: 0, Gruppo: 2
2014.02.22 11:23:13.205 - guard - 2119 2000.1.4 22:50:22 Area inserita - Utente: 0, Area: 1
2014.02.22 11:23:13.222 - guard - 2120 2000.1.4 22:50:22 Area inserita - Utente: 0, Area: 2
2014.02.22 11:23:16.449 - guard - 2121 2000.1.4 22:50:24 Gruppo disinserito - Utente: 0, Gruppo: 2
2014.02.22 11:23:16.450 - guard - 2122 2000.1.4 22:50:24 Area disinserita - Utente: 0, Area: 1
2014.02.22 11:23:16.451 - guard - 2123 2000.1.4 22:50:24 Area disinserita - Utente: 0, Area: 2
```

Other information about the Guardall driver are stored in the message.log daily file.

Release Notes

3.3.0

- added “logsize” server option
- object modified
- added guardall.ini configuration file

3.1.0

- added support for the PX80 panel
- modified log messages

3.0.3

- added the “ID.logon.N = error” event

3.0.2

- initial version release

Guardall is a registered trademark of UTC Fire & Security.

HSYCO Remote

The HSYCO Remote I/O Server is a powerful and secure server-to-server communication system that allows an HSYCO server read and write access to all the I/O data points and GUI variables of other remote HSYCO servers.

Using the HSYCO Remote I/O Server you can build a multi-level architecture with one or more HSYCO servers acting as concentrators for data points managed by other HSYCO servers.

Communication

The I/O Server communicates with the remote HSYCO server by establishing encrypted HTTPS connections to the HTTPS port of that server. The network infrastructure should be configured to allow these connections. The bandwidth and latency requirements depend on the specific application and the amount of data points on the remote HSYCO server.

Local HSYCO Configuration

Add an HSYCO I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: public IP address of the remote HSYCO server
- **IP Port**: HTTPS port of the remote HSYCO server, leave blank for default port 443.

Authentication

- **Password**: must correspond to the RemoteServerPassword parameter set in the remote HSYCO's settings (see below).

High Availability

- **Shutdown when inactive**: defaults to false.

Options

ID	Default	Values	Description
discovery	true	true	automatically creates the list of remote devices and individual data points in the systemtopo.txt file
		false	discovery is disabled
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
io	true	true	I/O data points of the remote HSYCO server are replicated on this server

		false	I/O data points of the remote HSYCO server are not used
ui	false	true	Web GUI elements of the remote HSYCO server are replicated on this server
		false	Web GUI elements of the remote HSYCO server are not used
drift	4	0...9999	used in combination with a remote SYSTEM I/O Server, sets the maximum clock drift between the local and the remote clocks, in seconds

Remote HSYCO Configuration

In the system settings of the remote HSYCO server the following parameters must be set:

- **RemoteServerPassword**: must correspond to the password set on the HSYCO I/O server's parameters on the local server
- **RemoteServerAddress** = IP address of the local HSYCO server
- **RemoteServerControl** = set to "true" to allow the local server to modify the datapoints values of this HSYCO server, "false" otherwise.

Datapoints

On the local HSYCO server, all datapoints of the remote server will be mapped with the original datapoint names, prefixed with the I/O server name and a separation point. For example, if the I/O server name is “remote”, a remote datapoint “dummy.light.1” will be mapped as “remote.dummy.light.1” on the local server.

User Interface

On the local HSYCO server, all GUI elements of the remote server will be mapped with the original names, prefixed with the I/O server name and a separation point. For example, if the I/O server name is “remote”, a remote GUI element “status.text” will be mapped as “remote.status.text” on the local server.

Release Notes

3.2.0

- added support of the SYSTEM I/O Server on remote systems for clock drift check
- the RemoteServerAddress option can now be set to * to allow any remote IP address
- new RemoteServerIOFilter and RemoteServerUIFilter options to filter the status data returned to the remote server

3.1.0

- initial release

Hephais

The SD3 HEPHAIS 1600 is a multi-zones fire panel. The integration with HSYCO can be accomplished via a standard Modbus RTU slave interface. This I/O Server requires HEPHAIS 1600 with protocol modbus JBUS V0.13.

Modbus Connections

The HEPHAIS 1600 has a Modbus interface.

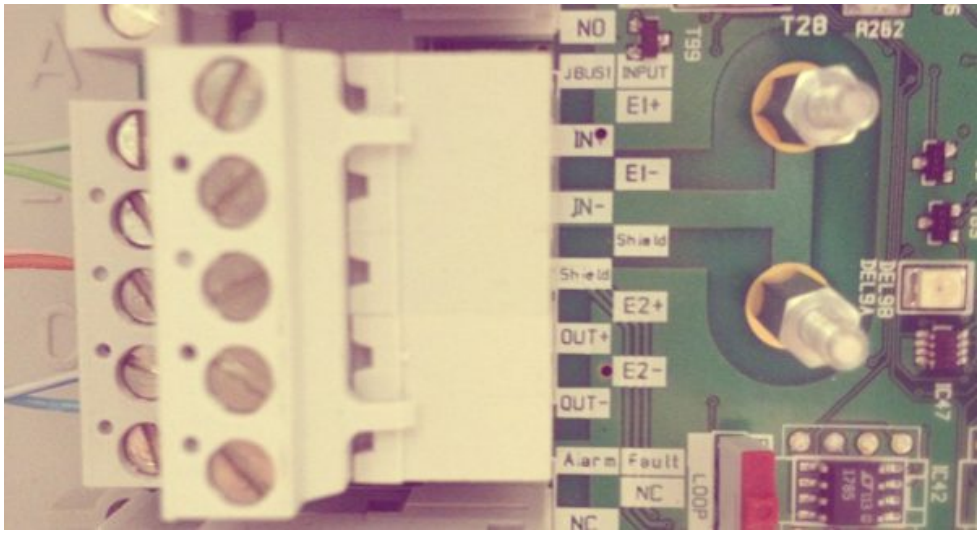
- Modbus address: 1 (can be set to 1 - 247)
- RS-422 parameters:

Baud rate	9600
Data bits	8
Stop bit	1
Parity	none
Flow control	none

Refer to the HEPHAIS 1600 manuals for installation, wiring and power supply requirements.

The modbus wires terminal is the “Block 11”. Modbus type is RS-422/JBUS (IN+/OUT+, IN-/OUT-, GND).

The Modbus-TCP/IP gateway used for the connection must support RS-422/JBUS.



HSYCO Configuration

You can define and connect more than one HEPHAIS panel on the same Modbus bus gateway or through different gateways.

Add an Hephais I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select the comm port connected to the panel.

High Availability

- **Shutdown when inactive**: defaults to false.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
address	1	1...247	the panel's Modbus address
pollinterval	3	n > 0	the data acquisition interval, in seconds
maxpoint	m4.l2.200	point id	id of the highest point used on the panel. The number after 'm' represents the number of the

			module, the number after 'l' represents the loop number of the module and the last number is the address of the point within the loop. Acceptable values range from "m1.l1.1" to "m9.l2.200"
hysteresis	0	<n>	if different than zero, the datapoints' values are checked <n> times before being set. Setting this option to a value different from zero will decrease the responsiveness of the driver

hephais.ini

The hephais.ini file is a specific configuration file located in the root directory. Here you can define the names of the users so that they will be automatically added to the user interface. Add a new line for each user using this format:

```
<io_server_id>.m<m>.l<l>.<addr>.name = <name>
```

Add a new line to specify the zone the point belongs using this format: <io_server_id>.m<m>.l<l>.<addr>.zone = <name>

Add a new line to define the name of a zone using this format: <io_server_id>.z<n>.name = <name>

For instance:

```
fire.m1.l1.1.name = Entrance
fire.m1.l1.1.zone = 1
fire.m1.l1.2.name = Corridor
fire.m1.l1.2.zone = 2

fire.z1.name = Zone 1
fire.z2.name = Zone 2
```

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
alarm	1	R	fire panel in alarm
	0	R	no alarm
fault	1	R	fire panel fault
	0	R	no fault
oos	1	R	at least one point out of service
	0	R	no point out of service

test	1	R	at least one zone in test mode
	0	R	no zone in test mode
error.sys	1	R	system fault
	0	R	no system fault
prealarm	1	R	fire panel in pre-alarm
	0	R	no pre-alarm
alarm.tec	1	R	fire panel in technical alarm
	0	R	no technical alarm
sound	1	R	H.S. speaker active
	0	R	H.S. speaker not active
error.com	1	R	fire panel communication error
	0	R	no fire panel communication error
error.battery	1	R	fire panel battery error
	0	R	no fire panel battery error
error.sector	1	R	fire panel sector fault
	0	R	no fire panel sector fault
z<n>.name	<text>	R	name assigned to zone <n>
z<n>.alarm	1	R	fire panel in alarm
	0	R	no alarm
z<n>.oos	1	R	fire panel in alarm
	0	R	no alarm
z<n>.fault	1	R	fire panel in alarm
	0	R	no alarm
z<n>.test	1	R	fire panel in alarm
	0	R	no alarm
m<m>.l<l>.<n>.name	<text>	R	name assigned to point P of module <m> in loop <l>
m<m>.l<l>.<n>.oos	1	R	point <n> of module <m> in loop <l> out of service
	0	R	point <n> of module <m> in loop <l> not out of service

m<m>.l<l>.<n>.alarm	1	R	point <n> of module <m> in loop <l> in alarm
	0	R	point <n> of module <m> in loop <l> not in alarm
m<m>.l<l>.<n>.fault	1	R	point <n> of module <m> in loop <l> out of fault
	0	R	point <n> of module <m> in loop <l> out not in fault
m<m>.l<l>.<n>.alarm.test	1	R	point <n> of module <m> in loop <l> alarm test
	0	R	point <n> of module <m> in loop <l> not in alarm test

User Interface

Hephais Object

UISET Actions

ID	Attribute	Set to	
connection.label.online	visible	true	when datapoint connection = online
		false	when datapoint connection = offline
connection.label.offline	visible	true	when datapoint connection = offline

		false	when datapoint connection = online
alarm.label.1	visible	true	when the fire panel is in alarm
		false	when the fire panel is not in alarm
alarm.label.0	visible	true	when the fire panel is not in alarm
		false	when the fire panel is in alarm
fault.label.1	visible	true	when the fire panel is in fault
		false	when the fire panel is not in fault
fault.label.0	visible	true	when the fire panel is not in fault
		false	when the fire panel is in fault
oos.label.1	visible	true	when at least one point is out of service
		false	when no one of the points is out of service
oos.label.0	visible	true	when no one of the points is out of service
		false	when at least one point is out of service
test.label.1	visible	true	when at least one zone is in test mode
		false	when there aren't zones in test mode
test.label.0	visible	true	when there aren't zones in test mode
		false	when at least one zone is in test mode
error.sys.label.1	visible	true	when there is a system fault
		false	when there is not a system fault
error.sys.label.0	visible	true	when there is not a system fault
		false	when there is a system fault
prealarm.label.1	visible	true	when the fire panel is in pre-alarm
		false	when the fire panel is not in pre-alarm

prealarm.label.0	visible	true	when the fire panel is not in pre-alarm
		false	when the fire panel is in pre-alarm
alarm.tec.label.1	visible	true	when the fire panel is in technical alarm
		false	when the fire panel is not in technical alarm
alarm.tec.label.0	visible	true	when the fire panel is not in technical alarm
		false	when the fire panel is in technical alarm
sound.label.1	visible	true	when H.S. sound speaker is active
		false	when H.S. sound speaker is not active
sound.label.0	visible	true	when H.S. sound speaker is not active
		false	when H.S. sound speaker is active
error.com.label.1	visible	true	when there is a communication fault
		false	when there is not a communication fault
error.com.label.0	visible	true	when there is not a communication fault
		false	when there is a communication fault
error.battery.label.1	visible	true	when there is a battery fault
		false	when there is not a battery fault
error.battery.label.0	visible	true	when there is not a battery fault
		false	when there is a battery fault
error.sector.label.1	visible	true	when there is a sector error
		false	when there is not a sector error
error.sector.label.0	visible	true	when there is not a sector error
		false	when there is a sector error
z<n>.alarm.label.1	visible	true	when the zone <n> is in alarm

		false	when the zone <n> is not in alarm
z<n>.alarm.label.0	visible	true	when the zone <n> is not in alarm
		false	when the zone <n> is in alarm
z<n>.oos.label.1	visible	true	when the zone <n> is out of service
		false	when the zone <n> is not out of service
z<n>.oos.label.0	visible	true	when the zone <n> is not out of service
		false	when the zone <n> is out of service
z<n>.fault.label.1	visible	true	when the zone <n> is in fault
		false	when the zone <n> is not in fault
z<n>.fault.label.0	visible	true	when the zone <n> is not in fault
		false	when the zone <n> is in fault
z<n>.test.label.1	visible	true	when the zone <n> is in test
		false	when the zone <n> is not in test
z<n>.test.label.0	visible	true	when the zone <n> is not in test
		false	when the zone <n> is in test

Release Notes

3.3.0

- initial release

SD3 and HEPHAIS are registered trademarks owned by SD3.

Hidedgesolo

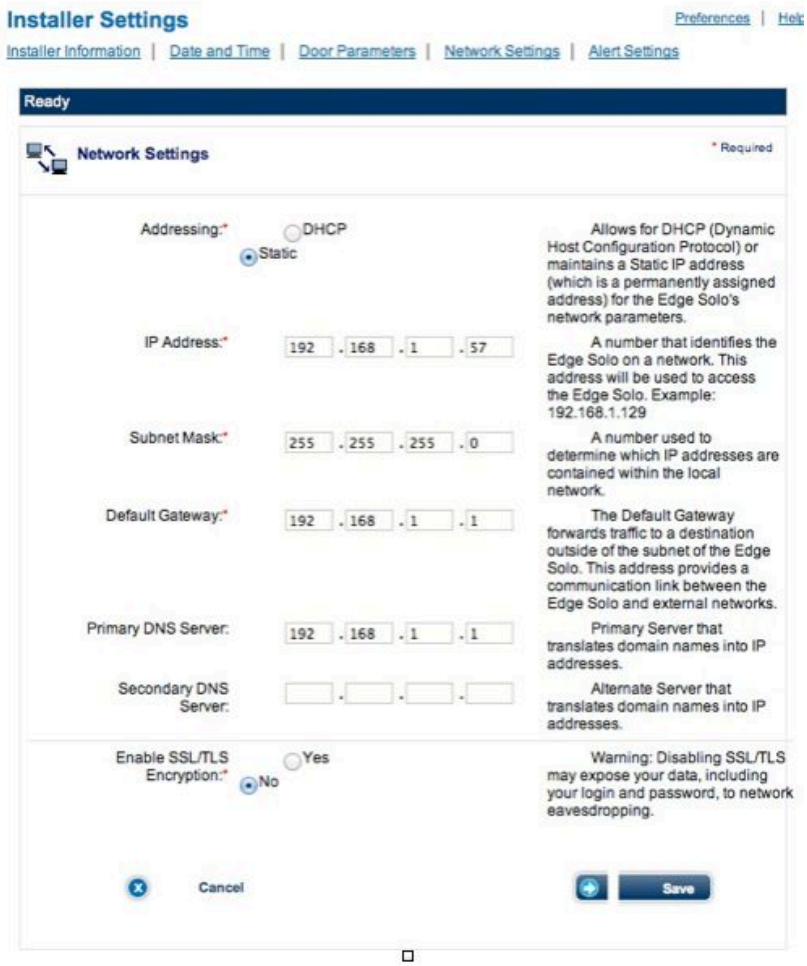
The HID Edge Solo **ESR40, ESRP40, ES400** are single-door IP access control solutions.

This I/O Server allows the integration of HID Edge Solo devices by using a TCP/IP connection.

More than one device can be controlled at the same time.

HID Edge Solo Configuration

The HID Edge Solo can be connected to a POE switch or to a normal ethernet switch. In this case the power supply must be provided with an external adapter. First of all, assign a static IP address to the Edge Solo, then it's mandatory to disable the SSL/TLS encryption in the installer settings interface.



HSYCO Configuration

Add an HIDEDGESOLO I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: IP address configured on the device
- **IP Port**: TCP/IP port to use, leave blank for default port 80.

Authentication

- **User**: username set in the installer settings
- **Password**: password set in the installer settings.

High Availability

- **Shutdown when inactive**: defaults to true.

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
card.notfound	<code>	R	badge <code> not found
pin.notfound	<pin>	R	<pin> not found
access.granted	<surname>	R	access granted to the user with surname <surname>
access.granted.extended	<surname>	R	access granted extended to the user with surname <surname>
access.denied.schedule	<surname>	R	access denied because badge/pin of the user <surname> was used outside scheduled time interval
access.denied.expired	<surname>	R	access denied because badge/pin of the user <surname> is expired
card.notassigned	<code>	R	access denied because the badge <code> is not assigned to an user.
access.denied.pin.expired	<surname>	R	access denied because the pin assigned to the user <surname> is expired
alarm.acknowledged	0	R	alarms reset
	1	R	normal conditions
door.lock.schedule	false	R	normal conditions
	true	R	door locked with a scheduled command
door.unlock.schedule	false	R	normal conditions
	true	R	door unlocked with a scheduled command

alarm.door.forced	unset	R	normal conditions
	set	R	door forced alarm
alarm.tamper	unset	R	normal conditions
	set	R	tamper alarm
access.granted.manual	false	R	normal conditions
	true	R	manual access granted
door.unlock	false	R	normal conditions
	true	R	door unlocked
state.alarm.tamper	unset	R	normal conditions
	set	R	tamper alarm
state.alarm.ac	unset	R	normal conditions
	set	R	power supply failure
state.alarm.battery	unset	R	normal conditions
	set	R	battery failure
state.alarm.door.held	unset	R	normal conditions
	set	R	door held
state.alarm.door.held	unset	R	normal conditions
	set	R	door held
state.alarm.door.forced	unset	R	normal conditions
	set	R	door forced
state.name	<name>	R	the door's name
state.relay	unset	R	door relay closed
	set	R	door relay open
state.relay.alert	unset	R	alert relay not active
	set	R	alert relay active
state.relay.aux	unset	R	aux relay not active
	set	R	aux relay active
command.access	grant	W	grant the access
command.door	lock	W	lock the door
	unlock	W	unlock the door

command.alarm	stop	W	alarms reset

Release Notes

3.0.0

- initial release

HID is a registered trademark of HID Global.

HwgMonitor

The HWGMONITOR I/O Server supports the following monitoring devices from HW group:

- Damocles
- Poseidon
- STE
- WLD.

HSYCO Configuration

Add an HWGMONITOR I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: host name or IP address of the device
- **IP Port**: TCP/IP port to use, leave blank to use default port 80.

High Availability

- **Shutdown when inactive**: defaults to true (these devices accept multiple connection and could be accessed by two HSYCO servers simultaneously).

Options

There are no available configuration options for the HWGMONITOR I/O Server.

Monitoring Devices Configuration

This I/O Server uses the XML format and HTTP requests to constantly poll the devices for the current status.

All HWg monitoring devices allow you to assign custom names to each sensor connected to a device. The name is important, as it is used by HSYCO as the id to retrieve the sensor's data. You should configure these names, using the configuration interface of the device.

Refer to the device manual for detailed configuration information.

Datapoints

The I/O Server generates asynchronous events based on the status of the devices input pins. The output data points can be written to control the output pins.

ID	Value	R/W	Description
model	<model_id>	R	the device model, as returned in the <model> tag of the XML response
<sensor_name>	<value>	R	the value returned for input <sensor_name>, as returned in the <value> tag of the XML response

<sensor_name>.cnt	<value>	R	the counter value returned for input <sensor_name>, as returned in the <cnt> tag of the XML response (for Damocles devices)
<sensor_name>.units	<units>	R	the units string returned for input <sensor_name>, as returned in the <units> tag of the XML response
<output_id>	0	W	set output <output_id> to off
	1	W	set output <output_id> to on

> Please note that while the assigned sensor name is used to identify sensors, the outputs are identified by their IDs, not the names.

Release Notes

3.0.1

- Fixed a bug that could block the update of the systemtopo.txt file at start-up.

2.8.4

- Support for the HWg Damocles line of products, with input lines counters.

2.8.0

- Initial support.

HWg is a registered trademark of HW group s.r.o.

Inim

Inim SmartLiving panels are multi-area intrusion detection systems.

The integration with HSYCO can be accomplished via direct control through the RS-232 port on the board.

The INIM I/O Server in HSYCO supports the integration of more than one unit within the same HSYCO interface.

Communication

Employ a female-female RS-232 crossed cable to connect the panel directly to HSYCO or to an Ethernet/RS-232 gateway.

RS-232 parameters:

Baud rate	57600
Data bits	8
Stop bit	1
Parity	even
Flow control	none

HSYCO Configuration

Add an INIM I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select the comm port the device is connected to.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system

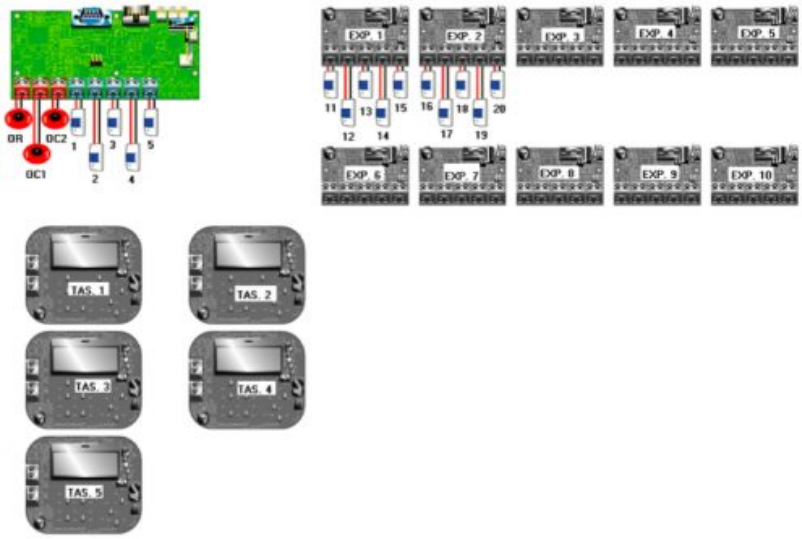
logsize	50	n ≥ 0	the number of log lines to display in the UI object
pollinterval	4	n > 0	the data acquisition interval, in seconds
mod		515	the panel model to control. This option is required
		1050	
		10100	
pin		<code>	use the specified user code to perform operations that require user authentication. If this option is not specified it will be required to manually insert the pin every time such operations are performed

inim.ini

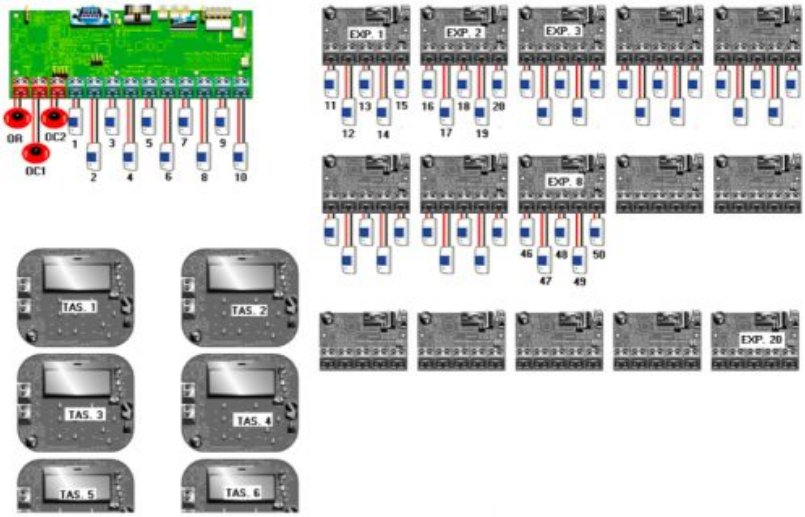
The inim.ini file is a specific configuration file located in the root directory.

This file is required to provide the I/O Server with the mapping between the board terminals and the respective zones or outputs when a configuration different from the default one is used.

The default configuration for the SmartLiving 515 is shown in the following diagram:



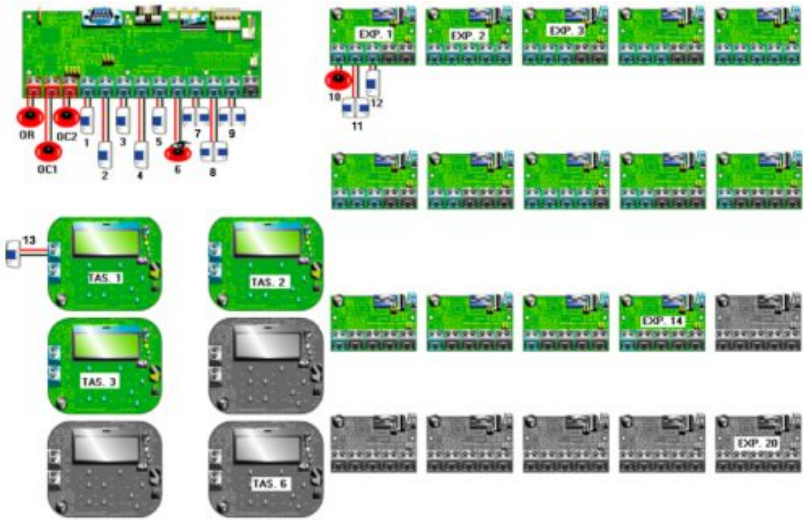
And here is default configuration for SmartLiving 1050 and 10100 (for the 10100 model this diagram must be extended to 100 terminals and 40 expanders):



If the inim.ini file is not created or it does not contain specifications for our I/O server, then the above configurations are used.

If, on the other hand, you are using a custom configuration, then, for each used terminal of the panel, expanders, and keypads you must specify its configuration.

For instance, consider the following configuration on a SmartLiving1050:



The corresponding configuration in inim.ini, assuming that the ID of the I/O Server is "inm", would be the following:

```
inm.t1 = z1
inm.t2 = z2
inm.t3 = z3
inm.t4 = z4
inm.t5 = z5
inm.t6 = z6,o6
inm.t7 = z7d
inm.t8 = z8d
inm.t9 = z9d

inm.e1.t1 = o10
inm.e1.t2 = z11d
inm.e1.t3 = o12

inm.k1.t1 = z13
```

Terminals on the panel itself are referred to using the notation “<ioserver_id>.t<n>” where <n> is the number of the board terminal from left to right. Terminals on expanders are identified by the string “<ioserver_id>.e<m>.t<n>” where <m> is the expander number and <n> the terminal number on the expander board. Finally, the syntax “<ioserver_id>.k<m>.t<n>” is used to identify terminals on keyboards.

When a terminal is configured as a simple zone (), it should be assigned the corresponding zone number (e.g. "`= z1`").

If the terminal is configured as double zone () then append “d” to the zone id (e.g. "`= z7d`"). This configuration allows to have two zones connected to the same terminal, thus the I/O Server will handle these two zones independently: the notation "z<n>" will identify datapoints related to base zones, while "z<n>d" identifies double zones.

If the terminal correspond to an output () specify the corresponding output number (e.g. "`= o10`"), while if the terminal is configured as an IO () then it will be seen as a zone as well as an output, so specify both of them (e.g. "`= z6,o6`").

Further, it is possible to assign a name to a zone terminal to be displayed in the user interface by adding it at the end of the line after an additional “=”. For instance:

```
inm.t4 = z4 = door
inm.t5 = z5 = corridor
inm.t6 = z6, o6 = alarm point
inm.e1.t2 = z11d = room 1, room 2
```

As you can see from the last line, when referring to a double zone you can specify two different names separated by a comma.

N.B. SmartLiving 515 panels have only 5 terminals on the board and the indexing of zones and outputs skips the numbers from 6 to 10 as we can see from the image above. But, from tests conducted on different configurations, sometimes these indexes are actually used, thus refinements on the configuration in inim.ini must be applied to map the terminals with the correct zone or output number.

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
armed	0	R	all areas are disarmed
	1	R	at least one area is armed
a<n>.armed	0	R	area <n> is disarmed
		W	disarm area <n>
	away	R	area <n> is armed in “away” mode
		W	arm area <n> in “away” mode
	stay	R	area <n> is armed in “stay” mode

		W	arm area <n> in “stay” mode
	inst	R	area <n> is armed in “instantaneous” mode
		W	arm area <n> in “instantaneous” mode
a<n>.alarm	0	R	no active alarm on area <n>
	1	R	alarm on area <n>
a<n>.tamper	0	R	no tamper alarm on area <n>
	1	R	tamper alarm on area <n>
a<n>.alarm.mem	0	R	no alarm in memory for area <n>
	1	R	alarm in memory for area <n>
a<n>.tamper.mem	0	R	no tamper alarm in memory for area <n>
	1	R	tamper alarm in memory for area <n>
a<n>.auto	0	R	auto-arming disabled on area <n>
	1	R	auto-arming enabled on area <n>
a<n>.reset	1	W	reset alarms on area <n>
z<n>.state z<n>d.state	closed	R	zone <n> (base or double) is closed
	open	R	zone <n> (base or double) is open
	tamper	R	zone <n> (base or double) is tampered
	short	R	zone <n> (base or double) is in short circuit
z<n>.bypassed z<n>d.bypassed	0	R	zone <n> is not bypassed
		W	unbypass zone <n>
	1	R	zone <n> is bypassed
		W	bypass zone <n>
z<n>.test z<n>d.test	0	R	zone <n> is not in test
	1	R	zone <n> is being tested

z<n>.alarm.mem z<n>d.alarm.mem	0	R	no alarm in memory for zone <n>
	1	R	alarm in memory for zone <n>
z<n>.tamper.mem z<n>d.tamper.mem	0	R	no terminal tamper alarm in memory for zone <n>
	1	R	terminal tamper alarm in memory for zone <n>
or	0	R	the relay output on the panel is not active
		W	deactivate the relay output on the panel
	1	R	the relay output on the panel is active
		W	activate the relay output on the panel
oc1	0	R	the open collector 1 on the panel is not active
		W	deactivate the open collector 1 on the panel
	1	R	the open collector 1 on the panel is active
		W	activate the open collector 1 on the panel
oc2	0	R	the open collector 2 on the panel is not active
		W	deactivate the open collector 2 on the panel
	1	R	the open collector 2 on the panel is active
		W	activate the open collector 2 on the panel
o<n> (if defined in inim.ini)	0	R	output <n> is not active
		W	deactivate output <n>
	1	R	output <n> is active
		W	activate output <n>
fault.fuse.zones	0	R	zones fuse OK
	1	R	fault on zones fuse
fault.fuse.ibus	0	R	IBUS fuse OK

	1	R	fault on IBUS fuse
fault.battery	0	R	battery OK
	1	R	fault on battery
fault.mains	0	R	mains OK
	1	R	panel mains fault
fault.phone	0	R	phone line OK
	1	R	fault on phone line
fault.jamming	0	R	no jamming
	1	R	radio jamming
fault.radio.battery	0	R	radio zones battery OK
	1	R	low battery on radio zone(s)
fault.radio.loss	0	R	radio zones OK
	1	R	loss of radio zone(s)
battery.volts	<val>	R	the voltage value of the battery is <val> Volts
tamper.panel	0	R	panel OK
	1	R	panel tampered
tamper.cover	0	R	panel cover OK
	1	R	panel cover tampered
tamper.reader	0	R	readers OK
	1	R	reader(s) tampered
tamper.keypad	0	R	keypads OK
	1	R	keypad(s) tampered
tamper.expander	0	R	FLEX5 expanders OK
	1	R	FLEX5 expander(s) tampered
reset	0	R	communication BUS ready
	1	R	ongoing reset procedure on communication BUS
internet	0	R	“internet access” option disabled
	1	R	“internet access” option enabled

service	0	R	service jumper in “RUN” position
	1	R	service jumper in “SERV” position (maintenance)
program	0	R	panel not in programming mode
	1	R	ongoing programming via keyboard
voice	0	R	no SmartLogos30M voice card detected
	1	R	SmartLogos30M voice card detected

User Interface

Inim Object

The user interface for the control and supervision of one area of the system, the zones and the visualization of the security log:

The Inim object is listed in the Project Editor’s new object list only when at least one INIM I/O Server is defined.

Parameters

- id - the id assigned to the INIM I/O Server in hsyco.ini
- area - the number of the area to be addressed
- pos - the object’s position. Use the pixels or rows and columns coordinates format.

Syntax

```
(inim <server id>; <area>; <position>)
```

E.g.

```
(inim serverid; area; x10y20)
```

UISET Actions

ID	Attribute	Set to
log	value	latest security log lines (as many as specified by the 'logsize' option)
logo	value	latest line of the security log when a new entry is added, set back to blank after a short period
z<n>.name z<n>d.name	value	name assigned to zone <n> (base or double) or its number if no name is defined

connection.label	visible	true	when datapoint connection = offline
		false	when datapoint connection = online
a<n>.armed.label.0	visible	true	when area <n> is disarmed
		false	when area <n> is armed
a<n>.armed.label.away	visible	true	when area <n> is armed in "away" mode
		false	when area <n> is not armed in "away" mode
a<n>.armed.label.stay	visible	true	when area <n> is armed in "stay" mode
		false	when area <n> is not armed in "stay" mode
a<n>.armed.label.inst	visible	true	when area <n> is armed in "instantaneous" mode
		false	when area <n> is not armed in "instantaneous" mode
a<n>.armed.label.1	visible	true	when area <n> is armed in any mode
		false	when area <n> is not armed
z<n>.state.label.closed z<n>d.state.label.closed	visible	true	when datapoint 'z<n>.state' has value 'closed'
		false	when datapoint 'z<n>.state' has not value 'closed'
z<n>.state.label.open z<n>d.state.label.open	visible	true	when datapoint 'z<n>.state' has value 'open'
		false	when datapoint 'z<n>.state' has not value 'open'
z<n>.state.label.tamper z<n>d.state.label.tamper	visible	true	when datapoint 'z<n>.state' has value 'tamper'
		false	when datapoint 'z<n>.state' has not value 'tamper'
z<n>.state.label.short z<n>d.state.label.short	visible	true	when datapoint 'z<n>.state' has value 'short'
		false	when datapoint 'z<n>.state' has not value 'short'
<datapoint_id>.label.1	visible	true	when datapoint with ID <datapoint_id> has value '1'

		false	when datapoint with ID <datapoint_id> has value '0'
<datapoint_id>.label.0	visible	true	when datapoint with ID <datapoint_id> has value '0'
		false	when datapoint with ID <datapoint_id> has value '1'

USER Commands

Name	Param	Action
a<n>.armed	0	disarm area <n>
	stay	arm area <n> in "stay" mode
	away	arm area <n> in "away" mode
	inst	arm area <n> in "instantaneous" mode
a<n>.reset	1	reset alarms on area <n>
z<n>.bypassed z<n>d.bypassed	0	unbypass zone <n> (base or double)
	1	bypass zone <n> (base or double)
or	0	deactivate the relay output on the panel
	1	activate the relay output on the panel
oc1	0	deactivate the open collector 1 on the panel
	1	activate the open collector 1 on the panel
oc2	0	deactivate the open collector 2 on the panel
	1	activate the open collector 2 on the panel
o<n>	0	deactivate output <n>
	1	activate output <n>

To perform any of the above operations it is necessary to enter the user code first. To enter the user code through the user interface use user buttons to create a virtual keypad and assign them name and param values according to this table:

Name	Param	Action
digit	{0 ... 9}	append the specified digit to the entered user code
	c	delete the last entered digit

To visualize the number of entered digits use up to 6 text objects (depending on the number of digits used for user codes) and set their IDs to:

ID	Description

pin.<n> (0 ≤ <n> ≤ 5)	shows the character '*' when the corresponding digit has been entered, shows '_' otherwise

The entered code will be cleared after a short period of inactivity.

Log Files

A permanent detailed record of all the INIM log messages is saved in a file called security.log in the logs/YYYY directory for the current year. This file is never overwritten or deleted by HSYCO.

Other information related to the INIM I/O Server are saved in the daily message.log files.

Release Notes

3.2.0

- initial version release

Inim and SmartLiving are registered trademarks of Inim Electronics s.r.l.

Mitsubishi

The MITSUBISHI air-conditioning system equipped with one of the centralized controllers among G-50, GB-50, or AG-150 allows for remote control and supervision, via Ethernet communication, of the whole framework.

This I/O Server complies with the XML API for the Mitsubishi M-NET proprietary protocol.

> In order to connect the Mitsubishi controller with HSYCO, the Web Monitor and Annual Schedule pin codes activation are required. On request, Mitsubishi will provide a document to be compiled for the activation.

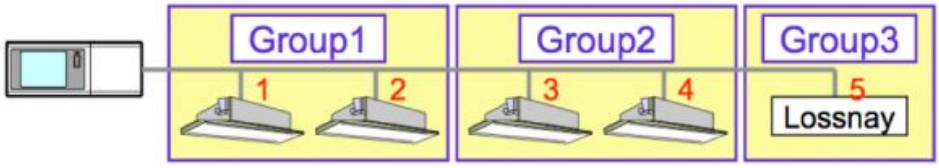
M-NET Architecture

All the devices comprising the M-NET architecture are assigned an address ranging from 0 to 250.

Conditioners and lossnay modules can be associated to 50 different groups (indexed from 1 to 50). The control of these modules is thus performed addressing the group they belong to.

Further, it is possible to expand the system using up to three ECs which allow to replicate a full M-NET architecture under each EC. In this case, modules are addressed specifying both the EC and the group they belong to.

The following image shows a possible configuration of a simple system:



The Mitsubishi controller replies on the port 80 (10 sockets) or 1025 (4 sockets) of its IP address.

HSYCO Configuration

Add an MNET I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: IP address of the controller
- **IP Port**: TCP/IP port to use, can be set to 80 or 1025, leave blank to use default port 80.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support

startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
pollinterval	20	n > 0	the data acquisition interval, in seconds
ec	0	0 - 3	specifies the number of ECs connected to the main controller. A value of 0 indicates that no EC is used

Datapoints

In the following tables, replace <target> with "system" for sending the command to all the defined groups. If ECs are used (i.e. the option "ec" is set to a value greater than 0), you can specify "ec<s>" (with <s> ranging from 1 to the specified value of ECs) as target to address a single sub-system.

Further, replace <n> in "g<n>" with a group number ranging from 1 to 50 and replace <m> in "a<m>" with an address from 0 to 250.

If ECs are used, it is necessary to specify the EC number we are addressing too. To this end, prepend the string "ec<s>." to the group or the address.

For instance, when an EC is used, the following datapoint ID is not correct:

```
g1.power
```

The correct one to be used is the following:

```
ec1.g1.power
```

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
<target>.power	1	R	at least one group is on
		W	turn all the groups on
	0	R	the whole system is off
		W	turn the whole system off
<target>.mode	cool dry fan heat auto	W	set the mode of all the groups to the specified value (Air conditioner modes)

	heatrecovery lcauto bypass	W	set the mode of all the groups to the specified value (LOSSNAY modes)
<target>.temp	17 - 30	W	set the temperature of all the groups to the specified value (if compliant with the allowed ranges[Note 1])
<target>.airdirection	horizontal mid1 mid2 vertical swing	W	set the air direction of all the groups to the specified value
<target>.fanspeed	high mid1 mid2 low	W	set the speed of the fan of all the groups to the specified value
<target>.ventilation	high low off	W	set the operation status of the interlocked LOSSNAY or OA processing unit present in the whole system to the specified value
<target>.prohibit.drive	1	W	prohibit power switching by remote control to all the groups
	0	W	allow power switching by remote control to all the groups
<target>.prohibit.mode	1	W	prohibit mode setting by remote control to all the groups
	0	W	allow mode setting by remote control to all the groups
<target>.prohibit.settemp	1	W	prohibit temperature setting by remote control to all the groups
	0	W	allow temperature setting by remote control to all the groups
<target>.prohibit.filter	1	W	prohibit filter signal reset by remote control to all the groups
	0	W	allow filter signal reset by remote control to all the groups
<target>.reset	error	W	reset the error signal of all the groups
	filter	W	reset the filter signal of all the groups
g<n>.power	1	R	this group is on
		W	turn this group on

	0	R	this group is off
		W	turn this group off
g<n>.mode	cool dry fan heat auto	R	the mode of this group is set to the reported value (Air conditioner modes)
		W	set the mode of this group to the specified value (Air conditioner modes)
	autoheat autocool	R	the mode of this group is set to the reported value (Air conditioner modes)
	heatrecovery lcauto bypass	R	the mode of this group is set to the reported value (LOSSNAY modes)
		W	set the mode of this group to the specified value (LOSSNAY modes)
g<n>.temp	17 - 30	R	the set temperature for this group correspond to the reported value (in °C)
		W	set the temperature of this group to the specified value (if compliant with the allowed ranges[Note 1])
g<n>.inlettemp	0 - 999	R	the inlet measured temperature for this group (in tenth of °C)
g<n>.airdirection	horizontal mid1 mid2 vertical swing	R	the air direction for this group is set to the reported value
		W	set the air direction of this group to the specified value
g<n>.fanspeed	high mid1 mid2 low	R	the fan speed for this group is set to the reported value
		W	set the speed of the fan of this group to the specified value
g<n>.ventilation	high low off	R	the operation status of the interlocked LOSSNAY or OA processing unit present in this group is set to the reported value
		W	set the operation status of the interlocked LOSSNAY or OA processing unit present in this group to the specified value

g<n>.prohibit.drive	1	R	the power switching by remote control is prohibited to this group
		W	prohibit power switching by remote control to this group
	0	R	the power switching by remote control is allowed to this group
		W	allow power switching by remote control to this group
g<n>.prohibit.mode	1	R	the mode setting by remote control is prohibited to this group
		W	prohibit mode setting by remote control to this group
	0	R	the mode setting by remote control is allowed to this group
		W	allow mode setting by remote control to this group
g<n>.prohibit.settemp	1	R	the temperature setting by remote control is prohibited to this group
		W	prohibit temperature setting by remote control to this group
	0	R	the temperature setting by remote control is allowed to this group
		W	allow temperature setting by remote control to this group
g<n>.prohibit.filter	1	R	the filter signal reset by remote control is prohibited to this group
		W	prohibit filter signal reset by remote control to this group
	0	R	the filter signal reset by remote control is allowed to this group
		W	allow filter signal reset by remote control to this group
g<n>.filter	1	R	the filter signal of this group is on
	0	R	the filter signal of this group is off

g<n>.error	1	R	the error signal of this group is on
	0	R	the error signal of this group is off
g<n>.reset	filter	W	reset the filter signal of this group
	error	W	reset the error signal of this group
a<m>.error	0	R	no error is reported for this address
	<xxxx>	R	the error with code <xxxx> is reported for this address
system.error	1	R	an error is detected on some address
	0	R	no error detected on any address

Note 1

For some modes the temperature must be set following these ranges:

- modes “cool” or “dry”: 19 - 30
- mode “heat”: 17 - 28
- mode “auto”: 19 - 28

If <target> is set to “system” or “ec<s>” and the specified temperature parameter doesn’t respect the above mentioned ranges for a group belonging to the target, then the temperature is not changed for that group and an error message is logged.

User Interface

UISET Actions

ID	Attribute	Set to
connection	visible	true when the device appears offline, false otherwise
system.power	value	the status of the system: ON: at least one group is on; OFF: all the system is off
ec<s>.power	value	the status of the sub-system connected to this EC: ON: at least one group is on; OFF: all the sub-system is off. Valid if the option “ec” is set to a value greater that 0.

g<n>.power	value	the status of the specified group: ON, OFF
g<n>.mode	value	the set mode for this group: COOL, DRY, FAN, HEAT, AUTO, AUTOHEAT, AUTOCOOL, HEATRECOVERY, LCAUTO, BYPASS
g<n>.temp	value	the set temperature for this group: 17 - 30 °C
g<n>.inlettemp	value	the inlet measured temperature for the specified group: 0.0 - 99.9 °C
g<n>.airdirection	value	the air direction setting for this group: HORIZONTAL, MID1, MID2, VERTICAL, SWING
g<n>.fanspeed	value	the speed of the fan for this group: HIGH, MID1, MID2, LOW
g<n>.ventilation	value	operation status of the interlocked LOSSNAY or OA processing unit: HIGH, LOW, OFF
g<n>.prohibit.drive	value	power switching by remote control prohibition: ON, OFF
g<n>.prohibit.mode	value	mode changing by remote control prohibition: ON, OFF
g<n>.prohibit.settemp	value	temperature setting by remote control prohibition: ON, OFF
g<n>.prohibit.filter	value	filter signal reset by remote control prohibition: ON, OFF
g<n>.error	visible	true if the error state for this group is set, false otherwise
g<n>.filter	visible	true if the filter signal state for this group is set, false otherwise
a<m>.error	visible	true if an error is detected for the specified address, false if no error is signaled
a<m>.error	value	error code for the specified address
system.error	visible	true if an error is detected on any address, false otherwise

USER Commands

Name	Param	Action
<target>.power	on	turn the target on
	off	turn the target off
<target>.mode	cool dry fan heat auto	air conditioner modes
	heatrecovery lcauto bypass	LOSSNAY modes
<target>.temp	17 - 30	set the target temperature[Note 1]
	up	increase the target temperature by 1°C (if the upper limit is not reached)[Note 1]
	down	decrease the target temperature by 1°C (if the lower limit is not reached)[Note 1]
<target>.airdirection	horizontal mid1 mid2 vertical swing	set the air direction of the target to the specified mode
g<n>.airdirection	up	change the air direction of this group according to the sequence: vertical - mid2 - mid1 - horizontal. If the current direction is set to swing, sets it to horizontal
	down	change the air direction of this group according to the sequence: horizontal - mid1 - mid2 - vertical. If the current direction is set to swing, set it to vertical
<target>.fanspeed	high mid1 mid2 low	set the fan speed of the target
g<n>.fanspeed	up	change the fan speed of this group according to the sequence: low - mid2 - mid1 - high
	down	change the fan speed of this group according to the sequence: low - mid2 - mid1 - high
<target>.ventilation	high low off	set the operation status of the interlocked LOSSNAY or OA processing unit
<target>.reset	error	reset the error signal of the terget
	filter	reset the filter signal of the target

<target>.prohibit.drive	on	prohibit power switching by remote control
	off	allow power switching by remote control
<target>.prohibit.mode	on	prohibit mode setting by remote control
	off	allow mode setting by remote control
<target>.prohibit.settemp	on	prohibit temperature setting by remote control
	off	allow temperature setting by remote control
<target>.prohibit.filter	on	prohibit filter signal reset by remote control
	off	allow filter signal reset by remote control
g<n>.prohibit	drive	switch the state (on/off) of the drive prohibition for this group
	mode	switch the state (on/off) of the mode prohibition for this group
	settemp	switch the state (on/off) of the temperature setting prohibition for this group
	filter	switch the state (on/off) of the filter reset prohibition for this group

Release Notes

3.0.0

- initial release version

Mitsubishi and M-NET are registered trademarks of MITSUBISHI ELECTRIC, Inc.

Modbus

The Modbus Protocol is a messaging structure. It is used to establish master-slave/client-server communication between intelligent devices. It is a de facto standard. The Modbus RTU specification defines the default communication protocol over serial lines. The Modbus TCP/IP specification takes the Modbus instruction set and wraps TCP/IP around it.

HSYCO supports both Modbus TCP and Modbus RTU, acting as a Modbus master in both cases.

This support is implemented as a set of Java APIs that allow read/write functions as described in the HSYCO Programming Handbook, and an I/O Server with a data point representation that you can use in the EVENTS language to poll Modbus data and generate events.

General Configuration

Modbus TCP

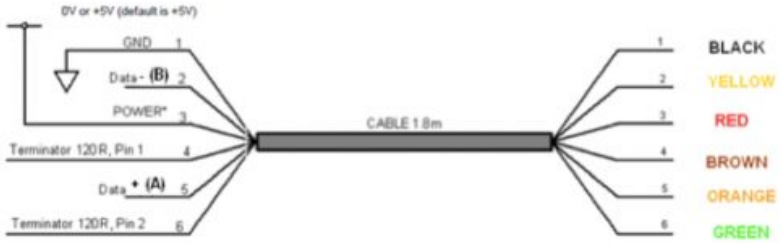
If a Modbus TCP/IP device is used to interface HSYCO to a Modbus RTU or ASCII bus over RS485, the gateway should be configured as Modbus TCP/IP slave and Modbus RTU/ASCII master.

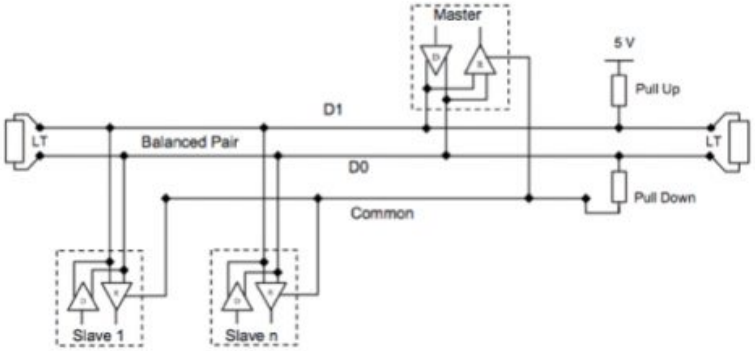
HSYCO can also directly access Modbus TCP slave devices.

Modbus RTU

HSYCO supports Modbus RTU over RS485, using the FTDI's USB-RS485 cable, a USB to RS485 levels serial UART converter cable incorporating FTDI's FT232RQ USB to serial UART interface IC device which handles all the USB signaling and protocols. The cable provides a fast, simple way to connect devices with a RS485 interface to USB.

The USB-RS485 cable connects to any USB port of the HSYCO server, and doesn't require external power supply. It offers an internal 120 Ohm resistor for line termination, and +5Vcc and ground level references that could become handy to connect pull-up and pull-down resistors for lines' polarization (See the MODBUS over Serial Line Specification & Implementation guide for additional information about line polarization and termination).





> The USB-RS485 cable doesn't provide opto-isolated RS485 lines. Transient currents and voltage levels exceeding the cable's specification could damage the USB-RS485 cable, HSYCO server and any Modbus device attached to the RS485 line.

HSYCO Configuration

Add a MODBUSTCP or MODBUSRTU I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

Modbus TCP

- **IP Address**: IP address of the gateway
- **IP Port**: TCP/IP port to use, leave blank to use default port 502.

Modbus RTU

- **Comm ID**: select the comm port the device is connected to.

When creating the used comm port, its id depends on your HSYCO server configuration, and should be "ttyUSB0" if there are no other devices connected to the USB ports. You could also use the "ftdi-n-n.n" alias, that is unique for each physical USB port on the server. In this case, you should also list the full device name in the CommPortsList parameter.

The communication parameters should be set to the 9600,8,1,2,0 defaults, unless you want to use a different speed.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
eventslog	false	true	if the general eventsLog option is also true in hsyco.ini, Modbus

			events for this gateway are written in the log files
		false	Modbus events for this gateway are not written in the log files, even if the eventsLog option is true in hsyco.ini
extendednames	false	true	when this option is set, data point names used to return values will have the postfix .FF, where FF is the 2-digit decimal function code
		false	data points' default naming convention is used

Modbus Integration

Because the Modbus protocol is not asynchronous, but is based on polling, you always have to request a read to tell the driver to actually read the data and eventually generate an event on the associated data point if the data has changed.

Datapoints

The naming convention HSYCO uses for the events is closely related to the Modbus addressing standards and data types.

The data point names always begin with the same format: ID.unitid.address, where ID is the I/O server id, unitid is the Modbus slave device address, from 1 to 247, and address is the initial register address you are reading.

The supported public function codes are:

- 01: Read Coils
- 02: Read Discrete Inputs
- 03: Read Holding Registers
- 04: Read Input Registers
- 05: Write Single Coil
- 06: Write Single Register
- 15: Write Multiple Coils
- 16: Write Multiple Registers
- 22: Mask Write Register (only available using the modbusWriteMaskRegister() JavaScript and Java function)
- 23: Read/Write Multiple Registers (only available using the modbusReadWriteMultipleRegisters() JavaScript and Java function).

Read operations with functions 03 and 04 support the following data types:

Type code	Description	Return range	Number of registers
ushort	unsigned short integer	0 ... 65535	1
uint	unsigned integer	0 ... (2^{32} - 1)	2
ulong	unsigned long integer	0 ... (2^{64} - 1)	4
short signed	short integer	-32768 ... +32767	1

int	signed integer	-2^{31} ... $(2^{31} - 1)$	2
long	signed long integer	-2^{63} ... $(2^{63} - 1)$	4
float	signed IEEE 754 floating point	$+/- 2^{-149}$ ... $(2-2^{-23})x2^{127}$	2

Read Coils

Reads from 1 to 2000 contiguous coils, using function code 01. Each coil returns a binary value, 0 or 1, in distinct data points, one for each coil.

ID	Value	R/W	Description
unitid.address	readcoils:<n>	W	executes the Read Coils function, reading <n> coils
unitid.address.1 ... unitid.address.<n> or unitid.address.01.<n> if extendednames is set	0	R	the status of the <n>-th coil starting at address is off
	1	R	the status of the <n>-th coil starting at address is on
unitid.address.error or unitid.address.01.error if extendednames is set	0, null, >0	R	the error status. 0: no error; null: unknown error; >0: modbus exception code

The <n> value after readcoils: can also be a variable, that will be replaced with its numeric content, for example:

```
IO modbus.27.400 = readcoils:$number
```

Read Discrete Inputs

Reads from 1 to 2000 contiguous discrete inputs, using function code 02. Each input returns a binary value, 0 or 1, in distinct data points, one for each coil.

ID	Value	R/W	Description
unitid.address	readdiscreteinputs:<n>	W	executes the Read Discrete Inputs function, reading <n> inputs
unitid.address.1 ... unitid.address.<n> or unitid.address.02.<n> if extendednames is set	0	R	the status of the <n>-th input starting at address is off
	1	R	the status of the <n>-th input starting at address is on
unitid.address.error or unitid.address.02.error if extendednames is set	0, null, >0	R	the error status. 0: no error; null: unknown error; >0: modbus exception code

The <n> value after readdiscreteinputs: can also be a variable, that will be replaced with its numeric content, for example:

```
IO modbus.27.400 = readdiscreteinputs:$number
```

Read Holding Registers

Reads one data point, using function code 03 to read the appropriate number of registers, based on the data type.

Each reading returns a single value, but you can also use an extended read function to read multiple consecutive values with one single Modbus read function, by adding :<n> at the end of “readholdingregisters:<format>”, where <n> is the number of values.

ID	Value	R/W	Description
unitid.address	readholdingregisters:ushort	W	executes the Read Holding registers function, reading 1 register and interpreting the data as an unsigned short integer
unitid.address	readholdingregisters:uint	W	executes the Read Holding registers function, reading 2 consecutive registers and interpreting the data as an unsigned integer
unitid.address	readholdingregisters:ulong	W	executes the Read Holding registers function, reading 4 consecutive registers and interpreting the data as an unsigned long integer
unitid.address	readholdingregisters:short	W	executes the Read Holding registers function, reading 1 register and interpreting the data as a signed short integer
unitid.address	readholdingregisters:int	W	executes the Read Holding registers function, reading 2 consecutive registers and interpreting the data as a signed integer
unitid.address	readholdingregisters:long	W	executes the Read Holding registers function, reading 4 consecutive registers and interpreting the data as a signed long integer
unitid.address	readholdingregisters:bitshort	W	executes the Read Holding registers function, reading 1 register and interpreting the data as 16 individual bits
unitid.address	readholdingregisters:bitint	W	executes the Read Holding registers function, reading 2 consecutive registers and interpreting the data as 32 individual bits

unitid.address	readholdingregisters:bitlong	W	executes the Read Holding registers function, reading 4 consecutive registers and interpreting the data as 64 individual bits
unitid.address	readholdingregisters:float	W	executes the Read Holding registers function, reading 2 consecutive registers and interpreting the data as a floating-point number using the IEEE Standard 754 representation
unitid.address or unitid.address.03 if extendednames is set	the returned value, according to the requested data type	R	this is the value read using the Read Holding Registers function
unitid.address.<n> or unitid.address.03.<n> if extendednames is set	the returned bit value, with <n> from 1 to 16, 32 or 64	R	this is the value read using the Read Holding Registers function with bitshort, bitint or bitlong format options
unitid.address.error or unitid.address.03.error if extendednames is set	0, null, >0	R	the error status. 0: no error; null: unknown error; >0: modbus exception code

Read Input Registers

Reads one data point, using function code 04 to read the appropriate number of registers, based on the data type.

Each reading returns a single value, but you can also use an extended read function to read multiple consecutive values with one single Modbus read function, by adding :N at the end of “readinputregisters:<format>”, where <n> is the number of values.

ID	Value	R/W	Description
unitid.address	readinputingregisters:ushort	W	executes the Read Input registers function, reading 1 register and interpreting the data as an unsigned short integer
unitid.address	readinputingregisters:uint	W	executes the Read Input registers function, reading 2 consecutive registers and interpreting the data as an unsigned integer
unitid.address	readinputregisters:ulong	W	executes the Read Input registers function, reading 4 consecutive registers and interpreting the data as an unsigned long integer
unitid.address	readinputregisters:short	W	executes the Read Input registers function, reading 1

			register and interpreting the data as a signed short integer
unitid.address	readinputregisters:int	W	executes the Read Input registers function, reading 2 consecutive registers and interpreting the data as a signed integer
unitid.address	readinputregisters:long	W	executes the Read Input registers function, reading 4 consecutive registers and interpreting the data as a signed long integer
unitid.address	readinputregisters:bitshort	W	executes the Read Input registers function, reading 1 register and interpreting the data as 16 individual bits
unitid.address	readinputregisters:bitint	W	executes the Read Input registers function, reading 2 consecutive registers and interpreting the data as 32 individual bits
unitid.address	readinputregisters:bitlong	W	executes the Read Input registers function, reading 4 consecutive registers and interpreting the data as 64 individual bits
unitid.address	readinputregisters:float	W	executes the Read Input registers function, reading 2 consecutive registers and interpreting the data as a floating-point number using the IEEE Standard 754 representation
unitid.address or unitid.address.04 if extendednames is set	the returned value, according to the requested data type	R	this is the value read using the Read Input Registers function
unitid.address.<n> or unitid.address.04.<n> if extendednames is set	the returned bit value, with <n> from 1 to 16, 32 or 64	R	this is the value read using the Read Input Registers function with bitshort, bitint or bitlong format options
unitid.address.error or unitid.address.04.error if extendednames is set	0, null, >0	R	the error status. 0: no error; null: unknown error; >0: modbus exception code

Write Coils

Writes a single coil, using function code 05, or multiple coils, using function 15.

ID	Value	R/W	Description

unitid.address	bit:0 bit:false bit:off	W	executes the Write Single Coil function, setting the coil to off
	bit:1 bit:true bit:on	W	executes the Write Single Coil function, setting the coil to on
	bits:<0|1|-...	W	executes the Write Multiple Coils function, setting consecutive coils based on the binary sequence. The left-most bit in the sequence is the first coil
unitid.address.<n> or unitid.address.05.<n> if extendednames is set	0	R	the write function returned a correct response, and the new status of the coil is off
	1	R	the write function returned a correct response, and the new status of the coil is on
unitid.address.error or unitid.address.05.error if extendednames is set	0, null, >0	R	the error status. 0: no error; null: unknown error; >0: modbus exception code

The value after bit: and bits: can also be a variable, that will be replaced with its numeric content, for example:

```
IO modbus.27.1220 = bit:$value
```

Write Registers

Writes registers, using function code 06 or 16, based on the data type.

ID	Value	R/W	Description
unitid.address	ushort:<value>	W	executes the Write Single Register function, writing an unsigned short number
unitid.address	uint:<value>	W	executes the Write Multiple Registers function, writing an unsigned integer number to two adjacent registers
unitid.address	ulong:<value>	W	executes the Write Multiple Registers function, writing an unsigned long number to four adjacent registers
unitid.address	short:<value>	W	executes the Write Single Register function, writing a signed short number
unitid.address	int:<value>	W	executes the Write Multiple Registers function, writing a

			signed integer number to two adjacent registers
unitid.address	long:<value>	W	executes the Write Multiple Registers function, writing a signed long number to four adjacent registers
unitid.address	float:<value>	W	executes the Write Multiple Registers function, writing a floating-point number using the IEEE Standard 754 representation to two adjacent registers
unitid.address or unitid.address.06 unitid.address.16 if extendednames is set	the returned value, according to the requested data type	R	the write function returned a correct response
unitid.address.error or unitid.address.06.error unitid.address.16.error if extendednames is set	0, null, >0	R	the error status. 0: no error; null: unknown error; >0: modbus exception code

The value after short:, int: etc. can also be a variable, that will be replaced with its numeric content, for example:

```
IO modbus.27.1220 = int:$value
```

Examples

Reading an integer value from two registers

```
TIME: IO modbus.27.400 = readholdingregisters:uint
IO modbus.27.400 : UISET mydata.text = IO modbus.27.400
```

The first line executes the read holding registers function on a Modbus device with address 27, reading two contiguous registers starting at address 400.

The second line sets a text object of the Web interface with the value read from the Modbus device.

Reading several values at once

```
TIME: IO modbus.27.220 Modbus.27.222 Modbus.27.224 = readinputregisters:float
IO modbus.27.220 : UISET mydata1.text = IO modbus.27.220
IO modbus.27.224 : UISET mydata1.text = IO modbus.27.224
```

The first line is a compact form for reading multiple data points.

Reading binary data

```
TIME: IO modbus.27.1000 = readdiscreteinputs:24
IO modbus.27.1000.1 = 0 : UISET mylabel1.color = g
```

```
IO modbus.27.1000.1 = 1 : UISET mylabel1.color = r
...
IO modbus.27.220.24 = 0 : UISET mylabel24.color = g
IO modbus.27.220.24 = 1 : UISET mylabel24.color = r
```

The first line executes the read discrete inputs function on a Modbus device with address 27, reading two contiguous registers starting at address 1000 and interpreting the first 24 bits as binary values.

The following lines change the color of some objects of the Web interface to green if the bit status is 0 or red if the status is 1.

Reading in extended mode

If the option extendednames was specified in System Settings:

```
TIME: IO modbus.27.1000 = readholdingregisters:ushort
IO modbus.27.1000.03 : UISET mydata.text = IO modbus.27.1000.03
```

The first line executes the read holding register function for an ushort value on a Modbus device with address 27, reading the registers 1000.

The following line changes the attribute text of a text object setting it to the value just read.

Writing a single coil

```
TIME : IO modbus.27.1220 = bit:1
```

Set coil 1220 to on.

Writing multiple coils

```
TIME : IO modbus.27.1220 = bits:100
```

Set three coils starting from 1220 to 1, 0, 0.

Writing registers

```
TIME : IO modbus.27.210 = ushort:2048
TIME : IO modbus.27.220 = float:-123.22
TIME : IO modbus.27.230 modbus.27.240 = long:200000000
```

The first line executes the write single register on a Modbus device with address 27, writing the register at address 210 with an unsigned short number.

The second line executes the write multiple registers on a Modbus device with address 27, writing two consecutive registers starting at address 220 with an IEEE-754 floating-point number.

The third line executes the write multiple registers on a Modbus device with address 27, writing four consecutive registers starting at address 230 and four at 240 with a signed long number.

Release Notes

3.3.1

- added support for function 23, Read/Write Multiple Registers:
  - modbusReadWriteMultipleRegisters() Java method and JavaScript function

3.3.0

- added support for Write Multiple Coils function:
  - writeMultipleCoils() Java method and JavaScript function
  - ID.unitid.address = bits:<0|1>... in EVENTS

3.2.1

- EVENTS: minor fixes to the readcoils and readdiscreteinputs actions

3.2.0

- reading multiple consecutive values with a single Modbus read is now supported for readholdingregisters and readinputregisters

3.1.1

- using format options bitshort, bitint and bitlong for readholdingregisters and readinputregisters, it is now possible to read 1, 2 or 4 registers and return the individual bit values

3.1.0

- new error events on Modbus read or write errors
- new option extendedNames. When this option is set, data point names used to return values will have the postfix .FF, where FF is the 2-digit decimal function code

3.0.3

- fixes to the Modbus write functions
- variables can now be used in value strings

3.0.0

- initial release

Modbus is a registered trademark of Modbus Organization, Inc.

Modbus Server

The Modbus Protocol is a messaging structure. It is used to establish master-slave/client-server communication between intelligent devices. It is a de facto standard. The Modbus RTU specification defines the default communication protocol over serial lines. The Modbus TCP/IP specification takes the Modbus instruction set and wraps TCP/IP around it.

HSYCO's MODBUS TCP SERVER I/O Server implements a Modbus TCP/IP server, allowing external Modbus clients to connect to HSYCO and perform any standard Modbus function.

This support is implemented as a simple Java API.

HSYCO Configuration

Add a MODBUSTCPSERVER I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: local IP address where to accept connections, leave blank to accept connection on any LAN port and IP address
- **IP Port**: TCP/IP port to use, leave blank to use default port 502.

> if more than one MODBUSTCPSERVER I/O servers are defined they have to be on different ports or IP addresses.

High Availability

- **Shutdown when inactive**: defaults to false.

Options

ID	Default	Values	Description
eventslog	false	true	if the general eventsLog option is also true in hsyco.ini, Modbus events for this gateway are written in the log files
		false	Modbus events for this gateway are not written in the log files, even if the eventsLog option is true in hsyco.ini

Modbus Server Integration

The Modbus TCP Server is implemented as a multi-threaded socket server that can accept multiple concurrent connections.

There are no hard-coded limits to the number of concurrent connections and no idle time-out mechanism to disconnect connections that don't do any traffic for a certain amount of time. A socket-level keepalive mechanism is in place to ensure that a socket thread is closed and resources freed if the client drops the connection without a graceful close.

It is the clients' responsibility to ensure the correct handling of connections to the server.

Whenever a client sends a Modbus request to the HSYCO server, the Modbus TCP Server will execute a Java callback method passing the complete PDU as a byte array, and respond with a PDU based on the returned byte array or, if the callback method is not defined or returns null, it will execute the Modbus read or write function using its internal data tables.

Datapoints

The naming convention HSYCO uses for the events is closely related to the Modbus addressing standards and data types.

The data point names always begin with the same format: <ioserver_id>.unitid.address, where *unitid* is the Modbus slave device address, from 1 to 247, and *address* is the initial register address you are reading.

The supported public function codes are:

- 01: Read Coils
- 02: Read Discrete Inputs
- 03: Read Holding Registers
- 04: Read Input Registers
- 05: Write Single Coil
- 06: Write Single Register
- 15: Write Multiple Coils
- 16: Write Multiple Registers
- 22: Mask Write Register
- 23: Read/Write Multiple Registers.

These functions read and write data on four distinct data tables:

Table	Suffix	Object Type	Mode	Functions
Discrete Inputs	.di	Single bit	Read Only	02
Coils	.co	Single bit	Read Write	01 05 15
Input Registers	.ir	16-bit word	Read Only	04
Holding Registers	.hr	16-bit word	Read Write	03 06 16 22 23

Operations on Input Registers and Holding Registers tables support the following data types:

Type code	Description	Return range	Number of registers
ushort	unsigned short integer	0 ... 65535	1
uint	unsigned integer	0 ... $(2^{32} - 1)$	2
ulong	unsigned long integer	0 ... $(2^{64} - 1)$	4
short signed	short integer	-32768 ... +32767	1
int	signed integer	-2^{31} ... $(2^{31} - 1)$	2
long	signed long integer	-2^{63} ... $(2^{63} - 1)$	4
float	signed IEEE 754 floating point	$+/- 2^{-149}$ ... $(2-2^{-23})x2^{127}$	2

Discrete Inputs Data Table

The internal discrete inputs data table can be read by a Modbus client, and written locally to set the discrete inputs' values.

ID	Value	R/W	Description
unitid.address.di	bit:0 bit:false bit:off	W	sets a single bit in the discrete inputs data table to off
	bit:1 bit:true bit:on	W	sets a single bit in the discrete inputs data table to on
	bits:<0|1|-... '-' characters in the sequence are skipped	W	sets multiple bits in the discrete inputs data table. The left-most bit in the sequence is the discrete input at <address>

Coils Data Table

The internal coils data table can be read or written by a Modbus client, and written locally to set the coils' values.

ID	Value	R/W	Description
unitid.address.co	bit:0 bit:false bit:off	W	sets a single coil in the coils data table to off
	bit:1 bit:true bit:on	W	sets a single coil in the coils data table to on
	bits:<0|1|-...> '-' characters in the sequence are skipped	W	sets multiple coils in the coils data table. The left-most bit in the sequence is the coil at <address>
	subscribe	W	subscribes one coil at <address> for event notification
	subscribe:<n>	W	subscribes <n> consecutive coils starting at <address> for event notification
	0 1	R	the status of the coil at <address>. Events are generated, only for coils that have been subscribed, when the coil is written by a Modbus client, even if there is no change of value

The <n> value after subscribecoils: can also be a variable, that will be replaced with its numeric content.

Input Registers Data Table

The internal input registers data table can be read by a Modbus client, and written locally to set the registers' values.

ID	Value	R/W	Description
unitid.address.ir	ushort:<value>	W	writes an unsigned short number to the input registers table
	uint:<value>	W	writes an unsigned integer number to the input registers table
	ulong:<value>	W	writes an unsigned long number to the input registers table
	short:<value>	W	writes a signed short number to the input registers table
	int:<value>	W	writes a signed integer number to the input registers table
	long:<value>	W	writes a signed long number to the input registers table
	float:<value>	W	writes a floating-point number using the IEEE Standard 754 representation to the input registers table
	hex:<value>	W	writes raw hex bytes to the input registers table
	bits:<0|1|-...> '-' characters in the sequence are skipped	W	writes 16 bits to a single register in the input registers table. The leftmost bit is the MSB

Holding Registers Data Table

The internal holding registers data table can be read or written by a Modbus client, and written locally to set the registers' values.

ID	Value	R/W	Description
unitid.address.hr	ushort:<value>	W	writes an unsigned short number to the holding registers table
	uint:<value>	W	writes an unsigned integer number to the holding registers table
	ulong:<value>	W	writes an unsigned long number to the holding registers table
	short:<value>	W	writes a signed short number to the holding registers table
	int:<value>	W	writes a signed integer number to the holding registers table

	long:<value>	W	writes a signed long number to the holding registers table
	float:<value>	W	writes a floating-point number using the IEEE Standard 754 representation to the holding registers table
	hex:<value>	W	writes raw hex bytes to the holding registers table
	bits:<0|1|-...> '-' characters in the sequence are skipped	W	writes 16 bits to a single register in the holding registers table. The leftmost bit is the MSB
	subscribe:ushort	W	subscribes an unsigned short at <address> for event notification
	subscribe:uint	W	subscribes an unsigned int starting at <address> for event notification
	subscribe:ulong	W	subscribes an unsigned long starting at <address> for event notification
	subscribe:short	W	subscribes a signed short integer at <address> for event notification
	subscribe:int	W	subscribes a signed int starting at <address> for event notification
	subscribe:long	W	subscribes a signed long starting at <address> for event notification
	subscribe:float	W	subscribes a floating-point number using the IEEE Standard 754 representation starting at <address> for event notification
	subscribe:hex	W	subscribes a single register at <address> for event notification using the hex notation
	subscribe:bits	W	subscribes the 16 individual bits of a single register at <address> for event notification
	data type value	R	the decimal or hexadecimal value of the data type at <address>. Events are generated, based on subscribed data types and addresses, when the data is written by a Modbus client, even if there is no change of value

unitid.address.hr.<n> <n> from 1 to 16; <n>=1 is the LSB of first byte; <n>=8 is MSB of first byte; <n>=9 is LSB of second byte; <n>=16 is MSB of second byte	0 or 1 bit value	R	the individual bits of a holding register at <address>. Events are generated, for bits subscriptions only, when the data is written by a Modbus client, even if there is no change of value

Java Callback Methods API

ModbusEvent

```
static byte[] ModbusEvent(String name, String InetAddress addr, int unitid, byte[] pdu)
```

This event is triggered by any valid Modbus request from a client connected to the IP address and port of the HSYCO's Modbus server, and can override the default server's response.

This method should return a valid PDU, that will be sent back to the client. If the method returns null, the server will perform the Modbus function on its internal data tables.

Parameters:

- name - the I/O Server name
- addr - the client's address
- unitid - the Modbus unit id set in the client's request
- pdu - the full PDU byte array, starting with the function code.

Examples

Subscribe request

Subscribing in required whenever you need to generate ad IO event for a specified register. For example writing:

```
io modbusserver.connection = online : io plc.1.1.hr = subscribe:bits
io modbusserver.connection = online : io plc.1.2.hr = subscribe:short
```

means that every time the registers 1 or 2 of the modbus unit with address 1 are changed the corresponding IO event is generated.

Write on the modbus table

Here's and example of how to write a value in the modbus server registers table.

$value : io modbusserver.1.500.hr modbusserver.1.501.hr modbusserver1.502.hrmodbusserver.1.503.hr modbusserver.1.504.hr modbusserver.1.505.hr = "int:"$value, log = "MODBUS SERVER WRITE!!"

Read Coils request

```
public static byte[] ModbusEvent(String name, InetAddress addr, int unitid, byte[] pdu) throws Exception {
        int registerAddress, quantity;

        if (unitid == 1) {
                if (pdu.length == 5 && pdu[0] == 1) {
```

```
                        registerAddress = ((pdu[1] & 0xff) << 8) + (pdu[2] & 0xff);
                        quantity = ((pdu[3] & 0xff) << 8) + (pdu[4] & 0xff);
                        if (quantity == 1) {
                                if (registerAddress % 2 == 0) {
                                        byte[] returnPDU = {0x01, 0x01, 0x00};
                                        return returnPDU;
                                } else {
                                        byte[] returnPDU = {0x01, 0x01, 0x01};
                                        return returnPDU;
                                }
                        }
                }
        }
        return null;
}
```

This quite useless code segment will respond to a Read Coils Modbus request to read a single coil value from unit 1. The response will be 0 when the requested coil address is an even number, and 1 for odd address numbers. The callback will respond with an exception PDU with exception code 04.

Release Notes

3.3.1

- added support for function 23, Read/Write Multiple Registers

3.3.0

- initial release version

Modbus is a registered trademark of Modbus Organization, Inc.

Notifier

The NOTIFIER AM-2000, AM-4000 and AM-6000 are multi-area fire panels. The integration with HSYCO can be accomplished via a standard Modbus RTU slave interface.

Communication

The NOTIFIER SIB 600 Modbus interface is required. The default connection parameters are:

Modbus address: 1 (can be set to 1-32)

RS-485 parameters:

Baud rate	9600
Data bits	8
Stop bit	1
Parity	none
Flow control	none

Refer to the Notifier manuals for installation, wiring and power supply requirements.

HSYCO Configuration

Add a NOTIFIER I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select a MODBUSTCP I/O server.

High Availability

- **Shutdown when inactive**: defaults to false.

Options

ID	Default	Values	Description
address	1	1 ... 32	the panel's Modbus address
gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
pollinterval	1	> 0	set to the highest zone number configured in the panel. Set to 0 to disable zone status polling and control commands
lastzone	150	0 ... 150	set to the highest zone number configured in the panel. Set to 0 to disable zone status polling and control commands
loglines	100	≥ 0	if GUI support is enabled, set to the maximum number of lines to show in the log text object. Set to 0 to disable the GUI log

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
annunciator.<n>.<m>.offline	1	R	annunciator <n>.<m> offline
	0	R	annunciator <n>.<m> online
annunciator.<n>.<m>.fault	1	R	annunciator <n>.<m>fault
	0	R	annunciator <n>.<m> no fault
annunciator.<n>.<m>.cfg.error	1	R	annunciator <n>.<m> configuration error
	0	R	annunciator <n>.<m> configured
lcd.<n>.offline	1	R	LCD <n> offline
	0	R	LCD <n> online

lcd.<n>.fault	1	R	LCD <n> fault
	0	R	LCD <n> no fault
lcd.<n>.cfg.error	1	R	LCD <n> configuration error
	0	R	LCD <n> configured
loop.<n>.short	1	R	loop <n> short circuit
	0	R	loop <n> no short circuit
loop.<n>.open	1	R	loop <n> open fault
	0	R	loop <n> no open fault
loop.<n>.voltage	1	R	loop <n> low voltage fault
	0	R	loop <n> no low voltage fault
module.<l>.<n>.normal	1	R	module <n> on loop <l> normal state
	0	R	module <n> on loop <l> error state
module.<l>.<n>.disabled	1	R	module <n> on loop <l> disabled
		W	disable module <n> on loop <l>
	0	R	module <n> on loop <l> enabled
		W	enable module <n> on loop <l>
module.<l>.<n>.disabled	1	R	module <n> on loop <l> output active (ON)
		W	turn ON out of module <n> on loop <l>
	0	R	module <n> on loop <l> output not active (OFF)
		W	turn OFF out of module <n> on loop <l>
module.<l>.<n>.fault	1	R	module <n> on loop <l> fault
	0	R	module <n> on loop <l> no fault
module.<l>.<n>.alarm	1	R	module <n> on loop <l> alarm
	0	R	module <n> on loop <l> no alarm
panel.fault.batt.disconn	1	R	batteries disconnected

	0	R	batteries connected
panel.fault.power	1	R	batteries disconnected
	0	R	batteries connected
panel.fault.batt.unbal	1	R	batteries disconnected
	0	R	batteries connected
panel.fault.batt.fail	1	R	batteries unbalanced charge
	0	R	batteries no unbalanced charge
panel.fault.bex	1	R	external batteries fault
	0	R	no external batteries fault
panel.fault.earth	1	R	ground connection fault
	0	R	no ground connection fault
panel.fault.undervoltage	1	R	AC power under-voltage fault
	0	R	no AC power under-voltage fault
panel.fault.overvoltage	1	R	AC power over-voltage fault
	0	R	no AC power over-voltage fault
panel.fault.loop.open	1	R	at least one loop is open
	0	R	no loops open
panel.fault.loop.short	1	R	at least one loop short circuit
	0	R	no loop short circuit
panel.fault.fuse.user	1	R	fuse fault
	0	R	fuse ok
panel.fault.batt.charger.undervoltage	1	R	battery charger under-voltage fault
	0	R	no battery charger under-voltage fault
panel.fault.batt.charger.overvoltage	1	R	battery charger over-voltage fault
	0	R	no battery charger over-voltage fault
panel.filesystem.clear.error	1	R	memory delete error
	0	R	no memory delete error
panel.filesystem.init.error	1	R	memory initialization error

	0	R	no memory initialization error
panel.reliab.par.mod	1	R	reliability parameters modified warning
	0	R	no reliability parameters modified warning
panel.hardkey.enabled	1	R	hardware programming key enabled
	0	R	hardware programming key not enabled
panel.power.on	1	R	power on phase
	0	R	power on phase completed
panel.wdt.reset	1	R	main CPU watchdog fault
	0	R	main CPU watchdog ok
panel.cfg.error	1	R	configuration error
	0	R	configuration ok
panel.filesystem.error	1	R	memory corruption
	0	R	memory ok
panel.hard.clock	1	R	internal hardware clock fault
	0	R	internal hardware clock ok
panel.arch.full	1	R	log archive full
	0	R	log archive not full
panel.arch.full.warning	1	R	log archive warning (80% full)
	0	R	no log archive warning
panel.zone.default	1	R	sensors not yet assigned to zones
	0	R	zones assignment done
panel.sib.offline	1	R	SIB board offline
	0	R	SIB board online
panel.sib.not.prg	1	R	SIB board not programmed
	0	R	SIB board programmed
panel.pc.invalid.cmd	1	R	invalid command from PC
	0	R	no invalid command from PC
panel.arch.disabled	1	R	log archive disabled

	0	R	log archive disabled
panel.lib.<n>.offline	1	R	LIB board <n> offline
	0	R	LIB board <n> online
panel.lib.<n>.cfg.error	1	R	LIB board <n> configuration error
	0	R	LIB board <n> configured
panel.lib.<n>.not.prg	1	R	LIB board <n> not programmed
	0	R	LIB board <n> programmed
panel.fire	silence	W	fire alarm silence
	reset	R	fire alarm reset
panel.fire.normal	1	R	fire panel normal state
	0	R	fire panel error state
panel.fire.exclusions	1	R	fire panel has one or more disabled components
	0	R	fire panel's all components are enabled
panel.fire.test	1	R	fire panel test
	0	R	fire panel no test
panel.fire.fault	1	R	fire panel fault
	0	R	fire panel no fault
panel.fire.alarm	1	R	fire panel alarm
	0	R	fire panel no alarm
panel.fire.alarm.silent	1	R	fire panel alarm silenced
	0	R	fire panel no alarm silenced
panel.fire.unknown	1	R	fire panel unknown events
	0	R	fire panel no unknown events
panel.tech	silence	R	technical alarm silence
	reset	R	technical alarm reset
panel.tech.normal	1	R	technical panel normal state
	0	R	technical panel error state
panel.tech.exclusions	1	R	technical panel has one or more disabled components

	0	R	technical panel's all components are enabled
panel.tech.test	1	R	technical panel test
	0	R	technical panel no test
panel.tech.fault	1	R	technical panel fault
	0	R	technical panel no fault
panel.tech.alarm	1	R	technical panel alarm
	0	R	technical panel no alarm
panel.tech.unknown	1	R	technical panel unknown events
	0	R	technical panel no unknown events
sensor.<l>.<n>.normal	1	R	sensor <n> on loop <l> normal state
	0	R	sensor <n> on loop <l> error state
sensor.<l>.<n>.disabled	1	R	sensor <n> on loop <l> disabled
		W	disable sensor <n> on loop <l>
	0	R	sensor <n> on loop <l> enabled
		W	enable sensor <n> on loop <l>
sensor.<l>.<n>.prealarm.1	1	R	sensor <n> on loop <l> pre-alarm 1
	0	R	sensor <n> on loop <l> pre-alarm 1 off
sensor.<l>.<n>.prealarm.2	1	R	sensor <n> on loop <l> pre-alarm 2
	0	R	sensor <n> on loop <l> pre-alarm 2 off
sensor.<l>.<n>.fault	1	R	sensor <n> on loop <l> fault
	0	R	sensor <n> on loop <l> no fault
sensor.<l>.<n>.alarm	1	R	sensor <n> on loop <l> alarm
	0	R	sensor <n> on loop <l> no alarm
zone.<n>.normal	1	R	zone <n> normal state
	0	R	zone <n> error state
zone.<n>.disabled	1	R	zone <n> disabled

		W	disable zone <n>
	0	R	zone <n> enabled
		W	enable zone <n>
zone.<n>.fault	1	R	zone <n> fault
	0	R	zone <n> no fault
zone.<n>.alarm	1	R	zone <n> alarm
	0	R	zone <n> no alarm

User Interface

UISET Actions

You can use any object that accepts a text attribute, usually text, but also marquee and others, to display the status of the system according to the following table:

ID	Attribute	Set to
log	value	a textual log of all relevant events. The last event is at the top, with older events scrolled down, then removed from the log past the log lines limit
log0	value	latest line of the security log when a new entry is added, set back to blank after a short period
offline.label	visible	true when HSYCO can't connect to the Notifier panel, false otherwise
annunciator.<n>.<m>.offline.label	visible	true when annunciator <n>.<m> is offline
annunciator.<n>.<m>.fault.label	visible	true when annunciator <n>.<m> fault
annunciator.<n>.<m>.cfg.error.label	visible	true when annunciator <n>.<m> has a configuration error
cd.<n>.offline.label	visible	true when LCD <n> is offline
lcd.<n>.fault.label	visible	true when LCD <n> is showing a fault condition
lcd.<n>.cfg.error.label	visible	true when LCD <n> has a configuration error
loop.<n>.short.label	visible	true when loop <n> short circuit
loop.<n>.open.label	visible	true when loop <n> is open
loop.<n>.voltage.label	visible	true when loop <n> voltage is too low

module.<l>.<n>.normal.label	visible	true when module <n> on loop <l> normal state
module.<l>.<n>.disabled.label	visible	true when module <n> on loop <l> disabled
module.<l>.<n>.out.label	visible	true when module <n> on loop <l> output is ON
module.<l>.<n>.fault.label	visible	true when module <n> on loop <l> fault
module.<l>.<n>.alarm.label	visible	true when module <n> on loop <l> alarm
panel.fire.normal.label	visible	true when fire panel normal state
panel.fire.exclusions.label	visible	true when fire panel has one or more disabled components
panel.fire.test.label	visible	true when fire panel test mode
panel.fire.fault.label	visible	true when fire panel fault
panel.fire.alarm.label	visible	true when fire panel alarm
panel.tech.normal.label	visible	true when technical panel normal state
panel.tech.exclusions.label	visible	true when technical panel has disabled components
panel.tech.test.label	visible	true when technical panel test mode
panel.tech.fault.label	visible	true when technical panel fault
panel.tech.alarm.label	visible	true when technical panel alarm
sensor.<l>.<n>.normal.label	visible	true when sensor <n> on loop <l> normal state
sensor.<l>.<n>.disabled.label	visible	true when sensor <n> on loop <l> disabled
sensor.<l>.<n>.prealarm.1.label	visible	true when Sensor <n> on loop <l> pre-alarm 1
sensor.<l>.<n>.prealarm.2.label	visible	true when Sensor <n> on loop <l> pre-alarm 2
sensor.<l>.<n>.fault.label	visible	true when sensor <n> on loop <l> fault
sensor.<l>.<n>.alarm.label	visible	true when sensor <n> on loop <l> alarm
zone.<n>.normal.label	visible	true when zone <n> normal state

zone.<n>.disabled.label	visible	true when zone <n> disabled
zone.<n>.fault.label	visible	true when zone <n> fault
zone.<n>.alarm.label	visible	true when zone <n> alarm

USER Commands

You can use the user object, or any other GUI object that generates user events, to execute any command that could be sent writing to a data point, as defined in the | Datapoints table. Set the user object's name field to the datapoint (event) name, prefixed by the IO Server id, and its param field to the value that should be written to execute the command.

For example, to disable zone 1, use the following object:

```
(user fire.zone.1.disabled; 1; r4c2; gr; zone 1 off)
```

To enable zone 1:

```
(user fire.zone.1.disabled; 0; r4c3; gr; zone 1 on)
```

Release Notes

3.3.0

- initial release

Honeywell and Notifier are registered trademarks owned by Honeywell International Inc., or its subsidiaries or affiliates.

Nuvo

NuVo Grand Concerto and Essentia E6G are multi-source, multi-zone audio distribution systems that may be integrated to work with external control systems and servers. The central unit is called Main Processor Unit (MPU).

It provides power supply, source audio routing to the zones, amplified stereo audio output for each zone and centralized control and interface for all zones.

Communication

Employ a DE-9 (often called DB-9) male-female RS-232 straight cable to connect the RS-232 MPU port to the serial port on HSYCO SERVER.

You can also define and connect to more than one NUVO MPU to control multiple systems.

RS-232 parameters:

Baud rate	57600
Data bits	8
Stop bit	1
Parity	none
Flow control	none

Nuvo Configuration

On the NUVO device, a number of contiguous sources starting from the source 1 must be configured.

One of the areas 17 to 20 must be reserved for communication with HSYCO.

HSYCO Configuration

Add a Nuvo I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select the comm port connected to the panel.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands

		false	disable UI support
menuzone	20	17 ... 20	defines which virtual zone will be used to access the menu system
menulines	10		the number of menu item lines

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established to the MPU
	offline	R	HSYCO can't connect to the MPU
system.version	<firmware id>	R	version information of the firmware installed on the MPU
system.power	1	R	at least one zone is on
	0	R	the ALLOFF command has been sent and all zones are off
		W	turn the system off. An “off” value is equivalent to 0
system.mute	1	R	system muted
		W	mute all zones. An “on” value is equivalent to 1
	0	R	system un-muted
		W	return all zones to the volume level as set before the system mute command was sent. An “off” value is equivalent to 0
system.page	1	R	page mode is on
		W	enter page mode. An “on” value is equivalent to 1
	0	R	page mode is off
		W	turn page mode off. An “off” value is equivalent to 0
system.message	<text>	W	send a text message to all Control Pads, up to 50 characters
system.response	<text>	R	the native protocol response message. This event is generated only if the message is

			ignored or not recognized by the driver
system.command	<text>	W	send a native protocol command to the MPU. Do not include the leading “*” character and trailing carriage return in the text value when setting this IO variable
z<n>.name	<name>	R	zone <n> name
z<n>.power	1	R	zone <n> is on
		W	turn zone <n> on. An “on” value is equivalent to 1
	0	R	zone <n> is off
		W	turn zone <n> off. An “off” value is equivalent to 0
z<n>.source	<s>	R	zone <n> audio source is <s> (from 1 to 6)
		W	set zone <n> to source
	next	W	set zone <n> to the next source
z<n>.volume	off	R	zone <n> volume is muted
		W	mute zone <n>. A 0 value is also interpreted as mute
	on	W	un-mute zone <n>
	<x%>	R	zone <n> volume level is 1% to 100%
		W	set zone <n> volume level to X%
	<x>	W	set the zone <n> volume level. <x> is a number between 0 and 100. Decimal numbers are accepted and truncated
	<x>/<y>	W	set the zone <n> volume level. <x>/<y> is a fractional number between 0 and 100. Decimal numbers are accepted and truncated
	up	W	increase volume (by one step) for zone <n>
	down	W	decrease volume (by one step) for zone <n>
z<n>.mute	1	R	zone <n> volume is muted

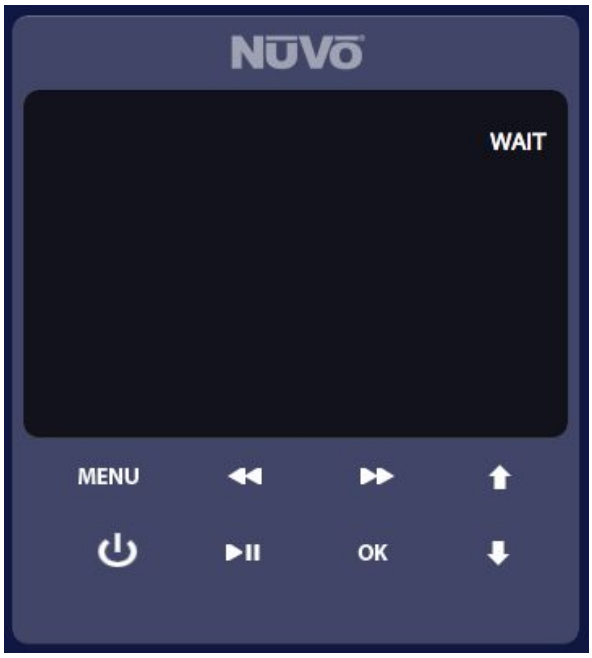
	0	W	mute zone <n>. An “on” value is equivalent to 1
z<n>.dnd	1	R	Do not disturb (DND) mode is on for zone <n>
		W	enable do not disturb mode for zone <n>. An “on” value is equivalent to 1
	0	R	Do not disturb (DND) mode is off for zone <n>
		W	disable do not disturb mode for zone <n>. An “off” value is equivalent to 0
z<n>.lock	1	R	zone <n> is locked
		W	enable lock mode for zone <n>. An “on” value is equivalent to 1
	0	R	zone <n> is not locked
		W	disable lock mode for zone <n>. An “off” value is equivalent to 0
	on	W	lock zone <n>
	<xxxx>	W	unlock zone <n>. <xxxx> is the four digit unlock code
z<n>.button	prev	R	the PREV [<<] button has been pressed on the Control Pad for zone <n>
		W	simulate zone PREV [<<] button on Control Pad
	next	R	the NEXT [<<] button has been pressed on the Control Pad for zone <n>
		W	simulate zone NEXT [<<] button on Control Pad
	playpause	R	the PLAY/PAUSE [<<] button has been pressed on the Control Pad for zone <n>
		W	simulate zone PLAY/PAUSE [<<] button on Control Pad
z<n>.macro	<m>	R	the IR macro number <m> is exectued
z<n>.party	1	W	make zone <n> the party host
	0	W	release party host mode for zone <n>

z<n>.playlist	<x>	W	select a favorite playlist for zone <n>. The <x> value should be 1 to 12
z<n>.bright	<x>	W	set the Control Pad display's brightness for zone <n>. The <x> value should be 1 (min) to 7 (max)
s<n>.name	<name>	R	source <n> name
s<n>.displine.1	<text>	R	the display line 1 text for source <n>
s<n>.displine.2	<text>	R	the display line 2 text for source <n>
s<n>.displine.3	<text>	R	the display line 3 text for source <n>
s<n>.displine.4	<text>	R	the display line 4 text for source <n>
s<n>.lenght		R	length of the song in 10th seconds. Only updated when the song starts playing
s<n>.time		R	current run time of the song in 10th seconds. Only updated when the song starts playing
s<n>.status	normal	R	normal mode for not NuVo Net sources
	idle	R	idle mode for not NuVo Net sources
	play	R	playing
	pause	R	paused
	ffd	R	fast forward
	rew	R	rewind
	playshuffle	R	playing, shuffle mode
	playrepeat	R	playing, repeat mode
	playshufflerepeat	R	playing, shuffle/repeat mode

User Interface

Nuvo Object

The user interface for the Nuvo multi-room system:



The Nuvo object is listed in the Project Editor's new object list only when at least one Nuvo I/O Server is defined.

Parameters

- **server id**: the server ID
- **zone number**: zone number of the Nuvo multi-room system
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

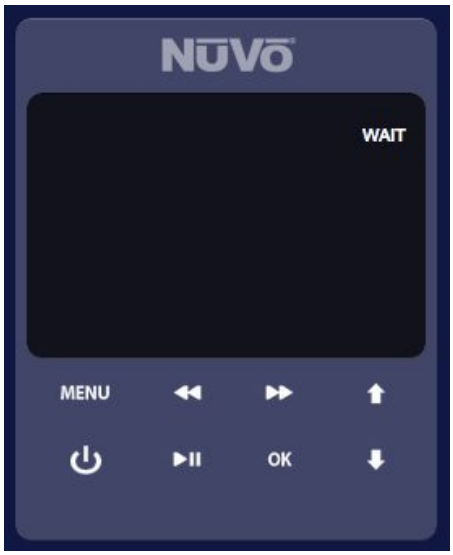
```
(nuvo <server id>; <zone number>; <position>)
```

E.g.

```
(nuvo serverid; 1; x10y20)
```

NuvoMini Object

The user interface for the Nuvo multi-room system:



The NuvoMini object is listed in the Project Editor's new object list only when at least one Nuvo I/O Server is defined.

Parameters

- **server id**: the server ID
- **zone number**: zone number of the Nuvo multi-room system
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(nuvomini <server id>; <zone number>; <position>)
```

E.g.

```
(nuvomini serverid; 1; x10y20)
```

UISET Actions

ID	Attribute	Set to
z<n>.zonedisplay	visible	true when the menu display for the zone is not active, false otherwise
z<n>.name	value	the zone name, retrieved from the MPU configuration
z<n>.source	value	the source number followed by the source name, as set in the MPU
z<n>.volume.status	value	invisible: zone off; MUTE: muted; <x%>: volume level, 1% to 100%

z<n>.displine.1	value	the line 1 of the source display, for the source currently in use by this zone
z<n>.displine.2	value	the line 2 of the source display, for the source currently in use by this zone
z<n>.displine.3	value	the line 3 of the source display, for the source currently in use by this zone
z<n>.displine.4	value	the line 4 of the source display, for the source currently in use by this zone
z<n>.status	value	ON: generic on for not NuVo Net sources; OFF: off; PAUSE: paused; PLAY: playing; PLAY [SH]: playing, shuffle mode; PLAY [REP]: playing, repeat mode; PLAY [SH][REP]: playing, shuffle repeat mode; FAST FORWARD: fast forward mode; REWIND: rewind mode
z<n>.menudisplay	visible	true when the menu display for the zone is active
menu.wait	visible	true when the NuVo MPU takes some time to retrieve a menu (this can happen when retrieving long lists of songs from the iPod, for example)
menu.name	value	menu title
menu.line.<n>	value	the menu items' text lines. <n> starts from 1, up to the number of menu display lines as set in the options (default is 10)
menu.line.header	value	this object shows a few dots (....) when you can scroll up the menu. Place this object above the first menu line
menu.line.footer	value	this object shows a few dots (....) when you can scroll down the menu. Place this object below the last menu line
z<n>.volume	value	0: mute or zone off; <x%>: volume level, 1% to 100%

USER Commands

Name	Param	Action

z<n>	power	zone power on/off button
z<n>	menu	simulate MENU button on Control Pad
z<n>	up	simulate UP button on Control Pad (increment volume level in approx. 5% steps, or move up one line in the menu)
z<n>	down	simulate DOWN button on Control Pad (decrement volume level in approx. 5% steps, or move down one line in the menu)
z<n>	playpause	simulate zone PLAY/PAUSE button on Control Pad
z<n>	ok	simulate zone OK button on Control Pad
z<n>	prev	simulate zone PREV [<<] button on Control Pad
z<n>	next	simulate zone NEXT [>>] button on Control Pad
z<n>	source	cycle zone sources
z<n>	source.	select source ~~(1 to 6) for this zone~~

Release Notes

3.2.1

- log more verbose in case of failure of initial reading

3.0.1

- various bug fixes
- adds support for z<n>.volume.status text field
- adds support for z<n>.volume.up and <ioserver_id>.z<n>.volume.down (user) buttons

3.0.0

- full integration in HSYCO 3.0.0

NuVo is a registered trademark of NuVo Technologies Inc.

Paradox

PARADOX® EVO48 and EVO192 are advanced, multi-area security systems integrated with a variety of wired and wireless sensors.

The EVO48 supports up to 48 zones in 4 areas (partitions), while the EVO192 supports 192 zones in 8 areas. Both units support programmable outputs (PGM) and virtual inputs.

The Paradox EVO driver in HSYCO supports the integration of more than one EVO unit within the same HSYCO interface.

The integration with HSYCO can be accomplished via direct control through the RS-232 port on the PRT3 Integration Module.

Communication

Employ a DE-9 (often called DB-9) male-female RS-232 straight cable to connect the PRT3 to the serial port on HSYCO SERVER.

RS-232 parameters:

Baud rate	same as defined on the PRT3 (default 9600)
Data bits	8
Stop bit	1
Parity	no parity
Flow control	none

Using the software Winload (default user: Master; default password: 1234) set the PRT3 speed to the appropriate rate setting options 2 and 3 of section 016. You can use any of the speeds that the PRT3 supports. 9600 or higher is recommended. Enable serial communication and home automation protocol (Ascii protocol).

HSYCO Configuration

Add a PARADOX I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select the comm port the device is connected to.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands

		false	disable UI support
language	system language	en	language of log and display messages. Supported values are it and en
		it	
zoneactivitylog	false	true	logs the open/close status of zones in the log page and security.log files
		false	disable zone status log
maxzone	0	0 ... 192	force zones status detect and data points initialization at startup (zones data points are initialized with events disabled) for zones 1 to maxzone value
userlogs	false	true	generates logs that can be viewed with the userlist object, and can be selected to generate user events
		false	logs for userlist objects disabled. Normal text logs are generated even when this option is false
areanamelog	true	true	adds the area name to all log messages
		false	remove the area name (if only one area is used, having the area name in the logs could be useless)
code		code	the user code to be used when sending IO commands to the EVO, if code is not explicitly set in the command
logsize	50	n ≥ 0	the number of log lines to display in the UI object

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
area.<n>	arm	W	arm area <n> using default code
	disarm	W	disarm area <n> using default code
	force	W	arm area <n> using default code

	stay	W	arm area <n> using default code
	instant	W	arm area <n> using default code
	arm.<c>	W	arm area <n> using code <c>
	disarm.<c>	W	disarm area <n> using code <c>
	force.<c>	W	forced arm area <n> using code <c>
	stay.<c>	W	stay arm area <n> using code <c>
	instant.<c>	W	instant arm area <n> using code <c>
area.<n>.mode	disarmed	R	the area is disarmed
	armed	R	the area is armed
	forced	R	the area is armed (force mode)
	stay	R	the area is armed (stay mode)
	instant	R	the area is armed (instant mode)
area.<n>.status	alarm	R	area in alarm
	strobe	R	area in strobe (silent alarm)
	reset	R	area alarm end
area.<n>.trouble	trouble	R	general trouble in the area
	reset	R	trouble end
zone.<n>.status	closed	R	zone is closed
	open	R	zone is open
	tamper	R	zone tampered
	fireloop	R	fire loop trouble
zone.<n>.alarm	alarm	R	zone alarm
	fire	R	zone fire alarm
	reset	R	alarm end
zone.<n>.trouble	radio	R	zone radio supervision lost
	battery	R	zone low battery warning
	reset	R	zone trouble end
trouble	online	R	connection established

	offline	R	connection failed
pgm.<n>	on	R	the PGM is activated
	off	R	the PGM is deactivated
input.<n>	open	RW	the virtual input is open
	0	W	set virtual input to open
	closed	RW	set virtual input to closed
	1	W	set virtual input to open

User Interface

Paradox Object

The user interface for the control and supervision of the Paradox EVO multi-area intrusion detection system:

File:UI Object paradox.png

The Paradox object is listed in the Project Editor's new object list only when at least one Paradox I/O Server is defined.

Parameters

- **server id**: the server ID
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(paradox <server id>; <position>)
```

E.g.

```
(paradox serverid; x10y20)
```

UISET Actions

ID	Attribute	Set to
areas	value	shows the status of all areas (up to 8)
log0	value	shows the latest entry of the security log for a short period
logs	value	shows the latest entries of the security log (as many as specified in hsyco.ini)
zones	value	shows the status of all known zones (zones are added to this list as soon as an event is detected on a zone)

connection.label.online	visible	true when the panel is connected to HSYCO, false otherwise
connection.label.offline	visible	true when the panel is not connected to HSYCO, false otherwise
area.<n>.mode.label.disarmed	visible	true when area <n> is disarmed, false otherwise
area.<n>.mode.label.armed	visible	true when area N is armed, false otherwise
area.<n>.mode.label.force	visible	true when area <n> is armed (force mode), false otherwise
area.<n>.mode.label.stay	visible	true when area <n> is armed (stay mode), false otherwise
area.<n>.mode.label.instant	visible	true when area <n> is armed (instant mode), false otherwise
area.<n>.status.label.alarm	visible	true when area <n> alarm is active, false otherwise
area.<n>.trouble.label.trouble	visible	true when trouble detected for area <n>, false otherwise
zone.<n>.status.label.closed	visible	true when zone N is closed, false otherwise
zone.<n>.status.label.open	visible	true when zone <n> is open, false otherwise
zone.<n>.status.label.tamper	visible	true when tamper detected for zone <n>, false otherwise
zone.<n>.status.label.fireloop	visible	true when fire loop detected for zone <n>, false otherwise
zone.<n>.alarm.label.fire	visible	true when zone <n> fire alarm is active, false otherwise
zone.<n>.alarm.label.alarm	visible	true when zone <n> alarm is active, false otherwise
zone.<n>.trouble.label.radio	visible	true when radio trouble detected for zone <n>, false otherwise
zone.N.trouble.label.battery	visible	true when battery trouble detected for zone <n>, false otherwise

Log Files

A permanent record of all the events shown in the log page is saved in a file called security.log in the logs/YYYY directory for the current year.

This file is never overwritten or deleted by HSYCO.

```
2014.05.12 08:48:59.182 - Inserimento
2014.05.12 08:48:59.195 - [Sensors] Inserimento istantaneo
2014.05.12 08:49:09.207 - [] Accesso remoto
2014.05.12 08:49:09.431 - [Sensors] Master: Disinserito
2014.05.12 09:10:06.703 - [Sensors] Kitchen: Aperta
2014.05.12 09:10:10.659 - [Sensors] Kitchen: Chiusa
2014.05.12 09:10:20.867 - [Sensors] Kitchen: Aperta
2014.05.12 09:10:24.843 - [Sensors] Kitchen: Chiusa
2014.05.12 09:17:26.867 - [] Accesso remoto
2014.05.12 09:17:27.151 - [Sensors] Master: Inserito
2014.05.12 09:17:27.343 - Inserimento
2014.05.12 09:17:35.807 - [] Accesso remoto
2014.05.12 09:17:36.059 - [Sensors] Master: Disinserito
2014.05.12 09:20:12.603 - [Sensors] Living: Aperta
2014.05.12 09:20:16.511 - [Sensors] Living: Chiusa
2014.05.12 09:20:26.791 - [Sensors] Living: Aperta
2014.05.12 09:20:30.675 - [Sensors] Living: Chiusa
```

Other information related to the ParadoxEVO driver are saved in the daily message.log files.

Release Notes

3.3.0

- ParadoxEVO driver converted to I/O Server
- Java API write command methods are now obsolete and deprecated, replaced by IO write commands
- new (paradox) object
- new option userlogs. Used to create (userlist) clickable logs

2.1.0

- integrated with HSYCO 2.10.0
- you can now embed the main keypad GUI components in a container with id IDmainkeypad, for example (container !evomainkeypad r1c1). This container will be made invisible when the code keypad is displayed, so that any custom graphic object that is embedded in the container will be made invisible as well, and visible again when you exit the code keypad
- added areanamelog option in paradox.ini

2.0.0

- full integration within HSYCO 2.9.0

Paradox and EVO are registered trademarks of Paradox Security Systems.

PowerOne

The Aurora Power-One inverters allow for remote supervision of the system providing with several measurement data. The integration with HSYCO allows for the visualization of such data and the creation of dynamic charts.

This driver complies with the Aurora communication protocol rel. 4.7.

Communication

The driver supports communication through RS-485 serial interface.

RS-485 parameters:

Baud rate	19200
Data bits	8
Stop bit	1
Parity	none
Flow control	none

HSYCO Configuration

Add a POWERONE I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select the comm port the device is connected to.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UISET actions
		false	disable UI support
pollinterval	20	n > 0	the data acquisition interval, in seconds
addr	2	2 ... 255	specifies the address number of the aurora inverter. If more inverters are connected, it is possible to specify the list of addresses separated by "/". For instance 2/3/4.

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the inverter
<n>.gridpower	<val>	R	the current power production of the inverter at address <n> is <val> Watt
<n>.gridpowerglob	<val>	R	the global power production received from the inverter at address <n> is <val> Watt
<n>.powerpeak	<val>	R	the historical power production peak of the inverter at address <n> is <val> Watt
<n>.powerpeaktoday	<val>	R	today's power production peak of the inverter at address <n> is <val> Watt
<n>.dayenergy	<val>	R	the daily energy production of the inverter at address <n> is <val> Watt hour
<n>.weekenergy	<val>	R	the weekly energy production of the inverter at address <n> is <val> Watt hour
<n>.monthenergy	<val>	R	the monthly energy production of the inverter at address <n> is <val> Watt hour
<n>.yearenergy	<val>	R	the annual energy production of the inverter at address <n> is <val> Watt hour
<n>.totenergy	<val>	R	the total energy production of the inverter at address <n> is <val> Watt hour
<n>.partenergy	<val>	R	the partial energy production of the inverter at address <n> is <val> Watt hour

User Interface

UISET Actions

ID	Attribute	Set to	
connection	visible	true	when datapoint connection = offline

		false	when datapoint connection = online
<n>.gridpower	value	the instant power production of the inverter at address <n> expressed in kW	
<n>.gridpowerglob	value	the instant power production received from the inverter at address <n> of the whole framework expressed in kW	
<n>.powerpeak	value	the historical power peak of the inverter at address <n> in kW	
<n>.powerpeaktoday	value	today's power peak of the inverter at address <n> in kW	
<n>.dayenergy	value	the daily energy production of the inverter at address <n> in kWh	
<n>.monthenergy	value	the monthly energy production of the inverter at address <n> in kWh	
<n>.yearenergy	value	the annual energy production of the inverter at address <n> in kWh	
<n>.totenergy	value	the total energy production of the inverter at address <n> in kWh	
<n>.partenergy	value	the partial energy production of the inverter at address <n> in kWh	

Release Notes

3.2.0

- initial version release

Power-One and AURORA are registered trademarks of Power-One, Inc.

RayCONTROL

RayCONTROL 30 is the control unit for the FCC Planterm temperature control system.

RayCONTROL 30 supports up to 30 individual zones.

Communication

RayNET is an interface module that provides a standard Modbus/RTU interface over RS485 for RayCONTROL 30. The default connection parameters are:

Modbus address: 1

RS-485 parameters:

Baud rate	9600
Data bits	8
Stop bit	1
Parity	none
Flow control	none

Refer to the RayNET and RayCONTROL 30 technical documentation for installation, wiring and power supply requirements.

HSYCO connects to the Modbus bus through any standard Modbus/TCP gateway.

HSYCO Configuration

Add a RAYCONTROL I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select a MODBUSTCP I/O server.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support

startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
pollinterval	30	1 ... 45	the data acquisition interval, in seconds
address	1	> 0	the Modbus address of the RayCONTROL device
readinterval	25	0 ... 1000	minimum interval between consecutive Modbus commands
systems	1	1 ... 2	set to 1 if using only system 1; set to 2 is system 2 is used
zones	15	1 ... 30	set to the highest zone number used
sensivity	5	>0	measured temperatures' changes are reported only if the difference from last reading is equal or greater than the sensivity option, in 10th of Celsius degrees
language	en	en, fr, it	the language used for text messages. If undefined, the global language option is used. English is used if the global language is also undefined

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
s<n>.status	off	R	system <n> is off
		W	turn off system <n>
	on	R	system <n> is on
		W	turn on system <n>
	maint	R	system <n> is in maintenance mode
s<n>.mode	winter	R	system <n> is in winter mode
		W	set system <n> to winter mode

	summer	R	system <n> is in winter mode
		W	set system <n> to summer mode
	towinter	R	system <n> is in summer mode, transitioning to winter mode
	tosummer	R	system <n> is in winter mode, transitioning to summer mode
s<n>.setpoint.winter.max	160 .. 300	R	system <n> current maximum winter temperature set-point
		W	set system <n> maximum winter temperature set-point
s<n>.setpoint.winter.min	150 .. 290	R	system <n> current minimum winter temperature set-point
		W	set system <n> minimum winter temperature set-point
s<n>.setpoint.summer.max	160 .. 300	R	system <n> current maximum summer temperature set-point
		W	set system <n> maximum summer temperature set-point
s<n>.setpoint.summer.min	150 .. 290	R	system <n> current minimum summer temperature set-point
		W	set system <n> minimum summer temperature set-point
s<n>.temp.out	-400 ... +800	R	outdoor temperature, in 10th of Celsius degrees (e.g. 210 corresponds to 21.0 °C)
s<n>.temp.inlet	-400 ... +800	R	inlet temperature, in 10th of Celsius degrees (e.g. 210 corresponds to 21.0 °C)
z<n>.mode	off	R	zone <n> is off
		W	turn off zone <n>
	maint	R	zone <n> is in maintenance mode
	eco	R	zone <n> is in energy saving mode
		W	set zone <n> to energy saving mode
	pgm	R	zone <n> is in program mode
		W	set zone <n> to program mode

	man	R	zone <n> is in manual mode
		W	set zone <n> to manual mode
	on	W	set zone <n> to the mode set before last off (ECO, PGM or MAN). If HSYCO starts with the zone already off, the on command sets the mode to ECO
z<n>.lock	0	R	zone <n> is not locked
		W	unlock zone <n>
	1	R	zone <n> is locked
		W	lock zone <n>
z<n>.temp	-400 ... 800	R	measured temperature, in 10th of Celsius degrees (e.g. 210 corresponds to 21.0 °C)
z<n>.humidity	1... 100	R	measured relative umidity, from 0 to 100
z<n>.setpoint.temp	150 ... 300	R	zone <n> set-point temperature, in 10th of Celsius degrees (e.g. 210 corresponds to 21.0 °C)
		W	zone <n> set-point temperature, in 10th of Celsius degrees (e.g. 210 corresponds to 21.0 °C)
	up	W	increase zone temperature set-point by 0.5 °C
	down	W	decrease zone temperature set-point by 0.5 °C
alarm.<c>	0	R	alarm code C (1-72) inactive
	1	R	alarm code C (1-72) active
alarm	0	R	no active alarms
	1	R	the system is reporting at least one active alarm

User Interface

UISET Actions

ID	Attribute	Set to	
s<n>.status	value	system <n> (1 or 2) status	

		ON: system is on OFF: system is off MAINT: maintenance mode
s<n>.mode	value	system <n> (1 or 2) mode: SUMMER: summer mode (cooling) WINTER: winter mode (heating) SUMMER*: summer, transitioning to winter mode WINTER*: winter, transitioning to summer mode
s<n>.temp.out	value	the outdoor measured temperature for system <n> (1 or 2), in Celsius degrees with one decimal digit, followed by " °C"
s<n>.temp.inlet	value	the inlet measured temperature for system <n> (1 or 2), in Celsius degrees with one decimal digit, followed by " °C"
z<n>.mode	value	zone <n> operating mode: OFF: zone is off ECO: energy saving mode PGM: program mode; the zone follows a set-point during a defined time period MAN: manual mode; constantly follows the temperature set-point MAINT: zone in maintenance mode
z<n>.lock	value	zone lock mode: LOCKED: zone is locked “”: zone is not locked"
z<n>.temp	value	the current temperature in Celsius degrees with one decimal digit, followed by " °C"
z<n>.humidity	value	the current relative humidity, from 0 to 100, followed by “ %”
z<n>.setpoint.temp	value	the zone temperature set-point in Celsius degrees with one decimal digit, followed by " °C"
alarm.<n>	value	text line <n> (1-30) of the alarms register, with start/end date and time and error code:

		“CODE NN: YYYY-MM-DD HH:MM:SS - YYYY-MM-DD HH:MM:SS”"
alarm.label	value	alarm flag. This object is normally invisible, and made visible when the rayCONTROL reports at least one active alarm

USER Commands

Name	Param	Action
z<n>	mode	cycle through the different operation modes (off, eco, pgm, man)
z<n>	mode.off	set zone <n> off
z<n>	mode.eco	set zone <n> to ECO (energy saving) mode
z<n>	mode.pgm	set zone <n> to PGM (program) mode
z<n>	mode.man	set zone <n> to MAN (manual) mode
z<n>	mode.on	set zone <n> to the mode set before last off (ECO, PGM or MAN). If HSYCO starts with the zone already off, the on command sets the mode to ECO
z<n>	temp.up	increase zone temperature set-point by 0.5 °C
z<n>	temp.down	decrease zone temperature set-point by 0.5 °C
s<n>	power	system <n> (1 or 2) power on/off button
s<n>	mode	system <n> (1 or 2) season button, for winter/summer mode selection

Release Notes

3.3.0

- new data points: s<n>.setpoint.winter.max s<n>.setpoint.winter.min s<n>.setpoint.summer.max s<n>.setpoint.summer.min z<n>.lock
- new readinterval option (defaults to 20 msec.)
- improved communication errors handling

3.0.0

- full integration in HSYCO 3.0.0

RayCONTROL is a registered trademark of Permasteelisa Impianti srl, FCC Planterm Division.

SSH

Secure Shell (SSH) is a cryptographic network protocol for secure data communication, remote command-line login, remote command execution, and other secure network services between two networked computers that connects, via a secure channel over an insecure network, a server and a client (running SSH server and SSH client programs, respectively).

The best-known application of the protocol is for access to shell accounts on Unix-like operating systems, but it can also be used in a similar fashion for accounts on Windows.

It was designed as a replacement for Telnet and other insecure remote shell protocols, which send information, notably passwords, in plaintext, rendering them susceptible to interception and disclosure using packet analysis.

The encryption used by SSH is intended to provide confidentiality and integrity of data over an unsecured network, such as the Internet.

The SSH I/O Server implements the SSH-2 version of the protocol, and supports remote command execution and the SCP protocol for bidirectional file transfer. It uses HSYCO data points to execute commands and return command status, making it easy to use the SSH protocol in EVENTS, as well as JavaScript and Java.

The SSH I/O Server establishes a single SSH connection to the server, using password-based authentication, and creates individual sessions to execute single commands.

💡 **Commands are executed sequentially, not in parallel.**

SSH Remote Server Configuration

The remote system should be configured to accept the SSH connection from the HSYCO server using the username and password defined in the I/O Server's options.

If you need to execute operating system's commands with superuser privileges (with the sudo program), the remote user name defined with the user option in the SSH I/O Server configuration must be allowed to execute "sudo" commands without asking for a password.

For example, if the user is hsyco, you could add the line:

```
hsyco ALL=(ALL:ALL) NOPASSWD: ALL
```

at the end of the /etc/sudoers file (use the visudo command to edit the /etc/sudoers file).

💡 **If the username and password are correct, but the remote server refuses the authentication (you will see a "Password authentication failed." message in the logs and the I/O Server will keep restarting), the most probable cause is that the remore SSH server is configured to refuse the "password" authentication type. To make it work, you should change the remote server's configuration to enable password authentication.**

HSYCO Configuration

Add an SSH I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: IP address of the remote device

- **IP Port**: IP port enabled for the SSH service on the remote device, leave blank for default port 22.

Authentication

- **User**: username to use on the remote device
- **Password**: password for the used username

High Availability

- **Shutdown when inactive**: defaults to false.

Options

ID	Default	Values	Description
stdout	true	true	when a remote command is executed, generate IO events containing the command's standard output and standard error streams
		false	do not generate IO events of the command's output and error streams
persistentconnection	true	true	the I/O Server establishes the SSH connection at start-up and keeps it open across multiple commands. Commands execution is faster, but a small amount of network traffic is present, even when no commands are executed, to keep the connection open
		false	the I/O Server will not establish a connection to the remote server until a command is executed. The connection is closed 30 seconds after the execution of the last command

Datapoints

ID	Value	R/W	Description
connection	online	R	SSH connection established (with successful authentication)
	offline	R	SSH connection closed
exec	<command string>	W	executes the command passed as value
	<exit status>	R	• the command's exit status number

			• "error" if the command could not be sent to the remote server • "null" if the remote server failed to return an exit status code (some SSH servers do not return the exit status)
err	<error stream>	R	the executed command standard error stream, with lines separated by " ", or an empty string if the command's error stream is empty
out	<error stream>	R	the executed command standard output stream, with lines separated by " ", or an empty string if the command's output stream is empty
get	<source> <destination>	W	retrieves the remote source file and writes it locally to the destination path name. The source and destination string in value should be separated by one or more spaces. Always use the "/" character as path separator. Use "\" as the escape character if the file names have spaces. The destination directory must exist and cannot be outside of the HSYCO's base directory, or the command will fail. If the destination name doesn't have a path, the file will be written to the HSYCO's base directory
	<exit status>	R	• the total number of transferred bytes (should be equal to the actual file size) • "error" if the file transfer failed
put	<source> <destination>	W	copy the local source file to the remote server's destination path name. The source and destination string in value should be separated by one or more spaces. Always use the "/" character as path separator. Use "\" as the escape character if the file names have spaces. The source directory must exist and cannot be outside of the HSYCO's base directory, or the command will fail. If the source name doesn't have a path, the file will be written to the HSYCO's

			base directory. The file will be saved to the remote server with 600 (owner's only read and write) permission
	<exit status>	R	• the total number of transferred bytes (should be equal to the actual file size) • "error" if the file transfer failed

💡 **The exec, err, out, get and put data points will trigger the IO events at the end of the command's execution even if the value is not changed. This allows you to easily intercept the exit status and other return information.**

Examples

💡 **In the following examples, we assume that the id of the SSH I/O Server is "ssh".**

Remote Command Execution

Reboot a remote system at midnight.

```
TIME 0000 : IO ssh.exec = "sudo reboot"
```

Turn off screen at midnight.

```
TIME 0000 : IO ssh.exec = "sudo vbetool dpms off"
```

Turn on screen at 7 o' clock.

```
TIME 0700 : IO ssh.exec = "sudo vbetool dpms on"
```

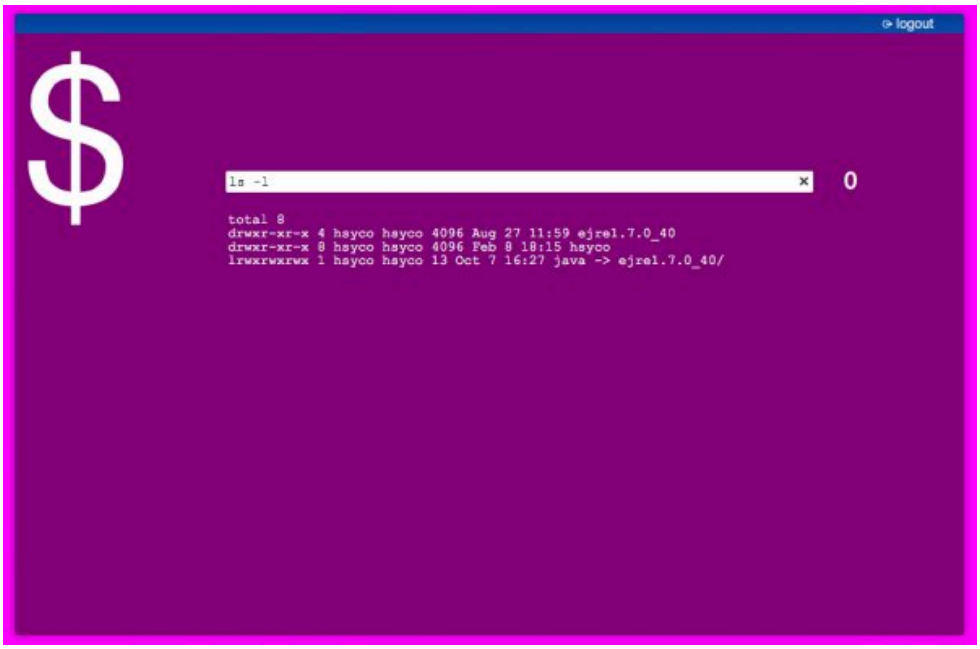
File Transfer

At midnight, copy HSYCO's internal database backup to a remote server.

```
TIME 0000 :     IO ssh.put = "data_backup/hsyco.data /tmp/data_backup/hsyco.data",
                IO ssh.put = "data_backup/hsyco.properties /tmp/data_backup/hsyco.properties",
                IO ssh.put = "data_backup/hsyco.script /tmp/data_backup/hsyco.script"
```

Interactive Remote Command Project

Create a simple interactive command console in HSYCO.



The project's file

```
(#skin blue)
(#language it)
(#size 980x640)
(#style body-background-color=#FF00FF; pages-background-color=#770077)
(#uiset $exec.eraseicon=true)

(header SSH)

(menu)
        (text!stderr r5c3;; (width:600px; text-align:left; height:400px; font-family:Courier New, Courier,
monospace))
        (text!stdout r5c3;; (width:600px; text-align:left; height:400px; overflow:auto; font-family:Courier
New, Courier, monospace))
        (input!$exec r4c3; (width:600px; font-family:Courier New, Courier, monospace); input)
        (text r1c1; $; font-size:200px)
        (text!exitcode r4c9;; font-size:24px)
(endofmenu)
```

Events logic

```
USER "$exec" : IO ssh.exec = $exec, UISET "$exec.focus" = true
IO ssh.out : UISET stdout.text = IO ssh.out
IO ssh.err : UISET stderr.text = IO ssh.err
IO ssh.exec : UISET exitcode.text = IO ssh.exec
```

Release Notes

3.3.0

- initial release

SamsungTV

The Samsung Smart TV driver for HSYCO allows you to control most recent Smart TV models from HSYCO web interfaces and application logic.

The driver can send most of the remote control commands, and a few extra commands not present on the remote, but can't turn on a TV. The driver is basically one-way, it sends commands but doesn't have any information of the current state of the TV, it only knows if the TV is on and can accept commands from HSYCO.

When you initially configure the Smart TV I/O server in HSYCO and the server restarts, you should have your TV on. The TV will then show a message asking to accept connections from HSYCO.

> We can't provide a list of Samsung Smart TV models that work with this driver. In general, most recent models from 2012 on should work. We do not guarantee that this driver will work with any specific model and firmware version.

HSYCO Configuration

Add a SAMSUNGTV I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: IP address of the TV
- **IP Port**: TCP/IP port to use, leave blank to use default port 55000.

High Availability

- **Shutdown when inactive**: defaults to false.

Datapoints

ID	Value	R/W	Description
send	<command_code>	W	send the specified control command (see table below)

Most Samsung Smart TV models should accept the following command codes. Other undocumented commands may also be supported.

Command code	Description
0	0
1	1
2	2
3	3
4	4
5	5

6	6
7	7
8	8
9	9
UP	up
DOWN	down
LEFT	left
RIGHT	right
ENTER	enter
INFO	info
EXIT	exit
RETURN	return
MENU	menu
TOOLS	tools
SOURCE	source
VOLUP	increase volume
VOLDOWN	decrease volume
MUTE	mute
CHUP	next channel
CHDOWN	previous channel
TV	select the internal TV tuner source
HDMI	select one of the HDMI sources (depends on the TV model)
POWEROFF	turn TV off (can't turn it back on)
PIP_ONOFF	picture-in-picture on/off
PIP_CHDOWN	next channel in PIP window
PIP_CHUP	previous channel in PIP window
PIP_SWAP	swaps PIP and main screen sources

User Interface

You can easily associate any user button to a specific command to be sent to the TV. Set *name* to the I/O Server name and *param* to the command code.

You can also enable the Repeat mode, so that multiple commands will be sent to the TV while the button is pressed.

Release Notes

3.3.0

- initial release version

Samsung and Samsung Smart TV are either trademarks or registered trademarks of Samsung Electronics Co., Ltd.

Sip

The HSYCO SIP I/O Server allows HSYCO to control standard VoIP client software (also called soft-phones), in order to integrate telephony features in HSYCO applications.

Using the SIP I/O Server, you will be able to use the audio capabilities of the HSYCO Server to implement a speakerphone with all calling features being controlled by HSYCO itself. This “phone” mode is particularly useful when HSYCO is running on a touch screen equipped with a microphone and speaker, and a kiosk mode Web browser, to create customized door-entry control panels with audio/video capabilities.

The SIP I/O Server also works as a basic IVR (interactive voice response) system, allowing HSYCO to interact with users via the standard telephony system through the use of voice and DTMF tones input on the phone’s keypad.

VoIP Applications

The SIP I/O Server currently works with the Linphone Voice Over IP software phone, on Linux systems only. Linphone must be installed and properly configured on the Linux system that runs the HSYCO Server.

The current version of the SIP I/O Server has been tested with Linphone 3.5.2.

> Linphone is a free software that you can use and distribute, currently licensed under the GNU GPL v2 license. You should verify with the Linphone owner that your intended use of this software is compliant with the license, or obtain a proprietary license of the software. Check www.linphone.org for additional information.

Linphone Installation

From the Linux command-line shell, install the software with the command:

```
sudo apt-get install linphone
```

After the installation, run Linphone for the hsyco user, and configure the audio devices and levels. This configuration will be used when Linphone is run as a daemon process and called by HSYCO.

The hsycolinphone script file must be unzipped, copied in the /usr/local/bin/ folder and set as executable.

Finally, if you are using the the PulseAudio system, ensure that the user that will run Linphone is added to the pulse-access group, while if you are using the ALSA audio system, the user that will run Linphone should be added to the audio group instead.

HSYCO Configuration

The VoIP client can be configured to work as a **stand-alone client**, or **registered to a VoIP PBX system**.

You can define only one HSYCO SIP I/O Server in System Settings.

Add a SIP I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

Stand-alone Client Configuration

Leave all the communication parameters blank.

Registered Client Configuration

- **IP Address**: host name or IP address of the VoIP PBX system
- **IP Port**: TCP/IP port to use, leave blank to use default port 5060
- **User**: username for the PBX account you are using
- **Password**: password for the PBX account you are using.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
user	hsyco	username	the operating system's user name that will be used to run the Linphone application, either locally or on a remote server
host		remote host name or address	the hostname or IP address of the remote server where Linphone is installed
password		password	this option is required when host is defined. It is the password for the remote user
mode	phone	phone	phone mode
		ivr	ivr mode

Remote System Configuration

When Linphone is run on a remote system, HSYCO will use the SSH protocol to execute the remote call. The remote system should be configured to accept the SSH connection from the HSYCO server using the username and password defined in the I/O Server's options.

If you also need to control the display backlight or execute generic operating system's commands, the remote user name defined with the user option in the SIP I/O Server configuration must be allowed to execute "sudo" commands without asking for a password.

For example, if the user is hsyco, you could add the line:

```
hsyco   ALL=(ALL:ALL) NOPASSWD: ALL
```

at the end of the /etc/sudoers file (Use the visudo command to edit the /etc/sudoers file).

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established

	offline	R	HSYCO can't connect to the VoIP client application
registration	0	R	the SIP client is not registered (note that this event is not guaranteed to reflect unregistration conditions caused by the SIP server)
	1	R	the SIP client is registered to a SIP server
mode	phone	R	the I/O Server is in phone mode
	ivr	R	the I/O Server is in IVR mode
call	<SIP URI string>	R	a call is being established or is in progress with the remote party identified by the value of this data-point
		W	start a call to a remote party
	answer	W	answer an incoming call
	hangup	W	hangup a call
	<empty string>	R	no call is currently being established or in progress
call.refused	<SIP URI string>	R	an incoming call is refused because another call is already in progress
status	ring	R	receiving an incoming call, not yet answered
	dial	R	dialing a call, not yet answered
	active	R	incoming or outbound call in progress
	hangup	R	no call is currently in progress
transfer	<SIP URI string>	W	blind transfer an active or incoming call to a remote party
mute	0	R	un-mute command sent successfully
		W	un-mute the microphone
	1	R	mute command sent successfully
		W	mute the microphone (note that when a call is terminated, the mute status is reset)
autoanswer	0	R	autoanswer is disabled

		W	disable autoanswer mode
	1	R	autoanswer is enabled
		W	enable autoanswer mode
volume	off	R	volume off command sent successfully
		W	turn off the speaker
	1 - 20	R	volume command sent successfully
		W	set the speaker volume
dtmf	0 - 9 # *	R	DTMF tone received
		W	send DTMF tones (you can send one or more tones with the same command)
generic	<string>	W	send a generic command to the Linphone application
play.file	<filename>	W	only in IVR mode, play back a file to an active call. The file pathname must be relative to HSYCO's root directory
play.voice.<voice>	<text>	W	only in IVR mode, use the TTS engine to play back a text message, using the voice set with <voice>
backlit	0	R	backlight off command sent successfully
		W	turn off the display's backlight
	1	R	backlight on command sent successfully
		W	turn on the display's backlight
exec	<command>	W	execute a generic operating system's command

> The SIP protocol uses a standard URI format to identify the calling and called parties, like:
>
> ```
> sip:user@host
> ```
>
> where *user* is the username and *host* is the remote host name or IP address.

The SIP I/O Server will automatically remove or imply the "sip:" prefix from data-point values. @host will also be removed or implied if host is an exact match of the I/O Server IP parameter.

When an incoming call is ringing or an outbound call is being dialed, the SIP I/O Server also generates a PBX event and calls the PBXCallEvent() Java callback method. When the phone is ringing the event will be:

```
PBX <remote URI> = <local URI>
```

and the Java callback:

```
PBXCallEvent("<ID>", "<remote URI>", "<local URI>")
```

When an outbound call is established, the event will be:

```
PBX <local URI>  = <remote URI>
```

and the Java callback:

```
PBXCallEvent("<ID>", "<local URI>", "<remote URI>")
```

💡 **Note that, if the SIP I/O Server is configured as a stand-alone client, the <local URI> is replaced with the I/O Server's ID.**

Release Notes

3.3.0

- improved performance of the SSH connection to remote servers
- support of password-based authentication

3.2.1

- text-to-speech performance improvements

3.2.0

- initial release

System

The HSYCO System Monitor I/O Server is used to monitor several key hardware and operating system parameters of an HSYCO server:

- Operating system boot time
- HSYCO start time
- Disk space
- Java memory (used, free and maximum available memory)
- Performance (CPU percent busy time, 1 minute load average, 5 minutes load average)
- Console error messages
- Internet connection status
- Cameras status
- I/O Servers status
- Clock drift of remote HSYCO servers monitored through the HSYCOREMOTE I/O server.

HSYCO Configuration

You can have only one HSYCO System Monitor I/O Server.

Add a SYSTEM I/O Server in the I/O Servers section of the Settings and set its parameters:

High Availability

- **Shutdown when inactive**: defaults to false.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support
pollinterval	30	n ≥ 10	the data acquisition interval, in seconds

Datapoints

ID	Value	R/W	Description
time	<val>	R	the system's timestamp, in milliseconds
start.elapsed	<val>	R	number of seconds since HSYCO start-up time
start.time	<date>	R	HSYCO start-up time, formatted as "YYYY-MM- DD hh:mm:ss",

			for example: "2012-09-29 22:36:43"
boot.elapsed	<val>	R	number of seconds since operating system boot time
boot.time	<date>	R	operating system boot time, formatted as "YYYY-MM-DD hh:mm:ss", for example: "2012-09-29 22:36:43"
cpu.usage	<val>	R	CPU percent busy time
oad.avg.1	<val>	R	the system's load average during the last 60 seconds
oad.avg.5	<val>	R	the system's load average during the last 5 minutes
oad.avg.15	<val>	R	the system's load average during the last 15 minutes
disk.free.percent	<val>	R	the free storage space as a percentage of total available space
disk.free	<val>	R	the free storage space in Mbytes
disk.used	<val>	R	the used storage space in Mbytes
memory.free	<val>	R	the amount of free run-time memory of the Java virtual machine, in Mbyte
memory.max	<val>	R	the maximum amount of run-time memory that could be used by the Java virtual machine, in Mbyte
memory.used	<val>	R	the amount of run-time memory used by the Java virtual machine, in Mbyte
camera.total	<val>	R	the total number of cameras defined
camera.errors	<val>	R	the number of cameras that HSYCO is unable to access
camera.errors.list	<list>	R	a comma separated list of the cameras having errors, or an empty string if there are no errors
io.total	<val>	R	the total number of I/O servers defined
io.errors	<val>	R	the number of I/O servers that HSYCO is unable to access

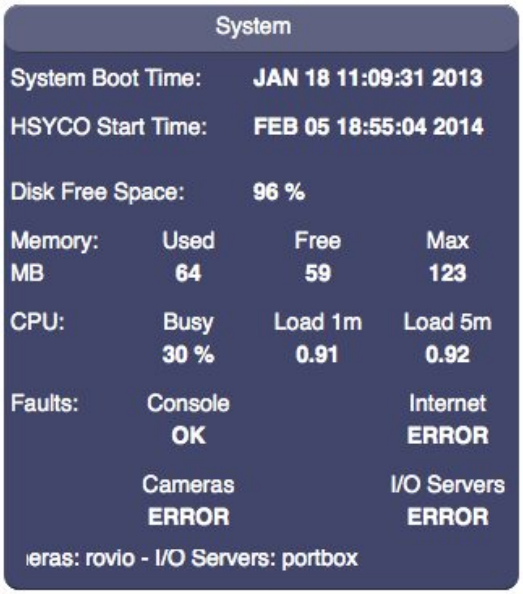
io.errors.list	<list>	R	a comma separated list of the I/O servers having errors, or an empty string if there are no errors
internet.connection	online	R	Internet access from HSYCO is working properly
	offline		HSYCO is unable to access the Internet
console.errors	0	R	the console.log file contains no new errors
	1		the console.log file contains new errors

User Interface

Hsycomonitor Object

A panel that monitors the system. These parameters are displayed:

- system boot time
- HSYCO start time
- disk free space
- memory in MBs: used, free, max
- CPU usage in %: busy, load 1m, load 5m
- faults: console, internet, cameras, I/O servers
- error messages



Parameters

- **server ID**: the ID of the server as specified in hsyco.ini
- **label**: the text label on the panel. Can contain HTML tags
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(hsycomonitor <server ID>; <label>; <position>)
```

E.g.

```
(hsycomonitor system; System; x10y20)
```

UISET Actions

ID	Attribute	Set to	
boot.time	value	operating system boot time	
start.time	value	HSYCO start-up time	
cpu.usage	value	CPU percent busy time	
load.avg.1	value	the system's load average during the last 60 seconds	
load.avg.5	value	the system's load average during the last 5 minutes	
disk.free.percent	value	the free storage space as a percentage of total available space	
	blink	true	when the free storage space is below 10%
		false	when the free storage space is equal or above 10%
memory.free	value	the amount of free run-time memory of the Java virtual machine, in Mbyte	
memory.max	value	the maximum amount of run-time memory that could be used by the Java virtual machine, in Mbyte	
memory.used	value	the amount of run-time memory used by the Java virtual machine, in Mbyte	
camera.errors	value	ERROR	when HSYCO cannot access one or more cameras
	blink	true	
	value	OK	when all cameras are connected
	blink	false	
io.errors	value	ERROR	when HSYCO cannot access one or more I/O servers
	blink	true	
	value	OK	when all I/O servers are connected
	blink	false	
internet.connection	value	ERROR	when HSYCO is unable to access the Internet
	blink	true	

	value	OK	when Internet access from HSYCO is working properly
	blink	false	
console.errors	value	ERROR	when the console.log file contains new error messages
	blink	true	
	value	OK	when the console.log file contains no new errors
	blink	false	
messages	value	the list of cameras and I/O servers with errors	

Release Notes

3.2.1

- the (hsycomonitor) object displayed an error on cameras if there were no cameras defined

3.2.0

- initial release

Tecnoalarm

The Tecnoalarm TP16-256, TP8-64BUS, TP8-96VIDEO, TP8-96GSM, TP16-512GSM and TP8-88 are advanced, multi-area security systems integrated with a variety of wired and wireless sensors.

The TECNO OUT driver in HSYCO supports the integration of more than one TP unit within the same HSYCO interface.

The integration with HSYCO can be accomplished with a TCP/IP network connection.

HSYCO Configuration

Add a TECNOALARM I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: the Tecnoalarm panel address
- **IP Port**: TCP/IP port enabled on the TECNO OUT.

High Availability

- **Shutdown when inactive**: defaults to true.

💡 **The code and model parameters in Options are actually mandatory. The I/O Server cannot start without these parameters.**

Options

ID	Default	Values	Description
code		code	the user code that the driver uses when sending commands to the Tecnoalarm system. It must be a valid user code, or the MASTER code when using the TP8-64BUS
model		string	the system's model. Use one of the following values: TP16-256, TP8-64BUS, TP8-96VIDEO, TP8-96GSM, TP16-512GSM, TP8-88
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
gui	true	true	enable support of the tecnoalarm Web interface object
		false	disable support of the tecnoalarm Web interface object

userlogs	false	true	generates logs that can be viewed with the userlist object, and can be selected to generate user events
		false	logs for userlist objects disabled. Normal text logs are generated even when this option is false
logsize	50	n ≥ 0	the number of log events shown in the tecnoalarm Web object. Set to 0 to hide the log in the tecnoalarm object
pass		string	the passphrase used to encrypt the TCP/IP traffic. If the Tecnoalarm system doesn't support the passphrase, you should omit this parameter
polltime	1000	n ≥ 0	the data polling interval, in milliseconds. Shorter time improve the response of the system, but increase the load on the CPU of both HSYCO and the Tecnoalarm system

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	connection failed
system.standby	1	R	general system stand-by status
	0	R	reset
system.fault	1	R	general system fault (low battery or no AC power)
	0	R	reset
system.lowbattery	1	R	general low battery fault
	0	R	reset
system.nopower	1	R	general no AC power fault
	0	R	reset
system.tamper	1	R	general tamper condition (central unit, zones, modules, modules loss of connection)
	0	R	reset

system.radiofault	1	R	general radio modules fault
	0	R	reset
system.duress	1	R	duress status
	0	R	reset
system.techzone	1	R	general technical zones open status
	0	R	reset
system.normal	1	R	normal status (AC power ok, battery ok, all zones closed, no tamper)
	0	R	reset
system.maintenance	1	R	maintenance mode
	0	R	reset
system.panic	1	R	panic alarm
	0	R	reset
system.indoorsiren	1	R	general status of indoor sirens
	0	R	reset
system.outdoorsiren	1	R	general status of outdoor sirens
	0	R	reset
zone.<n>.excluded	1	R	zone <n> disabled
	0	R	zone <n> enabled
zone.<n>.alarm	1	R	zone <n> alarm
	0	R	reset
zone.<n>.cablecutstatus	1	R	zone <n> cable cut
	0	R	reset
zone.<n>.cablecutalarm	1	R	zone <n> cable cut alarm
	0	R	reset
zone.<n>.lowbattery	1	R	zone <n> low battery
	0	R	reset
zone.<n>.noradio	1	R	zone <n> radio supervision lost
	0	R	reset

zone.<n>.active	1	R	zone <n> active
	0	R	reset
zone.<n>.radiolearned	1	R	zone <n> radio learned
	0	R	zone <n> radio not learned
zone.<n>	enable	W	sends a zone <n> enable command using default code
	disable	W	sends a zone <n> disable command using default code
	enable.C	W	sends a zone <n> enable command using code C
	disable.C	W	sends a zone <n> disable command using code C
program.<n>.status	off	R	program <n> reset
	preout	R	program <n> pre out status
	out	R	program <n> out status
	armed	R	program <n> armed
	outpartial	R	program <n> out partial status
	armedpartial	R	program <n> armed partial
	endpartial	R	program <n> end of partial
program.<n>.prealarm	1	R	program <n> pre-alarm
	0	R	reset
program.<n>.alarm	1	R	program <n> alarm
	0	R	reset
program.<n>.alarmmemory	1	R	program <n> alarm memory
	0	R	reset
program.<n>	arm	W	sends a program <n> arm command using default code
	disarm	W	sends a program <n> disarm command using default code
	partial	W	sends a program <n> partial arm command using default code
	full	W	sends a program <n> full arm command using default code

	arm.<C>	W	sends a program <n> arm command using code <C>
	disarm.<C>	W	sends a program <n> disarm command using code <C>
	partial.<C>	W	sends a program <n> partial arm command using code <C>
	full.<C>	W	sends a program <n> full arm command using code <C>
output.<n>.status	1	R	output line <n> on
	0	R	output line <n> off
output.<n>	1	W	sends an output <n> on command using default code
	on		
	0	W	sends an output <n> off command using default code
	off		
	1.<C>	W	sends an output <n> on command using code <C>
	on.<C>		
	0.<C>	W	sends an output <n> off command using code <C>
	off.<C>		

User Interface

Tecnoalarm Object

The user interface for the Tecnoalarm multi-area intrusion detection system:

File:UI Object tecnoalarm.png

The Tecnoalarm object is listed in the Project Editor's new object list only when at least one Tecnoalarm I/O Server is defined.

Parameters

- **server id**: the server ID
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(tecnoalarm <server id>; <position>)
```

E.g.

```
(tecnoalarm serverid; x10y20)
```

UISET Actions

ID	Attribute	Set to
connection.label.online	visible	true when the panel is connected to HSYCO, false otherwise
connection.label.offline	visible	true when the panel is not connected, false otherwise
program.<n>.status.label.off	visible	true when program <n> is off
program.<n>.status.label.preout	visible	true when program <n> is in pre out status, false otherwise
program.<n>.status.label.out	visible	true when program <n> is in out status, false otherwise
program.<n>.status.label.armed	visible	true when program <n> is armed, false otherwise
program.<n>.status.label.outpartial	visible	true when program <n> is in out partial status, false otherwise
program.<n>.status.label.armedpartial	visible	true when program <n> is partial armed, false otherwise
program.<n>.alarm.label	visible	true when program <n> alarm is active, false otherwise
zone.<n>.excluded.label	visible	true when zone <n> excluded, false otherwise
zone.<n>.alarm.label	visible	true when zone <n> alarm is active, false otherwise
zone.<n>.cablecut.label	visible	true when zone <n> cable cut detected (status or alarm), false otherwise
zone.<n>.lowbattery.label	visible	true when zone <n> has low battery, false otherwise
zone.<n>.noradio.label	visible	true when zone <n> radio supervision lost, false otherwise
zone.<n>.active.label	visible	true when zone <n> is active, false otherwise
lock.label	visible	true when the system is armed, false otherwise
alarm.label	visible	true when the alarm is active, false otherwise
tamper.label	visible	true when a tamper alarm is active, false otherwise

Log Files

A permanent record of all the events shown in the log page is saved in a file called security.log in the logs/YYYY directory for the current year.

This file is never overwritten or deleted by HSYCO.

```
2014.07.02 17:56:19.188 - tecno - 02/07/10 17:57:29 Activate Remote 6 Monitor
2014.07.02 17:56:22.211 - tecno - 02/07/10 17:57:32 Deactivate  Remote 6 Monitor
2014.07.02 17:56:48.174 - tecno - 02/07/10 17:57:58 Access Remote Master Keyboard
2014.07.02 19:00:06.825 - tecno - 02/07/10 19:01:16 Access Remote Master Keyboardd
2014.07.07 17:07:41.871 - tecno - 07/07/10 17:09:31 Activate Remote 6 Monitor
2014.07.07 17:07:43.669 - tecno - 07/07/10 17:09:32 Deactivate Remote 6 Monitor
```

Other information related to the Tecnoalarm Tecno Out driver are saved in the daily message.log files.

Release Notes

3.3.0

- added support for TP8-88
- Tecnoalarm Tecno Out driver converted to I/O Server
- Java API write command methods are now obsolete and deprecated, replaced by IO write commands
- new (tecnoalarm) object
- new option persistentconnection = <true|false>. If true (default is false) the I/O Server will not periodically disconnect and reconnect (periodic disconnection is used to avoid performance degradation on some firmware versions)
- new option userlogs. Used to create (userlist) clickable logs

3.2.1

- Tecno Out: workaround to address a performance issue on Tecno Out introduced with firmware 1.2.07

2.0.0

- full integration within HSYCO 2.9.0

1.1.4

- fixed a bug affecting the TCP/IP communication when using a passphrase

Tridonic

Tridonic is a lighting system based on the DALI bus technology. HSYCO integrates this system and communicates with it through the DALI SCI2 interface module and its communication protocol. The integration relies on a direct connection between the DALI SCI2 interface and HSYCO server's RS-232 serial port, or through a TCP/IP to serial port gateway.

Communication

The DALI SCI2 interface module communicates with HSYCO through a RS-232 interface.

RS-232 parameters:

Baud rate	38400 bps
Data bits	8
Stop bit	1
Parity	none

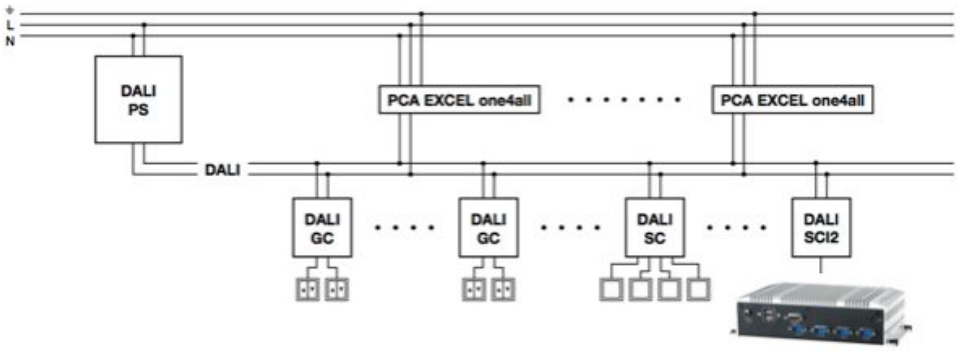
Further, the RS-232 signals RTS and DTR must be set to the following levels:

- RTS = +6 ... +12 V
- DTR = -6 ... -12 V.

Refer to the SCI2 manual for installation, wiring and power supply requirements.

Tridonic Configuration

An illustrative scenario for a DALI architecture looks as follows:



On one BUS, there can be up to 64 dimmers, addressed from 0 to 63, organized in up to 16 groups (0 - 15) or individually addressable. The driver also includes the support for the MSensor 2.0 motion detector and photometer functionalities. The MSensor 2.0 can be configured to work in two different modes:

- Direct-master mode (default mode): The MSensor has a DALI address and sends motion detection and light regulation commands to the group specified by its rotary switch (if the switch is set on the value X the controlled group will be X - 1, if the switch is set on "0" the commands are broadcasted). The light intensity value measured by the light sensor is only read upon request (see the datapoints table)

- Indirect-master mode: The MSensor doesn't have a DALI address and does not control any dimmer directly. The motion detector and the light sensor have two different eDALI (extended DALI) addresses from which they send motion and

light intensity asynchronous events.

HSYCO Configuration

Add a TRIDONIC I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: the comm port the device is connected to.

High Availability

- **Shutdown when inactive**: defaults to false.

Options

ID	Default	Values	Description
discovery	true	true	auto-detects all DALI devices and groups, and automatically lists them in the systemtopo.txt file
		false	auto-detect for devices is disabled
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system
broadcastrange	1-254	<min>-<max>	specifies the minimum <min> and maximum<max> allowed light levels for the broadcast command
groupsrange	g<n>:1-254	g<n>:<min>-<max>	specifies the minimum <min> and maximum<max> allowed light levels for the group <n>. If more than one group range is to be specified, then list the ranges separated by ";", e.g. groupsrange=g0:20-254;g1:40-250

luxdelta	0	≥ 0	option for MSensors. Specifies the minimum difference between light intensity readings required to generate a new event. I.e. when a “xxx.lux” event is generated, a new event (from the same sensor) will be generated only if the new value is greater than the old value plus the luxdelta value or smaller than the old value minus the luxdelta value

The Device Configuration Database

The systemtopo.txt file contains the list of all dimmers, groups and the broadcast address that could be directly associated to graphic objects in the Web-based user interface. This file can be filled manually or automatically by HSYCO at start-up. To enable automatic discovery and automatic generation of devices’ information in the systemtopo.txt file, use the discovery option in the hsyco.ini configuration file. This is an example of an automatically generated systemtopo.txt file:

```
(devices)
tr.0 : LIGHT ; DIMMER ;
tr.1 : LIGHT ; DIMMER ;
tr.2 : LIGHT ; DIMMER ;
tr.g1 : LIGHT ; DIMMER ; DALI group [1,2]
tr.g4 : LIGHT ; DIMMER ; DALI group [0,1]
tr.broadcast : LIGHT ; DIMMER ; DALI broadcast
```

You should then manually add comments and other optional parameters:

```
(devices)
tr.0 : LIGHT ; DIMMER ; door
tr.1 : LIGHT ; DIMMER ; corridor 1
tr.2 : LIGHT ; DIMMER ; corridor 2
tr.g1 : LIGHT ; DIMMER ; DALI group [1,2]; corridor group
tr.g4 : LIGHT ; DIMMER ; DALI group [0,1]; entrance group
tr.broadcast : LIGHT ; DIMMER ; DALI broadcast
```

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
<n> g<m> broadcast	on	W	set the light level of the target to the last level set before the off command
	off	R	the target is off
		W	switch the target off

	1...100%	R	the light level of the target corresponds to the reported value
		W	set the light level of the target to the specified value
	X/Y	W	set the light level of the target to the specified fraction (e.g. a value of 1/5 corresponds to 20%)
	min	W	set the light level of the target to the minimum level (1%) without fading
	max	W	set the light level of the target to the maximum level (100%) without fading
	up	W	set the light level of the target one step higher
	down	W	set the light level of the target one step lower
<n>.lux[Note 1]	read	W	request the reading of the light intensity measured by the targeted MSensor
	<lux>	R	the light intensity measured by the targeted MSensor corresponds to the reported value (in lux). The reading must be requested via the “read” function
broadcast.lux[Note 1]	read	W	request the reading of the light intensity on all the MSensors
e<x>.lux[Note 2] where <x> is the eDALI address of the light sensor of the MSensor	<lux>	R	the light intensity measured by the targeted MSensor corresponds to the reported value (in lux)
motion	g<m>[Note 1]	R	a motion event has been reported to group <m>
	broadcast[Note 1]	R	a motion event has been broadcasted
	e<x>[Note 2]	R	a motion event has been generated by the MSensor’s light sensor with eDALI address <x>

<n>.scene g<m>.scene broadcast.scene	0...15	W	set the light level of the target to the value stored for the specified scene number (from 0 to 15)
command	<xxyy>	R	the 2-byte hexadecimal DALI frame[Note 3] has been sent to the system
		W	sends the 2-byte hexadecimal DALI frame[Note 3] to the system
monitor	none	W	none of the DALI frames detected on the bus will be reported
	ignored	W	only the DALI frames detected on the bus not processed by the driver will be reported
	all	W	all the DALI frames detected on the bus will be reported
	<xxyy>	R	the reported 2-byte hexadecimal DALI frame[Note 3] has been detected on the bus
refresh	0	W	the secondary properties of the devices (minimum and maximum levels, fade time and fade rate) won't be rechecked on every monitor loop iteration
	1	W	the secondary properties of the devices (minimum and maximum levels, fade time and fade rate) will be rechecked on every monitor loop iteration
<n>.error g<m>.error broadcast.error	0	R	the target works properly
	1	R	the target is in an error state

Note 1
: Only available if the MSensor is configured to work in Direct-master mode (default mode).

Note 2
: Only available if the MSensor is configured to work in Indirect-master mode.

Note 3
: The 2-byte DALI frames consist of an 8-bit address part and an 8-bit command part.

The address byte is formatted as follows:

Type of Address	Byte Description
Individual address	0AAAAAAS (AAAAAA = 0 .. 63, S = 0/1)

Group Address	100AAAAS (AAAA = 0 .. 15, S = 0/1)
Broadcast Address	1111111S (S = 0/1)

The command byte is to be interpreted as an arc power level (light intensity) if bit S is 0. The arc power can range between 00 (off) and and FE (maximum level) expressed in hexadecimal. A 1 in the position S means that the command byte is to be interpreted as a DALI command. Refer to the following table for a list of all available commands:

Command	Description
00	Extinguish the lamp without fading
01	Dim up 200 ms using the selected fade rate
02	Dim down 200 ms using the selected fade rate
03	Set the actual arc power level one step higher without fading
04	Set the actual arc power level one step lower without fading
05	Set the actual arc power level to the maximum value
06	Set the actual arc power level to the minimum value
07	Set the actual arc power level one step lower without fading
08	Set the actual arc power level one step higher without fading
10 + scene	Set the light level to the value stored for the selected scene
20	Reset the parameters to default settings
21	Store the current light level in the DTR
2A	Store the value in the DTR as the maximum level
2B	Store the value in the DTR as the minimum level
2C	Store the value in the DTR as the system failure level
2D	Store the value in the DTR as the power on level
2E	Store the value in the DTR as the fade time
2F	Store the value in the DTR as the fade rate
40 + scene	Store the value in the DTR as the selected scene
50 + scene	Remove the selected scene from the slave unit
60 + group	Add the slave unit to the selected group
70 + group	Remove the slave unit from the selected group
80	Store the value in the DTR as a short address
90	Returns the status of the slave as XX

91	Check if the slave is working
92	Check if there is a lamp failure
93	Check if the lamp is operating
94	Check if the slave has received a level out of limit
95	Check if the slave is in reset state
96	Check if the slave is missing a short address
97	Returns the version number as XX
98	Returns the content of the DTR as XX
99	Returns the device type as XX
9A	Returns the physical minimum level as XX
9B	Check if the slave is in power failure mode
A0	Returns the current light level as XX
A1	Returns the maximum allowed light level as XX
A2	Returns the minimum allowed light level as XX
A3	Return the power up level as XX
A4	Returns the system failure level as XX
A5	Returns the fade time as X and the fade rate as Y
B0 + scene	Returns the light level XX for the selected scene
C0	Returns a bit pattern XX indicating which group (0-7) the slave belongs to
C1	Returns a bit pattern XX indicating which group (8-15) the slave belongs to
C2	Returns the high bits of the random address as HH
C3	Return the middle bit of the random address as MM
C4	Returns the lower bits of the random address as LL

Other than the above described commands, it is possible to send the following 2-byte DALI frames:

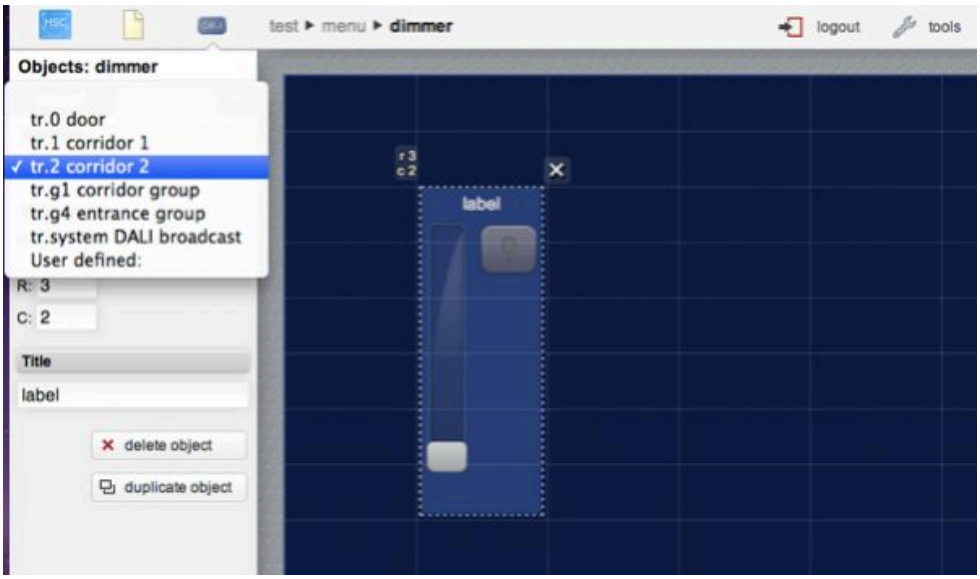
Frame	Description
A100	All special mode processes shall be terminated
A3XX	Store value XX in the DTR
A5XX	Initialize addressing commands for slaves with address XX
A700	Generate a new random address

A900	Compare the random address with the search address
AB00	Withdraw from the compare process
B1HH	Store value HH as the high bits of the search address
B3MM	Store value MM as the middle bits of the search address
B5LL	Store value LL as the lower bits of the search address
B7XX	Program the selected slave with short address XX
B9XX	Check if the selected slave has short address XX
BB00	The selected slave returns its short address XX
BD00	Go into physical selection mode

The value of the events ID.command and ID.monitor is expressed as a 4-digits string “XXYY” representing the hexadecimal value (upper case letters) of the two bytes. For instance, to send the command 00 to the group 1 of our system “tr”, the following syntax will apply: tr.command = 8300

User Interface

All the devices that have been defined in the systemtopo.txt database are automatically listed in the Web Editor. Adding a slider to control a dimmer requires just a few clicks and no additional EVENTS logic.



Release Notes

3.3.0

- added support for Tridonic MSensor 2.0 devices

3.0.0

- initial release

Tridonic is a registered trademark of Tridonic.

WXonline

This driver provides weather information and forecasts retrieved from the Yahoo! Weather service.

> The information provided by this driver must be used in compliance with Yahoo!'s Terms of Use.

HSYCO Configuration

Add an WXONLINE I/O Server in the I/O Servers section of the Settings and set its parameters:

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object and UISET actions
		false	disable UI support
pollinterval	10	n > 0	the data acquisition interval, in minutes
location		<woeid>	WOEID of the target location

The 'location' option is required. To obtain the WOEID of the desired location, go to the Yahoo! Weather home page and enter the location name in the search field. The WOEID will be at the end of the URL for the forecast page for that location. For example, if you search for Los Angeles, the forecast page for that city is "http://weather.yahoo.com/united-states/california/los-angeles-2442047". The WOEID is 2442047.

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
location	<location>	R	the target location is set to <location> (e.g. “New York, NY, United States”)
	<woeid>	W	set the location to the specified WOEID
	refresh	W	refresh the weather data for the current location

condition	<condition>	R	the current weather condition (refer to the conditions table for a complete list of possible values)
condition.clear	1	R	the current condition is categorized as “clear”
	0	R	the current condition is not categorized as “clear”
condition.precipitation	1	R	the current condition is categorized as “precipitation”
	0	R	the current condition is not categorized as “precipitation”
condition.storm	1	R	the current condition is categorized as “storm”
	0	R	the current condition is not categorized as “storm”
condition.partly_cloudy	1	R	the current condition is categorized as “partly cloudy”
	0	R	the current condition is not categorized as "partly cloudy”
condition.low_visibility	1	R	the current condition is categorized as “low visibility”
	0	R	the current condition is not categorized as “low visibility”
temp.c	<val>	R	the reported current temperature corresponds to <val> (expressed in °C)
temp.f	<val>	R	the reported current temperature corresponds to <val> (expressed in °F)
humidity	<val>	R	the reported current humidity percentage corresponds to <val>
wind.speed	<val>	R	the reported current wind speed corresponds to <val>
wind.dir	<val>	R	the reported current wind direction corresponds to <val> (expressed in degrees)
forecast.day.<n> (<n> = 1 or 2)	<day>	R	the week day identified by <n> corresponds to <day> (i.e. “mon”, "tue", "wed", ...)

forecast.temp.low.c.<n> (<n> = 1 or 2)	<val>	R	the forecasted lowest temperature for day <n> is <val> (expressed in °C)
forecast.temp.low.f.<n> (<n> = 1 or 2)	<val>	R	the forecasted lowest temperature for day <n> is <val> (expressed in °F)
forecast.temp.high.c.<n> (<n> = 1 or 2)	<val>	R	the forecasted highest temperature for day <n> is <val> (expressed in °C)
forecast.temp.high.f.<n> (<n> = 1 or 2)	<val>	R	the forecasted highest temperature for day <n> is <val> (expressed in °F)
forecast.condition.<n> (<n> = 1 or 2)	<condition>	R	the forecasted weather condition for day <n> corresponds to <condition> (refer to the conditions table for a complete list of possible values)
forecast.condition.clear.<n>	1	R	the forecasted condition for day <n> is categorized as "clear"
	0	R	the forecasted condition for day <n> is not categorized as "clear"
forecast.condition.precipitation.<n>	1	R	the forecasted condition for day <n> is categorized as "precipitation"
	0	R	the forecasted condition for day <n> is not categorized as "precipitation"
forecast.condition.storm.<n>	1	R	the forecasted condition for day <n> is categorized as "storm"
	0	R	the forecasted condition for day <n> is not categorized as "storm"
forecast.condition.partly_cloudy.<n>	1	R	the forecasted condition for day <n> is categorized as "partly cloudy"
	0	R	the forecasted condition for day <n> is not categorized as "partly cloudy"
forecast.condition.low_visibility.<n>	1	R	the forecasted condition for day <n> is categorized as "low visibility"
	0	R	the forecasted condition for day <n> is not categorized as "low visibility"

Conditions and categories

The following table shows the possible weather condition values and their categorization.

Categories are used to summarize relevant aspects of a specific condition so to simplify programming logic. The available categories are: clear (c), precipitation (p), storm (s), partly cloudy (pc), and low visibility (lv). For some conditions (e.g. “windy”) the categorization is not feasible; in such cases (denoted by "xx") the category datapoints will not be updated so to maintain the previous condition.

Condition	Categories
blowing_snow	p, lv
blustery	xx
clear_night	c
cloudy	
cold	xx
drizzle	p
dust	lv
fair_day	pc
fair_night	pc
foggy	lv
freezing_drizzle	p
freezing_rain	p
hail	p
haze	lv
heavy_snow	p,lv
hot	c
hurricane	p, s, lv
isolated_thundershowers	p, s, lv
isolated_thunderstorms	p, s, lv
light_snow_showers	p, lv
mixed_rain_hail	p
mixed_rain_sleet	p
mixed_rain_snow	p
mixed_snow_sleet	p

mostly_cloudy_day	
mostly_cloudy_night	
partly_cloudy	pc
partly_cloudy_day	pc
partly_cloudy_night	pc
scattered_showers	p, lv
scattered_snow_showers	p, lv
scattered_thunderstorms	p, s, lv
severe_thunderstorms	p, s, lv
showers	p, lv
sleet	p
smoky	lv
snow	p
snow_flurries	p
snow_showers	p, lv
sunny	c
thundershowers	p, s, lv
thunderstorms	p, s, lv
tornado	s
tropical_storm	p, s, lv
windy	xx
unknown	xx

User Interface

Weather Object

Shows a weather panel with current conditions and forecast.

Parameters

- **id**: the object's ID, used by UISets. Required
- **position**: the object's position. Use the pixels or rows and columns coordinates format

Syntax

```
(weather!<id> <position>)
```

E.g.

```
(weather!wx x10y20)
```

UISET Actions

ID	Attribute	Set to	
connection.label	visible	true	when datapoint connection = offline
		false	when datapoint connection = online
location	value	the target location (e.g. “New York, NY, United States”)	
temp.c	value	the reported current temperature followed by "°C"	
temp.c.nounit	value	the reported current temperature without unit (expressed in °C)	
temp.f	value	the reported current temperature followed by "°F"	
temp.f.nounit	value	the reported current temperature without unit (expressed in °F)	
humidity	value	the reported current humidity percentage	
wind.speed	value	the reported current wind speed in Km/h	
wind.speed	value	the reported current wind direction in degrees	
forecast.day.<n> (<n> = 1 or 2)	value	the week day identified by <n> (i.e. “Mon”, “Tue”, “Wed”, ...)	
forecast.temp.low.c.<n> (<n> = 1 or 2)	value	the forecasted lowest temperature for day <n> followed by "°C"	
forecast.temp.low.c.<n>.nounit (<n> = 1 or 2)	value	the forecasted lowest temperature for day <n> without unit (expressed in °C)	
forecast.temp.low.f.<n> (<n> = 1 or 2)	value	the forecasted lowest temperature for day <n> followed by "°F"	
forecast.temp.low.f.<n>.nounit (<n> = 1 or 2)	value	the forecasted lowest temperature for day <n> without unit (expressed in °F)	

forecast.temp.high.c.<n> (<n> = 1 or 2)	value	the forecasted highest temperature for day <n> followed by "°C"
forecast.temp.high.c.<n>.nounit (<n> = 1 or 2)	value	the forecasted highest temperature for day <n> without unit (expressed in °C)
forecast.temp.high.f.<n> (<n> = 1 or 2)	value	the forecasted highest temperature for day <n> followed by "°F"
forecast.temp.high.f.<n>.nounit (<n> = 1 or 2)	value	the forecasted highest temperature for day <n> without unit (expressed in °F)
forecast.condition.<n> (<n> = 1 or 2)	value	the forecasted weather condition for day <n> (e.g. “SUNNY”)
condition.label	img	weather/<current_condition_image>
forecast.condition.<n>.label (<n> = 1 or 2)	img	weather/<forecasted_condition_image>

Weather conditions images

The following table shows the mapping between each condition and the corresponding image.

To customize the images add a folder named "weather" inside the "img" folder of your project and add your images named accordingly to the table.

Condition	Image name	Default image
blowing_snow	snow.png	
blustery	windy.png	
clear_night	sunny.png	
cloudy	cloudy.png	
cold	icy.png	
drizzle	drizzle.png	
dust	dust.png	
fair_day	partly_cloudy.png	
fair_night	partly_cloudy.png	
foggy	fog.png	
freezing_drizzle	drizzle.png	
freezing_rain	rain.png	

hail	hail.png	
haze	haze.png	
heavy_snow	snow.png	
hot	sunny.png	
hurricane	storm.png	
isolated_thundershowers	chance_of_storm.png	
isolated_thunderstorms	chance_of_storm.png	
light_snow_showers	snow.png	
mixed_rain_hail	hail.png	
mixed_rain_sleet	snow.png	
mixed_rain_snow	snow.png	
mixed_snow_sleet	snow.png	
mostly_cloudy_day	cloudy.png	
mostly_cloudy_night	cloudy.png	
partly_cloudy	partly_cloudy.png	
partly_cloudy_day	partly_cloudy.png	
partly_cloudy_night	partly_cloudy.png	
scattered_showers	chance_of_rain.png	
scattered_snow_showers	chance_of_snow.png	
scattered_thunderstorms	chance_of_storm.png	
severe_thunderstorms	storm.png	
showers	rain.png	
sleet	snow.png	
smoky	dust.png	
snow	snow.png	
snow_flurries	snow.png	
snow_showers	snow.png	
sunny	sunny.png	

thundershowers	storm.png	
thunderstorms	storm.png	
tornado	tornado.png	
tropical_storm	storm.png	
windy	windy.png	
unknown	unknown_weather_condition.png	

Release Notes

3.1.0

- changed weather provider
- modified I/O Datapoints
- modified I/O Server options

3.0.1

- added weather object

3.0.0

- initial release

Yahoo! and Yahoo! Weather are registered trademarks of Yahoo! Inc.

Wyrestorm

The Wyrestorm MX-0808-PP, MX-0606-PP and MX-0404-PP modules are matrix switchers that allow any of their HDMI inputs to be routed over distance to any combination of their display outputs via a single UTP cable. The integration with HSYCO can be accomplished via direct control through the RS-232 port on the module.

Communication

Use a DE-9 (often called DB-9) female-male RS-232 straight cable to connect the RS-232 port on the Wyrestorm module to the serial port on HSYCO server.

RS-232 parameters:

Baud rate	9600
Data bits	8
Stop bit	1
Parity	none
Flow control	none

HSYCO Configuration

Add a Wyrestorm I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **Comm ID**: select the comm port connected to the device.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for the UI object, UISET actions and USER commands
		false	disable UI support
startupevents	false	true	generate IO events also during the driver's start-up phase
		false	start generating events only after HSYCO is aligned with the current status of the system

outputs	4	4 | 6 | 8	number of outputs on the module

wyrestorm.ini

The wyrestorm.ini file is an optional configuration file located in the root directory.

Here you can define the names to assign to the inputs so that they will be automatically added in the objects of the UI interface. Add a new line for each name using the format:

```
<ioserver_id>.in.<n> = <name>
```

where `<ioserver_id>` is the ID of the referred I/O server and `<n>` is the input number.

For instance:

```
wyr.in.1 = TV
wyr.in.2 = DVD
wyr.in.5 = PC
```

Datapoints

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the device
<n>	<m>	R	output <n> has input <m> as source
		W	set input <m> as source for output <n>
	next	W	select the next input number for output <n>
	prev	W	select the previous input number for output <n>

User Interface

Wyrestorm Object

The Wyrestorm object lets you control one output of a Wyrestorm I/O Server.

The object displays the input selected on the addressed output. If an entry is available in wyrestorm.ini for the selected input, then the specified name will be reported, otherwise the object will show the input number. This object is listed in the Project Editor's new object list only when at least one Wyrestorm I/O Server is defined.

Parameters

- **server id**: the Wyrestorm I/O server ID
- **output**: the output number to be addressed
- **position**: the object's position. Use the pixels or rows and columns coordinates format
- **CSS style**: the CSS style to apply to the inner text (optional)

Syntax

```
(wyrestorm <server id>; <output>; <position>; <CSS style>)
```

E.g.

```
(wyrestorm serverid; 2; x10y20; font-size:50px)
```

UISET Actions

ID	Attribute	Set to	
connection.label	visible	true	when datapoint connection = offline
		false	when datapoint connection = online
<n>.label	value	the number of the input routed on the output <n>. If wyrestrom.ini contains an entry for the current input then the number is replaced by the specified name	

USER Commands

Name	Param	Action
<n>	<m>	route the input <m> on output <n>
	next	select the next input number for output <n>
	prev	select the previous input number for output <n>

Release Notes

3.2.0

- initial version release

WyreStorm is a registered trademarks of WyreStorm Technologies

Yamaha RX-V

Yamaha RX-V1067, RX-V2067, and RX-V3067 are multi-source, multi-zone AV receiver/amplifier that may be integrated to work with external control systems and servers via ethernet connection.

Yamaha Configuration

In order to be able to control the device with HSYCO SERVER some initial configuration of the Yamaha device is needed.

Using the supplied remote control, go to the "home screen" and select the "setup" menu. From here, configure the network setting a fixed IP address compatible with the network the device is connected to. Further, make sure the option "network standby" is set to "on".

HSYCO Configuration

Add a Yamaha I/O Server in the I/O Servers section of the Settings and set its parameters:

Communication

- **IP Address**: the IP address of the device.

High Availability

- **Shutdown when inactive**: defaults to true.

Options

ID	Default	Values	Description
gui	true	true	enable support for UISET actions and USER commands
		false	disable UI support
pollinterval	1	n > 0	the data acquisition interval, in seconds

Datapoints

For datapoints addressing a specific zone, replace <zone> with the name of the zone, i.e. use "main" for the main zone and "z2", "z3" or "z4" for the other zones (the number of zones depends on the Yamaha model).

ID	Value	R/W	Description
connection	online	R	connection established
	offline	R	HSYCO can't connect to the panel
system.power	1	R	at least one zone is on

		W	turn all zones on
	0	R	all zones are off
		W	turn all zones off
system.party	1	R	party mode is on
		W	set party mode on
	0	R	party mode is off
		W	set party mode off
<zone>.power	1	R	zone <zone> is on
		W	turn zone <zone> on
	0	R	zone <zone> is off
		W	turn zone <zone> off
<zone>.sleep	off	R	the sleep time of this zone is set to off
		W	set the sleep time of this zone to off
	30	R	the sleep time of this zone is set to 30 minutes
		W	set the sleep time of this zone to 30 minutes
	60	R	the sleep time of this zone is set to 60 minutes
		W	set the sleep time of this zone to 60 minutes
	90	R	the sleep time of this zone is set to 90 minutes
		W	set the sleep time of this zone to 90 minutes
	120	R	the sleep time of this zone is set to 120 minutes
		W	set the sleep time of this zone to 120 minutes
<zone>.source	<src>[Note 1]	R	the source for this zone is set to <src>[Note 1]
		W	set the source for this zone to <src>[Note 1]

<zone>.source.name	<name>	R	the current source on this zone is named <name>
<zone>.volume	off	R	this zone volume is muted
		W	mute this zone
	on	W	un-mute this zone
	1% ... 100%	R	this zone's volume is set to the reported value
		W	set this zone's volume to the specified value (the '%' character can be omitted)
	up	W	increase this zone's volume
	down	W	decrease this zone's volume
<zone>.mute	1	R	this zone's volume is muted
		W	mute this zone
	0	R	this zone's volume is not muted
		W	un-mute this zone
main.scene	1 ... 12	W	set the main zone to the specified scene number
z<n>.scene	1 ... 4	W	set zone <n> to the specified scene number
main.surround	<prg>[Note 2]	R	the surround program is set to <prg>[Note 2]
		W	set the surround program to <prg>[Note 2]
main.enhancer	1	R	the enhancer is on
		W	set the enhancer on
	0	R	the enhancer is off
		W	set the enhancer off

Note 1

The main zone accepts the following parameters as sources: "av1", "av2", "av3", "av4", "av5", "av6", "av7", "tuner", "phono", "vaux", "multich", "ipod", "pc", "netradio", "usb", "audio1", "audio2", "audio3", "audio4". The other zones accept the following parameters as source: "av1", "av2", "av3", "av4", "tuner", "phono", "vaux", "ipod", "pc", "netradio", "usb", "audio1", "audio2", "audio3", "audio4".

Note 2

The following is the list of accepted surround programs for the main zone: "straight", "hallinmunich", "hallinvienna", "chamber", "cellarclub", "theroxytheatre", "thebottomline", "sports", "actiongame", "roleplayinggame", "musicvideo", "standard", "spectacle", "scifi", "adventure", "drama", "monomovie", "2chstereo", "7chstereo", "surrounddecoder".

User Interface

UISET Actions

ID	Attribute	Set to	
connection	visible	true	when datapoint connection = offline
		false	when datapoint connection = online
system.power	value	ON	when at least one zone is active
		OFF	when all zones are off
system.party	value	ON	when party mode is on
		OFF	when party mode is off
	visible	true	when the device is on
		false	when the device is off
<zone>.power	value	ON	when the zone is on
		OFF	when the zone is off
<zone>.source	value	the source name, as set in the Yamaha device	
	visible	true	when the zone is on
		false	when the zone is off
<zone>.source.name	value	the user-defined source name	
	visible	true	when the zone is on
		false	when the zone is off
<zone>.sleep	value	OFF	when the sleep time of this zone is set to off
		30 min	when the sleep time of this zone is set to 30 min
		60 min	when the sleep time of this zone is set to 60 min
		90 min	when the sleep time of this zone is set to 90 min
		120 min	when the sleep time of this zone is set to 120 min
	visible	true	when the zone is on
		false	when the zone is off

<zone>.volume.status	value	MUTE	when tis zone is muted
		1% ... 100%	when the volume is set to the shown value
	visible	true	when the zone is on
		false	when the zone is off
main.surround	value	the active surround program for the main zone	
	visible	true	when the zone is on
		false	when the zone is off
main.enhancer	value	ON	when the enhancer on the main zone is on
		OFF	when the enhancer on the main zone is off
	visible	true	when the zone is on
		false	when the zone is off

USER Commands

Name	Param	Action
system	power	switch the state of the device on/off
	on	turn all zones on
	off	turn all zones off
	party	switch the party mode
<zone>.volume	0%	mute the zone
	1% ... 100%	set the volume of the zone to the specified value
	up	increase volume of the zone by 5%
	down	decrease volume of the zone by 5%
<zone>	power	switch the zone on/off
	on	turn the zone on
	off	turn the zone off
	sleep	switch the sleep time of the zone (off → 30 min → 60 min → 90 min → 120 min → off)
	mute	mute/unmute the zone
<zone>.mute	on	mute the zone

	off	unmute the zone
<zone>.source	<src>[Note 1]	set the source for this zone to <src>[Note 1]
main.scene	1 ... 12	set the main zone to the specified scene number
z<n>.scene	1 ... 4	set zone <n> to the specified scene number
main.surround	<prg>[Note 2]	set the surround program to <prg>[Note 2]
main	enhancer	switch the enhancer of the main zone on/off
main.enhancer	on	activate the enhancer on the main zone
	on	deactivate the enhancer on the main zone

Release Notes

3.0.0

- initial version release

Yamaha is a registered trademark of Yamaha Corporation