

HSYCO SERVER

CONFIGURATION AND PROGRAMMING HANDBOOK

HSYCO 3.1.2

DOC REL. EN 1

© 2007-2012 Home Systems Consulting SpA. All rights reserved.

HSYCO® Configuration and programming handbook

Any unauthorized reproduction, copy with electronic or mechanical means, or translation is forbidden.

The content of this manual is protected by the copyright law and international treaties, even when not distributed with a software accompanied by a license for the end user. This handbook is provided for instructional purposes only and can be subject to any modification without preventive communication.

Home Systems Consulting expressly denies any responsibility for any mistake contained in this handbook.

Installation	19
Initial Set-Up	20
The Configuration Console	21
The Administration Pages	23
Accounts Management	26
System Password	27
Network Settings	29
Clock Settings	31
The Manager	32
The File Manager	34
Upgrading and restarting HSYCO	36
The Log Viewer	37
Files Organization	38
hsyco/access.ini	39
hsyco/acl.ini	39
hsyco/console.log	39
hsyco/hsyco.ini	40
hsyco/hsyco.jar	40
hsyco/hsyco.keys	40
hsyco/keys.data	40
hsyco/license.txt	41
hsyco/systemtopo.txt	41
hsyco/timers.data	41
hsyco/com/hsyco	42

hsyco/www	42
hsyco/www/<project>/index.hsm	42
hsyco/www/<project>/img	43
hsyco/www/<skin>	43
hsyco/www/img	43
hsyco/www/public	43
hsyco/motion	44
hsyco/logs	44
hsyco/ir	45
hsyco/dataloggers.ini	45
hsyco/data	47
hsyco/Scripts	47
Software License Key	48
Access to the Operating System	50
Start and Stop the HSYCO Server	51
Access to Files	52
Access to Files on Microsoft Windows	52
Access to Files on Mac OS X	55
Configuration	57
Security and Users Management	58
hsyco.ini Initial Settings	59
Introduction to Field Systems	61
The hsyco.ini File	64
Access Control	64
Log Levels	67

High Availability	68
Network	69
Clock	70
DMX Gateway	72
Cameras	73
I/O Servers	79
Remote HSYCO Servers	80
Serial Communication Ports	81
IRTrans	83
Squeezebox	84
Public Announcement	85
Timers	85
Location Services	86
Data Loggers	87
Email	90
Additional Parameters	91
Customization	93
The Configuration Database	94
(location)	95
(devices)	96
The Web Pages	98
The Project Editor	99
Creating a New Project	100
Page Size and Coordinates	100
Editing a Project	102

Menus, Pages and Pop-Ups	104
Page's Objects	105
Containers	107
The index.hsm project's source file	108
Objects Reference	109
Objects	109
General directives	110
(#skin <skin name>)	110
(#language <language id>)	110
(#size <page size>)	111
(#scale <factor>)	111
(#kiosk <mode>)	112
(#locked <mode>)	112
(#deviceimage disable)	112
(#location <list>)	112
(#cameralist <list>)	113
(#cameragridlist <list>)	113
(#cameraoverlay <overlay list>)	114
(#include <file name>)	114
Main Menu	114
(header <title>)	114
(menu)	115
(menu#landscape)	115
(menu#portrait)	115
(endofmenu)	115
Pages	116
(link <pos>; <color>; <page>; <label>)	116

(linkmini <pos>; <color>; <page>; <label>)	116
(linkmicro <pos>; <color>; <page>; <label>)	116
(dlink <pos>; <color>; <page>; <label>)	117
(page <id>; <description>; <protection>)	118
(page#landscape <id>; <description>; <protection>)	118
(page#portrait <id>; <description>; <protection>)	118
(popup <id>; <description>)	119
(endofpage)	119
(container <pos>; <visibility>)	119
(endofcontainer)	120
(selector <file>; <pos>; <width>; <height>; <container id>; <group>; <text>; <css>)	120
(background <file>)	121
Text	121
(text <pos>; <text>; <css>)	121
(marquee <pos>; <width>; <height>; <dir>; <speed>; <text>; <css>)	122
Graphic Elements	122
(panel <pos>; <width>; <height>; <color>)	122
(hbar <pos>; <width>)	123
(vbar <pos>; <height>)	123
(image <file>; <pos>; <width>; <height>; <text>; <css>)	123
(imagelink <file>; <pos>; <width>; <height>; <page>; <text>; <css>)	124
(chart!id <pos>; <width>; <height>; <attributes>)	125
(datalogger <id>; <pos>; <width>; <height>; <label>; <attributes>)	127
(video <src>; <pos>; <width>; <height>; <mode>)	129
Administration	130
(adminlink <function>; <pos>; <color>; <label>)	130
Lighting and Automation	131

(button <address>; <pos>; <label>)	131
(3button <address>; <pos>; <label>)	131
(buttonicon <address>; <pos>; <color>)	132
(buttonimage <address>; <pos>; <width>; <height>; <unknown img>; <on img>; <off img>; <up img>; <down img>; <offup img>; <offdown img>; <text>; <css>)	132
(dimmer <address>; <pos>; <label>)	134
(dmx <address>; <pos>; <label>)	134
(dmxrgb <address>; <pos>; <label>)	135
Temperature Control	136
(temp <server>; <address>; <label>; <pos>)	136
(tempmini <server>; <address>; <label>; <pos>)	136
Timers and Schedulers	137
(timer <id>; <pos>; <label>)	137
(scheduler <pos>; <width>; <height>; <mode>; <names>)	137
Cameras	140
(cameralink <cam id>; <pos>; <color>; <label>; <dest panel id>)	140
(camera <cam id>; <pos>; <width>; <height>; <dest panel id>)	140
(camerapanel <cam id>; <pos>; <width>; <height>; <cam list>)	141
IRTrans	142
(ir <irid>; <com>; <pos>; <color>; <label>)	142
(irmini <irid>; <com>; <pos>; <color>; <label>)	142
(irmicro <irid>; <com>; <pos>; <color>; <label>)	142
Music Players	143
(music <id>; <pos>; <label>)	143
(musicsync <pos>)	143
(nuvo <id>; <zone>; <pos>)	144
(nuvomini <id>; <zone>; <pos>)	144
User Buttons	145

(user <name>; <param>; <pos>; <color>; <label>)	145
(usermini <name>; <param>; <pos>; <color>; <label>)	145
(usermicro <name>; <param>; <pos>; <color>; <label>)	145
(userrgb <name>; <param>; <pos>; <color>; <label>)	146
(userimage <file>; <pos>; <width>; <height>; <name>; <param>; <text>; <css>)	147
(slider!id <name>; <pos>; <label>)	148
(sliderbutton!id <name>; <pos>; <label>)	149
Forms	150
(input!id <pos>; <css>)	150
(submit!id <pos>; <color>; <label>)	150
(submitmini!id <pos>; <color>; <label>)	150
(submitmicro!id <pos>; <color>; <label>)	150
(submitimage <file>; <pos>; <width>; <height>; <text>; <css>)	151
(keypad!id <pos>; <min>; <max>; <digits>; <decimals>; <type>; <label>; <css>)	152
Weather	153
(weather!id <pos>)	153
Web Interface Options	154
Web Pages Translation	155
The Skins Interpreter	156
Inserting HTML Code in the Pages	158
Events Programming	159
EVENTS	160
The EVENTS Language	161
Comments	162
Other General Rules	162
Stable and transient events	163

One Event, More Rules	165
Actions	166
Variables	166
Variables and Actions	167
Variables and Events	169
Persistent and Volatile Variables	169
Predefined Variables	170
Events	172
I/O Servers	172
IO	172
IOSTART	173
DMX	174
DMXSTART	174
DMX	175
Cameras	176
CAMERA	176
CAMERACOMMAND	177
Security	178
SECURITY	178
IRTrans	179
IR	179
Squeezebox	180
MUSIC	180
PBX	181
PBX	181
Network Services	182

LOCATION	182
PING	182
URL	183
Timers and Schedulers	184
TIMER	184
Leak Detector	185
LEAK	185
Internal Events	186
HSYCOSTART	186
USER	186
PROGRAMTIMER	187
TIME	187
DAY	188
NIGHT	188
SUNAZIMUTH	189
SUNELEVATION	189
POWER	190
Actions	191
I/O Servers	191
IO	191
DMX	192
Cameras	193
CAMERA	193
CAMERAREC	193
CAMERARECFULL	194
Serial Communication Ports	195
COMM	195

IRTrans	196
IR	196
Squeezebox	197
MUSIC	197
Public Announcement	198
AUDIO	198
Mail	200
MAIL	200
Log	202
LOG	202
Network Services	203
PING	203
URL	203
Data Logger	205
DATALOGGER	205
Leak Detector	207
LEAK	207
Internal Actions	209
WAIT	209
USER	209
POWER	210
PROGRAMTIMER	210
UISET	212
Java Programming	219
Predefined Constants	220
Callback Methods	221

I/O Servers	221
public static void IOStartupEvent(int serverIndex)	221
public static void IOEvent(String id, String value)	221
Timers and Schedulers	223
public static boolean UserTimerEvent(String name, boolean active)	223
DMX	224
public static void DmxStartupEvent(int serverIndex)	224
public static void DmxEvent(int channel, int state)	224
public static int DmxFilter(int channel, int state, boolean reverse)	224
Cameras	226
public static void CameraMotionEvent(String eventName, long remoteTime)	226
public static int cameraCommandEvent(String function, String action, String camera)	226
IRTrans	228
public static void IREvent(boolean received, int irtransid, String event)	228
Squeezebox	229
public static void SlimPowerEvent(int index, int power)	229
public static void SlimStatusEvent(int index, int status)	229
public static void SlimVolumeEvent(int index, int volume)	229
PBX	230
public static boolean PBXCallEvent(String host, String caller, String called)	230
Network Services	231
public static void LocationEvent(long MAC, InetAddress IP, int zoneId)	231
System Methods	232
public static void StartupEvent()	232
public static String getUserVersion()	232
public static void TimeEvent(long time)	232

public static void programTimerEvent(String name)	232
public static String userCommand(String name, String param)	233
public static void DaylightEvent(boolean day)	233
public static int PowerEvent(int power)	234
public static void SunPositionEvent(int azimuth, int elevation)	234
public static String WebRootRequestEvent(InetAddress addr, boolean secure, String useragent)	234
static void SchedulerEvent(String groupname, String schedulename)	235
Command and Utility Methods	236
I/O Servers	236
static void ioSet(String id, String value)	236
static String ioGet(String id)	236
DMX	237
static int dmxSet(int channel, int state)	237
static int dmxSet(int[] channels, int[] state)	237
static int dmxOff(int channel)	238
static int dmxOff(int[] channels)	238
static int dmxOn(int channel)	239
static int dmxOn(int[] channels)	239
static int dmxMerge(int from, int to, boolean merge)	240
static int dmxMerge(int[] channels, boolean merge)	240
static int dmxGet(int channel)	240
static void dmxOffTimerSet(int channel, int seconds)	241
static void dmxOffTimerClear(int channel)	241
static void dmxOffTimerPreset(int channel, int seconds)	242
static void dmxOffTimerReset(int channel, int seconds)	242
Cameras	243
static void cameraRecTrigger(String camera, String source, int seconds)	243

static void cameraRecTriggerFull(String camera, String source, int seconds)	243
static int cameraCommand(String function, String action, String camera)	244
static void cameraMode(String camera, boolean enabled)	244
static void cameraRecMode(String camera, boolean enabled)	245
Serial Communication Ports	246
static int writeComm(String portName, String data)	246
static String readComm(String portName, int len)	246
static int closeComm(String portName)	247
Plugins	248
static Bentel getBentelPlugIn(String id)	248
static ParadoxEVO getParadoxEVOPlugIn(String id)	248
static Tecnoalarm getTecnoalarmPlugIn(String id)	248
Modbus	250
static byte[] modbusReadCoils(String name, int unit, int address, int quantity)	250
static byte[] modbusReadDiscreteInputs(String name, int unit, int address, int quantity)	250
static byte[] modbusReadHoldingRegisters(String name, int unit, int address, int quantity)	251
static byte[] modbusReadInputRegisters(String name, int unit, int address, int quantity)	252
static byte[] modbusWriteSingleCoil(String name, int unit, int address, boolean value)	252
static byte[] modbusWriteSingleRegister(String name, int unit, int address, byte[] bytes)	253
static byte[] modbusWriteMultipleRegisters(String name, int unit, int address, byte[] bytes)	254
static byte[] modbusWriteMaskRegister(String name, int unit, int address, byte[] mask, byte[] bytes)	254
IRTrans	256
static int irtransCommand(String irlId, String command)	256
Squeezebox	257

static String slimCommand(int index, String command)	257
static String slimCommand(String playerName, String command)	257
static int slimButton(int index, String button)	257
static int slimButton(String playerName, String button)	258
Public Announcement	259
static int audioPlay(String where, File file)	259
static int audioPlay(String where, String voice, String text)	260
Data Logger	261
static boolean dataLoggerRefresh(String name)	261
static boolean dataLoggerUpdate(String name, double value)	261
static boolean dataLoggerClear(String name)	261
static boolean dataLoggerSave(String type, String[] names, String path, boolean timestamp)	262
static boolean dataLoggerOptions(String name, String param, String value)	262
Leak Detector	263
static void leakDetectorClear(String name)	263
static void leakDetectorOptions(String name, String param, String value)	263
public static int leakDetectorUpdate(String name, double value)	264
Mail	265
static int sendMail(String to, String from, String subject, String body)	265
static int sendMail(String to, String from, String subject, Vector<String> body)	266
Log	268
static void messageLog(String message)	268
static void errorLog(String message)	268
Network Services	269
ping(String host, int timeout)	269
static int wakeOnLan(String broadcast, String address)	269
System Methods	270

static boolean haActiveState()	270
static boolean isDaylight()	270
static long getNextSunrise(long now, boolean withoffset)	270
static long getNextSunset(long now, boolean withoffset)	270
static void programTimerSet(String name, int seconds)	271
static void programTimerClear(String name)	271
static void programTimerReset(String name, int seconds)	271
static void programTimerRepeat(String name, int seconds)	272
static void sleep(long millis)	272
static void schedulerRegister(String groupname, String schedulename, int interval)	272
static void SchedulerRemove(String groupname, String schedulename)	272
static void user(String name, String param)	273
static String varGet(String name)	273
static void varSet(String name, String value)	273
static String uiGet(String id, String attr)	274
static void uiSet(String id, String attr, String value)	274
static void powerSet(int power)	280

Appendix	281
Data Loggers	281
Counter Data Loggers	282
Range Data Loggers	286
Cameras Configuration	289
Users Configuration	289
Frames Acquisition	289
Motion Detection	289
PTZ Control	290

Serial Ports Servers Configuration	291
IRTrans	291
HW Group's Controllers	291
Generic Serial Gateways	292
Wi-Fi Client Location Services	293
Telephony Services	294
Public Announcement	295
Playing Audio Files	295
The Text-to-Speech Engine	295
eSpeak	296
Acapela	296
Mac OS X	296
Log Files Format	298
Starting HSYCO Without a License Key	298
HSYCO Normal Start-up Process	299
I/O Servers Connection Errors	300
IRTrans Connection Errors	300
Writing Errors of the Cameras Images	300
Security Messages	301
Commands Sent from the Web Interface	302
Web Wi-Fi Clients Location services	302
Phone Calls Log	303
The security.log File	303
Access Control List	304
SSL Certificates for Cryptography	306

SQL Database	309
Connect	310
Disconnect	310
Tables	310
Statement	311
Query	312
Insert	313
Update	313
The access.ini File	314
Release Notes	317
3.1.2	317
3.1.1	318
3.1.0	319
3.0.3	321
3.0.2	322
3.0.1	322
3.0.0	324
2.10.4	328
2.10.3	328
2.10.2	329
2.10.1	329
2.10.0	329
2.9.0	333
2.8.4	337
2.8.3	338

2.8.2	338
2.8.1	339
2.8.0	340
2.7.1	343
2.7.0	344
2.6.1	346
2.6.0	346
2.5.4	348
2.5.3	349
2.5.2	350
2.5.1	351
2.5.0	351
2.4.3	352
2.4.2	352
2.4.1	352
2.4.0	352
2.3.0	353
2.2.2	353
2.2.1	353
2.2.0	353
2.1.7	353
2.1.6	354
2.1.5	354
2.0	355

Installation

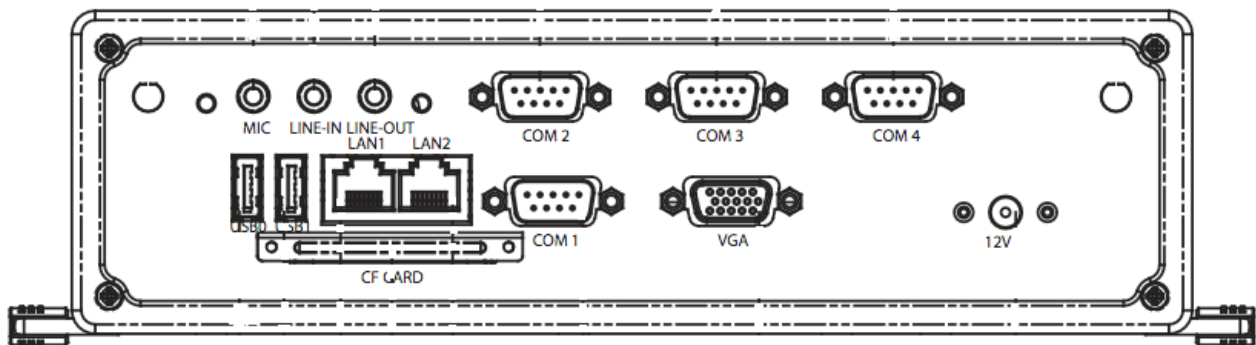
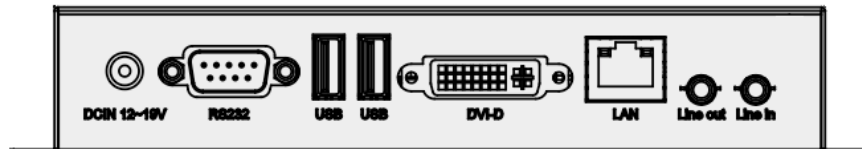
HSYCO SERVER ships pre-installed with the latest version of the HSYCO software, on a Linux Ubuntu 10.04 LTS Server distribution stripped of all not-essential services. Java SE Runtime Environment version 6 is also installed. No other software is required.



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

There are two standard hardware configurations, a compact server with one Ethernet port and one Serial port, and a larger, expandable server with two Ethernet ports and four serial ports.



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.

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.

The Administration Pages

HSYCO SERVER is pre-configured with an administration menu page that gives you access to some administration functions, including the ability to change the network settings and the operating system's password, and HSYCO users management.

To access the administration menu, enter the following URL in your Web Browser:

<https://192.168.0.50/hsycoserver/admin>

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.

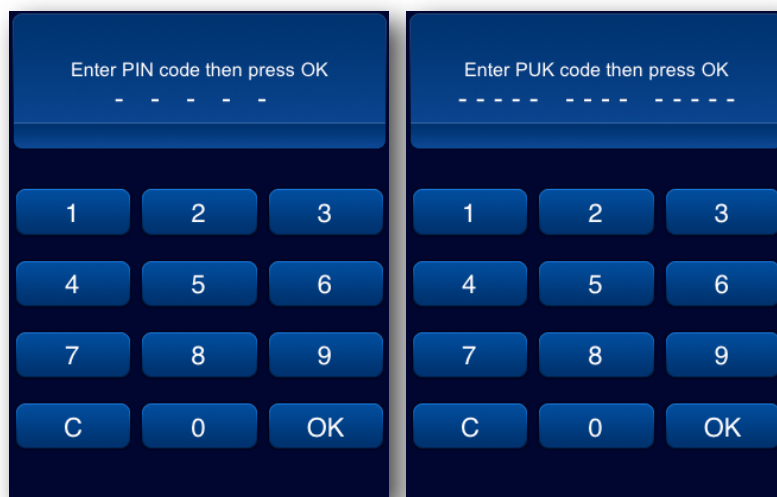


On the iPod touch, iPhone or iPad press Continue, on other browsers follow the dialog boxes to permanently accept the certificate.

You will see the PIN and PUK authentication page. Enter the default PIN and PUK, and press OK:

PIN: 0000

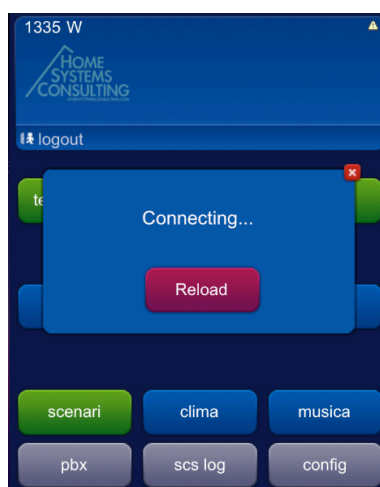
PUK: 00000 0000 00000



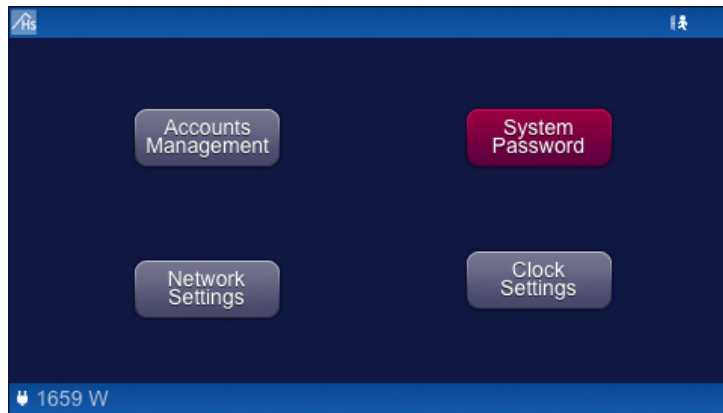
The default pre-configured PIN and PUK are the same for all HSYCO SERVERS, to let you conveniently access the user interface during the initial installation. Both the PIN and PUK should be changed immediately to different, hard to guess numbers.

If a yellow triangle appears at the top right corner of the display, the browser can't communicate with HSYCO server. One of the reasons can be the loss of the server certificate.

To regenerate the certificate press the yellow triangle or the Reload button to force a refresh of the Web page, then press Continue on the Server Identity pop-up.



The standard administration page has four buttons. Later on you will learn how to reconfigure the administration page, as well as any other page in HSYCO, to fit your specific needs.



Accounts Management

The accounts management 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.



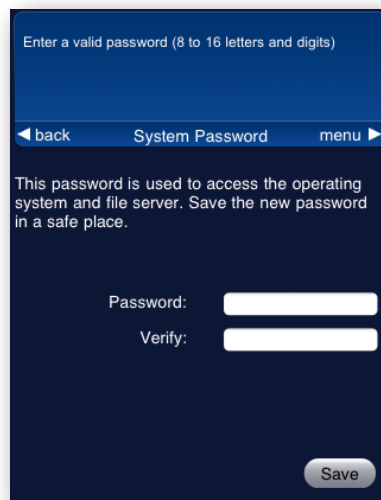
To change the predefined PIN and PUK, select **user**, then enter the new PIN and PUK codes in the appropriate fields, and press **Save** to confirm.

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 System Password button in the administration page 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.

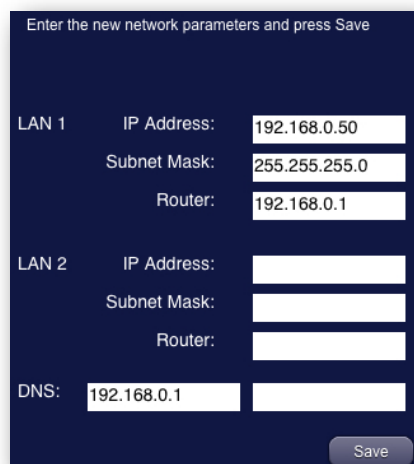
You should save the password in a safe place. If you forget the password, but still have access to HSYCO, you can simply go to the password page and type a new password.

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.

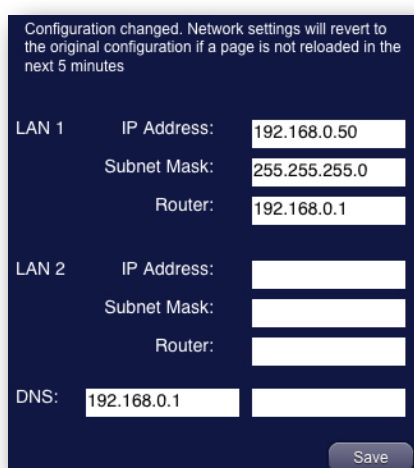


Enter the new network parameters and press Save

LAN 1	IP Address:	192.168.0.50
	Subnet Mask:	255.255.255.0
	Router:	192.168.0.1
LAN 2	IP Address:	
	Subnet Mask:	
	Router:	
DNS:	192.168.0.1	

Save

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.



Configuration changed. Network settings will revert to the original configuration if a page is not reloaded in the next 5 minutes

LAN 1	IP Address:	192.168.0.50
	Subnet Mask:	255.255.255.0
	Router:	192.168.0.1
LAN 2	IP Address:	
	Subnet Mask:	
	Router:	
DNS:	192.168.0.1	

Save

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.

Time settings:

Year	Month	Day
2012	07	25

Hours	Minutes
12	04

Daylight Saving Time: on

Time Zone: Europe/Rome (Central European Summer Time)

Automatic Time Set: on

Save

1689 W

To change the time settings, enter the new parameters and press **Save**.

All the other parameters are read-only from the clock page. It is possible to set them in the hsyco.ini file.

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.

The Manager

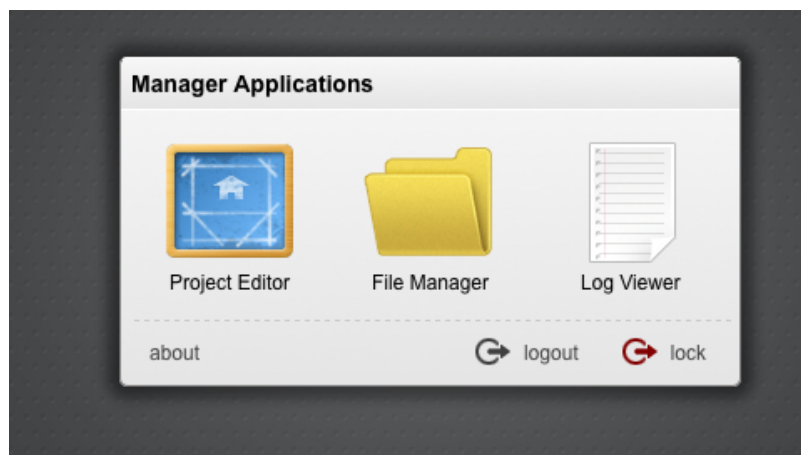
The Manager is the Web-based toolkit of applications that you will use to create customized graphic user interfaces and access all system's configuration files.

The Manager runs on most modern Web browsers, including tablets like Apple's iPad. It 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.

To access the Manager, enter the following URL in your Web Browser:

`https://192.168.0.50/hsycoserver/manager`

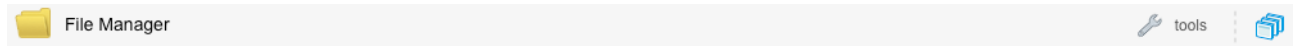
You need to authenticate using a PIN with administrator privileges to use the Manager.



The Manager Toolkit has three application: the Project Editor is the visual editor you use to customize the user interface; the 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; and the Log Viewer, a real-time viewer of HSYCO's logs.

Touch “about” in the bottom left corner to show HSYCO’s current version information.

You can switch between applications touching the  icon on the top bar.

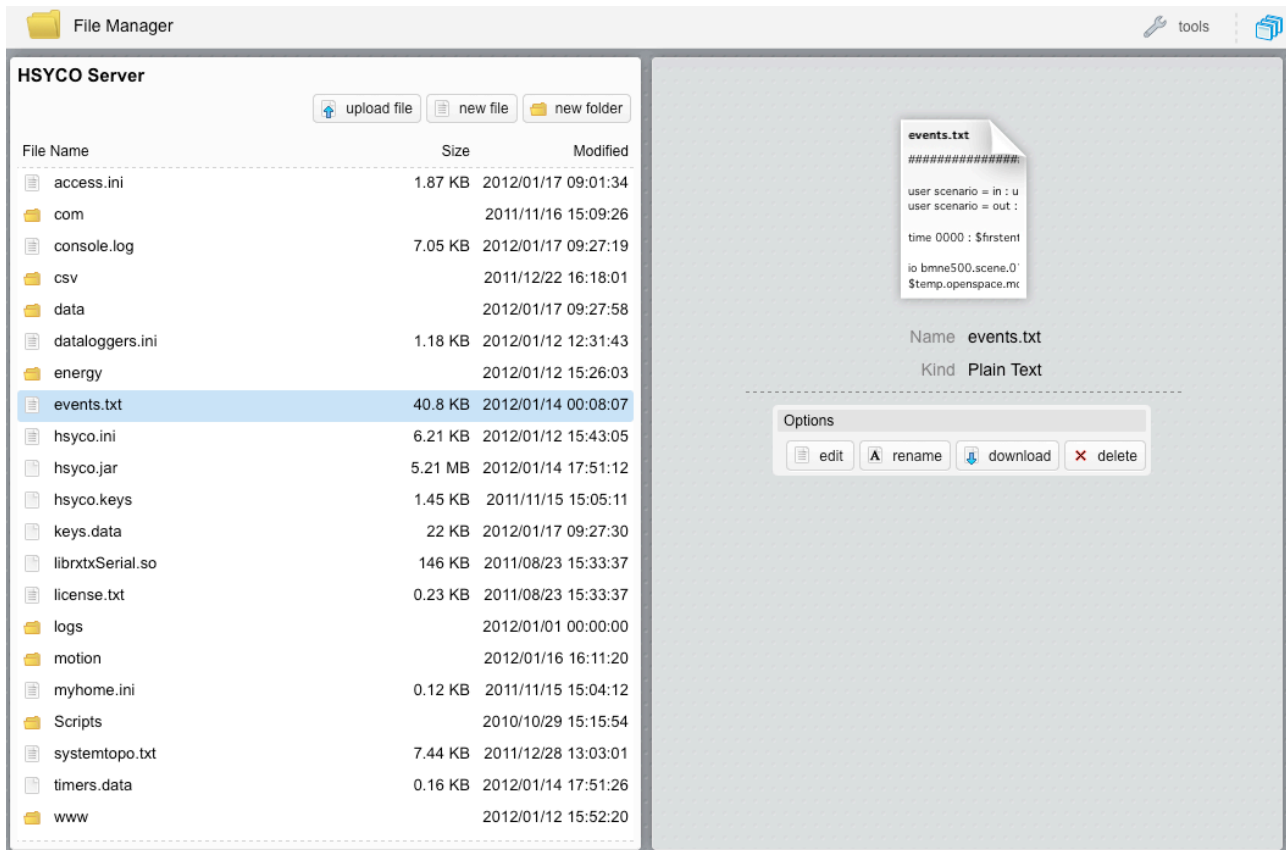


Using a PC with a physical keyboard, you can press [Ctrl] and [Shift] at the same time for fast switching between applications. An application selection panel appears until you keep the [Ctrl] key pressed; select the application you need by repeatedly pressing [Shift], then releasing [Ctrl].

The Project Editor will be discussed later, in the Customization chapter.

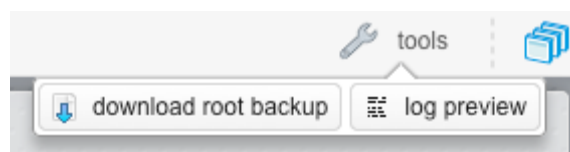
The File Manager

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.

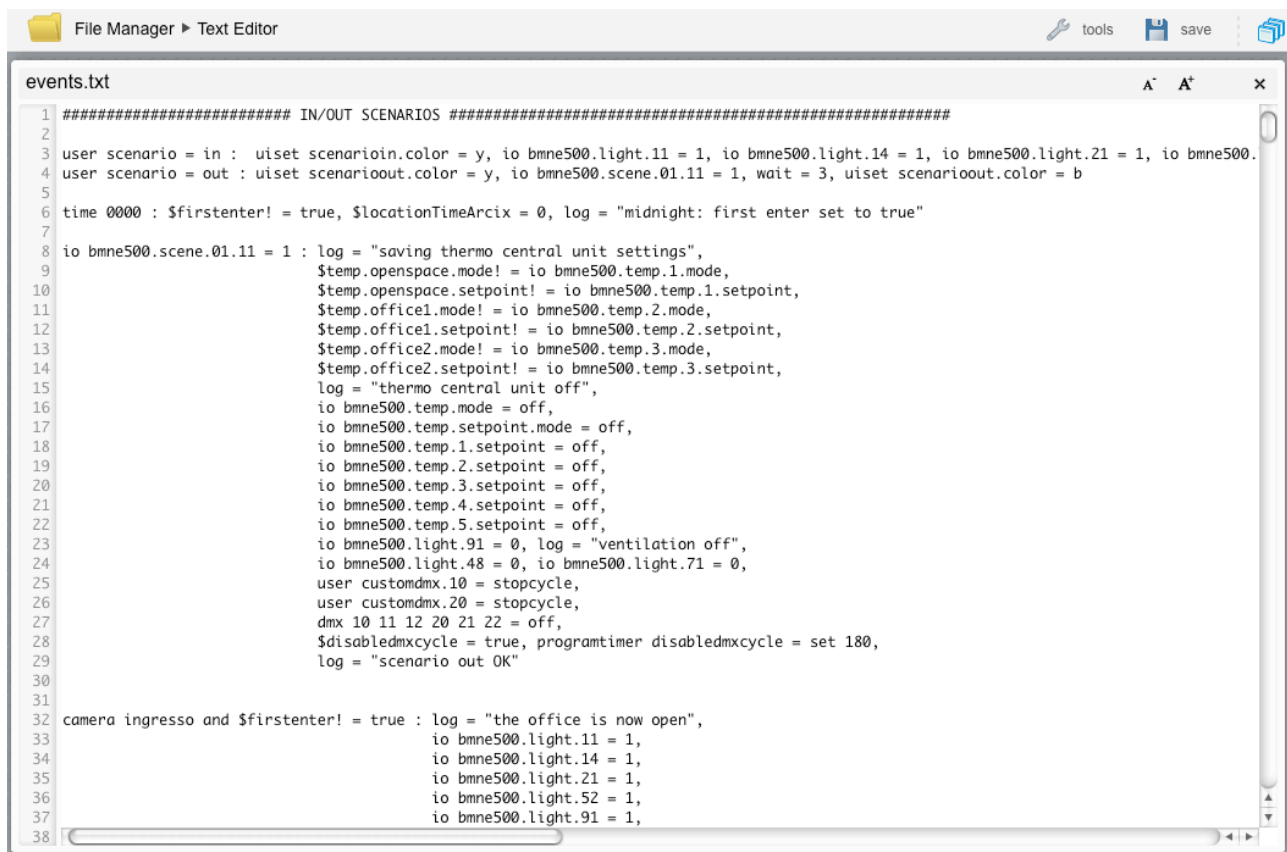


You can download and upload individual files or ZIP directories, and create, rename and delete files and directories.

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.

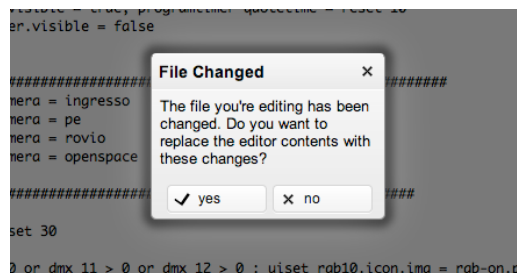


```

1 ##### IN/OUT SCENARIOS #####
2
3 user scenario = in : uiset scenarioin.color = y, io bmne500.light.11 = 1, io bmne500.light.14 = 1, io bmne500.light.21 = 1, io bmne500.
4 user scenario = out : uiset scenarioout.color = y, io bmne500.scene.01.11 = 1, wait = 3, uiset scenarioout.color = b
5
6 time 0000 : $firstenter! = true, $locationTimeArcix = 0, log = "midnight: first enter set to true"
7
8 io bmne500.scene.01.11 = 1 : log = "saving thermo central unit settings",
9     $temp.openspace.mode! = io bmne500.temp.1.mode,
10     $temp.openspace.setpoint! = io bmne500.temp.1.setpoint,
11     $temp.office1.mode! = io bmne500.temp.2.mode,
12     $temp.office1.setpoint! = io bmne500.temp.2.setpoint,
13     $temp.office2.mode! = io bmne500.temp.3.mode,
14     $temp.office2.setpoint! = io bmne500.temp.3.setpoint,
15     log = "thermo central unit off",
16     io bmne500.temp.mode = off,
17     io bmne500.temp.setpoint.mode = off,
18     io bmne500.temp.1.setpoint = off,
19     io bmne500.temp.2.setpoint = off,
20     io bmne500.temp.3.setpoint = off,
21     io bmne500.temp.4.setpoint = off,
22     io bmne500.temp.5.setpoint = off,
23     io bmne500.light.91 = 0, log = "ventilation off",
24     io bmne500.light.48 = 0, io bmne500.light.71 = 0,
25     user customdmx.10 = stopcycle,
26     user customdmx.20 = stopcycle,
27     dmx 10 11 12 20 21 22 = off,
28     $disabledmxcycle = true, programtimer disabledmxcycle = set 180,
29     log = "scenario out OK"
30
31
32 camera ingresso and $firstenter! = true : log = "the office is now open",
33     io bmne500.light.11 = 1,
34     io bmne500.light.14 = 1,
35     io bmne500.light.21 = 1,
36     io bmne500.light.52 = 1,
37     io bmne500.light.91 = 1,
38

```

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.

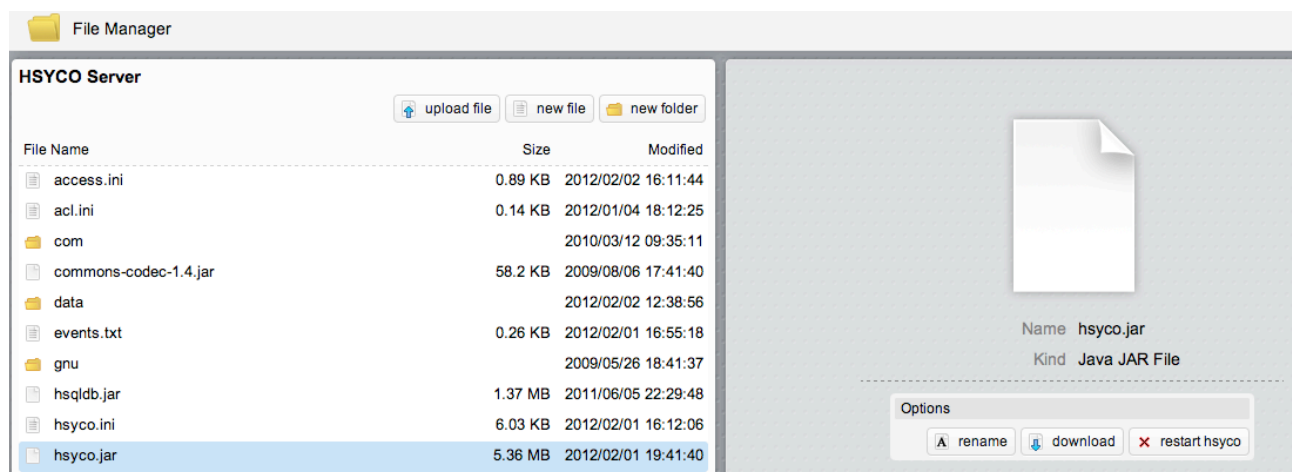


Your changes will be lost if you accept to reload the new version of the file.

Upgrading and restarting HSYCO

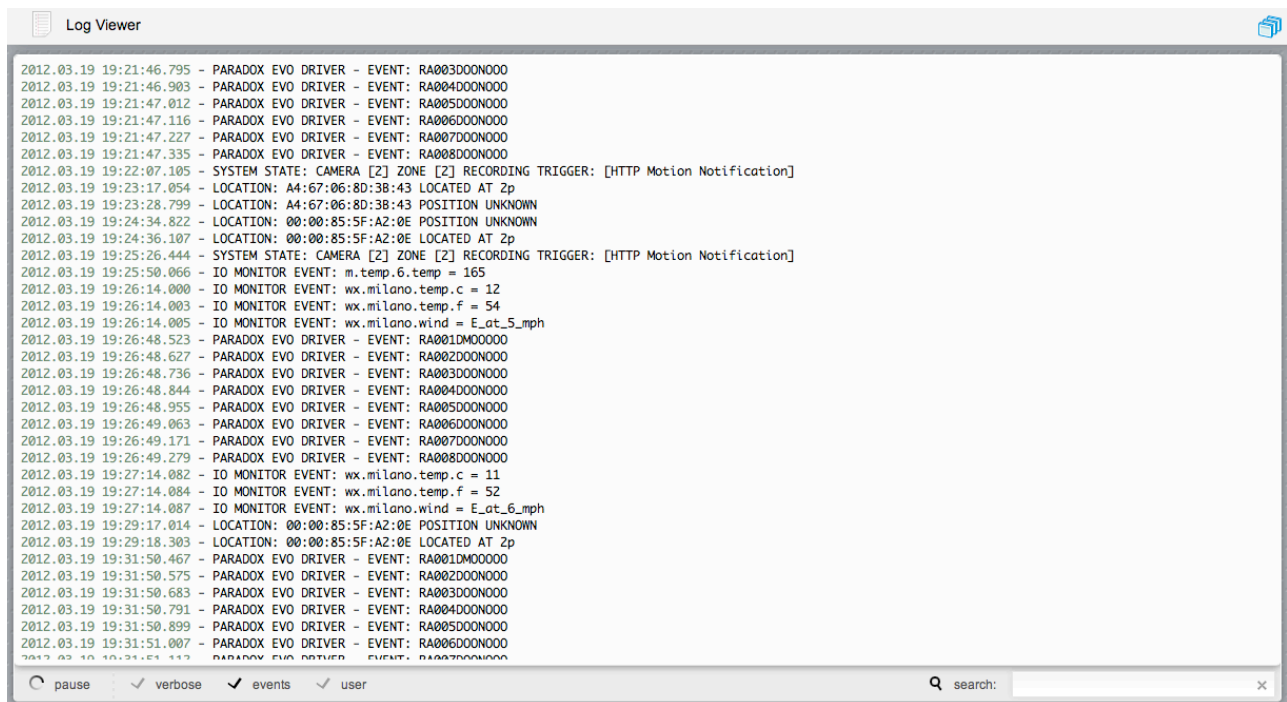
In order to upgrade the HSYCO software version, you should simply upload a new version of the hsyco.jar file to the root directory. After the upload, the server will restart and the Manager will reload on the browser.

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.




The Log Viewer

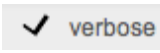
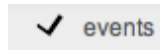

The Log Viewer application shows a scrollable live view of the last 1000 lines from the daily log file.



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  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.

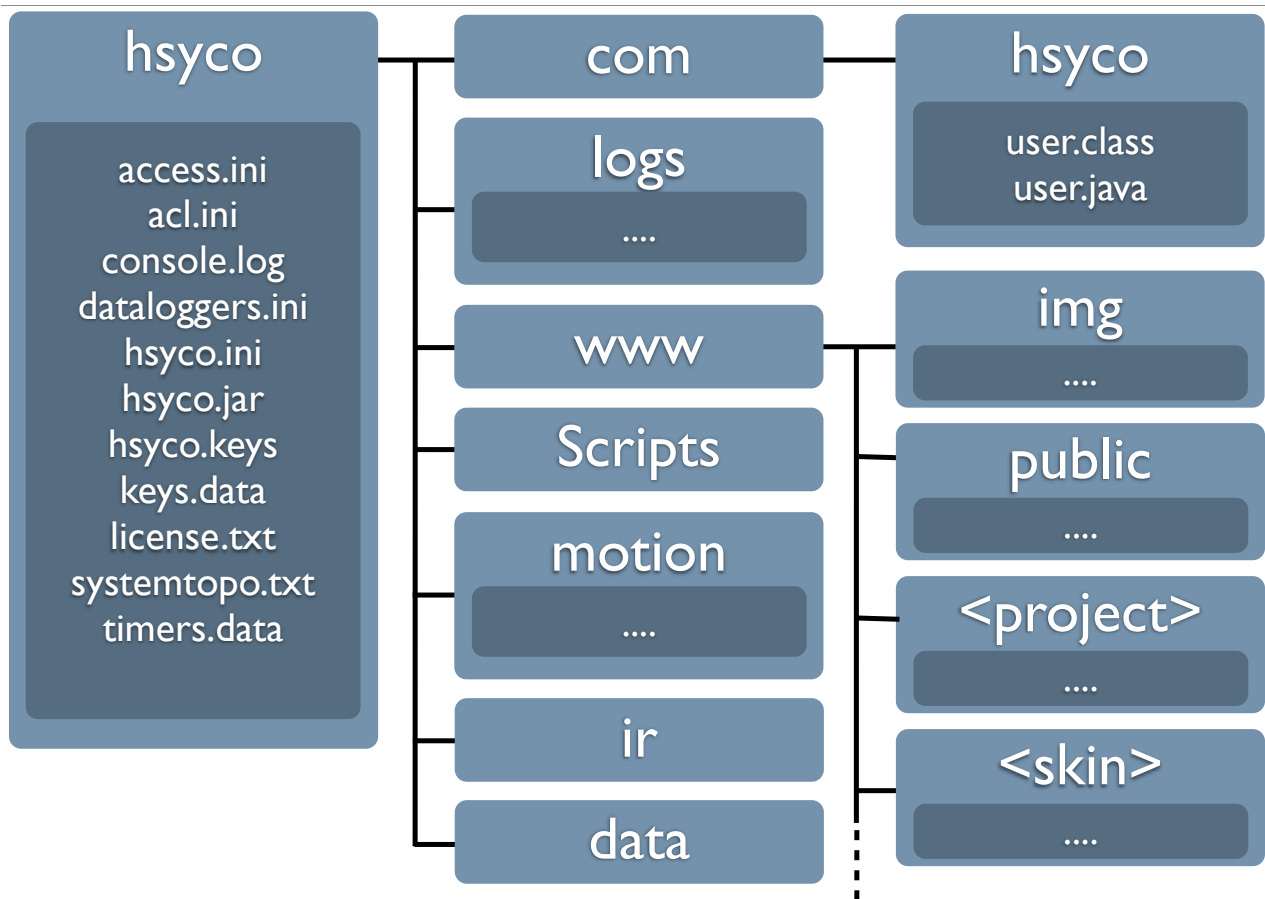
You can also temporarily select which log levels you want, without having to modify the hsyco.ini main configuration file.

Touch ,  or  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.

Touch  to toggle between text search and exclude mode.

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.

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 chapter in the Appendix 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)**¹. 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.

¹ For example Verisign or Thawte.

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** location zones².

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.

² That is, the list of the different Access Point devices that constitute the local wireless network.

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 hsyco.ini 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. Some of the most important log messages are listed in the Appendix chapter.

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
<i>datalogger_name</i>	String	specifies which data logger this slot is applied to
<i>slot_id</i>	Positive integer	identifier for the slot. More lines can refer to the same slot identifier
<i>time_range</i>	hh:mm-hh:mm	specifies the time range in which the slot applies. An asterisk (*) in this field will always result in a match
<i>days_of_week</i>	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
<i>date</i>	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
<i>rate</i>	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.

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. 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 started³.

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

³ HSYCO SERVER is based on a Linux Ubuntu 10.04 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.

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 3.0.0
#Sat Dec 10 11:49:06 CET 2011
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 3.0.0
#Sat Dec 10 12:20:00 CET 2011
name=Company Name
hwid=2080334b6283f450283bd4bbc3389bf66dc03dda
key=302D021500898B20F13ABE66DF3D2577809C6A53343A23A5CD0214540FD580
A1E3DA1CBEB18D11E98CAAE453D402BF
```

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**⁴ to access the console, and **Bitvise Tunnelier**⁵ and **WinSCP**⁶ 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**⁷ is one of the most widespread utilities to manage files through SFTP.

⁴ <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

⁵ <http://www.bitvise.com/download-area>

⁶ <http://winscp.net/eng/index.php>

⁷ <http://cyberduck.ch/>

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 ...
```

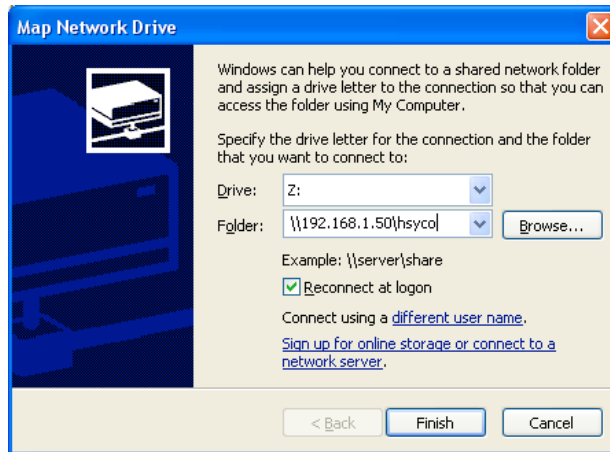
Access to Files

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.

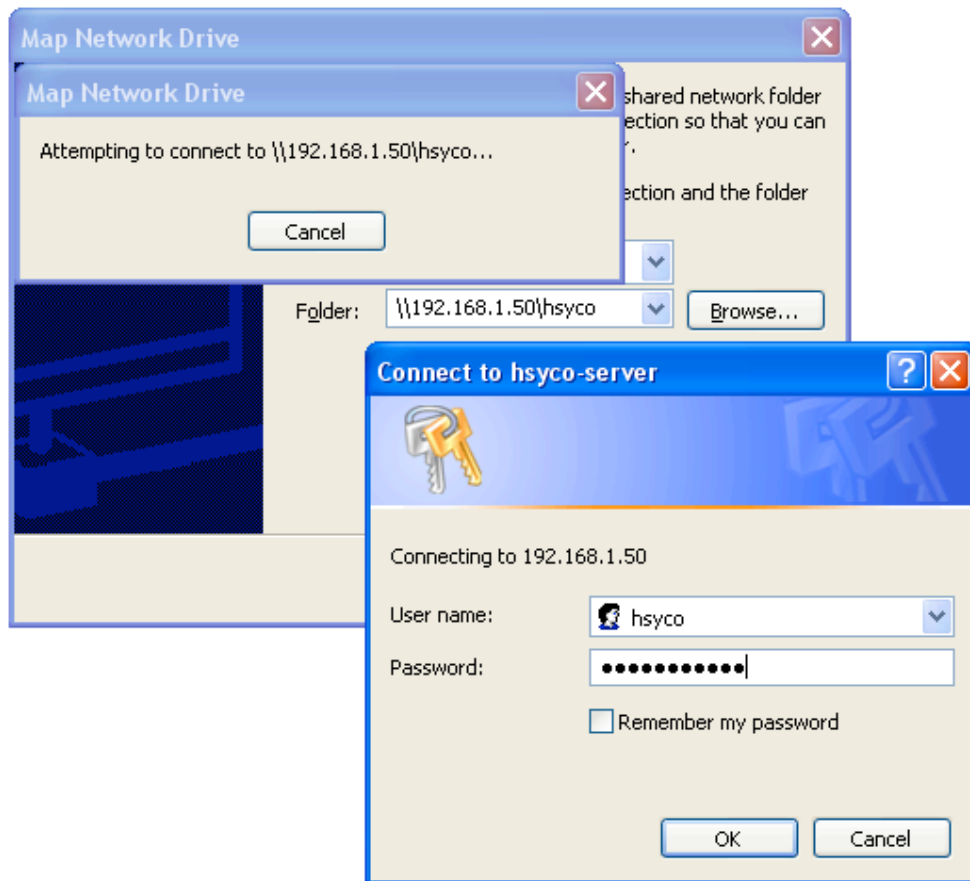
Access to Files on Microsoft Windows

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**.



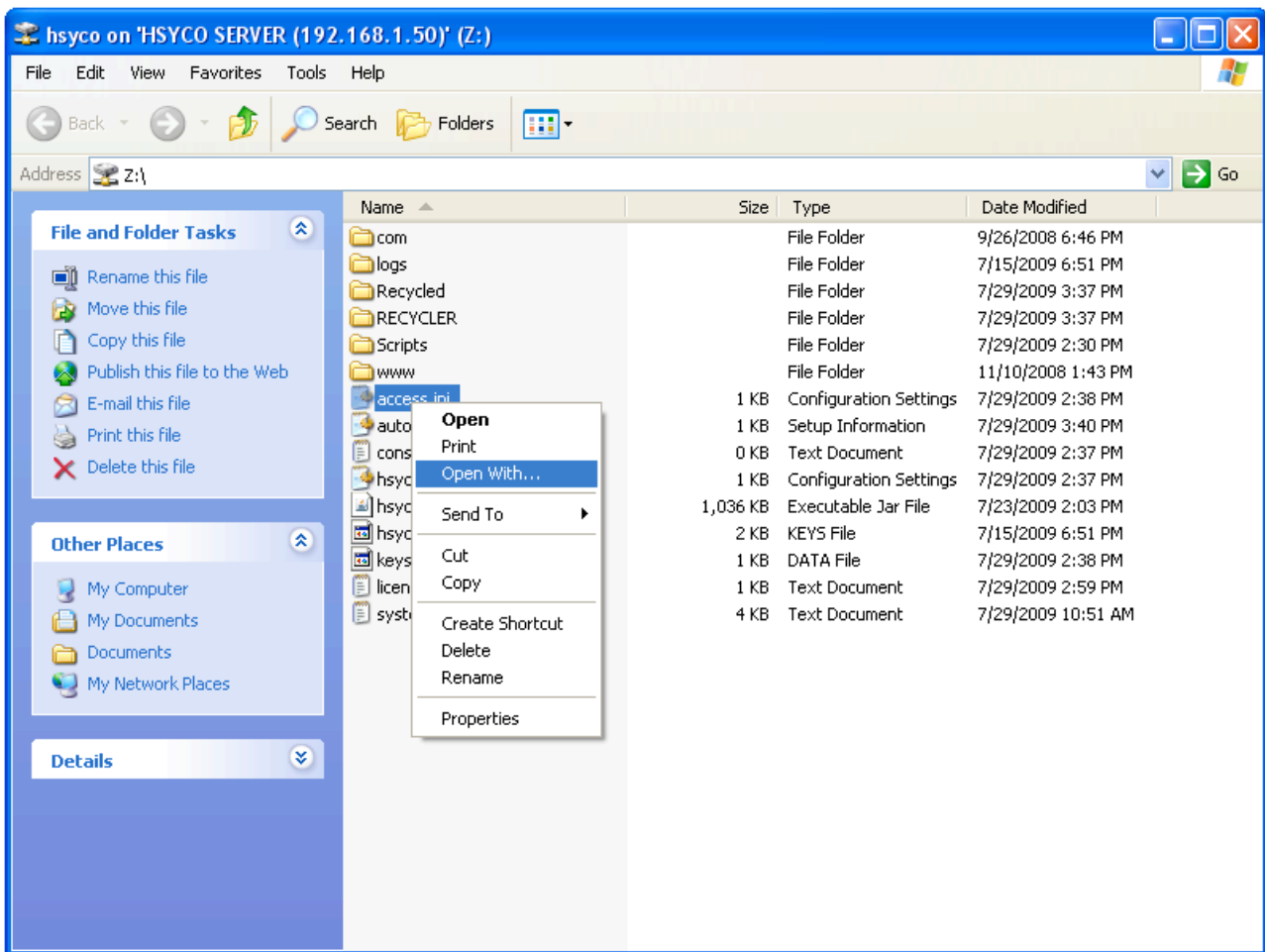
All the HSYCO files are available and accessible in the shared folder window, and you have full read and write permissions on them.

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 saving⁸ 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.

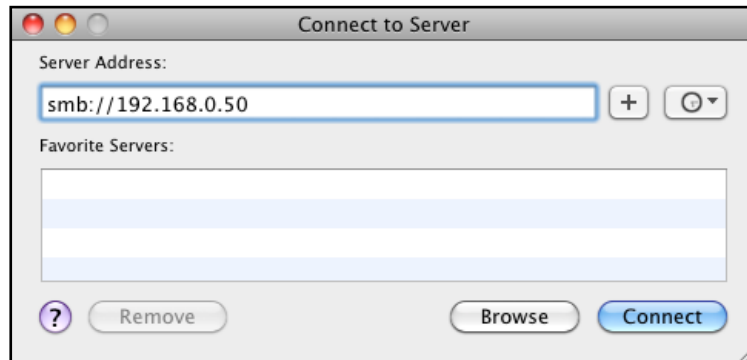
⁸ Do not select the *Remember my password* option in the window where the username and password are required.

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.



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*.

Configuration

The configuration of HSYCO is based on the ***hsyco.ini*** file.

It contains the general configuration parameters and all specific parameters related to the field devices.

Security and Users Management

The protection of any system from unauthorized access is extremely important. HSYCO uses several security mechanisms, and it is therefore extremely safe to use through the Internet or inside a local network.

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 `hsyco.ini` file.

The URLKey is not a secret password, but only an additional protection feature.

Therefore, it is necessary to modify some parameters of the `hsyco.ini` file that affect the basic aspects of the system's safety and users authentication.

hsyco.ini Initial Settings

The content of hsyco.ini before the configuration is the following:

```
verboseLog = false
eventsLog = true
trustedNet = local
KeysTrustedValidityHours = 100000
KeysNotTrustedValidityHours = 1
AutoKillFiles = hsyco.ini,hsyco.jar,com/hsyco/user.class
openServersDiscovery = true
HTTPServerPort = 80
HTTPSSLServerPort = 443
ServerName = hsyco
URLKey = hsyncoserver
SysLogServerPort = 514
Latitude = 45
Longitude = 10
```

All the existing parameters will be fully described later; the parameters which are strictly related to the original safety settings are **ServerName** and **URLKey**.

In **trustedNet** it is necessary to specify the IP addresses considered private and safe. 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 addresses given are considered fairly safe by HSYCO. A different session time-out can be defined for these addresses, and it is usually longer than that of any other Internet address.

ServerName specifies the name used to generate the **SSL** certificate, necessary for the cryptography of HTTPS Web traffic, and should correspond to the name of the domain through which the HSYCO server is accessible via the Internet.

The certificate is 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**. Otherwise, HSYCO simply uses the certificate contained in this file, which could have also been generated by an official **Certification**

Authority (CA). If you are not using an official certificate, this file is thus created and managed by HSYCO without any manual intervention.

If the name defined in `ServerName` is modified, HSYCO has to restart before the change becomes effective.

Before using HSYCO it is necessary to modify the **URLKey**, changing the default *hsyco* key into a new one, of at least 8 characters, composed by numbers and letters. It is possible to specify more than one key, separated by a comma; in this case all the specified URLKeys will be valid for the web access to HSYCO.

The URLKey must be at least 8 characters long. Shorter keys will be ignored. If your key is shorter, you will not be able to access HSYCO's Web pages.

Introduction to Field Systems

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 “IO Servers”. An IO Server is a driver 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.

Refer to the Application Notes documentation for detailed configuration and interfacing information for IO Servers.

The following table lists all systems that are currently supported as IO Servers.

System Type	Description
Airzone Innobus	Airzone Innobus HVAC control system
Aritech	Aritech Comfort CSx75 and Master ATS security panels
Aton	Aton AH66T multi-room music system
Axis Decoder	Control and streaming to the AXIS Video Decoder
BACnet	BACnet/IP client
Dali	Dali protocol via the Tridonic SCI2 interface module
Daikin VRF	Daikin's VRF systems with Intelligent Controller
Duemmegi Contatto	proprietary bus building automation system
Duemmegi Domino	proprietary bus home automation system
Ekahau Positioning Engine	Ekahau's RTLS server
EL.MO.	EL.MO. ETR100
Elsner P03/3 weather station	Elsner's Modbus version of the P03/3 weather station

System Type	Description
Epson ESC/VP projectors	Epson's Networked projectors that support the ESC/VP control protocol
GSM modem	standard GSM modems for SMS messages and voice call events
Guardall	Guardall QX32i, PX80 and PX500 security panels
HID Edge Solo	HID's Edge Solo stand-alone networked EdgeReader access control solution
HSYCO remote	access remote HSYCO servers data-points
HW Group I/O devices	HW Group's I/O Controllers and ER-xx devices
HW Group data acquisition devices	HW Group's STE, Poseidon and Damocles families
Interlogix FP2000	GE UTC Interlogix fire panels based on the FP2000 protocol
KNX/EIB	KNX/EIB protocol via the Weinzierl KNX/IP BAOS 771/772 gateway with Object Server 2.0
ISMG inverters	Carlo Gavazzi ISMG inverters
Mitsubishi M-NET	Mitsubishi's VRV systems that support the M-NET proprietary protocol
My Home	BTicino / Legrand My Home proprietary bus home automation system
Modbus TCP servers	HSYCO acts as Modbus TCP client and connects to servers configured as gateways to RS-485 Modbus RTU or Modbus ASCII slave devices
NuVo Concerto and Essentia	NuVo multi-room audio
PowerOne inverters	Aurora's PowerOne inverters with support for Aurora proprietary communication protocol
RayCONTROL	HVAC control system by FCC Planterm
T-Lab	T-Lab Web 144 security panels
Yamaha RXV A/V receivers	Yamaha's RXV-x067 family of networked A/V receivers
Weather Online	Internet-based weather conditions and forecast data acquisition using Yahoo!'s data

A special IO Server is the DUMMY server, that doesn't interface to any actual system, but simply simulates lighting and automation devices and can be used for demonstrations, prototyping and as a virtual IO Server to implement complex control logic.

A few systems are integrated with HSYCO not as IO Servers but with a dedicated API, either because of special requirements or for compatibility with older versions of HSYCO. The following table lists all systems that are currently supported via a specific API.

System Type	Description
Bentel KYO 320	burglar alarm control panel
DMX	DMX protocol via the KISS-BOX gateway
IP video cameras	IP video cameras, with JPEG and MJPEG video feeds, from AXIS, Mobotix, Panasonic, Samsung, Vivotek, Rovio and any IP camera that can serve individual JPEG frames or an MJPEG stream via HTTP
IRTrans infrared emitters/receivers	networked IR emitters/receivers
Netstreams Musica	Netstreams Musica multi-room audio
Paradox EVO 48/192	burglar alarm control panel
RS-232	the local RS-232 serial ports on HSYCO Server, and remote ports through IP-based serial gateways
Squeezebox	Logitech's Squeezebox network music players
Tecnoalarm Tecno-Out	burglar alarm control panel
Wi-Fi Access Points	Wi-Fi APs from several vendors, used to provide Wi-Fi clients' basic location informations

The configuration settings for most of these systems are in the `hsyco.ini` configuration file. Some systems also require their own configuration files.

The hsyco.ini File

This is HSYCO's main configuration file.

Any change to this file causes the automatic restart of HSYCO after a few seconds, making all the changes effective.

The general format of hsyco.ini is:

parameter = value

Comment lines can be added, and will be ignored. A comment line is identified by the “#” character at the beginning of the line.

You can separate the parameter lines with empty lines to make the file easier to read.

Access Control

Name	Default	Format
URLKey	hsycoserver	string of at least 8 characters

Before starting HSYCO it is necessary to modify the URLKEY, changing the default hsyncoserver key with a different one, of at least 8 characters, composed by numbers and letters. 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

Name	Default	Format
WebAdminNetConfigLock	false	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

Name	Default	Format
trustedNet	local	nn.nn.nn.nn-nn.nn.nn.nn local

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`

Name	Default	Format
KeysTrustedValidityHours	24	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

Name	Default	Format
KeysNotTrustedValidityHours	1	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

Name	Default	Format
KeysInactivityHours		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

Name	Default	Format
KeysInactivityMode	browser	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

Name	Default	Format
HTTPServerLowSecurityEnabled	false	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

Nome	Default	Formato
HTTPServerPublicDirectory		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

Log Levels

Name	Default	Format
eventsLog	false	true or false
if true, the log of events received from field devices is enabled, for example the events related to the IO Servers		

Name	Default	Format
userLog	false	true or false
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		

Name	Default	Format
verboseLog	false	true, false or silent
<p>if true, the extended log is enabled.</p> <p>It is useful for debugging, or during the advanced customization phase or the development of Java code.</p> <p>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</p>		

High Availability

Name	Default	Format
haMode		master or slave
<p>Enables the high availability configuration of HSYCO.</p> <p>The high availability configuration uses two different HSYCO systems, one is configured as the master system, the other as slave</p>		

Name	Default	Format
haMasterIP	mandatory	address IP: nn.nn.nn.nn
<p>mandatory parameter for the high availability configuration. It specifies the IP address of the HSYCO master system</p>		

Name	Default	Format
haSlaveIP	mandatory	IP address: nn.nn.nn.nn
<p>mandatory parameter for the high availability configuration. It specifies the IP address of the HSYCO slave system</p>		

Name	Default	Format
haDisableCommandEvents	mandatory	true or false
<p>mandatory parameter for the high availability configuration.</p> <p>If true, the method <i>irtransCommand()</i> and the IR action in EVENTS, and commands sent to IO Servers that support the high availability configuration 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).</p> <p>Sending of DMX commands from the inactive system, generated either by user.java or events.txt and by the Web interface is always disabled, even when this parameter is set to false</p>		

Network

Name	Default	Format
HTTPServerPort		integer number < 65535
TCP/IP port of HSYCO HTTP Web Server. If not defined, the HTTP server is not enabled. Normally set to 80		

Name	Default	Format
HTTPSSLServerPort		integer number < 65535
TCP/IP port of HSYCO HTTPS Web Server. If not available, the HTTPS server is not enabled. Normally set to 443		

Name	Default	Format
OffLineCache	false	true or false
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		

Name	Default	Format
ServerName	hsyco	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 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		

Name	Default	Format
SysLogServerPort		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		

Clock

Name	Default	Format
Latitude	0	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		

Name	Default	Format
Longitude	0	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		

Name	Default	Format
SunriseOffsetMinutes	0	positive or negative integer number
deviation in minutes from the astronomical sunrise time (in advance if negative, delayed if positive)		

Name	Default	Format
SunsetOffsetMinutes	0	positive or negative integer number
deviation in minutes from the astronomical sunset time (in advance if negative, delayed if positive)		

Name	Default	Format
TimeAutoUpdate	true	true, false, or 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

Name	Default	Format
TimeZone		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:

```
2012.07.10 15:28:30.113 - HSYCO Ver. 3.1.0 Build 0097 (USER Ver. No User Code) started
2012.07.10 15:28:30.425 - Time zone is: Europe/Rome (Central European Summer Time)
```

DMX Gateway

Name	Default	Format
dmxServers		comma separated list of IDs (only letters and numbers)
<p>list of identifiers associated to the DMX512 gateways. The IDs must be unique, and different from the IO Servers IDs.</p> <p>For each ID, you should also define the <i>dmxServersIP.id</i> parameter</p>		

Name	Default	Format
dmxServersIP.id	mandatory for KissBox	nnn.nnn.nnn.nnn
<p>IP address of the KissBox DMX gateway. Replace <i>id</i> with one of the names defined in the <i>dmxServers</i> parameter. Mandatory for each ID defined in <i>dmxServers</i></p>		

Name	Default	Format
dmxServersPort.id	mandatory for KissBox	positive integer < 65535
<p>TCP/IP port of the KissBox DMX512 gateway</p>		

Name	Default	Format
dmxServersComm.id	mandatory for ENTTEC	string
<p>Virtual serial port name assigned to the ENTTEC DMX USB PRO DMX gateway. See the ENTTEC DMX USB PRO Application Note for additional information</p>		

Cameras

Name	Default	Format
Cameras		comma separated list of IDs (only letters and numbers)

this parameter is necessary to acquire images from IP cameras and contains the list of identification names of the cameras. The IDs must be unique, and different from the IO Servers IDs.

For each identification name you also have to define the mandatory *Camera.id.URL* parameter, and a few other optional parameters

Name	Default	Format
CamerasRecordingMotionTriggerSeconds	5	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

Name	Default	Format
CamerasRefreshMillis	1000	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

Name	Default	Format
CamerasResizedQuality	0.7	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

Name	Default	Format
Camera.id.URL	mandatory	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`

Name	Default	Format
Camera.id.URL.Small		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

Name	Default	Format
Camera.id.PTZ	0	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)

Name	Default	Format
Camera.id.Type		string (letters and numbers)
defines the camera model type for cameras that support this option. For example, <i>Camera.samsung.Type = SND-5080</i>		

Name	Default	Format
Camera.id.IO		<i>enabled</i>
for some camera models, you can enable I/O and VA (Visual Analysis) option for a camera. For example, <i>Camera.<camid>.IO=enabled</i> . If you want to enable I/O only, set it to “ <i>enabled:io</i> ”, or to “ <i>enabled:va</i> ” to enable VA only		

Name	Default	Format
Camera.id.User		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		

Name	Default	Format
Camera.id.Password		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		

Name	Default	Format
Camera.id.DroppedFrames	0	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		

Name	Default	Format
Camera.id.MaxAge	30d	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

Name	Default	Format
Camera.id.MotionBuffer	0	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

Nome	Default	Format
Camera.id.RemoteRequest Password		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

Nome	Default	Format
Camera.id.TrustedRequestPassword		string (letters and numbers)
<p>Similar to RemoteRequestPassword, you can use this optional parameter to set remote passwords for clients inside the trusted range of IO addresses.</p> <p>Note that requests with the TrustedRequestPassword value will be also served via HTTP, without SSL</p>		

Name	Default	Format
Camera.id.Rotate	0	integer number
<p>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</p>		

Name	Default	Format
CameraGrid.id		list of cameras id
<p>with this parameter it is possible to specify up to 99 different combinations of cameras to create matrix displays of multiple cameras. The id should be starting from 1 and up to 99 with consecutive numbers.</p> <p>For each grid, a list of cameras needs to be specified line by line from left to right and from top to bottom of the display grid, separating the cameras IDs on the same line using spaces and the rows using a comma.</p> <p>For example, to create a 2X2 matrix:</p> <p style="text-align: center;"><i>CameraGrid.1 = c1 c2, c3 c4</i></p> <p>and for a 3X3 matrix:</p> <p style="text-align: center;"><i>CameraGrid.2 = c1 c2 c3, c4 c5 empty, c6 c7 c8</i></p> <p>The word <i>empty</i> can be used to replace a normal ID to identify an empty position</p>		

Name	Default	Format
CameraGrid.<i>id</i>.Resolution		<i>width X height</i> 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. For example, to set a maximum resolution of 640x480 pixels for grid 3:

CameraGrid.3.Resolution = 640x480

Nome	Default	Format
CameraGrid.<i>N</i>.RemoteRequestPassword		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

Nome	Default	Format
CameraGrid.<i>N</i>.TrustedRequestPassword		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

I/O Servers

Name	Default	Format
ioServers		comma separated list of IDs (only letters and numbers)
list of identifiers associated to the I/O Servers. You can use any unique name. For each id, you should also define the <i>ioServersIP.id</i> and <i>ioServersType.id</i> parameters		

Name	Default	Format
ioServersIP.id	mandatory	nnn.nnn.nnn.nnn
IP address of an I/O Server. Replace <i>id</i> with one of the IDs defined in <i>ioServers</i> . Mandatory for each ID defined in <i>ioServers</i>		

Name	Default	Format
ioServersPort.id		positive integer < 65535
TCP/IP port of the I/O Server. Use this parameter to specify a specific port number for each server		

Name	Default	Format
ioServersComm.id		string (letters and numbers)
the communication port id used by the I/O Server. Used only when <i>ioServersIP</i> is not defined		

Name	Default	Format
ioServersType.id	mandatory	interface type code
type of the I/O Server. See the Application Notes for available types. Mandatory for each I/O Server		

Name	Default	Format
ioServersOptions.id		
some I/O Server could require additional configuration parameters, as defined in the Application Notes		

Name	Default	Format
ioServersAuth.id.User		string (letters and numbers)
some I/O Server could require additional access control parameters, as defined in the Application Notes		

Name	Default	Format
ioServersAuth.id.Password		string (letters and numbers)
some I/O Server could require additional access control parameters, as defined in the Application Notes		

Remote HSYCO Servers

Name	Default	Format
RemoteServerPassword		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)		

Name	Default	Format
RemoteServerAddress		comma separated list of IP addresses
set to the IP addresses of the remote HSYCO servers that should be allowed to access this HSYCO server		

Name	Default	Format
RemoteServerControl	false	true or false
set to true to allow the remote server to modify the data points values of this HSYCO server		

Serial Communication Ports

Name	Default	Format
CommPorts		comma separated list of IDs (only letters and numbers)

list of identifiers associated to the serial ports. You can use any unique name. For each id, you should also define the *CommPort.id.Id*, *CommPort.id.Type* and *CommPort.id.Params* parameters.

If both verboseLog = true and userLog = true, the full trace of received and sent bytes is written to the log file

Name	Default	Format
CommPort.id.Id	mandatory for each name defined in CommPorts	system name of the communication port

complete or partial system name of the associated communication port. For example ttyS0 or /dev/ttyS0.

For serial ports based on I/O network servers, use the ID of the I/O Server (ad defined in the *ioServers* list).

For the serial ports of the IRTrans devices, use the id of the IRTrans server (ad defined in the *IRTrans* list)

Name	Default	Format
CommPort.id.Type	mandatory	“serial” “server” “io” “irtrans”

defines the driver name, corresponding to the type of serial port device. For local serial ports, set the value to “serial”, and set the CommPort.id.Params with the appropriate handshake and speed parameters. Mandatory for each name defined in CommPorts.

For remote serial ports using a supported serial to IP converter, set the value to “server”, and the CommPort.id.IP and CommPort.id.Port parameters with the converter’s IP and port configuration.

For serial ports based on I/O network servers, set the value to “io”, and set CommPort.id.Id to the id of the I/O server.

For the serial ports of the IRTrans devices, set the value to “irtrans”, and set CommPort.id.Id to the id of the IRTrans server

Name	Default	Format
CommPort.<i>id</i>.Params	mandatory for each name defined in CommPorts, types “serial” and “io”	communication parameters, comma separated

defines the communication parameters for each serial port. The format is: speed, bits, stop, parity, flow control.

The parameters must be set to values supported by the drivers and the hardware of each port.

The speed is expressed as an integer number (baud rate). The number of bits can be 8, 7, 6 or 5. The number of stop bits can be 1, 1.5 or 2. The parity can be 0 (no parity), 1 (odd parity), 2 (even parity), 3 (mark parity) or 4 (space parity). The flow control can be 0 (no control), 1 (XON/XOFF) or 2 (RTS/CTS).

This parameter is not used for serial ports types “ir” and “server”

Name	Default	Format
CommPort.<i>id</i>.IP	mandatory for each comm port ID defined as “server” type	nnn.nnn.nnn.nnn or, for failover configurations: nnn.nnn.nnn.nnn, mmm.mmm.mmm.mmm

IP address of the serial IP gateway. Replace *id* with one of the IDs defined in *CommPorts*

Name	Default	Format
CommPort.<i>id</i>.Port	mandatory for each comm port ID defined as “server” type	positive integer < 65535

port number of the serial IP converter. Replace *id* with one of the IDs defined in *CommPorts*

Name	Default	Format
CommPortsList	optional	comma separated list of local serial port system names

Name	Default	Format
this 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: CommPortsList=/dev/ttyS0,/dev/ttyS1		

IRTrans

Name	Default	Format
IRTrans		comma separated list of IDs (only letters and numbers)
<p>list of identifiers associated to the IRTrans modules. You can use any unique name.</p> <p>The IRTrans are used to send and receive IR commands (infrared).</p> <p>For each id, the <i>IRTransIP.id</i> parameter should also be defined</p>		

Name	Default	Format
IRTransIP.id	mandatory for every IRTrans module defined	nnn.nnn.nnn.nnn
<p>IP address of the IRTrans module.</p> <p><i>id</i> is one of the identifiers defined in the <i>IRTrans</i> list</p>		

Squeezebox

Name	Default	Format
slimServerName		nnn.nnn.nnn.nnn
IP address of the Squeezebox Server to control the Squeezebox music players		

Name	Default	Format
slimPlayers		comma separated list of IDs (only letters and numbers)
list of identifiers associated to the Squeezebox modules. You can use any unique name. For each listed item, the <i>slimPlayerId.id</i> parameter must also be defined		

Name	Default	Format
slimPlayerId.id	mandatory for each Squeezebox	identification string of the music player provided by the SqueezeCenter server
identification string of each Squeezebox, assigned automatically or defined on the Squeezebox Server. It is usually the MAC Address of each player, in the <i>hh:hh:hh:hh:hh:hh</i> format, where hh is two digits 0-9 and/or letters a-f. Replace <i>id</i> with one of the IDs defined in the <i>SlimPlayers</i> parameter		

Public Announcement

Name	Default	Format
AudioServerTTS		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 Public Announcement Appendix for additional information		

Name	Default	Format
AudioServerVolume		integer number
change the default output volume for pre-recorded audio files and the text-to-speech engine		

Name	Default	Format
AudioServerQuality		integer number
change the default quality for the text-to-speech engine		

Name	Default	Format
AudioServerSpeed		integer number
change the default speed for pre-recorded audio files and the text-to-speech engine		

Timers

Name	Default	Format
Timers		comma separated list of IDs (only letters and numbers)
this parameter contains the list of identification names of the user timers. A user timer can be set through the Web interface, and will generate on/off events based on its schedule. The IDs can be arbitrary, but should be unique		

Location Services

Name	Default	Format
LocationBases		comma separated list of IDs (only letters and numbers)

this parameter is required for the the location services of mobile devices used to access the HSYCO Web pages through the Wi-Fi LAN. Lists the IDs names of the Wi-Fi access points. You can use any unique name. For each item in the list, you should also define the *LocationBaseIP.id* parameter. These names are also used in the systemtopo.txt file to associate each location with a user friendly description

Name	Default	Format
LocationBaseIP.id	mandatory	nnn.nnn.nnn.nnn
IP address of the Access Point. Mandatory for each name defined in LocationBases		

Data Loggers

Name	Default	Format
DataLoggers		comma separated list of IDs (only letters and numbers)
list of IDs for each data logger. You can use any unique name. For each id, you should also define the <code>DataLoggers.id.Type</code> parameter		

Name	Default	Format
DataLoggerCsvSeparator	comma	tab comma semicolon
this parameter defines the field separator type for CSV files generated by the data loggers		

In the following tables replace *id* with one of the names defined in DataLoggers.

Name	Default	Format
DataLoggers.id.Type	mandatory	counter range
type of the data logger, use “counter” or “range”		

Name	Default	Format
DataLoggers.id.UseCharts	false	true or false
if you want a data logger to set the discrete UI objects for charts and totals, set this parameter to true. Set it to false if you are displaying the data logger’s data using the (datalogger) object or for data acquisition only		

Name	Default	Format
DataLoggers.id.Decimals	1	integer number
the number of decimal digits to be used for the gathered data		

Name	Default	Format
DataLoggers.id.HourInterval	1	integer number
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		

Name	Default	Format
DataLoggers.id.CounterUpperLimit	0	float number
option for data loggers of type counter. 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		

Name	Default	Format
DataLoggers.id.CounterMaxDelta		float number
option for data loggers of type counter. 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		

Name	Default	Format
DataLoggers.id.SeparateCharts	false	true false
option for data loggers of type counter using time slots. Specifies whether or not to use different monthly and yearly charts for each defined time slot		

Name	Default	Format
DataLoggers.id.SlotAlign	true	true false
option for data loggers of type counter using time slots with separate charts. Specifies whether or not to align the scale of the slot charts belonging to the same period		

Name	Default	Format
DataLoggers.id.RatesLogFile		file path. e.g.: hsyco/ rates.csv
option for data loggers of type counter 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		

Name	Default	Format
DataLoggers.id.PeriodAlign	true	true false
option for data loggers of type range. Specifies whether or not to align the scale of the charts (min, max, and avg) belonging to the same period		

Name	Default	Format
DataLoggers.id.Origin	0	float number
option for data loggers of type range. Specifies the origin of the charts		

Name	Default	Format
DataLoggers.id.Range		range: low:up
<p>option for data loggers of type range. 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.</p> <p>If the parameter is omitted, the charts will range from the lowest processed value to the highest one</p>		

Name	Default	Format
DataLoggers.id.OutOfRangeMode	cut	cut ignore
option for data loggers of type range with a static 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		

Email

Name	Default	Format
SmtpName		SMTP server name or IP address
<p>SMTP server name or numeric IP address.</p> <p>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</p>		

Name	Default	Format
SmtpPort	25, 465 or 587	positive integer number
<p>SMTP server's port number, if different from the default port.</p> <p>The default port is:</p> <ul style="list-style-type: none"> • 25 for not encrypted traffic • 465 for SMTP over SSL • 587 for ESMTP 		

Name	Default	Format
SmtpUser		string
account's username to authenticate the connection to the SMTP server		

Name	Default	Format
SmtpPassword		string
account's password to authenticate the connection to the SMTP server		

Name	Default	Format
SmtpSSL	false	false true esmtp
<p>set to "true" for SMTP server that support SMTP over SSL (default port is 465).</p> <p>Set to "esmtp" for SMTP server that support ESMTP (default port is 587).</p> <p>Set to "false" for SMTP servers that don't support traffic encryption.</p>		

Additional Parameters

Name	Default	Format
AutoKillFiles		List of files' names separated by commas
<p>this parameter is usually set as follows:</p> <p><i>hsyco.ini,hsyco.jar,com/hsyco/user.class</i></p> <p>forcing the automatic restart after the changes of the three files listed</p>		

Name	Default	Format
Language	en	en fr it
<p>Some I/O Servers and other core services use localized text messages. This parameter defines the default system language for these services</p>		

Customization

Customizing HSYCO basically means adapting the system to the network of devices connected to it, and developing a specific Web interface that controls these devices.

HSYCO has a few standard graphic formats, called skins, optimized to offer a uniform and effective interface on most popular Web browsers for personal computers and mobile devices.

The quickest and easiest way to customize HSYCO, which does not require any particular skill in programming or HTML pages development, offers nonetheless freedom of choice in the organization of the interface and allows to personalize a complex system in a very short time.

The configuration of HSYCO is exclusively based on normal text files, which can be easily modified with any text editor on a PC; no other dedicated software is required for the configuration activity.

It is also possible to create alternative skins that differ from the standard ones, or work directly at the HTML level and possibly JavaScript to have the highest flexibility when creating the user interface.

This chapter explains the details of the configuration based on the standard skins first, and an introduction to some of the aspects of lower level configuration.

The 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 will 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, for example when HSYCO is only used for video-surveillance, to manage Squeezebox, or for IR virtual remote control. 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>; ; <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 configuration⁹ of the actuator associated to the bus address.

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*, *HSHT*, *LOCK*, *ONOFF*, *SCENE*, corresponding to the graphics shown below¹⁰ when using the standard “blue” skin.

Object Type							
Status	LIGHT	STD	VSHUT	HSHT	LOCK	ONOFF	SCENE
ERROR							
OFF							
ON							
UP							
DOWN							
GOING UP							
GOING DOWN							
UNKNOWN							

The **** field is optional. The following line has therefore a valid format:

k.12 : AUTOM; VSHUT

⁹ For the actuators configured in interlocking mode, it must be set to AUTOM.

¹⁰ 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.

The `` field can be used to associate an image or a camera to the device¹¹. 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
```

¹¹ 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.

The Web Pages

Web pages are defined using a simple, text-based description format that allows you to define multiple pages and the navigation links in a single file, called ***index.hsm***. You can have several index.hsm files, each file in its own subdirectory inside the www directory. This way you can implement different navigation layouts, functions and graphics for each user or group of users.

It is not possible to have more description files of the Web interface within the same directory.

Whenever HSYCO detects any change to the files in www and in its sub-directories, it forces the reload of the Web page 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>

You can create and edit pages by manually typing the index.hsm files, but in most cases you will rather use the Project Editor.

The Project Editor

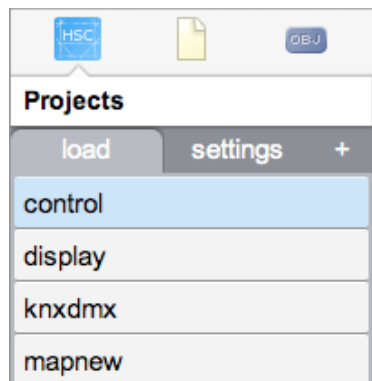
The Project Editor is one of the Manager's applications, and a very powerful Web-based visual editor you can use to create the HSYCO Web interface.

To access the Manager, enter the following URL in your Web Browser:

```
https://192.168.0.50/hsycoserver/manager
```

then touch the Project Editor icon.

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 or 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, touch the + character to show a blank project's settings form.

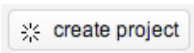
Here you should enter a unique name for this project, this is used for both the path under www and the URL to access the project from the Web browser.

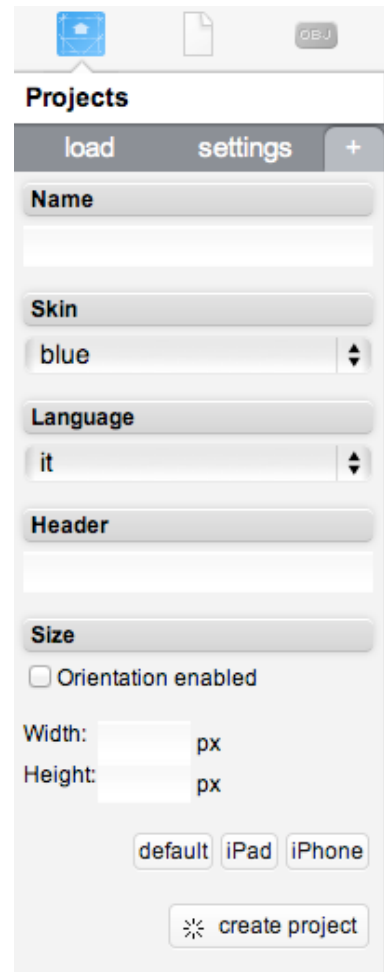
The “blue” skin is automatically selected. This is the standard skin in HSYCO 3.0. If you have other custom skins, they will appear in the Skin selector.

The Language selector offers the languages that the skin supports, Italian, English and French for the blue skin. 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).

We'll discuss the size options in more detail in the next chapter.

When done, touch the  button to confirm the new project. The Project Editor will create a directory for this project with the index.hsm skeleton file.



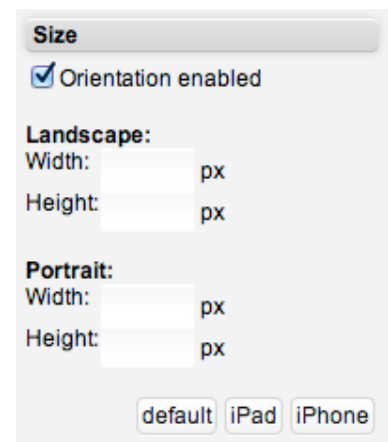
Page Size and Coordinates

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, touch 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

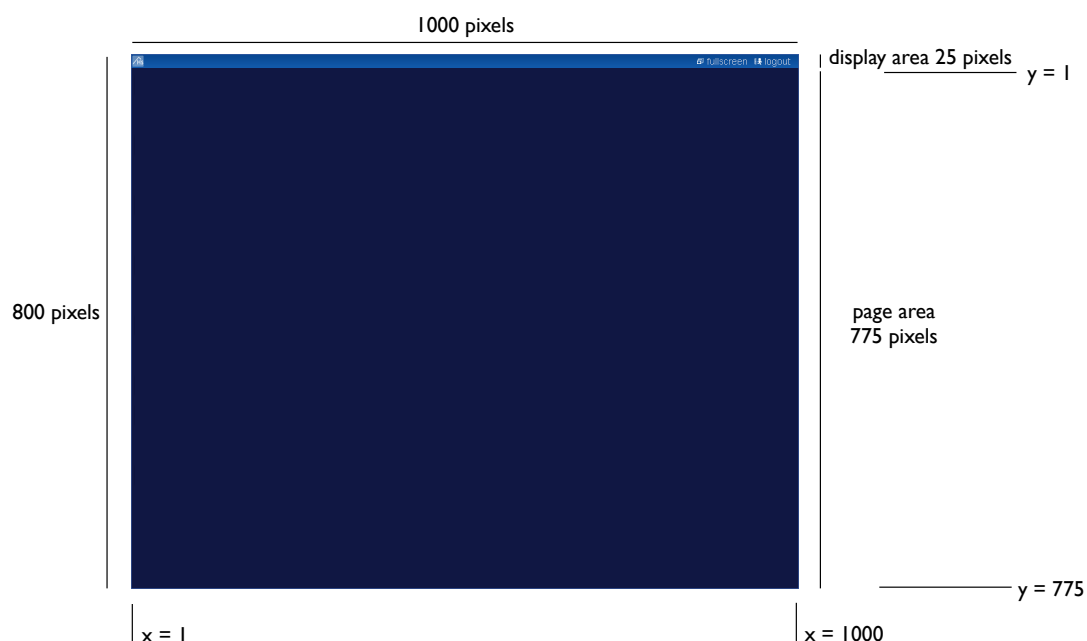
width, as you would usually do for desktop or portable traditional computers, or a larger height, that normally fits the iPhone, iPod touch or other mobile phones better.

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 Web objects:

- **X-Y pixel coordinates.** For example, x100y200 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.** r12c14 places the object at row 12, column 14 based on the reference grid. Because variable size skins can have more than 9 rows and columns, you have to use the letters R and C (uppercase or lowercase) to specify rows and columns. r1c1 represents the top-left position in the grid. The reference grid depends on the size of the object using it.

Editing a Project

Touching the settings tab of an existing project, you have access to a 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.

The screenshot shows a vertical list of settings for a project. Each setting has a title bar and a control element:


- Device Image:** A dropdown menu currently showing "enabled".
- Kiosk:** A dropdown menu currently showing "disabled".
- Scale:** A text input field, currently empty.
- Locked:** A section header with a checkbox labeled "Project locked", which is currently unchecked.
- Background:** A text input field with a three-dot menu icon to its right.
- Camera List:** A text input field containing the text "Front", "Back", "Walk" with a three-dot menu icon to its right.
- Camera Overlay:** A text input field with a three-dot menu icon to its right.
- Camera Grid Headers:** A text input field with a three-dot menu icon to its right.

Selecting “Project locked”, disables any action that could be sent pressing buttons and commands in this project’s Web interface.

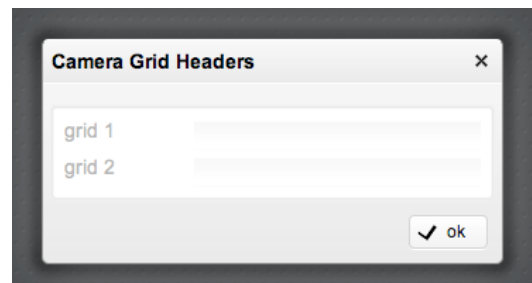
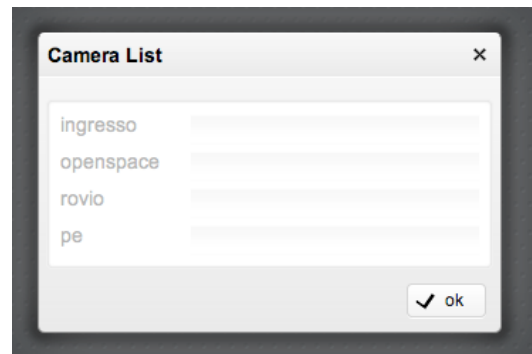
The Background parameters is used to select a custom image as the general background for all pages in this project.

Touching 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.

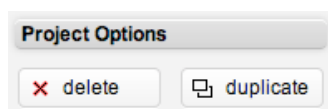
Just leave the description text field blank if you don’t want to show a camera in this project.

Touching 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.

Finally, just like the Camera List section, Camera Grid allows you to enter friendly descriptions for camera grids defined in hsyco.ini.



At the bottom of the panel you have buttons to delete or duplicate the project.



Menus, Pages and Pop-Ups

We have three types of pages in the Web interface, menus, regular pages and pop-ups. To create a new page, touch the pages tab, then the + character.

Menus are just like ordinary pages, but a menu is also the home page of a project. You can have only one menu in a project, or actually two, considering the landscape and portrait versions of it for projects with orientation enabled.

Touch menu to create a menu, then select the orientation, if enabled. There are no additional parameters for menus.

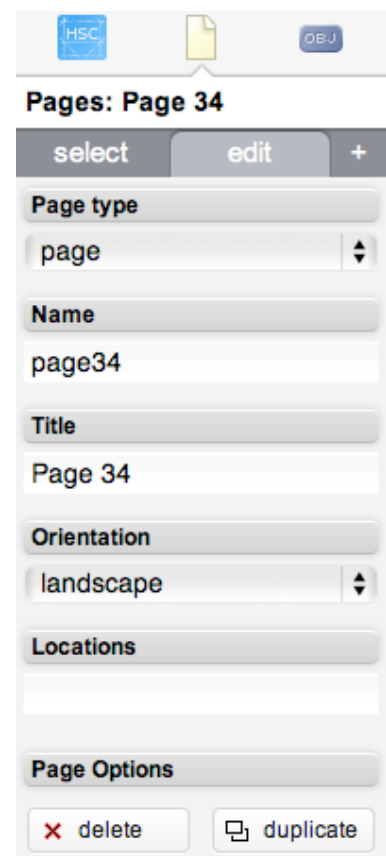
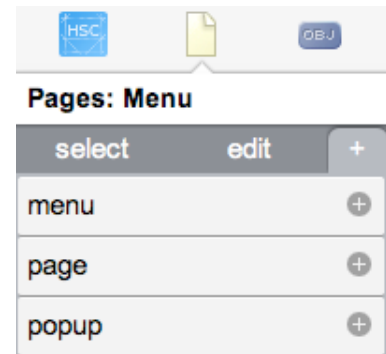
Touch page to create a new page, or pick a page from the selection list to edit.

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.

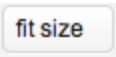
A page can be deleted or duplicated to an identical page, with a different name (ID) using the two buttons at the bottom of the panel.

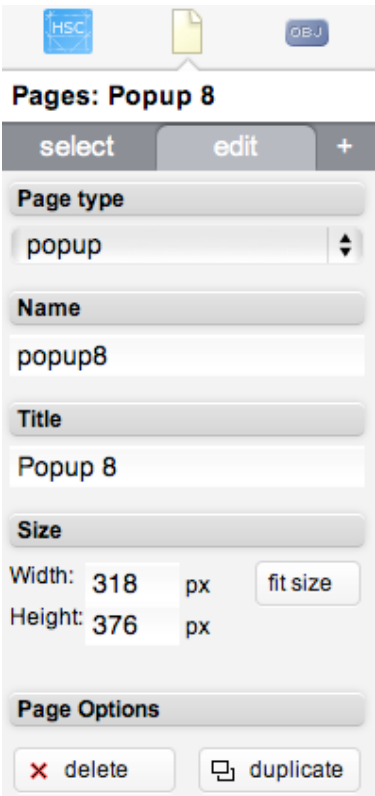


Touch popup to create a new pop-up.


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 pop-up, and also set its width and height in pixels.

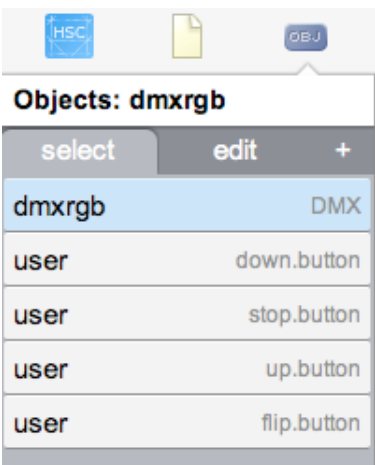
The  button automatically adjusts the pop-up's size based on the objects that you added to the pop-up.



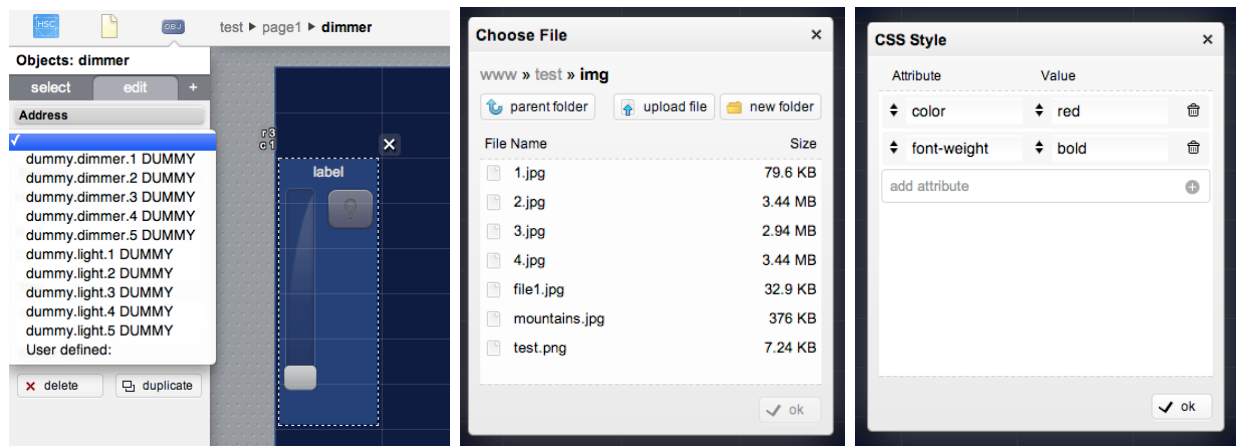
Page's Objects

You can add and edit graphic and control objects in a page. Touch the  button and you will see the select tab with the list of all objects in the page. Touch any object in the list to highlight it in the page, then touch the edit tab to edit its attributes.

Touch the + character 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.

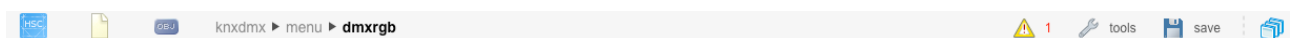


When you have an object highlighted, go to the select tab to change the order of the object in the page. You can move an object up or down the list.



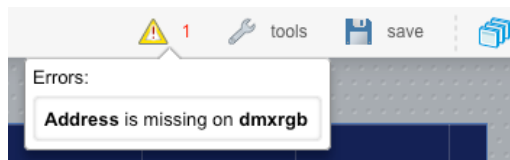
Objects on top will graphically appear above others in the Web page.

You have a top toolbar with functions that are specific of the Project Editor.

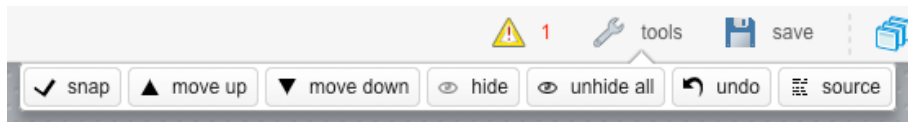


Next to the projects, pages, and objects tabs, see the navigation hierarchy, with project's name, current page name, and selected object. You can click these items to quickly jump back to the objects' listing or pages' listing.

A warning icon appears if there are errors in the page, like objects with missing parameters. Touch the icon to list all warnings, then touch a warning message to highlight the object and show its attributes.



The tools icon shows a number of useful editing functions.



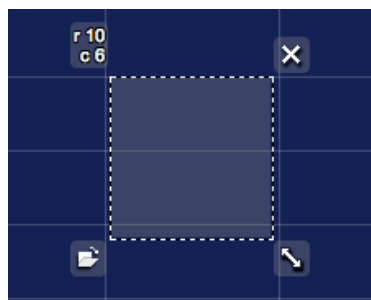
Snap enables or disables the snap to grid function. Undo allows you to revert changes you've made since last save.

Finally, the hide button hides an object from view in the Project Editor, not affecting the actual Web 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. Touch the unhide button to show all objects again.

Containers

A container is not a real object. It 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, touch the + character, then choose container from the list.



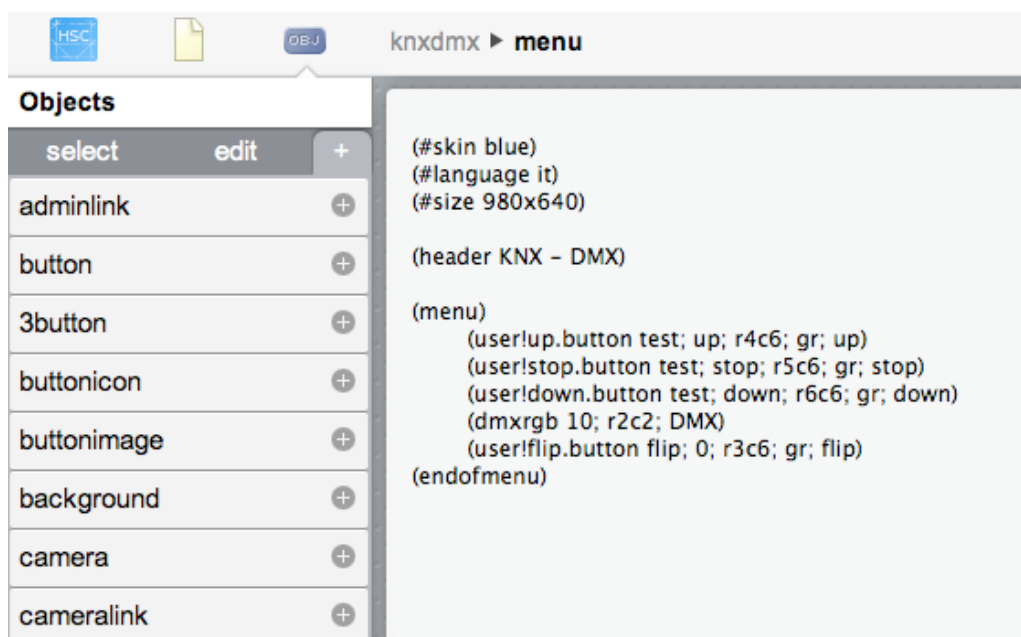
A container box appears in the page. You can resize it by touching the bottom right corner and dragging, or open the container touching the bottom left corner.



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 index.hsm project's source file

Touching source in the tools bar, the Project Editor will show the index.hsm source text.



Using the File Manager you can open it for editing and directly modify the project's content working on text. Going back to the Projects Editor, if that project was open, you will be asked to reload the changes.

Objects Reference

Objects

The index.hsm file is based on graphic objects having the following standard format:

(object_name <parameter1>; <parameterN>)

Errors in the index.hsm file are reported in the log file with the message:

Page Parser: Syntax Error, line: nnn

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 dynamic modification of the text, visibility, color several other attributes with the `uiSet()` Java method or the UISET action.

The identified objects should be defined with an id string that is unique in the `index.hsm` file. The presence of more objects with the same id can cause undesired effects.

To identify an object, append a `!` character and the id at the end of the object type, for example:

```
(text!mytextid r1c1; This is my identified text object)
```

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.

(#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 use 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.

Main Menu**(<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.

Parameters:

<title> - the title of the page.

(menu)

Defines the beginning of the main menu page, shown after the login. You can only have one menu page in the index.hsm file.

(menu#landscape)

Defines the landscape orientation version of the menu page. On devices that support orientation, you can create distinct portrait and landscape versions of the menu page. The appropriate version is automatically used based on the physical orientation of the device.

(menu#portrait)

Defines the portrait orientation version of the menu page.

(endofmenu)

Defines the end of the main menu page.

Pages

(link <pos>; <color>; <page>; <label>)

!id

(linkmini <pos>; <color>; <page>; <label>)

!id

(linkmicro <pos>; <color>; <page>; <label>)

!id

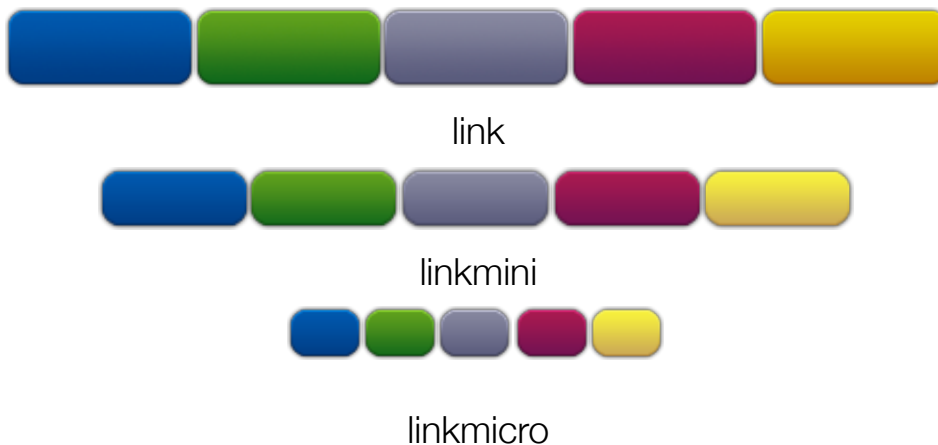
This object can be used in the menu page and in any other page. It creates a button to access another page, a pop-up, or an external Web page.

The (link), (linkmini), (linkmicro) objects have the identified versions:

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

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

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



Parameters:

<pos> - position. Use the pixels or rows and columns format

<color> - the color of the button. It can assume one of the values:

b - blue; g - green; gr - gray; r - red; y - yellow

<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 text label in the button. The text field can contain HTML tags.

(dlink <pos>; <color>; <page>; <label>)

!id

Similar to the (link) object, it differs in the graphic, which is larger than a normal button.

The (dlink) object has the identified version:

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



Parameters:

<pos> - position. Use the pixels or rows and columns format

<color> - the color of the button. It can assume one of the values:

b - blue; r - red;

<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 text label in the button. The text field can also contain HTML tags.

(page <id>; <description>; <protection>)

(page#landscape <id>; <description>; <protection>)

(page#portrait <id>; <description>; <protection>)

This object represents a generic page.

The `page#landscape` object defines the landscape (horizontal) orientation version of a page, and the `page#portrait` defines the portrait (vertical) version.

Some names are reserved and can't be used, not even as the initial part of a longer name. The reserved names are: *menu*, *temp*, *music*, *cameras*, *timer*, *logout*. While it is normal to have more link objects directed to the same page, it is an error to have more than one (page) object with the same name; this error can cause unpredictable behavior of the Web Browser, even if this type of semantic error is not detected by HSYCO Web Server.

Parameters:

<id> - the id of the page

<description> - the description of the page, that appears in the central area of the top bar when the page is shown

<protection> - this is an optional parameter. Can be set to "pin" or "puk", to force PIN or PIN and PUK re-authentication when entering the page¹².

Please note that the page-level PIN or PUK authentication is a client-side protection, and could be circumvented by a skilled person using Web software tools, to gain access to the protected page's commands. Nevertheless it is a good protection from accidental access to specific pages.

¹² The authentication is forced only if at least 60 seconds have passed since the last successful authentication.

Always use the server-side ACL features to ensure server-side protection of critical commands.

(popup <id>; <description>)

This object is very similar to the (page) object. It shows a pop-up on top of the original page.

The (#size) directive could be used to define the size of pop-up windows. It should be placed just after the (popup) object, before any other object defined inside the pop-up. If the size is not explicitly set with the (#size) directive, a standard size will be used for the pop-up window.

Parameters:

<id> - the id of the popup

<description> - the description of the pop-up that appears in the central area of the top bar when the page is shown.

(endofpage)

Defines the end of a (page) or (popup) section.

(container <pos>; <visibility>)

!id

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

The (container) object has the identified version:

(container!id <pos>; <visibility>)

Parameters:

<pos> - 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.

(endofcontainer)

Defines the end of a container.

(selector <file>; <pos>; <width>; <height>; <container id>; <group>; <text>; <css>)

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:

<file> - the name of the file to use as image

<pos> - position. Use the pixels or rows and columns coordinates format

<width> - the width of the text box, in pixels

<height> - the height of the text box, in pixels

<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

<text> - optional text. Can contain HTML tags

<css> - optional CSS style.

(background <file>)**!id**

This object creates a background image. The source image width and height should match the Web interface container size. It should be placed just after the (header) object to define the same background for the menu and all subpages, or after the (menu), (page) or (popup) objects to have different backgrounds for each page.

The (background) object has the identified version:

(background!id <file>)

Parameters:

<file> - the name of the file to use as background image, from the www/img directory.

Text**(text <pos>; <text>; <css>)****!id**

This object shows a generic text.

The (text) object has the identified version:

(text!id <pos>; <text>; <css>)

Parameters:

<pos> - position. Use the pixels or rows and columns coordinates format

<text> - the text. Can contain HTML tags

<css> - this parameter is optional, and defines the CSS style.

(marquee <pos>; <width>; <height>; <dir>; <speed>; <text>; <css>) !id

This object shows a generic text just like the (text) object, but with a scrolling effect.

The (marquee) object has the identified version:

(marquee!id <pos>; <width>; <height>; <dir>; <speed>; <text>; <css>)

Parameters:

<pos> - position. Use the pixels or rows and columns coordinates format

<width> - the width of the text box, in pixels

<height> - the height of the text box, in pixels

<dir> - 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> - this parameter is optional, and defines the CSS style.

Graphic Elements

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

This object displays a background panel.

The (panel) object has the identified version:

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

Parameters:

- <pos> - the position of the panel. Use the pixels or rows and columns coordinates format
- <width> - the width of the panel, in pixels
- <height> - the height of the panel, in pixels
- <color> - optional panel's color. If not set, the panel will be transparent. It can assume the following values:
b: blue; g: green; gr: gray; r: red; y: yellow.

(hbar <pos>; <width>)

This object creates a horizontal bar.

Parameters:

- <pos> - the position of the bar. Use the pixels or rows and columns coordinates format
- <width> - the width of the bar, in pixels.

(vbar <pos>; <height>)

This object creates a vertical bar.

Parameters:

- <pos> - the position of the bar. Use the pixels or rows and columns coordinates format
- <height> - the height of the bar, in pixels.

(image <file>; <pos>; <width>; <height>; <text>; <css>)**!id**

This object 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.

The (image) object has the identified version:

```
(imagelid <file>; <pos>; <width>; <height>; <text>; <css>)
```

Parameters:

<file> - the name of the file to use as image, from the www/img directory

<pos> - the position of the image. Use the pixels or rows and columns coordinates format

<width> the width of the image, in pixels

<height> the height of the image, in pixels

<text> - the optional text label

<css> - this parameter is optional, and defines the CSS style.

(imagelink <file>; <pos>; <width>; <height>; <page>; <text>; <css>) !id

This object creates a clickable image using a custom image file. 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.

The (imagelink) object has the identified version:

```
(imagelink!id <file>; <pos>; <width>; <height>; <page>; <text>; <css>)
```

Parameters:

<file> - the name of the file to use as image, from the www/img directory

<pos> - the position of the image. Use the pixels or rows and columns coordinates format

<width> the width of the image, in pixels

<height> the height of the image, in pixels.

<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

<text> - the optional text label

<css> - this parameter is optional, and defines the CSS style.

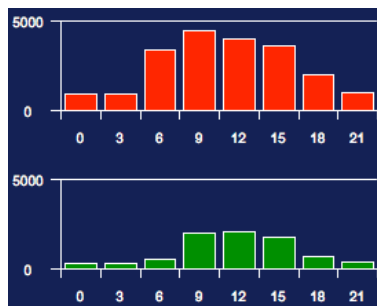
(chart!id <pos>; <width>; <height>; <attributes>)

!id

This object creates a bar chart. The chart's parameters and data are set with the uiSet() Java function or UISET action in EVENTS.

The (chart) object has the identified version only.

When charting a series of values, you have the choice of bar or point charts.



The “gauge” type chart is a special type of chart, representing a single value.



```
(type=gauge; orientation=horizontal; value=1; valuerange=0,10; bgcolor=black)
(type=gauge; orientation=vertical; value=4; valuerange=0,10; bgcolor=green;
barborder=false)
```

Charts can have a vertical orientation, so that the values axis is vertical, or horizontal, as the charts below.

Parameters:

<pos> - position. Use the pixels or rows and columns coordinates format

<width> - the width of the chart area, in pixels

<height> - the height of the chart area, in pixels

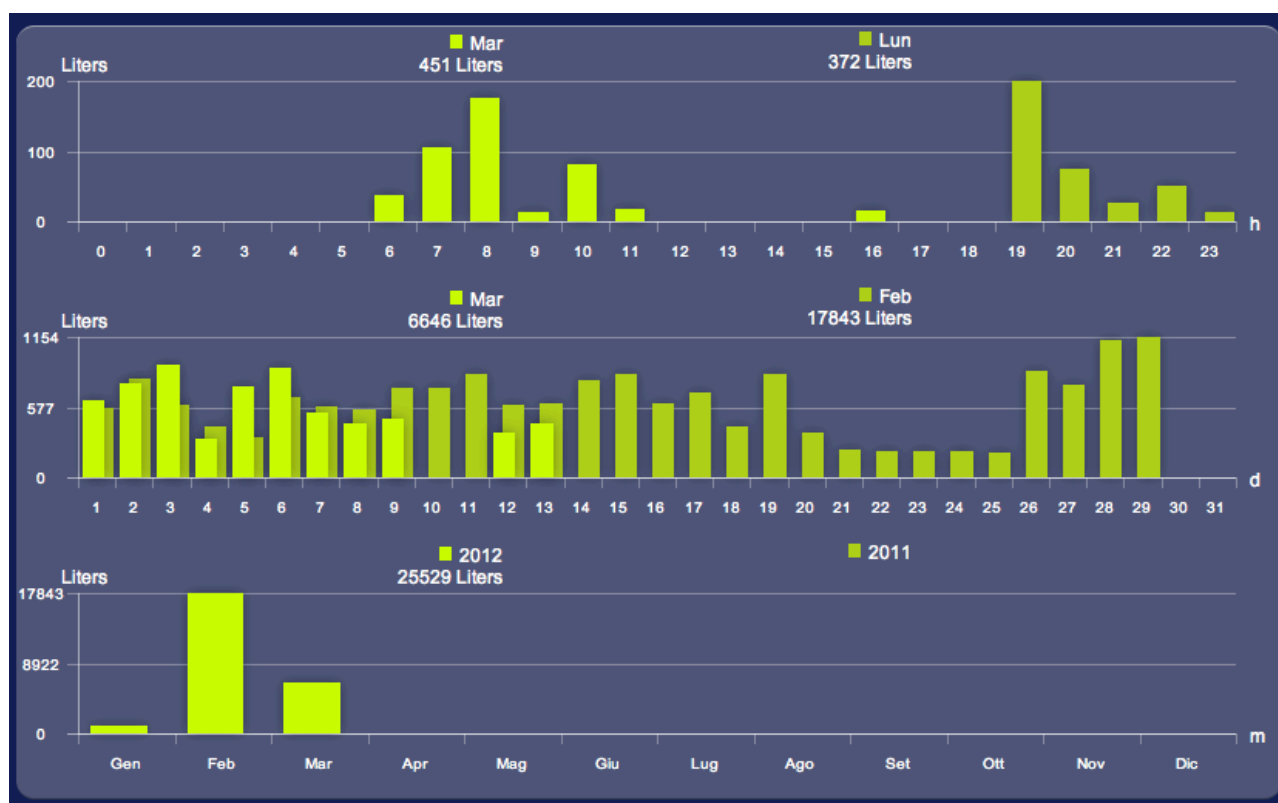
<attributes> - optional parameter to statically set the chart's attributes.

attribute	value
visible	<i>true</i> : visible; <i>false</i> : not visible
pos	the object position
type	chart type; supported types: <i>bars</i> , <i>points</i> , <i>gauge</i> . gauge charts only support the following attributes: value, orientation, valuerange, bgcolor, barcolor, barborder; the valuerange attribute is required for a correct graphic representation of a value
orientation	vertical or horizontal orientation of the value axis; supported values: <i>vertical</i> , <i>horizontal</i>
value	the value for gauge charts
values	the comma separated series of values; you can omit some values in the series (e.g. values="1, 3, 2, , 4, , , 5")
valuerange	minimum and maximum y-axis value, separated by comma. They indicate the graphic range. For example, "-100,100" will show a chart with values between -100 and 100
axislabels	labels on the x-axis (for vertical charts) or on the y-axis (for horizontal charts). It doesn't need to correspond to the number of values in values, and it can contain empty labels (in this case only the notches are shown)
drawaxis	specifies if the axis (notches and labels) are shown or not; possible values: <i>true</i> o <i>false</i>
origin	moves the origin to a value different from 0 (e.g. origin=3.2)
axisoffset	the axis offset (y if vertical, x if horizontal) expressed as percentage or as the value series index (ex. "50%" or "3")
baroffset	shifts the value bars or points by the given amount of pixels, so you can partially overlap multiple charts
notches	number of notches on the x-axis (for vertical charts) or on the y-axis (for horizontal charts)
spacing	spacing between the bars expressed as a percentage (use for the bar chart type)
pointsize	points size (use for the points chart type)

attribute	value
vlabelsstyle	value labels style; supported values: <i>inside</i> , <i>outside</i> , <i>none</i> . Values are not shown if this attribute is not defined
bgcolor	background color; use standard HTML color codes
barcolor	series of HTML colors, one for each data point in <i>values</i> ; use standard HTML color codes
barborder	bars/points border visibility; supported values: <i>true</i> , <i>false</i>
axiscolor	axis HTML color; use standard HTML color codes
notchcolor	HTML color of the axis notches; use standard HTML color codes
labelcolor	HTML color of the axis and bars labels; use standard HTML color codes

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

This object 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 data logger source id. Must be the same id used in hsyco.ini to define the data logger server
- <pos> - position. Use the pixels or rows and columns coordinates format
- <width> - the width of the object's area, in pixels
- <height> - the height of the object's area, in pixels
- <label> - the values axis label
- <attributes> - optional attributes.

attribute	value
avgvalues	<i>true</i> : show average values; <i>false</i> : hide average values (only range loggers). Default is true
daily	<i>true</i> : show daily charts; <i>false</i> : hide daily charts. Default is true
hourly	<i>true</i> : show hourly charts; <i>false</i> : hide hourly charts. Default is true
legend	<i>false</i> : do not show the values legend; <i>true</i> : show the legend with minimum, average and maximum summary values (for range loggers), for the current and previous period. Set legend to any comma separated combination of <i>min</i> , <i>avg</i> and <i>max</i> to show a legend with summary for the corresponding data sets. Default is true
maxvalues	<i>true</i> : show maximum values; <i>false</i> : hide maximum values (only range loggers). Default is true
minvalues	<i>true</i> : show minimum values; <i>false</i> : hide minimum values (only range loggers). Default is true
monthly	<i>true</i> : show monthly charts; <i>false</i> : hide monthly charts. Default is true
panel	<i>true</i> : show background panel; <i>false</i> : hide background panel. Default is true
pastbarscolor	past period bars' HTML color; use standard HTML color codes. Use a list of three comma-separated colors for range data loggers
pastperiod	<i>true</i> : show past period chart; <i>false</i> : hide past period chart. Default is true
presentbarscolor	present period bars' HTML color; use standard HTML color codes. Use a list of three comma-separated colors for range data loggers
tabs	<i>true</i> : show hourly, daily and monthly charts one at a time; <i>false</i> : show all charts. Default is false

attribute	value
totals	if the legend is visible, set to <i>true</i> to show period's totals or <i>false</i> to show min, avg and max labels. Default is true
axiscolor	axis HTML color; use standard HTML color codes
bgcolor	background color; use standard HTML color codes
labelcolor	HTML color of the axis and bars labels; use standard HTML color codes
notches	number of notches on the x-axis
notchcolor	HTML color of the axis notches; use standard HTML color codes
vlabelstyle	value labels style; supported values: <i>inside</i> , <i>outside</i> , <i>none</i> . Values are not shown if this attribute is not defined

(video <src>; <pos>; <width>; <height>; <mode>)



This object 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.

The (video) object has the identified version:

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

It allows dynamic control of position, visibility, play/pause and mode.

Parameters:

<src> - the name of the file, from the www/img directory

<pos> - the position of the video. Use the pixels or rows and columns coordinates format

<width> the width of the video, in pixels

<height> the height of the video, in pixels

<mode> - optional, selects the playback 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.

Administration

(adminlink <function>; <pos>; <color>; <label>)

This object creates the access button to the standard configuration pages. HSYCO supports four configuration pages: users, clock, network and password. Use the <function> field of the adminlink object to go to each of these pages.

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:

<function> - the function of the button. It can assume one of the values:
users, clock, network, password

<pos> - the position of the link button. Use the pixels or rows and columns coordinates format

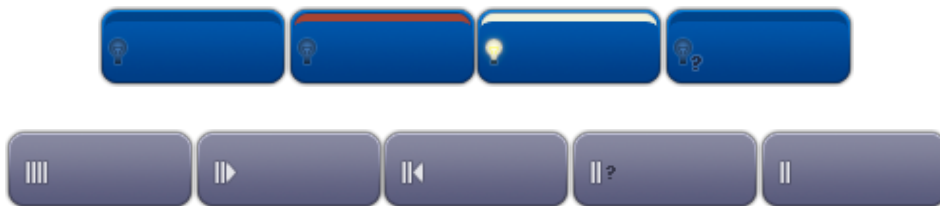
<color> - the color of the button. It can assume one of these values:

b - blue; g - green; gr - gray; r - red; y - yellow
 <label> - the text label on the button. Can contain HTML tags.

Lighting and Automation

(button <address>; <pos>; <label>)

Controls a light or automation device.



When used to control a VSHUT or HSHUT automation interlocked device, pressing this button will cycle as follows:

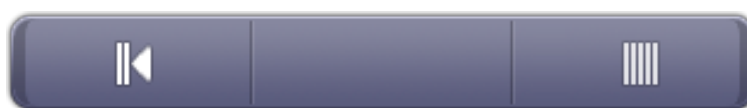
UNKNOWN -> DOWN -> STOP -> UP -> STOP -> DOWN

Parameters:

<address> - device data point name
 <pos> - the position of the button. Use the pixels or rows and columns coordinates format
 <label> - the text label on the button. Can contain HTML tags.

(3button <address>; <pos>; <label>)

(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

<pos> - the position of the button. Use the pixels or rows and columns coordinates format

<label> - the text label on the button. Can contain HTML tags.

(buttonicon <address>; <pos>; <color>)

(buttonicon) controls a light or automation device, and has the same function of (button).

**Parameters:**

<address> - device data point name

<pos> - the position of the button. Use the pixels or rows and columns coordinates format

<color> - the color of the button. It can assume one of these values:

gr - standard gray

glass - a semi-transparent graphic effect that looks nice on top of background images.

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

(buttonimage) controls a light or automation device, and has the same function of (button), but you can choose custom graphic images to 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.

Parameters:

- <address> - device data point name
- <pos> - the position of the button. Use the pixels or rows and columns coordinates format
- <width> - the object's width, in pixels
- <height> - the object's height, in pixels
- <unknown img> - the file name of the image associated to the unknown state
- <on img> - the file name of the image associated to the on state
- <off img> - the file name of the image associated to the off state
- <up img> - the file name of the image associated to the going up state
- <down img> - the file name of the image associated to the going down state
- <offup img> - the file name of the image associated to the stable/off up state
- <offdown img> - the file name of the image associated to the stable/off down state
- <text> - the optional text label
- <css> - this parameter is optional, and defines the CSS style.

All images should be located in the `www/img` or `www/<project>/img` directories.

(dimmer <address>; <pos>; <label>)

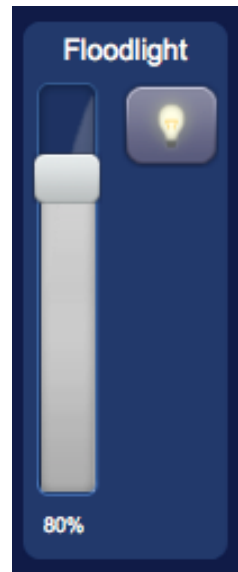
This object corresponds to a dimmer light actuator.

Parameters:

<address> - device data point name

<pos> - the object's position. Use the pixels or rows and columns coordinates format

<label> - the text label on the button. Can contain HTML tags.



(dmx <address>; <pos>; <label>)

Control 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.

<pos> - the object's position. Use the pixels or rows and columns coordinates format

<label> - the text label on the button. Can contain HTML tags.



(dmxrgb <address>; <pos>; <label>)

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

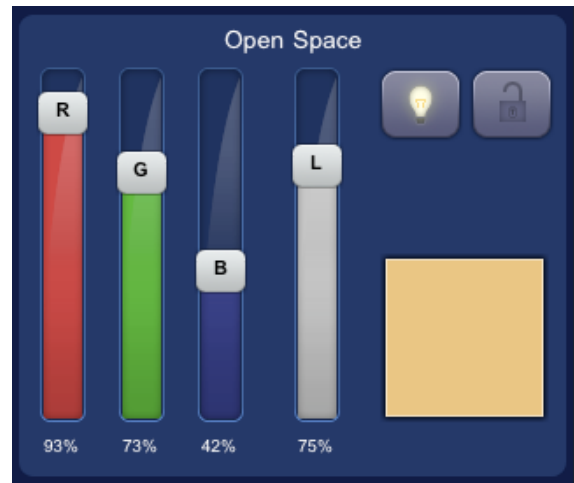
Parameters:

<address> - using just one DMX gateway, address should be a number between 1 and 512, set to the address of the red channel.

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.

<pos> - the object's position. Use the pixels or rows and columns coordinates format

<label> - the text label on the button. Can contain HTML tags.



Temperature Control

(temp <server>; <address>; <label>; <pos>)

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

<pos> - the object's position. Use the pixels or rows and columns coordinates format.

(tempmini <server>; <address>; <label>; <pos>)

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

<pos> - the object's position. Use the pixels or rows and columns coordinates format.

Timers and Schedulers

(timer <id>; <pos>; <label>)

This object corresponds to the mini-display for user timers control. The mini-display shows the current status of the timer, and touching it opens the timer set-up page or pop-up.

The (timer) object generates TIMER events and userTimerEvent() Java callbacks on activation and reset.

**Parameters:**

<id> - the timer id, as defined in the Timers parameter in the hsyco.ini file

<pos> - the object's position. Use the pixels or rows and columns coordinates format

<label> - the text label on the button. Can contain HTML tags.

(scheduler <pos>; <width>; <height>; <mode>; <names>)

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 name. This name is used to generate the events, and is not unique, as you can define several rules under the same name. In this case the scheduler will evaluate the rules having the same name 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 name. So, rules on top have priority within the group of rules with the same name.

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.

The image displays two side-by-side screenshots of the HSYCO configuration interface. The left screenshot shows a summary view of a 'daily' rule. It includes a date range from 16/06/2012 to 16/06/2012 and a time range from 10:32 to 10:32. Below this, it shows a 'daily' rule with the description 'exec every day' and a date range from 12/06/2012 to 18/06/2013. The right screenshot shows the configuration form for the 'daily' rule. It includes fields for 'rule name' (daily), 'description' (exec every day), 'enabled' (checked), 'daily mode' (checked), 'from' and 'to' date ranges (12 Jun 2012 to 18 Jun 2013), 'from' and 'to' time ranges (12:00 to 14:00), 'days of the week' (M, T, W, T, F, S, S), and 'mode' (on, interval, off).

Parameters:

- <pos> - the object's position. Use the pixels or rows and columns coordinates format
- <width> - object's width, in pixels
- <height> - object's height, in pixels
- <mode> - set to "edit" to enable the creation or deletion of individual schedules. Set to "noedit" to only allow editing of already defined schedules
- <names> - optional, comma separated list of schedule names that are shown in this object. If the list is not defined or set to "*", the object shows all schedules.

Cameras

(cameralink <cam id>; <pos>; <color>; <label>; <dest panel id>)

This object represents a button, graphically identical to (link), to access the view and control page of a single camera or a grid of cameras.

To work correctly, this object requires the (#cameralist) directive as well as (#cameragridlist) for grids.

Parameters:

- <cam id> - 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
- <pos> - the object's position. Use the pixels or rows and columns coordinates format
- <color> - the color of the button. It can assume one of these values:
b - blue; g - green; gr - gray; r - red; y - yellow
- <label> - the text label on the button. Can contain HTML tags
- <dest panel id> - if set to "default" will show this camera in a dynamic, full-size panel. If set to a panel id, will show the camera on that camera panel.

(camera <cam id>; <pos>; <width>; <height>; <dest panel id>)

!id

This object displays a real-time view of a camera or grid of cameras.

The (camera) object has the identified version:

(camera!id <cam id>; <position>; <width>; <height>; <dest panel id>)

Parameters:

- <cam id> - 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
- <pos> - the object's position. Use the pixels or rows and columns coordinates format
- <width> - the width of the camera view, in pixels
- <height> - the height of the camera view, in pixels
- <dest panel id> - optional. If set to "default" will show this camera in a dynamic, full-size panel. If set to a panel id, will show the camera on that camera panel.

(camerapanel <cam id>; <pos>; <width>; <height>; <cam list>)

!id

This object displays a real-time view of a camera or grid of cameras. Unlike the (camera) object, the image is on a panel from which the PTZ functions and access to the recorded images can be controlled.

The (camerapanel) object has the identified version:

(camerapanellid <cam id>; <position>; <width>; <height>; <cam list>)

Parameters:

- <cam id> - 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
- <pos> - the object's position. Use the pixels or rows and columns coordinates format
- <width> - the width of the camera panel, in pixels
- <height> - the height of the camera panel, in pixels
- <cam 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.

IRTrans

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

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

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

This object is used to send infrared (IR) commands through the IRTrans IR emitter/receiver network devices.

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”

<pos> - the object's position. Use the pixels or rows and columns coordinates format

<color> - the color of the button. It can assume one of these values:

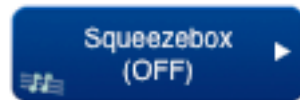
b - blue; g - green; gr - gray; r - red; y - yellow

<label> - the text label on the button. Can contain HTML tags.

Music Players

(music <id>; <pos>; <label>)

This object is used to display a mini-display to manage a Squeezebox music player.



Parameters:

- <id> - the number that identifies the Squeezebox, starting from 0, in the list defined by the slimPlayers parameter in hsyco.ini
- <pos> - the object's position. Use the pixels or rows and columns coordinates format
- <label> - the text label on the button. Can contain HTML tags.

(musicsync <pos>)

This object is used to insert the music [Sync] button in a music page.

Parameters:

- <pos> - the object's position. Use the pixels or rows and columns coordinates format.

(nuvo <id>; <zone>; <pos>)

(nuvomini <id>; <zone>; <pos>)

The (nuvo) object is used to display a zone control panel for the NuVo Grand Concerto and Essentia multi-room system.

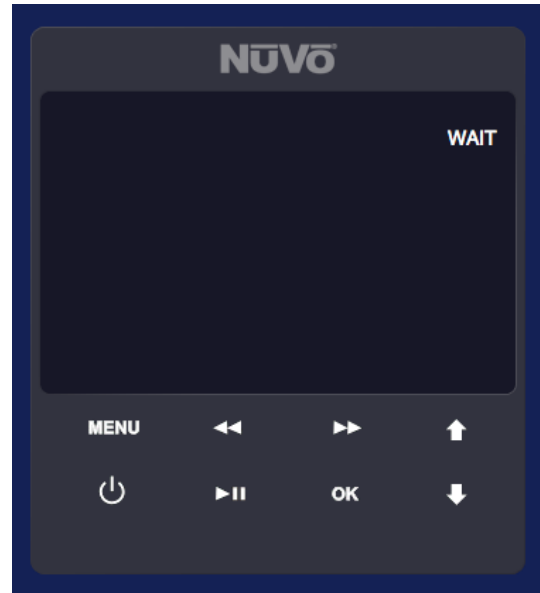
(nuvomini) offers the same functions and display of (nuvo), but in a smaller graphic footprint, that fits better the standard page size used for mobile devices.

Parameters:

<id> - the I/O Server id assigned to the NuVo system. See the Application Notes for additional information

<zone> - the zone number

<pos> - the object's position. Use the pixels or rows and columns coordinates format.



User Buttons

(user <name>; <param>; <pos>; <color>; <label>)

!id

(usermini <name>; <param>; <pos>; <color>; <label>)

!id

(usermicro <name>; <param>; <pos>; <color>; <label>)

!id

Clicking these objects, you can execute customized Java code or generate events for the EVENTS interpreter.

Pressing the button created with the *(user)* object, 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. This function is described in details in the Programming chapter.

The *(user)*, *(usermini)* and *(usermicro)* objects have the identified version:

(user!id <name>; <param>; <pos>; <color>; <text>)

(usermini!id <name>; <param>; <pos>; <color>; <text>)

(usermicro!id <name>; <param>; <pos>; <color>; <text>)



user



usermini



usermicro

Parameters:

<name> - name passed to the Java callback method and event in EVENTS

<param> - parameter passed to the Java callback method and event in EVENTS

<pos> - the object's position. Use the pixels or rows and columns coordinates format

<color> - the color of the button. It can assume one of these values:

b - blue; g - green; gr - gray; r - red; y - yellow

<label> - the text label on the button. Can contain HTML tags.

(userrgb <name>; <param>; <pos>; <color>; <label>)

!id

Clicking this object, you can execute customized Java code or generate events for the EVENTS interpreter. It is similar to the (user) object, with a different graphic presentation that allows you to use an arbitrary color in the center area of the button.

The (userrgb) object has the identified version:

(userrgb!id <name>; <param>; <pos>; <color>; <label>; <text>)



Parameters:

<name> - name passed to the Java callback method and event in EVENTS

`<param>` - parameter passed to the Java callback method and event in EVENTS

`<pos>` - the object's position. Use the pixels or rows and columns coordinates format

`<color>` - the color of the button. It must be specified in the hexadecimal format RRGGBB, where RR, GG and BB are the hexadecimal values, from 00 to FF, of the red, green and blue colors, for example:

FF0000 - red

00FF00 - green

0000FF - blue

000000 - black

FFFFFF - white

FFFF00 - yellow

`<label>` - the text label on the button. Can contain HTML tags.

(userimage `<file>`; `<pos>`; `<width>`; `<height>`; `<name>`; `<param>`; `<text>`; `<css>`) 

Clicking this object, you can execute customized Java code or generate events for the EVENTS interpreter. It is similar to the (user) object, with a different graphic presentation that allows you to use an arbitrary image. 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.

The (userimage) object has the identified version:

(userimage!id `<file>`; `<pos>`; `<width>`; `<height>`; `<name>`; `<param>`; `<text>`; `<css>`)

Parameters:

- <file> - the name of the file to use as image, from the www/img directory
- <pos> - the object's position. Use the pixels or rows and columns coordinates format
- <width> the width of the image, in pixels
- <height> the height of the image, in pixels
- <name> - name passed to the Java callback method and event in EVENTS
- <param> - parameter passed to the Java callback method and event in EVENTS
- <text> - the optional text label
- <css> - this parameter is optional, and defines the CSS style.

(slider!id <name>; <pos>; <label>)**!id**

Clicking 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.

The (slider) object has the identified version only.

Parameters:

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



(sliderbutton!id <name>; <pos>; <label>)

Clicking 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.

The (slider) object has the identified version only.

Parameters:

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



Forms

(input!id <pos>; <css>)

!id

This object shows a text input field.

The (input) object has the identified version only.



Parameters:

<pos> - the object's position. Use the pixels or rows and columns coordinates format

<css> - optional. Defines the CSS style for the input field.

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

!id

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

!id

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

!id

Clicking this object, you can execute customized Java code, passing the values of all the input fields in the same page, pop-up or container.

HSYCO calls the method:

userCommand(String name, String param)

in the *user.class* class, passing:

name - the id of the (submit) object

param - a string of all the objects (input) ids and their text content.

The (submit), (submitmini), (submitmicro) objects have the identified version only.



If the (submit!id) object's id starts with \$, then the server will automatically set variables named \$<id>.<input id> for each input field to the values in the input fields.

Parameters:

- <pos> - the object's position. Use the pixels or rows and columns coordinates format
- <color> - the color of the button. It can assume one of these values:
b - blue; g - green; gr - gray; r - red; y - yellow
- <label> - the text label on the button. Can contain HTML tags.

(submitimage <file>; <pos>; <width>; <height>; <text>; <css>)

!id

Just like (submit), this object is used to execute customized Java code, passing the values of all the input fields in the same page, pop-up or container.

The graphic representation is the same as the (image) object, with a custom image and a text label.

The (submitimage) object has the identified version:

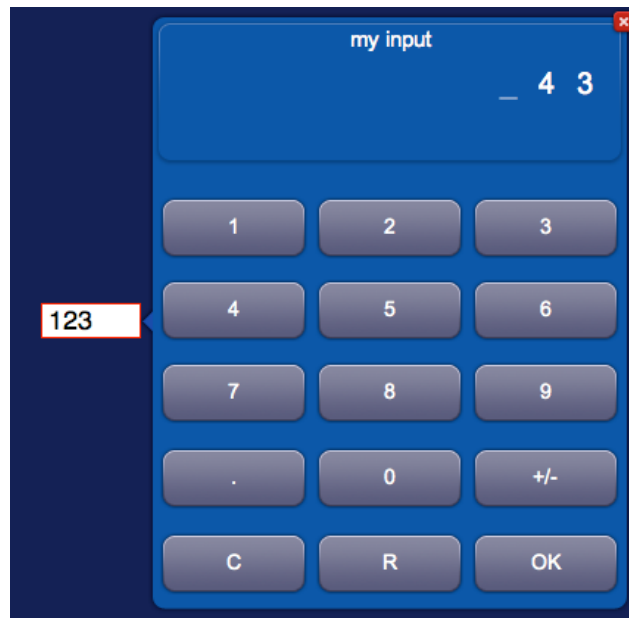
(submitimage <file>; <pos>; <width>; <height>; <text>; <css>)

Parameters:

- <file> - the name of the file to use as image
- <pos> - the object's position. Use the pixels or rows and columns coordinates format
- <width> the width of the image, in pixels
- <height> the height of the image, in pixels
- <text> - optional text label
- <css> - optional CSS style.

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

This object shows a numeric input field. Touching the field, a keypad pop-up is opened automatically.



The keypad performs range and number of digits validation of the input data. Touching the OK button or pressing the Enter key on the physical keyboard is equivalent to touching the (submit) object of a form; HSYCO will call the method:

userCommand(String name, String param)

in the *user.class* class, passing:

name - the id of the (keypad) object

param - the numeric value entered via the keypad.

The USER <id> event will be generated as well. If the (keypad!id) object's id starts with \$, then the server will also automatically set a variable named \$<id> to the appropriate value. The (keypad) object can also be used in forms, just like an (input) object. The (keypad) object has the identified version only.

Parameters:

- <pos> - 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> - a free text shown at the top of the keypad
- <css> - optional. Defines the CSS style for the input field.

Weather**(weather!id <pos>)****!id**

This object 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.
The (weather) object has the identified version only.

Parameters:

- <pos> - the object's position. Use the pixels or rows and columns coordinates format.

Web Interface Options

The built-in “blue” skin support some options that can modify the behavior of the Web interface.

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>/<name sub-directory>/pin.<urlkey>?<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.

Web Pages Translation

The creation of Web interface pages in different languages is very easy. All the localized custom text is in the index.hsm file, so you should create an index.hsm file for each language, remembering that each file must have a dedicated www sub-directory.

The (#language <language id>) directive determines the language of all system messages. HSYCO standard skins provide support for English, Italian and French, respectively *en*, *it*, and *fr*. To add support for a different language, you should simply create a **text_<language id>.txt** file in the skin directory.

The text_<language id>.txt file contains all the system messages used by HSYCO. The format is:

name = value

The blank spaces before and after the name and value are ignored. Empty lines and those that start with the “#” character, which can be used for comments, are also ignored.

To ensure you are creating a correct file for the new language, we suggest to start from one of the existing files, and translate each and every string in it.

Some examples:

```
weekdayslist = "L","M","M","G","V","S","D"
pressctocont = Premere C per continuare
enterpuk = Inserire il PUK e premere OK

weekdayslist = "M","T","W","T","F","S","S"
pressctocont = Press C to continue
enterpuk = Enter PUK code then press OK
```

The Skins Interpreter

The format of the index.hsm file, described in this chapter, and all its objects, are not strictly imposed by HSYCO. Actually, all that has been written about the the Web interface pages can be completely modified.

The objects that we have described so far are nothing but simple macros defined in the skin, that can be modified, extended with new objects or completely replaced with different objects.

To understand how the system works, we will briefly discuss the parser architecture and how the skins are used to generate native HTML code from the index.hsm files.

The index.hsm file format is based on objects, whose generic representation is:

(object_type <parameter1>;<parameterN>)

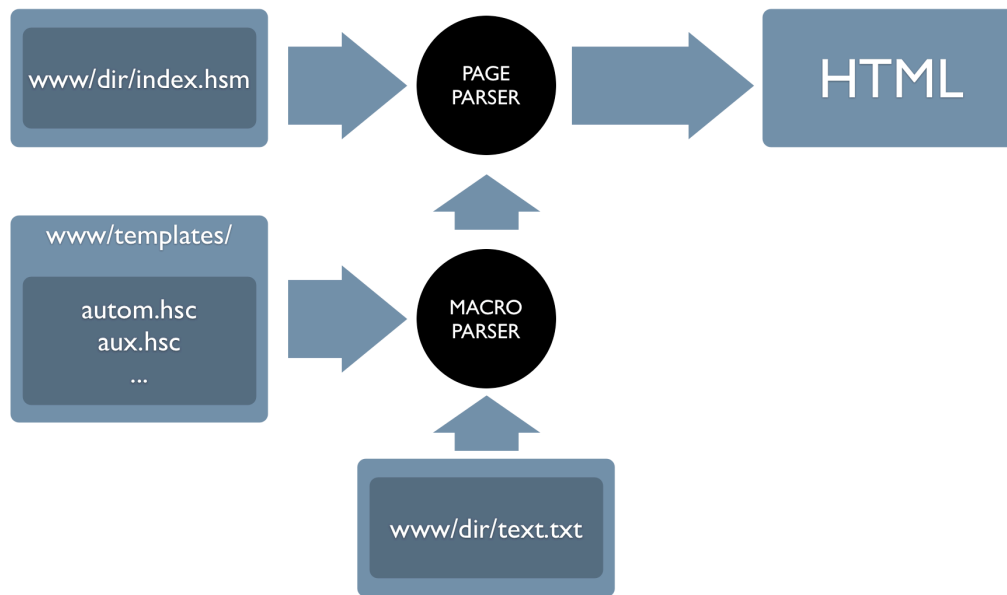
Every type of object can require one or more parameters, separated by the “;” character. At every request received by the Web Server, the appropriate index.hsm file is read by the **Page Parser**, which translates the file as shown in the diagram.

For each object in the index.hsm file, the Page Parser performs the translation to HTML by invoking the **Macro Parser**.

The work done by Macro Parser is quite simple, it consists in reading from the `www/<skin name>` directory the content of the **object_type.hsc** file¹³ and replacing the parameters’ placeholders with actual values.

The skin to be used for an index.hsm file is defined by the **(#skin <skin name>)** meta object (also called directive) at the beginning of the file, before the (header) object.

¹³ As stated before, HSYCO always looks for files first in the www sub-directories and, only if the file is not available, it looks for them inside hsyco.jar. Therefore a built-in file in HSYCO can always be replaced with a new, customized file.



The Macro Parser just replaces all the **\$N** strings (where N is a digit between 1 to 9¹⁴) with the corresponding parameter in the object tag defined in index.hsm. It also replaces all the **\$name;** strings with the value associated to *name* in the file **text_<language id>.txt** which has to be located in the skin's directory.

Use two **\$\$** characters to obtain a plain "\$" in the hsc files. A single "\$" character is an error.

Besides these placeholders, the content of the hsc files is mostly HTML or JavaScript code.

The final result is pure HTML code, built from many objects that all together describe the Web interface.

Most of the hsc files are just a few lines. Others are bigger, like **header.hsc**, for example, which contains most of the CSS definitions for the skin. Another important file is **main.hsc**; it contains the HTML code of the standard pages and of the display area, and the JavaScript code necessary to the multilingual management of texts and other functions, as well as the loading of the main JavaScript file containing the actual Web application.

¹⁴ It is not possible to define objects with more than 9 parameters.

At last, the ***pin.hsc*** file contains both the HTML code and JavaScript for the management of the PIN/PUK pages. When the Web Server receives an access request to the Web interface, if the Web Browser has not already been authenticated, the content of *pin.hsc* is returned, not the complete HTML code that implements the full interface. The fully functional Web interface is loaded only after the correct authentication.

A skilled developer can leverage the skin interpreter and the HSYCO Web interface architecture to achieve a complete customization of the user interface.

Inserting HTML Code in the Pages

In some cases it may be necessary to write HTML custom code, besides using the standard objects, and integrate this HTML code in *index.hsm*.

However, since the HSYCO Web Server is not designed to serve pure HTML files¹⁵ and the content of *index.hsm* must strictly be in the object format, the recommended way to insert HTML code is to create one or more files with the *hsc* extension in the skin directory.

For example, the HTML content¹⁶ of a *www/classic/custom/myHtml.hsc* file can be inserted in the appropriate position inside *index.hsm* by just writing:

```
(custom/myHtml)
```

Nonetheless, it is recommended to follow clear and straightforward rules in naming the objects, which should allow a simple management of the files associated to the objects, remembering that files in the *www* subdirectories are used instead of same name files that are built-in the *hsyco.jar* package.

¹⁵ An exception to this is the public Web Server function, that you can enable to serve pure HTML pages to anonymous users, without authentication and access control.

¹⁶ *hsc* files cannot contain a single isolated “\$” character, but it must be written as \$\$\$. Moreover, in the content of *index.hsm* and in the names of variables in the *hsc* file, the small and capital letters are significant, for example (menu) and (MENU) are two different objects, the second does not exist in the standard skin.

Events Programming

HSYCO basic features can be extended thanks to two programming systems: the Java and the EVENTS programming environments.

The standard Java language programming environment provides an API where you can create custom Java code that executes when a field event is detected, or based on user interaction with the Web interface. The API also has a number of methods for device control and user interface interaction. Besides, the standard Java environment allows you to tightly integrate HSYCO with any custom Java code or library.

EVENTS is a proprietary language that offers a very simple, powerful and high-performance solution that allows you to easily associate control commands to any field event or state condition.

HSYCO's architecture is *multi-threaded*¹⁷ to guarantee, as much as possible, quick response times to any managed event. Every call to the Java functions or to the events method is executed in an independent **thread**, so that it doesn't stop all other concurrent calls. However, the hardware resources being shared and limited, heavy processing might have a negative impact on the system in terms of performances and stability.

In order to carry out the normal activities of supervision and control, HSYCO constantly monitors the status of all field devices. After a change of status, such as a light being turned off or the temperature change in a zone, HSYCO calls the appropriate "callback" Java method, and also checks all events declarations in EVENTS and executes actions of all matching conditions.

¹⁷ The different activities of the system are executed by different parallel branches of execution. This way, even if a branch of execution requires a very long time to be completed, this does not necessarily jeopardize the response times of the other branches of execution.

EVENTS

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.

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 : 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:

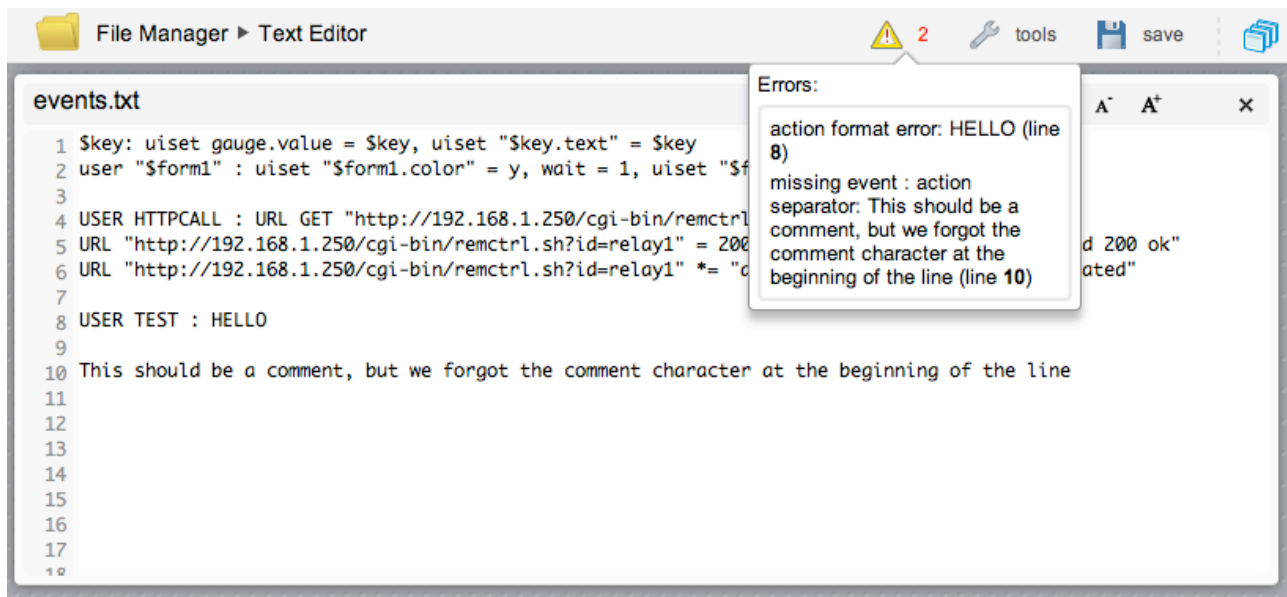
```
DAY : IO k.33 k.34 k.35 k.36 = 0
```

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.



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 SECURITY = ALARM
```

is an event that occurs when the camera “entrance” starts recording, but only if the anti-intrusion system is not in the alarm state.

Note:

```
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
```

Note:

```
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.

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
```

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.

Double quotes can be used to delimit text strings, especially when strings contain special characters, such as , : or #.

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 expressions 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 remainder (%) and rounding to an arbitrary¹⁸ number of digits (ROUND n) are supported.

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.

¹⁸ You can round a number from 0 to 9 decimal digits. Rounding to 0 decimal digits rounds to the nearest integer.

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`¹⁹ class documentation.

The pattern can also 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 "#.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.

¹⁹ <http://docs.oracle.com/javase/6/docs/api/java/text/DecimalFormat.html>

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:

`= > < >= <= *=20`

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!
```

²⁰ *= is a contains operator, matching any left-side string that contains the right-side string.

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:

`$TIME$`

`$SUNSET$`

`$SUNRISE$`

`$TIME:HMS$`

`$DATE:YMD$`

`$DATE:DOW$`

`$POWER$`

Predefined variables are different from normal ones because their names end with the \$ character.

`$TIME$`, `$SUNSET$` 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.

These variable 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.

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.

Power example:

`$POWER$`

is translated to the current power load - an integer value in Watt. This is the same number that appears in the Web interface.

You can also set the power level with an action, so it appears in the top or bottom bars of the Web interface:

`$TIME : POWER = IO m.100`

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`

Events

The format of an event is generally the event keyword and the required attributes, like the address of a device, and the condition, if appropriate.

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.cl = 1
IOSTART 2: IO relays.door = 1
```

DMX

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
```

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
```

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


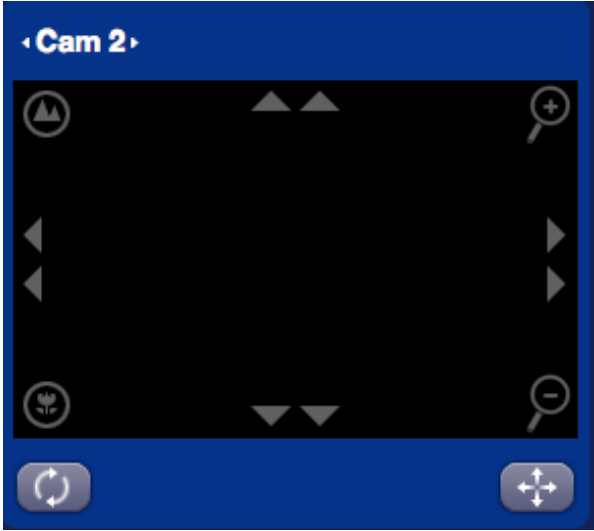







Triggered by a camera control input from the Web interface. It can be used to execute arbitrary actions when the user clicks on the active areas of the camera view.

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.

	function	
	focusfar	
	focusnear	
	zoomtele	
	zoomwide	
	moveleft	
	moveright	
	moveup	
	movedown	
Center	movestop	

Examples:

```
CAMERACOMMAND entrance = focusfar
```

Security

SECURITY

This event is deprecated, and will not supported in future releases.

HSYCO fully integrates some state-of-the-art anti intrusion units (Paradox EVO, Bentel KYO320 and Tecnoalarm) that generate many proprietary IO events. To simplify programming and handling of simple events, some SECURITY events are generated too, associated to activation, deactivation and alarm.

Event	State	Description
SECURITY id = ALARM n	stable	alarm status area/partition/program N
SECURITY id = ON n	stable	area/partition/program N active
SECURITY id = OFF n	stable	area/partition/program N inactive

Parameters:

id - the anti intrusion unit name; corresponds to the ids defined in the configuration files. id cannot be omitted

n - a positive integer number identifying the area number (Paradox), the partition number (Bentel) or the program number (Tecnoalarm). n cannot be omitted.

Examples:

```
SECURITY evo = ALARM 1
SECURITY kyo = ON 2
SECURITY kyo = OFF 2
```

IRTrans

IR

This event is triggered when an IRTrans receives or sends an IR command from its local IR database.

Event	State	Description
IR name = command	transient	the IR code must be in the local IRTrans FLASH memory database

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
```

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
```


PBX

PBX

Called when a call notification is sent to HSYCO by the PBX system.

Event	State	Description
PBX from = to	transient	the event occurs when the number is dialed, even if nobody answers or if the dialed number doesn't exist

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
```

Network Services

LOCATION

This event is triggered, if the location service is active, when a change in the association of a client to the Wi-Fi Access Points is detected. It is a stable condition that represents the current connection state and 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

Event	State	Description
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"
```

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
```

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:

```
TIME: 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"
```

Internal Events

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
```

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.

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
```

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
```

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
```

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
```

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
```


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 = 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 = 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
```

POWER

Triggered by a change of the current power load level, as set using the powerSet() Java API 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. The power must be defined in Watt

Examples:

```
POWER > 2000 : IO k.70 = 0
POWER : LOG "NEW POWER:" $POWER$
```

Actions

The format of an action is generally composed by the action keyword and the required attributes, like the address of a device, and status values.

I/O Servers

IO

Writes to I/O servers variables or data points.

Action	Description
IO name = value	sets the variable 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 and the 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

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
```

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
```

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
```

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 = true` and `userLog = 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
```

IRTrans

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
```


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
```

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 Public Announcement Appendix 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 Public Announcement Appendix 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)

Audio Destination	Description
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:

TIME : audio speaker = file "audio/gong.mp3"

The gong.mp3 audio file is played every minute through the server's audio line out or internal speaker.

TIME : audio snom@239.255.255.245:5555 = 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 axis@cam1 = file "audio/gong.mp3"

Same as above, but played to the Axis camera with id "cam1".

TIME : audio axis@cam1 = "voice:en" "the time is " \$time:h:m\$

Converts a text message to speech using the "en" voice of the text-to-speech engine, and play the audio on an Axis camera.

Mail

MAIL

Sends an email message. HSYCO SERVER sends the mail either directly to the recipient's domain mail server if the SmtplibName 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.

This method does not retry sending the message if the destination mail server is not available when the method is called.

The mail body can be a simple text or a multi-part message, with live or recorded camera images, and file attachments.

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:

```
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
```

In this example, we send a message every minute, with the measured power load, to two email addresses.

```
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"
```

40 seconds after the alarm event, an email from the address hsyco@hsyco.net is sent to email1@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.

```
TIME = 0000: MAIL email1@hsyco.com = hsyco@hsyco.net "Sent
from HSYCO" "Energy daily report." file:data/energy.csv
at midnight we are sending the energy.csv file in the data sub-directory as
an attachment.
```

Log

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

Examples:

```
LOG = log message example
```

```
LOG = "Time: " $TIME:H-M-S$ " DATE: " $DATE:D/M/Y$
```

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

Action	Description
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"
```


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 chapter in the Appendix 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 = 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.

Example 1:

```
$value : DATALOGGER energy = $value,  
DATALOGGER energy = FILE LOG hsyco/energylog.csv TIMESTAMP,  
DATALOGGER energy = FILE STAT hsyco/energystat.csv
```

Example 2:

```
TIME : DATALOGGER temperature = IO ste.1  
TIME : DATALOGGER humidity = IO ste.2
```

Leak Detector

LEAK

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 whenever a flow counter is incremented. 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.

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 an updated flow value. Using variables or references to IO data points is supported

Action	Description
LEAK name = PERIOD t	<p>changes the time base. The default is 1200 seconds.</p> <p>Set the period to 0 to temporarily disable the detector.</p> <p>The time base can be changed anytime, even when the detector is already processing data</p>
LEAK name = DEVIATION d	<p>changes the deviation percentage. d should be a number between 1 and 100. The default is 50.</p> <p>Set the deviation to 0 to temporarily disable the detector.</p> <p>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.</p> <p>The deviation can be changed anytime</p>
LEAK name = CLEAR	<p>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</p>

Parameters:

name - the name of a leak detector. Using different names you can implement several independent leak detectors.

Examples:

```

TIME: 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"

```

Internal Actions

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

Examples:

```
WAIT = 5
WAIT = 0.3
```

USER

Triggers a USER event and the `userCommand(String name, String param)` Java method. Can be used like a function call in EVENTS, and as a calling mechanism between EVENTS and Java.

Action	Description
USER name = param	executes the <code>user.java</code> method: <i>userCommand (String name, String param)</i> 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

```

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

Action	Description
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.

Examples:

```
SECURITY evo = ON 1 : PROGRAMTIMER presence = REPEAT 3600  
SECURITY evo = OFF 1 : PROGRAMTIMER presence = CLEAR
```

UISET

Changes the dynamic attributes of the (text), (user), (link) and the other identified objects.

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; see the table below

value - attribute value; see the table below. You can use multiple unquoted words, quoted strings and variables, that will be appended to form the value parameter.

object type	attribute	value
project	page	set id to the directory name of the index.hsm file you want to control, or to "*" to control all Web clients. Setting value to a page id or to "menu" will automatically force all Web clients that have that index.hsm visible to switch to the specified page (if a project doesn't have that page, the command is ignored). Setting value to an empty string will automatically revert to the previous page
	lock	set id to the directory name of the index.hsm file you want to control, or to "*" to control all Web clients. Setting value to a page id or to "menu" will automatically force all Web clients that have that index.hsm visible, and Web clients that are loading that index.hsm in the future, to switch to the specified page, also disabling user's navigation. Setting value to an empty string unlocks the navigation

object type	attribute	value
	pageback	set id to the directory name of the index.hsm file you want to control, or to "*" to control all Web clients. Set value to a page id to force Web clients that are currently displaying that page to go to the previous page. Set value to a pop-up id to force Web clients that have that pop-up open to close it
page	blink	<i>false</i> : stops blinking of all objects in the page
background	img	name of a customized file with the image of the object, saved in the www/img directory, or a full URL (starting with http:// or https://) to show an image from a remote website
camera	camera	id of the camera to show in this object
camerapanel	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
chart	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	type	chart type; supported types: <i>bars</i> , <i>points</i> , <i>gauge</i> . gauge charts only support the following attributes: value, orientation, valuerange, bgcolor, barcolor, barborder; the valuerange attribute is required for a correct graphic representation of a value
	orientation	vertical or horizontal orientation of the value axis; supported values: <i>vertical</i> , <i>horizontal</i>
	value	the value for gauge charts
	values	the comma separated series of values; you can omit some values in the series (e.g. values="1, 3, 2, ,4, , , 5")
	valuerange	minimum and maximum y-axis value, separated by comma. They indicate the graphic range. For example, "-100,100" will show a chart with values between -100 and 100

object type	attribute	value
	axislabels	labels on the x-axis (for vertical charts) or on the y-axis (for horizontal charts). It doesn't need to correspond to the number of values in values, and it can contain empty labels (in this case only the notches are shown)
	drawaxis	specifies if the axis (notches and labels) are shown or not; possible values: <i>true</i> o <i>false</i>
	origin	moves the origin to a value different from 0 (e.g. origin=3.2)
	axisoffset	the axis offset (y if vertical, x if horizontal) expressed as percentage or as the value series index (ex. "50%" or "3")
	baroffset	shifts the value bars or points by the given amount of pixels, so you can partially overlap multiple charts
	notches	number of notches on the x-axis (for vertical charts) or on the y-axis (for horizontal charts)
	spacing	spacing between the bars expressed as a percentage (use for the bar chart type)
	pointsize	points size (use for the points chart type)
	vlabelsstyle	value labels style; supported values: <i>inside</i> , <i>outside</i> , <i>none</i>
	bgcolor	background color; use standard HTML color codes
	barcolor	series of HTML colors, one for each data point in <i>values</i> ; use standard HTML color codes
	barborder	bars/points border visibility; supported values: <i>true</i> , <i>false</i>
	axiscolor	axis HTML color; use standard HTML color codes
	notchcolor	HTML color of the axis notches; use standard HTML color codes
	labelcolor	HTML color of the axis and bars labels; use standard HTML color codes
container	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
dlink	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	text	the text label

object type	attribute	value
	color	<i>r</i> : red, <i>g</i> : green, <i>b</i> : blue, <i>gr</i> : gray, <i>y</i> : yellow
image	visible	<i>true</i> : visible; <i>false</i> : not visible
imagelink	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	img	name of a customized file with the image of the object, saved in the www/img directory, or a full URL (starting with http:// or https://) to show an image from a remote website
	text	the text label
input	visible	<i>true</i> : visible; <i>false</i> : not visible
keypad	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	text	the text
	color	the text color; use standard HTML color codes
	size	the text font size, in pixels
	width	width, in pixels
	height	height, in pixels
	weight	the text boldness: <i>lighter</i> , <i>normal</i> , <i>bold</i> , <i>bolder</i>
	style	the text style: <i>normal</i> , <i>italic</i> , <i>oblique</i>
link	visible	<i>true</i> : visible; <i>false</i> : not visible
linkmini	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
linkmicro	pos	the object position
	text	the text label
	color	<i>r</i> : red, <i>g</i> : green, <i>b</i> : blue, <i>gr</i> : gray, <i>y</i> : yellow
marquee	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	text	the text
	color	the text color; use standard HTML color codes
	size	the text font size, in pixels
	width	the text width, in pixels
	align	the text alignment: <i>left</i> , <i>right</i> , <i>center</i> , <i>justify</i>

object type	attribute	value
	weight	the text boldness: <i>lighter</i> , <i>normal</i> , <i>bold</i> , <i>bolder</i>
	style	the text style: <i>normal</i> , <i>italic</i> , <i>oblique</i>
	height	height of the scrolling area
	paused	true: pause scrolling; false: resume scrolling
panel	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	color	<i>r</i> : red, <i>g</i> : green, <i>b</i> : blue, <i>gr</i> : gray, <i>y</i> : yellow. Set color to an empty string to reset the panel's color to the transparent default
slider	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	value	<i>on</i> , <i>off</i> , value from 0% to 100%, from 0.00 to 1.00, from 0/<max> to <max>/<max>
submit	visible	<i>true</i> : visible; <i>false</i> : not visible
submitmini	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
submitmicro	pos	the object position
	text	the text label
	color	<i>r</i> : red, <i>g</i> : green, <i>b</i> : blue, <i>gr</i> : gray, <i>y</i> : yellow
submitimage	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	img	name of a customized file with the image of the object, saved in the www/img directory
text	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	text	the text
	color	the text color; use standard HTML color codes
	size	the text font size, in pixels

object type	attribute	value
	width	the text width, in pixels
	align	the text alignment: <i>left, right, center, justify</i>
	weight	the text boldness: <i>lighter, normal, bold, bolder</i>
	style	the text style: <i>normal, italic, oblique</i>
user	visible	<i>true</i> : visible; <i>false</i> : not visible
usermini	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
usermicro	pos	the object position
	text	the text label
	color	<i>r</i> : red, <i>g</i> : green, <i>b</i> : blue, <i>gr</i> : gray, <i>y</i> : yellow
userimage	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	img	name of a customized file with the image of the object, saved in the <code>www/img</code> directory
	text	the text label
userrgb	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	text	the text label
	color	the button's color; use standard HTML color codes
video	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	play/pause	play/pause of video
	mode	play mode: <i>stop, auto</i> and <i>loop</i>

Examples:

```
UISET mytext.style = italic
UISET mychart. barcolor = "#000000"
```


Java Programming

The 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, that is 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) 2011 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 standard setting of the ***AutoKillFiles*** parameter in hsyco.ini cause the automatic restart of HSYCO when user.class is changed.

This chapter 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 not use other public methods provided by other classes within the HSYCO package.

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	action and status OFF
ON	action and status ON
PAUSE	status PAUSE (only Squeezebox music player)
PLAY	status PLAY (only Squeezebox music player)
UNKNOWN	unknown status value

Callback Methods

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; for example:

```
OpenWebNetMonitor - Exception in user event call: ...
```

I/O Servers

public 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.

Parameters:

serverIndex - 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.

public static void IOEvent(String id, String value)

Called by a status change of any data port of an I/O server.

Parameters:

id - the server id and the 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 the ioServers parameter in hsyco.ini, followed by a dot and the input/output port name
value - the new value. For binary inputs or outputs, the returned values are "0" or "1".

Timers and Schedulers

public static boolean UserTimerEvent(String name, boolean active)

This method is 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 - timer name or scheduler's rule name

active - *true* when activating and *false* when deactivating the timer or scheduler's rule.

DMX

public 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.

It is a blocking method. The execution of this method must be completed before the monitor thread is started.

Parameters:

serverIndex - number of the DMX bus, starting from 0.

public static void DmxEvent(int channel, int state)

Called after changes to DMX-512 channels.

Parameters:

channel - DMX channel address, from 1 to 512 which might be preceded by the DMX gateway number, starting from 0
state - the new channel value, from 0 to 255.

public 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 - the DMX channel address, from 1 to 512 which might be preceded by the DMX gateway number, starting from 0
- state - the unfiltered value of the channel, from 0 to 255
- reverse - *false* if called when writing to the gateway; *true* if called when reading from the gateway.

Cameras

public static void CameraMotionEvent(String eventName, long remoteTime)

Called when HSYCO starts recording video from a camera.

Recording could start automatically 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 call or the CAMERAREC or CAMERARECFULL actions in EVENTS.

Parameters:

eventName - the event identification string, in the <name>:<id> format

remoteTime - the current time in milliseconds.

public 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.


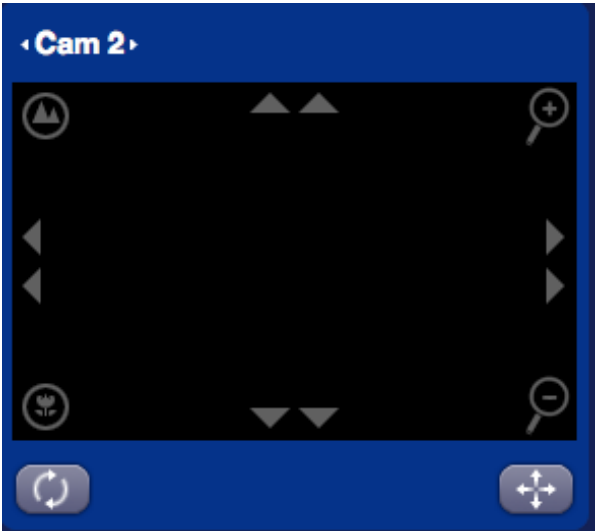







If the method returns -1, the standard PTZ command associated to the PTZ driver is skipped, otherwise it is executed normally.

Parameters:

function - see the table below

action - see the table below

camera - the camera name.

	function	action	
	focus	far	
	focus	near	
	zoom	tele	
	zoom	wide	
	move	left	
	move	right	
	move	up	
	move	down	
Center	move	stop	

IRTrans

public 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.

Parameters:

received - *true* if the command has been received²¹ by the IRTrans, *false* if sent

irtransid - IRTrans number, starting from 0 for the first device, based on the listing order of the IRTrans parameter in hsyco.ini

event - the string of the command received or issued, formatted as:

<remote control>,<command>

²¹ That is, if the devices detected a command issued by a normal infrared remote control.

Squeezebox

public static void SlimPowerEvent(int index, int power)

Called when a Squeezebox player is turned on or off.

Parameter:

index - player number, starting from 0 for the first player, based on the listing order of the slimPlayers parameter in hsyco.ini
power - ON or OFF.

public static void SlimStatusEvent(int index, int status)

Called when a Squeezebox player changes its playback mode.

Parameters:

index - player number, starting from 0 for the first player, based on the listing order of the slimPlayers parameter in hsyco.ini
status - ON, OFF, PLAY, PAUSE, UNKNOWN.

public static void SlimVolumeEvent(int index, int volume)

Called when a Squeezebox player changes its volume level.

Parameters:

index - player number, starting from 0 for the first player, based on the listing order of the slimPlayers parameter in hsyco.ini
volume - an integer number representing the volume change²² in a scale from 0 to 100; positive numbers correspond to a volume increase, negative numbers to a decrease.

²² This parameter does not return the absolute audio level.

PBX

public 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.

Parameters:

host - the IP address of the PBX server

caller - the caller number

called - the called number.

Network Services

public static void LocationEvent(long MAC, InetAddress IP, int zoneId)

This method is 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 - the MAC address of the client

IP - 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 - 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.

System Methods

public static void StartupEvent()

This method is 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.

public 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.
`public static void TimeEvent(long time)`

public 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 - the current time in milliseconds.

public static void programTimerEvent(String name)

This method is called when a program timer is activated. Program timers are set using `programTimerSet()`, `programTimerReset()`, `programTimerRepeat()`, or using the corresponding actions in the EVENTS.

Parameters:

name - program timer name.

public 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 userCommand() Java method.

It is also called when a (submit!id), (submitmini!id) or (submitmicro!id) button is pressed.

The method returns a string. If it returns null, an [ERROR] status is logged in the log file, while any other 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 - the <name> field of the (user) object. If called on the (submit!id) object, this parameter is set to the id of the object

param - the <param> of the (user) object. If called on the (submit!id) object, this parameter is set to the id-value pairs of all (input) fields, with the "@" character as separator.

public 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 - *true* at sunrise, *false* at sunset.

public 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.

If PowerEvent() returns -1 or if not available in user.java, HSYCO status is updated with the detected power value, the value returned by PowerEvent() is otherwise used. Thanks to this method it is possible to alter the power value shown in the Web interface, for example to aggregate power readings acquired from other sensors.

Parameters:

power - the power level, in Watts.

public static void SunPositionEvent(int azimuth, int elevation)

This method is called when the Sun changes its height with respect to the horizon or its angle from true north.

Parameters:

azimuth - the current Sun angle from true north, in decimal degrees

elevation - the current Sun elevation in decimal degrees from the horizon.

Elevation is negative at night.

public 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 - the HTTP client's address

secure - true if this is an HTTPS request

useragent - the Web browser's user agent information.

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 - the scheduler's group name

schedulename - the scheduler's name.

Command and Utility Methods

These methods are used to control field devices, to acquire status information and to execute general utility functions.

Commands that execute relevant functions will be logged in the daily log file:

2011.07.30 11:39:16.133 - USERCODE: <method> - <parameter>

I/O Servers

static void ioSet(String id, String value)

Writes to I/O servers variables or data points.

Parameters:

id - the server id and the 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 the ioServers parameter in hsyco.ini, followed by a dot and the input/output port name, for example: "contacts.o1"

value - the new value.

static String ioGet(String id)

Returns the current status of an input or output of an I/O gateway.

Parameters:

id - the server id and the 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 the ioServers parameter in hsyco.ini, followed by a dot and the input/output port name.

DMX

static int dmxCSet(int channel, int state)

Controls the value of a single DMX-512 channel.

Returns 1 if successful, 0 in case of errors.

Parameters:

channel - 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
state - the new value of the channel, from 0 to 255.

static int dmxCSet(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. Returns 1 if successful, 0 in case of errors.

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 - 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 - an array that contains the list of values for each channel, from 0 to 255.

static int dmxCOff(int channel)

Sets a channel on the DMX-512 bus to 0. It also saves the previous value, so it can be restored with dmxCOn().

Returns 1 if successful, 0 in case of errors.

Parameters:

channel - 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.

static int dmxCOff(int[] channels)

Sets a list of DMX-512 channels to 0, also saving the previous values, so they can be restored with dmxCOn(). 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.

Returns 1 if successful, 0 in case of errors.

Parameters:

channels - 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. 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.

static int dmxBOn(int channel)

Sets a DMX-512 channel to the value preceding the last call to dmxBOff().
Returns 1 if successful, 0 in case of errors.

Parameters:

channel - 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.

static int dmxBOn(int[] channels)

Sets a list of DMX-512 channels to the values preceding the last call to dmxBOff() 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. Returns 1 if successful, 0 in case of errors.

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 - 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. 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.

static int dmxFMerge(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.

Returns 1 if successful, -1 if *to* < *from*, 0 in case of a generic error.

Parameters:

from - 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 - last address of the range

merge - *true* to enable merge mode, *false* to return to normal mode.

static int dmxFMerge(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.

Returns 1 if successful, 0 in case of errors.

Parameters:

channels - 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. 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.

static int dmxFGet(int channel)

Returns the current status of a channel on the DMX-512 bus, with values between 0 and 255, or UNKNOWN if the status is unknown.

Parameters:

channel - 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.

static void dmxCOffTimerSet(int channel, int seconds)

This method sets a power off timer for a DMX-512 channel. If the channel's value is greater than 0 when the method is called, HSYCO will set the channel to 0 exactly after the number of seconds specified by the *seconds* parameter.

The method is ignored if another timer is already set for the same channel. A timer is deleted automatically when the channel changes status. Consequently it is not useful to set a timer on a channel whose current value is 0.

Parameters:

channel - when using one DMX gateway, address will be a standard DMX-512 address between 1 and 512.

seconds - the channel will be turned off after the specified number of seconds.

static void dmxCOffTimerClear(int channel)

Cancels a power off timer for a DMX-512 channel.

Parameters:

channel - when using one DMX gateway, address will be a standard DMX-512 address between 1 and 512.

static void dmxOffTimerPreset(int channel, int seconds)

This method is similar to `dmxOffTimerSet()`, but can be executed when the channel is set to 0. Whenever another command turns the channel ON, the pre-set timer will be triggered, causing an OFF command after the specified number of seconds.

This method will be ignored if another timer is already set for the same channel.

Parameters:

channel - 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

seconds - the channel will be turned off after the specified number of seconds.

static void dmxDmxCmdTimerReset(int channel, int seconds)

Similar to `dmxOffTimerPreset()`, but if another timer is already set for the same device, the previous timer is cancelled and replaced by the new one.

Parameters:

channel - 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

seconds - the channel will be turned off after the specified number of seconds.

Cameras

static void cameraRecTrigger(String camera, String source, int seconds)

Records video from a camera.

Parameters:

- camera - the camera's id, as defined by the Cameras parameter in hsyco.ini
- source - a string used to generate a log message, to easily identify the source that triggered the recording
- seconds - recording time in seconds. If set on 0, stops the current recording.

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 - the camera's id, as defined by the Cameras parameter in hsyco.ini
- source - a string used to generate a log message, to easily identify the source that triggered the recording
- seconds - recording time in seconds. If set on 0, stops the current recording.

static int cameraCommand(String function, String action, String camera)









Sends PTZ and focus commands to cameras. The returned value is included in the log message, and has no other purpose.

Parameters:

function - see the table below

action - see the table below

camera - the camera name.

	function	action
	focus	far
	focus	near
	zoom	tele
	zoom	wide
	move	left
	move	right
	move	up
	move	down
Center	move	stop

static void cameraMode(String camera, boolean enabled)

Disables or enables a camera. When the live view is disabled, recording is also disabled.

Parameters:

camera - the camera name

enabled - *true* to enable the camera, *false* to disable.

static void cameraRecMode(String camera, boolean enabled)

Disables or enables recording from a camera, not affecting live view.

Parameters:

camera - camera name

enabled - *true* to enable recording, *false* to disable.

Serial Communication Ports

static int writeComm(String portName, String data)

Sends a sequence of bytes to a communication port.

Returns 0 in case of errors, or the number of bytes sent if successful.

If both `verboseLog = true` and `userLog = true`, the full trace of received and sent bytes is written to the log file.

Parameters:

`portName` - the name of the communications port, as defined in the `CommPorts` parameter in `hsyco.ini`

`data` - a string with the hexadecimal representation of the sequence of bytes to be sent. For example "45AB4400BA".

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 port²³, until the maximum number specified in the `len` parameter.

If `len` is set to 0, the function doesn't return data, but it empties the communication port's buffer from older data.

Returns `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".

If both `verboseLog = true` and `userLog = true`, the full trace of received and sent bytes is written to the log file.

²³ There is a time-out of 2 seconds on data reading. In case the buffer is empty at the moment of execution of the function, this remains in hold till time-out.

Parameters:

portName - the name of the communications port, as defined in the CommPorts parameter in hsyco.ini

len - maximum number of bytes to be read from the port, or 0 to empty the buffer without reading data.

static int closeComm(String portName)

Closes a serial port connection when the serial port is defined as 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 - the name of the communications port, as defined in the CommPorts parameter in hsyco.ini.

Plugins

static Bentel getBentelPlugIn(String id)

Retrieves the object instance of a BENTEL security system, and allows you to use the Bentel add-on class methods in your Java code. Refer to the Bentel Application Note document for additional information.

Returns the Bentel object instance, or *null* in case of error.

Parameters:

id - the id of the Bentel unit, as set in the bentel.ini add-on configuration file.

static ParadoxEVO getParadoxEVOPlugIn(String id)

Retrieves the object instance of a Paradox EVO security system, and allows you to use the ParadoxEVO add-on class methods in your Java code. Refer to the Paradox EVO Application Note document for additional information.

Returns the ParadoxEVO object instance, or *null* in case of error.

Parameters:

id - the id of the EVO unit, as set in the paradox.ini add-on configuration file.

static Tecnoalarm getTecnoalarmPlugIn(String id)

Retrieves the object instance of a Tecnoalarm security system, and allows you to use the Tecnoalarm add-on class methods in your Java code. Refer to the Tecnoalarm Application Note document for additional information.

Returns the Tecnoalarm object instance, or *null* in case of error.

Parameters:

id - the id of the Tecnoalarm unit, as set in the tecnoalarm.ini add-on configuration file.

Modbus

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. If there is an error, the first byte is set to 0x81 and the second byte to the error code.

This method 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 method to access gateways that are not defined as I/O servers.

Parameters:

- name - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit - the MODBUS slave unit identifier (1 to 247)
- address - the starting address of the block of coils
- quantity - the quantity of coils you are reading (1 to 2000).

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. If there is an error, the first byte is set to 0x82 and the second byte to the error code.

This method 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 method to access gateways that are not defined as I/O servers.

Parameters:

- name - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit - the MODBUS slave unit identifier (1 to 247)
- address - the starting address of the block of discrete inputs
- quantity - the quantity of discrete inputs you are reading (1 to 2000).

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 method 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 method to access gateways that are not defined as I/O servers.

Parameters:

- name - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit - the MODBUS slave unit identifier (1 to 247)

address - the starting address of the block of registers

quantity - the quantity of registers you are reading (1 to 125).

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 method 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 method to access gateways that are not defined as I/O servers.

Parameters:

name - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway

unit - the MODBUS slave unit identifier (1 to 247)

address - the starting address of the block of registers

quantity - the quantity of registers you are reading (1 to 125).

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 method 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 method to access gateways that are not defined as I/O servers.

Parameters:

- name - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit - the MODBUS slave unit identifier (1 to 247)
- address - the coil's address
- value - true to turn the coil on, false to turn it off.

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 method 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 method to access gateways that are not defined as I/O servers.

Parameters:

- name - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit - the MODBUS slave unit identifier (1 to 247)
- address - the register's address
- bytes - data to be written in the register, using big-endian encoding (the most significant byte is sent first).

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 method 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 method to access gateways that are not defined as I/O servers.

Parameters:

- name - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit - the MODBUS slave unit identifier (1 to 247)
- address - the starting address of the block of registers
- bytes - data to be written in the registers, using big-endian encoding (the most significant byte is sent first).

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 method 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 method to access gateways that are not defined as I/O servers.

Parameters:

- name - the id of the I/O Server as defined in hsyco.ini, or the host name or IP address of the MODBUS gateway
- unit - the MODBUS slave unit identifier (1 to 247)
- address - the register's address
- mask - the AND mask to be applied to the register
- bytes - data to be written in the register, using big-endian encoding (the most significant byte is sent first).

IRTrans

static int irtransCommand(String irlId, String command)

Sends a command to an IRTrans. Returns 0 in case of errors.

Parameters:

irlId - identifies the IRTrans, as defined with the IRTrans parameter in hsyco.ini

command - the sequence of IR commands to be sent. Every command is composed by the name that identifies the commands database²⁴, 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. For example:

```
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 command²⁵. 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.

²⁴ 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.

²⁵ Even if a @N delay before an IR command is not specified, this method will wait 500ms before sending each command; with @0 this pause will be reduced to 1ms for the following commands.

Squeezebox

static String slimCommand(int index, String command)

Controls the Squeezebox players. It returns *null* in case of errors, otherwise the protocol answer to the command is returned.

Parameters:

index - the Squeezebox number, starting from 0, in the order defined by the *slimPlayers* parameter in *hsyco.ini*
command - the Squeezebox command²⁶.

static String slimCommand(String playerName, String command)

Controls the Squeezebox players. It returns *null* in case of errors, otherwise the protocol answer to the command is returned.

Parameters:

playerName - the Squeezebox name, as listed in the *slimPlayers* parameter in *hsyco.ini*
command - the Squeezebox command.

static int slimButton(int index, String button)

Controls the Squeezebox players, simulating a remote control button. Returns 0 in case of errors, 1 if successful.

Parameters:

index - the Squeezebox number, starting from 0, in the order defined by the *slimPlayers* parameter in *hsyco.ini*

²⁶ The commands are defined in the documentation of the Command Line Interface (CLI) of SqueezeCenter.

button - the commands listed in the table below, or any command supported by the **button** instruction of the Command Line Interface of the Squeezebox Server. You can send a list of commands separated by the “;” character.

button	Description
x_volupxN	increases the volume by N steps
x_voldnxN	decreases the volume by N steps
x_poweron	It turns the Squeezebox on
x_poweroff	It turns the Squeezebox off
x_pauseon	pause
x_pauseoff	resumes playing, if possible
unsync	cancels the synchronization of this player with another and turns it off
syncN	synchronizes player N with this player, turning both players on

static int slimButton(String playerName, String button)

Controls the Squeezebox players, simulating a remote control button. Returns 0 in case of errors, 1 if successful.

Parameters:

playerName - the Squeezebox name, as listed in the *slimPlayers* parameter in hsyco.ini
 button - see the table above.

Public Announcement

The `audioPlay()` methods are 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 Public Announcement Appendix 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 <code>hsyco.ini</code>
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)

static int audioPlay(String where, File file)

Plays a pre-recorded audio file. It returns -1 in case of errors, or the index of this message in the play back queue.

Parameters:

where - the audio destination; see the table above
 file - the audio file.

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. It returns -1 in case of errors, or the index of this message in the play back queue.

Parameters:

where - the audio destination; see the table above

voice - the voice name for the text-to-speech engine; see the Public Announcement Appendix for additional information

text - the text message for text-to-speech conversion.

Data Logger

static boolean dataLoggerRefresh(String name)

Forces the data logger to check for data consistency with respect to the current time, i.e. it will perform the shift of the data to the previous period charts whenever necessary.

Parameters:

name -data logger name.

static boolean dataLoggerUpdate(String name, double value)

Supplies the specified data logger with a new value to be processed.

Parameters:

name - data logger name

value - value to be processed.

static boolean dataLoggerClear(String name)

Clears the data gathered in the specified data logger.

Parameters:

name - data logger name.

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**: the method will 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 - log type (*log* or *stat*)

names - data logger names

path - path of the file to be created

timestamp - if true and the type is *log* adds a timestamp column in the CSV file for the logged values.

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 - data logger name

param - option to be set (“origin” is the only option you can currently set with this method)

value - value for the specified option.

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 whenever a flow counter is incremented. 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.

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 - leak detector name.

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.

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.

Both period and deviation parameters can be changed anytime, even when the detector is already processing data.

Parameters:

name - leak detector name

param - set to “period” or “deviation”

value - the period or deviation value, in String format.

public static int leakDetectorUpdate(String name, double value)

Calls the leak detector passing an updated flow value.

Returns -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.

Parameters:

name - leak detector name

value - the flow counter value.

Mail

```
static int sendMail(String to, String from, String subject, String body)
```

Sends a simple 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.

This method does not retry sending the message if the destination mail server is not available when the method is called.

This method returns 1 if the message is sent successfully, 0 if a mail server is not available for the destination address, or -1 on generic errors.

Parameters:

to - 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 - a valid email address to be used as the sender address

subject - the message subject

body - the message text.

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 SmtplibName 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.

This method does not retry sending the message if the destination mail server is not available when the method is called.

This method returns 1 if the message is sent successfully, 0 if a mail server is not available for the destination address, or -1 on generic errors.

Parameters:

to - 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 - a valid email address to be used as the sender address

subject - the message subject

body - 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.

Log

static void messageLog(String message)

Generates an information message in the daily log file.

Parameters:

message - the text of the message.

static void errorLog(String message)

Generates an error message in the daily log file.

Parameters:

message - the text of the error message.

Network Services

ping(String host, int timeout)

Sends an IP network ping command to the host specified.

Returns *true* if the host is reachable, *false* otherwise.

Parameters:

host - the hostname or IP address

timeout - timeout in milliseconds

static int wakeOnLan(String broadcast, String address)

Sends a Wake on LAN command to the LAN specified in the broadcast Returns -1 in case of errors, 1 if successful.

Parameters:

broadcast - the LAN broadcast address. For example "192.168.0.255"

address - 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 will have to be on to allow HSYCO to determine the MAC address.

²⁷ The association cache between IP and MAC address is saved in the wol.data file in the HSYCO main directory.

System Methods

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*.

static boolean isDaylight()

Returns *true* during the day, *false* at night. The result computed based on the latitude and longitude data set in hsyco.ini and the SunriseOffsetMinutes and SunsetOffsetMinutes parameters.

static long getNextSunrise(long now, boolean withoffset)

This method 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 - the reference time in milliseconds

withoffset - *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 time²⁸.

static long getNextSunset(long now, boolean withoffset)

This method 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.

²⁸ Based on civil dawn, begins when the geometric center of the sun is 6° below the horizon.

Parameters:

now - the reference time in milliseconds

withoffset - *true* to take the sunset offset into account, as defined with the `SunsetOffsetMinutes` in `hsyco.ini`, *false* to ignore the offset and return the actual sunset time²⁹.

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 - the program timer name

seconds - the number of seconds to the timer's execution.

static void programTimerClear(String name)

Cancels a program timer.

Parameters:

name - the program timer name.

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 - the program timer name

seconds - the number of seconds to the timer's execution.

²⁹ Based on civil dusk, ends when the geometric center of the sun reaches 6° below the horizon.

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 - the program timer name

seconds - the interval in seconds of the timer's execution.

static void sleep(long millis)

Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.

Parameters:

millis - the length of time to sleep in milliseconds.

static void schedulerRegister(String groupname, String schedulename, int interval)

This method registers a new scheduled callback; interval is defined in milliseconds.

Parameters:

groupname - the group name

schedulename - the schedule name

interval - interval in milliseconds

static void SchedulerRemove(String groupname, String schedulename)

This method deletes a schedule.

Parameters:

groupname - the group name

schedulename - the schedule name

static void user(String name, String param)

This method triggers a USER event and the userCommand(String name, String param) Java method. It can be used as a calling mechanism between Java and EVENTS.

Parameters:

name - the name parameter, passed to userCommand()

param -the param parameter, passed to userCommand()

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 - the variable name. Names are case-insensitive.

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

user.java 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 - the variable name. Names are case-insensitive

value - the value to be assigned to the program variable.

static String uiGet(String id, String attr)

Returns the current value of an attribute of an identified UI object, set with the uiSet() method or the UISET action in EVENTS. Returns *null* if the object attribute has not been previously set with these methods.

Parameters:

id - object id

attr - attribute name

value - attribute value.

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, for example:

```
(text!thisobject ... )
```

Parameters:

id - object id

attr - attribute name; see the table below

value - attribute value; see the table below.

object type	attribute	value
project	page	set id to the directory name of the index.hsm file you want to control, or to “*” to control all Web clients. Setting value to a page id or to “menu” will automatically force all Web clients that have that index.hsm visible to switch to the specified page (if a project doesn’t have that page, the command is ignored). Setting value to an empty string will automatically revert to the previous page
	lock	set id to the directory name of the index.hsm file you want to control, or to “*” to control all Web clients. Setting value to a page id or to “menu” will automatically force all Web clients that have that index.hsm visible, and Web clients that are loading that index.hsm in the future, to switch to the specified page, also disabling user’s navigation. Setting value to an empty string unlocks the navigation
	pageback	set id to the directory name of the index.hsm file you want to control, or to “*” to control all Web clients. Set value to a page id to force Web clients that are currently displaying that page to go to the previous page. Set value to a pop-up id to force Web clients that have that pop-up open to close it
page	blink	<i>false</i> : stops blinking of all objects in the page
background	img	name of a customized file with the image of the object, saved in the www/img directory, or a full URL (starting with http:// or https://) to show an image from a remote website
camera	camera	id of the camera to show in this object
camerapanel	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
chart	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position

object type	attribute	value
	type	chart type; supported types: <i>bars</i> , <i>points</i> , <i>gauge</i> . gauge charts only support the following attributes: value, orientation, valuerange, bgcolor, barcolor, barborder; the valuerange attribute is required for a correct graphic representation of a value
	orientation	vertical or horizontal orientation of the value axis; supported values: <i>vertical</i> , <i>horizontal</i>
	value	the value for gauge charts
	values	the comma separated series of values; you can omit some values in the series (e.g. values="1, 3, 2, ,4, , , 5")
	valuerange	minimum and maximum y-axis value, separated by comma. They indicate the graphic range. For example, "-100,100" will show a chart with values between -100 and 100
	axislabels	labels on the x-axis (for vertical charts) or on the y-axis (for horizontal charts). It doesn't need to correspond to the number of values in values, and it can contain empty labels (in this case only the notches are shown)
	drawaxis	specifies if the axis (notches and labels) are shown or not; possible values: <i>true</i> o <i>false</i>
	origin	moves the origin to a value different from 0 (e.g. origin=3.2)
	axisoffset	the axis offset (y if vertical, x if horizontal) expressed as percentage or as the value series index (ex. "50%" or "3")
	baroffset	shifts the value bars or points by the given amount of pixels, so you can partially overlap multiple charts
	notches	number of notches on the x-axis (for vertical charts) or on the y-axis (for horizontal charts)
	spacing	spacing between the bars expressed as a percentage (use for the bar chart type)
	pointsize	points size (use for the points chart type)
	vlabelsstyle	value labels style; supported values: <i>inside</i> , <i>outside</i> , <i>none</i>
	bgcolor	background color; use standard HTML color codes
	barcolor	series of HTML colors, one for each data point in <i>values</i> ; use standard HTML color codes
	barborder	bars/points border visibility; supported values: <i>true</i> , <i>false</i>
	axiscolor	axis HTML color; use standard HTML color codes

object type	attribute	value
	notchcolor	HTML color of the axis notches; use standard HTML color codes
	labelcolor	HTML color of the axis and bars labels; use standard HTML color codes
container	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
dlink	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	text	the text label
	color	<i>r</i> : red, <i>g</i> : green, <i>b</i> : blue, <i>gr</i> : gray, <i>y</i> : yellow
image imagelink	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	img	name of a customized file with the image of the object, saved in the www/img directory, or a full URL (starting with http:// or https://) to show an image from a remote website
	text	the text label
input keypad	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	text	the text
	color	the text color; use standard HTML color codes
	size	the text font size, in pixels
	width	width, in pixels
	height	height, in pixels
	weight	the text boldness: <i>lighter</i> , <i>normal</i> , <i>bold</i> , <i>bolder</i>
	style	the text style: <i>normal</i> , <i>italic</i> , <i>oblique</i>
link	visible	<i>true</i> : visible; <i>false</i> : not visible

object type	attribute	value
linkmini	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
linkmicro	pos	the object position
	text	the text label
	color	<i>r</i> : red, <i>g</i> : green, <i>b</i> : blue, <i>gr</i> : gray, <i>y</i> : yellow
marquee	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	text	the text
	color	the text color; use standard HTML color codes
	size	the text font size, in pixels
	width	the text width, in pixels
	align	the text alignment: <i>left</i> , <i>right</i> , <i>center</i> , <i>justify</i>
	weight	the text boldness: <i>lighter</i> , <i>normal</i> , <i>bold</i> , <i>bolder</i>
	style	the text style: <i>normal</i> , <i>italic</i> , <i>oblique</i>
	height	height of the scrolling area
	paused	<i>true</i> : pause scrolling; <i>false</i> : resume scrolling
panel	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	color	<i>r</i> : red, <i>g</i> : green, <i>b</i> : blue, <i>gr</i> : gray, <i>y</i> : yellow. Set color to an empty string to reset the panel's color to the transparent default
slider	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	value	<i>on</i> , <i>off</i> , value from 0% to 100%, from 0.00 to 1.00, from 0/<max> to <max>/<max>
submit	visible	<i>true</i> : visible; <i>false</i> : not visible
submitmini	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
submitmicro	pos	the object position
	text	the text label

object type	attribute	value
	color	<i>r</i> : red, <i>g</i> : green, <i>b</i> : blue, <i>gr</i> : gray, <i>y</i> : yellow
submitimage	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	img	name of a customized file with the image of the object, saved in the www/img directory
text	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	text	the text
	color	the text color; use standard HTML color codes
	size	the text font size, in pixels
	width	the text width, in pixels
	align	the text alignment: <i>left</i> , <i>right</i> , <i>center</i> , <i>justify</i>
	weight	the text boldness: <i>lighter</i> , <i>normal</i> , <i>bold</i> , <i>bolder</i>
	style	the text style: <i>normal</i> , <i>italic</i> , <i>oblique</i>
user	visible	<i>true</i> : visible; <i>false</i> : not visible
usermini	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
usermicro	pos	the object position
	text	the text label
	color	<i>r</i> : red, <i>g</i> : green, <i>b</i> : blue, <i>gr</i> : gray, <i>y</i> : yellow
userimage	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	img	name of a customized file with the image of the object, saved in the www/img directory
	text	the text label
userrgb	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position

object type	attribute	value
	text	the text label
	color	the button's color; use standard HTML color codes
video	visible	<i>true</i> : visible; <i>false</i> : not visible
	blink	<i>true</i> : slow blinking; <i>slow</i> : slow blinking; <i>fast</i> : fast blinking; <i>false</i> : stop blinking
	pos	the object position
	play/pause	play/pause of video
	mode	play mode: <i>stop</i> , <i>auto</i> and <i>loop</i>

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 - the power in Watt units.

Appendix

Data Loggers

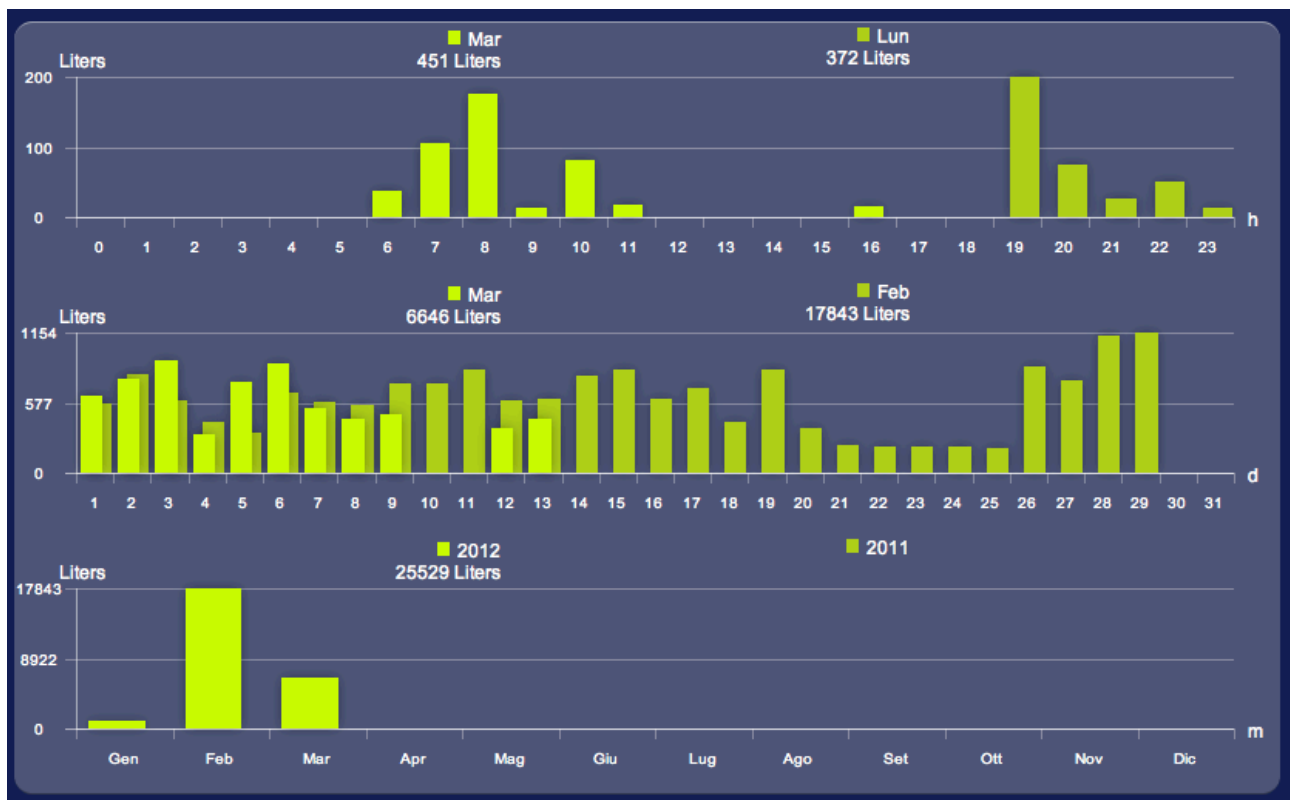
Data loggers allow to gather, process and visualize statistical data on the variations of a data set. They can be used to automatically generate periodic charts of its trend and/or to log data as CSV files.

Data loggers collect data and group them in order to visualize the trends for each hour (or group of hours) of the current and past day, each day of the current and past month, and each month of the current and past year. Further, they allows for the logging of every processed value.

There are two types of data logger:

- **counter**: suitable for the creation of statistics of an incremental value (e.g. energy consumption or production). It calculates the variation (delta) of the value for each time interval with respect to the previous one. Further, it is possible to specify time slots and the relative rates to calculate for instance the costs for energy consumption.
- **range**: aimed at the computation of the maximum, minimum, and average of a fluctuating value (e.g. a measured temperature or humidity) for each time interval.

Once a data logger is defined in *hsyco.ini*, you can easily pass data to it using a simple action or Java method, and display statistical data using the (datalogger) graphic object.



HSYCO will also optionally³⁰ update a set of generic charts, allowing you greater control on the graphic representation of your data than the standard (datalogger) object. The available charts depend on the type of the defined data logger.

Counter Data Loggers

The first step to create a counter data logger is to initialize it in *hsyco.ini*. Refer to the data loggers section in the configuration chapter for a complete description of all the available parameters and options. Here we declare a counter data logger named “energy” with default parameters.

In *hsyco.ini* add the following lines:

```
DataLoggers = energy
DataLoggers.energy.Type = counter
```

³⁰ Charts are automatically updated if `DataLoggers.<id>.UseCharts = true`.

Now, we need to provide the data logger with the incremental value we want to process. Assuming we have an I/O server named “elektro” which provides the energy consumption value, we write the following lines in *events.txt*:

```
IO elektro : DATALOGGER energy = IO elektro,
            DATALOGGER energy = FILE LOG energy.csv
```

Whenever the I/O server will provide a new value, the data logger will be updated and a log entry will be added to the file “energy.csv”.

Finally, in order to display the data logger’s charts, we should use the (datalogger) object, for example:

```
(datalogger energy; x80y60; 820; 500; kWh;)
```

For a complete reference of all the available actions on data loggers refer to the relative section in the Events programming chapter. The same actions can be performed using Java code (see section Datalogger in the Java programming chapter).

Adding the following line in *hsyco.ini*:

```
DataLoggers.energy.UseCharts = true
```

HSYCO will automatically create a set of charts with the following IDs:

```
energy.counter.chart.day (chart of the current day)
energy.counter.chart.day.past (chart of the past day)
energy.counter.chart.month (chart of the current month)
energy.counter.chart.month.past (chart of the past month)
energy.counter.chart.year (chart of the current year)
energy.counter.chart.year.past (chart of the past year)
```

Thus, by simply adding a chart in our Web interface and setting its ID field to one of the above listed, will result in having an automatically updated chart showing the data relative to the specified period.

Also, by setting the ID field of a textual object (e.g. *text*) to the same ID of a chart extended with “.total” (e.g. *energy.counter.chart.day.total*) will result in having a label showing the total value variation in the specified period.

We could also create a data logger to monitor the expenses relative to energy consumption, initializing a new counter data logger and declaring the time slot rates in the file *dataloggers.ini*.

In *hsyco.ini* we declare our new data logger:

```
DataLoggers = cost
DataLoggers.cost.Type = counter
```

Then, we set the elektro I/O server to update the data logger. In *events.txt* we add this line:

```
IO elektro OR TIME : DATALOGGER cost = IO elektro
```

Note that this time we are updating the data logger not only when a new value is available, but also every minute. This is necessary to have an accurate calculation of the expenses.

Now we need to declare our time slots in the file *dataloggers.ini*, for instance:

```
cost; 0; 00:00-11:59; *; *; 0.3
cost; 1; 12:00-16:59; *; *; 1.4
cost; 2; 17:00-23:59; *; *; 0.8
```

These rules state that every day from midnight to 11:59:59 we are in slot 0 and the rate to be applied is 0.3, from 12:00:00 until 16:59:59 we are in slot 1 and the applicable rate is 1.4, later on slot 2 applies with a rate of 0.8.

We can include some exceptions by adding some lines before these rules:

```
cost; 0; *; *; 2000/12/10-2000/12/31; 0.3
cost; 2; *; 67; *; 0.8
cost; 0; 00:00-11:59; *; *; 0.3
cost; 1; 12:00-16:59; *; *; 1.4
cost; 2; 17:00-23:59; *; *; 0.8
```

The previous rules will still apply, except on Saturdays (6) and Sundays (7) when during all day slot 2 will apply. Further, during the period between December 10th and December 31st of 2000, slot 0 will always apply despite the day of the week or the time.

Refer to the section about the `dataloggers.ini` file for a complete reference on how to write time slots rules.

At this point we should add the (datalogger) object to the Web interface to display the charts and totals. The user interface also allows you to filter data by rate.

We could also add the individual charts showing the periodical energy consumption expenses by simply setting the appropriate ID among the following:

```
cost.counter.chart.day
cost.counter.chart.day.past
cost.counter.chart.month
cost.counter.chart.month.past
cost.counter.chart.year
cost.counter.chart.year.past
```

If desired, we can have different monthly and yearly charts for each defined slot. This is achieved by setting the parameter *SeparateCharts* to true in *hsyco.ini*:

```
DataLoggers = cost
DataLoggers.cost.Type = counter
DataLoggers.cost.UseCharts = true
DataLoggers.cost.SeparateCharts = true
```

By doing so HSYCO will provide the following charts (considering we defined the slots 0,1, and 2 in *dataloggers.ini*) :

```
cost.counter.chart.day
cost.counter.chart.day.past
cost.counter.slot0.chart.month
cost.counter.slot0.chart.month.past
cost.counter.slot1.chart.month
cost.counter.slot1.chart.month.past
cost.counter.slot2.chart.month
cost.counter.slot2.chart.month.past
cost.counter.slot0.chart.year
cost.counter.slot0.chart.year.past
cost.counter.slot1.chart.year
cost.counter.slot1.chart.year.past
```

```
cost.counter.slot2.chart.year
cost.counter.slot2.chart.year.past
```

Further, we can have a detailed report of all the processed data by specifying the path of a log file in *hsyco.ini*:

```
DataLoggers = cost
DataLoggers.cost.Type = counter
DataLoggers.cost.RatesLogFile = counterRates/log.csv
```

This will create a CSV file listing a row for each update of the data logger divided in the following columns:

data_logger_name, update_time, provided_value, calculated_cost, slot_id, applied_rate

Range Data Loggers

To declare a range data logger, named for instance “temperature”, we add these lines to *hsyco.ini*:

```
DataLoggers = temperature
DataLoggers.temperature.Type = range
```

Note that the average is calculated as the sum of the values divided by the number of records. It is therefore advised to provide the data logger with periodical values so to have a correct calculation.

Now, we set the data logger to follow the temperature measured by a probe identified by the I/O server “temp”.

We specify in events that every minute it must be updated, plus we log the measured value in the file “temp.csv”:

```
TIME : DATALOGGER temperature = IO temp,
      DATALOGGER temperature = FILE LOG temp.csv
```

The created charts will automatically be scaled to range from the lowest measured value to the highest and charts belonging to the same period will be aligned. If for instance we only want to show the chart showing the lowest values, it might be better not to have this chart aligned to the others. To this end

we can specify in *hsyco.ini* to scale the charts of the same period independently:

```
DataLoggers.temperature.PeriodAlign = false
```

Further, by default, the origin of the charts is set to correspond to the lowest measured value. One might prefer to set the origin to a specified value (for instance zero) so to highlight the difference between values below and above it. We can achieve this by setting the *Origin* option:

```
DataLoggers.temperature.Origin = 0
```

Finally, in order to display the data logger's charts, we should use the (datalogger) object, for example:

```
(datalogger temperature; x80y60; 820; 500; C;)
```

Adding the following line in *hsyco.ini*:

```
DataLoggers.temperature.UseCharts = true
```

HSYCO will create charts with the following IDs:

```
temperature.min.chart.day (chart of lowest values of the current day)
temperature.min.chart.day.past (chart of lowest values of the past day)
temperature.avg.chart.day (chart of average values of the current day)
temperature.avg.chart.day.past (chart of average values of the past day)
temperature.max.chart.day (chart of highest values of the current day)
temperature.max.chart.day.past (chart of highest values of the past day)

temperature.min.chart.month (chart of lowest values of the current month)
temperature.min.chart.month.past (chart of lowest values of the past month)
temperature.avg.chart.month (chart of average values of the current month)
temperature.avg.chart.month.past (chart of average values of the past month)
temperature.max.chart.month (chart of highest values of the current month)
temperature.max.chart.month.past (chart of highest values of the past month)

temperature.min.chart.year (chart of lowest values of the current year)
temperature.min.chart.year.past (chart of lowest values of the past year)
temperature.avg.chart.year (chart of average values of the current year)
```

temperature.avg.chart.year.past (chart of average values of the past year)

temperature.max.chart.year (chart of highest values of the current year)

temperature.max.chart.year.past (chart of highest values of the past year)

Just like the counter example, appending “.total” to the ID and assigning it to a textual object, will show the overall minimum, maximum or average of the relative period.

Cameras Configuration

HSYCO interacts with cameras in three ways:

1. using a url defined in the hsyco.ini file, to acquire individual frames, only when necessary³¹ and with the required frequency;
2. receiving, in the form of HTTP request sent by the camera to HSYCO Web server, the motion detection events notifications;
3. sending pan, tilt, zoom and focus commands to cameras with PTZ support.

Users Configuration

You should configure the cameras to allow anonymous or authenticated access.

User and password information for each camera are configured in the hsyco.ini file.

Frames Acquisition

The camera must be enabled to serve JPEG frames requests coming from the HSYCO server.

Motion Detection

To receive motion detection events notifications, HSYCO expects that cameras generate HTTP requests to the HSYCO Web Server, in the following format:

http://192.168.0.50/x/camerarec?camera=<name>

³¹ To reduce the processing load on the cameras and network traffic, HSYCO acquires frames from each camera only in the following cases: 1) after motion detection events; 2) when the Web interface requests the live view of a camera; 3) on recording requests by the customized (user.class) Java code.

or:

http://192.168.0.50/x/camerarec?camera=<name>&zone=<id>

The *camera* and *zone* parameters can be separated by "&", ":" or ";".

<name> is the camera name, corresponding to one of the cameras listed in the *Cameras* parameter in *hsyco.ini*.

<id> is an optional parameter that identifies the active zone of the motion area.

This parameter is optional and, if present, is passed as a parameter to the Java method in *user.class* associated to the motion detection events and can be used at the application level.

For example:

```
http://192.168.0.50/x/camerarec?camera=back
```

```
http://192.168.0.50/x/camerarec?camera=back&zone=1
```

PTZ Control

The camera must be configured to allow PTZ control from the HSYCO server.

Serial Ports Servers Configuration

HSYCO supports remote serial ports through several networked devices, like the IRTrans, HW Group's IO Controller and ERxx, and generic serial gateways that support bi-directional serial port communication via a TCP connection.

IRTrans

The IRTrans only supports serial out traffic. To configure a comm port to send data to the serial port of an IRTrans device, you should set the comm port type to "ir", its id to the IRTrans id, and also set the serial port handshake parameters.

For example:

```
IRTrans = theater
IRTransIP.theater = 192.168.0.130
CommPorts = serial
CommPort.serial.Id = theater
CommPort.serial.Type = ir
CommPort.serial.Params = 115200,8,1,0,0
```

HW Group's Controllers

HW Group's IO Controller and ERxx support bi-directional serial traffic. To configure a comm port to send data to the serial port of an HW Group device, you should set the comm port type to "io", its id to the HW Group Controller's I/O Server id, and also set the serial port handshake parameters.

For example:

```
ioServers = er84
ioServersIP.er84 = 192.168.1.199
ioServersType.er84 = HWGIO
CommPorts = serial
CommPort.serial.Id = er84
CommPort.serial.Type = io
```

Generic 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.

For example:

```
CommPorts = eki  
CommPort.eki.Type = server  
CommPort.eki.IP = 192.168.0.187  
CommPort.eki.Port = 5300
```

HSYCO supports a high-availability, failover configuration for serial gateways. You enable failover by adding a second IP address to the IP parameter:

```
CommPorts = eki  
CommPort.eki.Type = server  
CommPort.eki.IP = 192.168.0.187, 192.168.0.188  
CommPort.eki.Port = 5300
```

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 Java `closeComm(String portName)` method 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.

Wi-Fi Client Location Services

To enable the Wi-Fi client location services, you should define the appropriate parameters in the hsyco.ini file, that is **LocationBases** with the list of all the Access Points and **LocationBaseIP.<name>** with the IP address of each Access Point.

The SYSLOG server must also be enabled, specifying the IP listening port³², with the **SysLogServerPort=<port>** parameter.

Finally, configure each Access Point so that it sends its log messages to the SYSLOG server at the HSYCO server IP address.

³² The standard port for the SYSLOG service on UDP protocol is 514.

Telephony Services

HSYCO has the ability to process call notifications from telephony systems. When using the Grandstream PBX systems, the HSYCO SYSLOG server must be enabled, and the PBX should be configured to send its logs, at **Info** level, to the HSYCO SYSLOG server.

To interface with other telephony systems, each telephony device or the central 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 and causes the execution of the *PBXCallEvent()* callback method and the PBX event in EVENTS.

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 eSpeak³³, with the possibility to use the optional Acapela engine that offers a wide selection of very high quality voices.

On Mac OS X, HSYCO uses the integrated text-to-speech service and all its voices.

³³ Go to <http://espeak.sourceforge.net/> for more information about the eSpeak software.

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 parameters in hsyco.ini 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, 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` parameters in `hsyco.ini` 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).

Log Files Format

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

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 time³⁴, 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 BETA 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.

³⁴ The local time is always used according to the server settings.

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 the 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 device³⁵ 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:

³⁵ The device should be running a Web Browser with an active HSYCO session.

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
```

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 use 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

Field	Format	Description
command	string * matches any string or substring	<p>matching command string.</p> <p>“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.</p> <p>“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.</p> <p>“timer” commands are formatted as <timer_name>=<action>.</p>

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 parameter in hsyco.ini.

SSL Certificates for Cryptography

HSYCO supports the High-Grade cryptography at 256 bits (AES-256) to protect the communication between the Web Browser used by HSYCO Web interface and the HSYCO Web server. The AES-256 algorithm is considered secure for commercial applications (for example, on-line banking and e-commerce) and for the exchange of classified information³⁶.

When the SSL certificate is generated by HSYCO, and it is therefore not signed by an official Certification Authority (CA) recognized by the Web Browser, it is normal for the browser to display a security message during the first access to the site or periodically at the beginning of a new session.

This message asks the user a confirmation about the reliability of the server with which it is trying to start a secure session; since the certificate is not signed by a known CA, the browser can't guarantee the identity of the server. By accepting the certificate, the connection is established and it is possible to proceed normally.

Anyway, even when the certificate is generated by HSYCO, all the exchanged data are protected by the AES-256 cryptography, as with signed certificates. The auto-generated certificate guarantees the same level of cryptographic protection as an equivalent certificate signed by a CA.

The **ServerName** parameter in the **hsyco.ini** configuration file specifies the name used to generate the **SSL** certificate, necessary for the cryptography of the HTTPS Web traffic, and must correspond to the domain name through which HSYCO is accessible via the Internet.

The certificate is contained in the **hsyco.keys** file. When HSYCO is started, if this file is not available, a new SSL certificate is automatically generated

³⁶ In June 2003, the USA government certified the possibility to use the AES-256 algorithm for the protection of information classified at TOP SECRET level.

according to the name defined in `ServerName`. Otherwise, HSYCO simply uses the certificate contained in this file, which could have also been generated by an official ***Certification Authority (CA)***. If an official certificate has not been chosen, this file is thus created and managed by HSYCO without any manual intervention.

- If the name defined in `ServerName` is modified, HSYCO will automatically
- recreate a new `hsyco.keys` file the next time it restarts.

In order to use certificates that are generated by a Certification Authority, you should follow a command line based procedure, using the cryptography tools available in HSYCO SERVER.

In the following example, we'll use the StartCom Ltd³⁷ Certification Authority.

Access to the console via SSH:

```
ssh hsyco@192.168.0.50
```

and execute the following commands in HSYCO main directory, in which the hsyco.keys file is located, or better in a temporary directory, from which it will be sufficient to copy hsyco.keys in HSYCO's main directory at the end of the procedure.

- These commands require a password for the protection of the hsyco.keys file
- and of the certificate contained in it. However, since HSYCO must be able to
- automatically read this file, the password should be the same for all installations.

Always use the password:

hsycopass

in the following commands.

³⁷ Start Commercial (StartCom) Ltd. - <http://www.startssl.com/>

Since this password is not secret, it is important to remember that the signed certificate could be potentially used by anyone having access to the file. The protection of the HSYCO console, using a long and complex password for the hsyco user, becomes for this reason even more important.

1) generate the key for the domain name:

```
keytool -genkey -keyalg rsa -alias hsyco -dname "cn=www.domain.com,
o=domain, ou=.com" -keystore hsyco.keys
```

2) create the certificate request file:

```
keytool -certreq -alias hsyco -file www.domain.com.csr -keystore hsyco.keys
```

3) import the keys of the chosen CA in the file hsyco.keys:

```
wget http://www.startssl.com/certs/ca.crt
```

```
keytool -import -alias startcom.ca -file ca.crt -keystore hsyco.keys
```

```
wget http://www.startssl.com/certs/sub.class1.server.ca.crt
```

```
keytool -import -alias startcom.ca.sub -file sub.class1.server.ca.crt -
keystore hsyco.keys
```

4) once you have the certificate signed by the CA, transfer the content in a file, for example:

```
cat > www.domain.com.signed.crt
```

5) finally, import the certificate in the hsyco.keys file and copy this file to the HSYCO main directory:

```
keytool -import -alias hsyco -file www.domain.com.signed.crt -keystore
hsyco.keys
```


SQL Database

HSYCO has an integrated SQL, HSQLDB³⁸ (HyperSQL Database) database engine you can use in your applications via the standard JDBC API³⁹.

It 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.

³⁸ 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. All HSQLDB documentation is available at: <http://hsqldb.org/>.

³⁹ 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>

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 of 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.

The following examples show how to make a query, an insert and an update operation in a database table.

Query

```
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) {}
```

Insert

```
try {  
    PreparedStatement stmt =  
dbConnection.prepareStatement("insert into mytable values  
(?, ?)");  
    stmt.setString(1, id);  
    stmt.setString(2, value);  
    stmt.execute();  
} catch (Exception e) {}
```

Update

```
try {  
    PreparedStatement stmt =  
dbConnection.prepareStatement("update mytable set value=?  
where id=?");  
    stmt.setString(2, id);  
    stmt.setString(1, value);  
    stmt.execute();  
} catch (Exception e) {}
```

The access.ini File

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

It is possible to temporarily disable access for each user, and define access rights to the www subdirectories of the web interface.

All the manual changes affecting the access.ini file are loaded by HSYCO within 15 seconds after saving the file, it is therefore not necessary to restart HSYCO to make these changes effective.

Each line in access.ini defines the access parameters for one user. The format is:

user_name = parameters

There are no limits to the number of users. Every user must have a unique identification name and PIN code. The PIN is strictly a 5-digit code (0-9). Creating two different users with the same PIN code is a configuration error; HSYCO ignores users with duplicate PIN codes.

user_name can be preceded by the “*” character to identify a user with administrator rights⁴⁰.

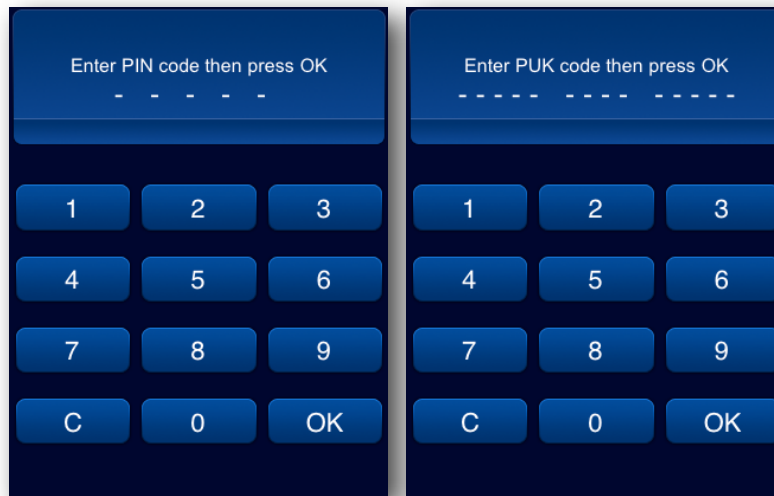
You can create an access.ini file from scratch, or add new users to an existing file, simply adding lines with the user name and PIN, for example:

```
*user=00000
```

⁴⁰ All the administrator rights allow the administrator to create, modify and manage the access rights of other users of the HSYCO Web interface.

After saving the access.ini file with the new users, you can complete the registration through the Web interface:

<https://192.168.0.50/small/pin.hsycoserver>



Type one of the new PIN codes and press [OK], then enter a 14-digit PUK code and press [OK]. This code becomes the PUK code associated to the PIN and is saved by HSYCO in the access.ini file in an encrypted format.

For security reasons, the PIN and PUK codes are stored in a not-readable format, and it is not easy to retrieve the codes associated to a user. It is therefore recommended to write down these codes in a safe place.

In case a user's PIN/PUK codes are forgotten or lost, it will be then necessary to reset the codes, deleting the user and creating a new one.

Leaving PIN codes readable in the access.ini file, that is, without having completed the PUK code association procedure, creates a security risk. You should always complete the procedure for all new users defined in access.ini.

In order to guarantee the highest security, the access.ini file must be treated as a secret document⁴¹, avoiding copying or sending it via not-secure systems and restricting any access by unauthorized subjects.

You can revoke a user's access by simply deleting the entire user definition line from the access.ini file, without restarting HSYCO. The access is revoked within a few seconds after the access.ini file is modified, even for the ongoing active sessions.

To temporarily disable access for a single user without deleting the whole definition, just add the “#” character right after *user_name*=, as in the following example:

```
#Generated by HSYCO 3.0.0
#Wed Jan 04 14:25:29 CET 2012
user=#55634e926185944768b4f3,d29995f54ef9b1b169e7c,
1230989129162,0,192.168.0.180
```

To restrict access of a user to one or more www subdirectories of the Web interface, specify the names of the authorized subdirectories, separated by spaces, in the user definition, right after the comma of the second parameter:

```
user=55634e926168b4f3,d29995f54e7c,garage garden,
1230989129162,0,192.168.0.180
```

In this example, the PIN code associated to the user can only access the Web interfaces defined in the subdirectories www/garage and www/garden.

Up to three different subdirectories can be specified for each user.

⁴¹ Even though the content of access.ini is partly encrypted, it is theoretically possible to retrieve the PIN and PUK codes with special decryption procedures.

Release Notes

3.1.2

server updates:

- Mitsubishi MNET I/O Server: performance improvements
- HSYCOREMOTE I/O Server: some data-point change events were not propagated correctly to the supervising HSYCO server
- new URL format for PBX calls notifications: /x/vcall?from=<from>&to=<to>
The old format, /x/vcall/<from>/<to> continues to be valid too
- the HTTP camera recording trigger API now also supports the following syntax: /x/camerarec?camera=<name>&zone=<id>
- new CommPortsList option in hsyco.ini, allows explicit definition of operating system's local serial ports

bug fixes:

- minor fix to the EVENTS language parser for the IO action
- scheduler time bug (on new, it showed "00:" on time fields)
- camerapanel motion popup showed up in wrong position when in a popup page
- the camera client is now more tolerant to HTTP content where lines are not always terminated with CR LF (as seen with GANZ cameras)
- Domino I/O Server: the (temp) object's DFCT configuration function was broken in HSYCO 3.1.0
- Domino I/O Server: energy metering data point were not updated
- manual clock update admin page wasn't working
- fixes a compatibility problem with latest versions of Firefox and Opera, that affected the select objects in admin pages
- fixes a bug in the Web page reload process, that could cause performance problems on HSYCO updates or www files changes

3.1.1

server updates:

- Modbus I/O Server: writing readholdingregisters:bitshort, readholdingregisters:bitint, readholdingregisters:bitlong, readinputregisters:bitshort, readinputregisters:bitint and readinputregisters:bitlong, it is now possible to read 1, 2 or 4 registers returning the individual bit values
- Domino I/O Server: added support for DF4RP/I and DFDT

EVENTS

- the update of variables is now executed atomically for the list of actions associated to an event condition. For example, time : \$v + 1, \$v + 1 used to trigger two distinct variable change events, because \$v was changed twice on the time event. Now the same action still causes \$v to be increased by two every minute, but only a single \$v event is triggered, after both actions have been executed

GUI updates:

- improved browser performance when showing data loggers
- (scheduler) object: time from and time to now show 00:00 if not set, instead of the current time

bug fixes:

- (scheduler) object error on some dates; entering dates of the month of December, and few other days in other month, caused a server-side validation error that prevented the scheduler from saving the change
- fixes a login issue in the Manager that caused all login requests after a failed one to fail again until the Manager page was manually reloaded
- (csx75) object of the CSX75 I/O Server: the PIN entry procedure was not working properly in some cases
- GSM I/O Server: the I/O Server occasionally restarted after transient error responses from the modem
- changes to the MJPEG streaming to address a problem caused by the latest versions of Safari browser
- WXONLINE I/O Server: minor fixes
- fixes to the (dmx) object
- minor fixes to the dimmer slider control

3.1.0

server updates:

- new BACnet I/O Server
- new HSYCO I/O Server: remote, hierarchical access to HSYCO's data-points
- new scheduler server
- audio support: text to speech and files playback to the server's line-out, multicast to SNOM phones, Web clients, Axis cameras and Rovio
- new Interlogix FP2000 I/O Server
- new EL.MO. security panels I/O Server
- new ATON AH66 I/O Server: multi-room audio system
- new GSM I/O Server: send GSM messages using any standard AT-protocol serial GSM modem
- added support for the ENTTEC DMX USB PRO DMX gateway
- DOMINO I/O Server: added support for the DFANA energy meter module
- Guardall I/O Server: added support for the PX80 panel, and changed log with zone names instead of numbers
- Modbus I/O Server: new error events on Modbus read or write errors
- Modbus I/O Server: 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
- WXONLINE I/O Server: added support for Yahoo data sources; removed support for Google
- integrated network time synchronization and time-zone configuration (TimeZone and TimeAutoUpdate params in hsyco.ini)
- improved database disk usage
- changes to the network configuration setting procedure

EVENTS:

- new AUDIO action for text-to-speech and audio files play
- string concatenation support on values in COMM, IO and USER actions
- the CAMERA event now supports the optional zone information: CAMERA name [= zone_id]
- assignment of an empty string "" to a variable is now allowed. Events like \$var="" match both if \$var doesn't exist and if \$var was explicitly assigned an empty string.
- \$var = IO datapoint where datapoint doesn't exist also sets \$var to the empty string, while it didn't change the variable's value in previous versions
- external URL support for (background) and (image) UISET img attribute
- the MAIL action can now send regular files as attachments, and supports sending messages via a authenticated email accounts, also with SSL

- the DATALOGGER name = FILE ... actions now accept multiple strings, including variables, as the file name
- Leak Detector:
 - commands: LEAK <name> = <value>, LEAK <name> = PERIOD <seconds>, LEAK <name> = DEVIATION <percent>, , LEAK <name> = CLEAR
 - event: LEAK <name> = on|off

Java API:

- the sendMail() method can now send regular files as attachments, and supports sending messages via a authenticated email accounts, also with SSL
- Leak Detector APIs: LeakDetectorClear(), LeakDetectorOptions() and LeakDetectorUpdate()
- new audioPlay() methods for text-to-speech and audio files play

GUI updates:

- new (scheduler) object
- new (selector <file>; <pos>; <width>; <height>; <container id>; <group>; <text>; <css>) object
- new (submitimage) object
- new (bacnetbrowser) object
- new UISET object's "blink" attribute
- (panel) now has a default transparent color
- (container) now has an optional attribute to set the container's default visibility
- left CSS style attribute added for text fields in (buttonimage), (image), (imagelink), (text), (userimage)
- (datalogger) now has additional options for the legend format. The new legend attribute can be set to "false", "true", or any combination of "min", "avg" and "max". The totals attribute can be set to "true" to show summary values, or "false" to only show text labels in the legend
- (image), (imagelink), (userimage): text can now be set with UISET

Manager / Project Editor:

- the video object now has a video file selection pop-up
- the image and video file selection pop-ups now default to www/img
- choose file pop-up now remembers last opened folder
- tabs in project panel renamed to select, edit and +

general:

- improved server restart detection and automatic page reload on hsyco.jar changes

bug fixes:

- High Availability: the data directory should not be synced from master to slave
- MAIL action and sendMail() method: sending a mail with an embedded reference to not-existing or disabled camera caused an internal error preventing the whole message from being sent
- improved UTF-8 character set support
- URL bar in Android browser is now properly removed
- Manager: the log viewer text can now be selected for copy&paste when paused
- Data Loggers: fixed handling of erroneous values passed to a counter data logger
- fixes to the license limits validation logic

3.0.3

Core:

- CONTATTO and DOMINO I/O Servers: optimized performance of registers status polling
- minor changes to the Guardall QX32i I/O Server
- DataLogger: refresh is now automatic; refresh is automatically performed at the beginning of each hour. The explicit refresh action is now obsolete
- new I/O Server for the Aritech CSx75 (Comfort) security panels
- variables can now be used in value strings

EVENTS:

- new URL actions and events with HTTP/HTTPS support
- new \$DATE:DOW\$ predefined variable returns a number that represents the current day of week
- new *= string contains operator. The string1 *= string2 condition is true if string2 is contained in string1. This operator is case-insensitive
- concatenated strings are not supported as values passed to the UISET action

index.hsm / GUI:

- new (datalogger <logger id>; <pos>; <width>; <height>; <val label>; <attributes>) object
- added label text with CSS attributes to (image), (imagelink), (buttonimage) and (userimage)

Manager:

- using a PC with a physical keyboard, you can press [Ctrl] and [Shift] at the same time for fast switching between applications
- Log Viewer controls added to File Manager mini-logs
- Log Viewer's log lines buffer increased to 1000 lines
- Log Viewer's new pause button
- a confirmation pop-up is shown if the user tries to leave the Manager's web page while uploading a file or changes have not been saved in the Project's Editor or File Editor
- when editing events.txt, a new warning is shown if the file contains errors, also listing all errors
- when events.txt is uploaded using the File Manager, reloading occurs immediately
- support for UTF-8 characters in the File Manager's text editor has been improved

Bug-fix:

- improved page switching, to prevent pages overlapping effect on very slow browsers
- backgrounds in manager appear in the same position as they do on the client (top: -5px)
- CONTATTO I/O Server: the server failed to initialize when an unsupported device was configured in the MCP
- saving a file with the File Manager and immediately shutting off the server could cause the loss of data. Risk of data loss is now significantly reduced
- fixes to the Modbus write functions
- minor fixes to the Data Logger engine
- minor fixes to (chart) object
- minor fixes to the EVENTS interpreter

3.0.2

Core:

- new I/O Server for the Guardall QX32i security system

Bug-fix:

- back broken on (music) and (timer)
- keys malfunction in timer panels on iOS and Android

3.0.1

Core:

- DOMINO and CONTATTO: 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
- DOMINO: added support for DF8IL, DFDI2B and DFRHT modules
- Data Loggers: if you want a Data Logger to set the discrete UI objects, you should now set `DataLoggers.<id>.UseCharts = true`
- the ACL system now allows you to define rules based on the client's IP address being inside the trusted range (local) or outside (remote)

EVENTS:

- if one of the two operands is a number and the other is "off", replace the latter with 0 before comparing
- new ROUND operator for rounding numeric values in variables to an arbitrary number of decimal digits (from 0 to 9). E.g.: `$MYVAR ROUND 2`
- new FORMAT operator formats numeric values in variables based on a format pattern. E.g.: `$MYVAR FORMAT #####.00`
- new POWER transient event (so you can set an event that reacts to any change in power, without the condition)

Java API:

- `PowerEvent()` changed to `public static int PowerEvent(int power)`

index.hsm / GUI:

- pages have a new "protected" parameter that requires the user to login again when accessing the page
- new "gauge" type for the (chart) object
- (chart) object default colors changed: background is now transparent, and labels are white
- new (keypadlid <pos>; <min>; <max>; <digits>; <decimals>; <type>; <label>; <css>) object
- we now have a local graphic effect when a button is pressed

Manager:

- about popup with version information
- when `hsyco.ini` is changed, it reloads the configuration after the server has restarted
- new HSYCO restart button when the `hsyco.jar` file is selected
- search or exclude now works with multiple words in the search field, matching any word you have entered
- when `systemtopo.txt` is updated, objects' graphics change (e.g. button appearance changes from LIGHT to AUTOM)
- changes to the new project panel

Bug-fix:

- I/O Servers HWGIO, HWGMONITOR, EKAHAU, HIDEDEGESOLO, SAMSUNG VIDEO ANALYSIS and SAMSUNG CAMERA I/O could block the update of the systemtopo.txt file at start-up
- Tridonic/DALI: dimmer values were not returned with ending % character
- (adminlink) objects failed to show the admin pages
- Tecnoalarm driver:
 - in the high-availability configuration, the not-active server should not connect to the Tecnoalarm units
 - in some cases, a communication problem could force the driver to erroneously reload all past logs
- (buttonimage) caused an error if some parameters were not defined; now they are optional
- access.ini and other system's files are written to disk using a more robust procedure
- light_rain icon for (weather) was missing
- (cameralink) not working with grids and the default destination panel id
- minor fixes to the Data Logger
- various fixes to prevent the manager to stop working when the server is offline (like switching from file manager to project editor or saving a project)
- reversed value series for (chart) objects with horizontal orientation
- DOMINO: fix to the (temp) object's set-point programming function
- fixes a problem playing recorded video on Firefox with MJPEG streaming mode enabled
- fixes to the (nuvo) object

3.0.0**Core:**

- support for DUEMMEGI's DOMINO home automation system
- support for DUEMMEGI's CONTATTO home automation system
- support for NuVo's Grand Concerto and Essentia E6G multi-room system
- support for FCC Planterm RayCONTROL
- support for YAMAHA RX-Vxx67 AV Receivers/Amplifiers
- new AXISDEC I/O Server for control and streaming to the AXIS P7701 Video Decoder
- extended license manager to support additional license limits options (CAM, DMX, SEC, IOS, COM, IRT, SQU, DOMINO, CONTATTO, KNX)
- support for TCP/IP serial servers with server failover option, including the Advantech EKI serial port IP gateway; configuration parameters:
CommPort.<id>.Type = SERVER

CommPort.<id>.IP = <IP address or server name> [, <failover server IP address or name>]

CommPort.<id>.Port = <port number>

- support for PowerOne inverters
- support for Mitsubishi M-NET gateways
- support for KNX BAOS Object Server gateway
- enhanced Project Editor
- new File Manager
- new Log Viewer
- support for license key auto-provisioning
- support for server-side access control lists for Web-based user commands. The optional acl.ini file defines the rules to allow or reject commands
- new dummy I/O Driver. It's a fake driver that could be useful for projects development and demonstrations. Supports lighting and automation
- new URL query option "nocache" disables the HTML5 persistent cache for the Web session, overriding the default settings
- support for Tridonic DALI SCi2 RS-232 DALI gateway
- support for Carlo Gavazzi ISMG inverters
- redesigned My Home IO Server
- new WXONLINE IO Server retrieves local weather data from Google
- in hsyco.ini, setting trustedNet = local or not defining this parameter, HSYCO will consider all local network addresses as trusted
- HTTPS call to /x/hsyco.pem returns the self-generated root certificate

EVENTS:

- new PING event and action. A "PING <hostname1> [<hostnameN>] [=<timeout>]" action 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
- you can now assign the current value of IO and DMX devices to variables and UI objects , for example: \$myvar = IO gate.o22.1, UISET uiobject.text = DMX 33
- new DATALOGGER action to easily create counters and range historical charts (hourly, daily, monthly charts for present and past periods)
- new Modbus actions and events: it is now possible to trigger the Modbus read functions on arbitrary addresses, with IO events generated when data is read:
 - actions:
 - IO <modbus IO gwid>.<unit id>.<register address> = READCOILS:<quantity of coils>

- IO <modbus IO gwid>.<unit id>.<register address> = READDISCRETEINPUTS:<quantity of inputs>
- IO <modbus IO gwid>.<unit id>.<register address> = READINPUTREGISTERS:<USHORT | UINT | ULONG | SHORT | INT | LONG | FLOAT>
- IO <modbus IO gwid>.<unit id>.<register address> = READHOLDINGREGISTERS:<USHORT | UINT | ULONG | SHORT | INT | LONG | FLOAT>
- events:
 - for READCOILS and READDISCRETEINPUTS: IO <modbus IO gwid>.<unit id>.<register address>.<coil/input number> = <0 | 1>
 - for READINPUTREGISTERS and READHOLDINGREGISTERS: IO <modbus IO gwid>.<unit id>.<register address> = <value>
- added support for Modbus write actions (single coil, single/multiple registers with short, int, long and float numbers)
- new POWER action, equivalent to the powerSet() method, sets the electric power state

Java API:

- if you force a page or popup display by returning a page:... string on the userCommand() callback, this method will be called again when that popup or page is closed, with "/close" appended to param
- enhanced uiSet(id, attr, value):
 - id = * is a broadcast to all open projects. Only supported for attr = page, attr = lock and attr=pageback
 - when used with attr=page, will force all browsers that have any index.hsm visible, to switch to the specified page id, or menu. Setting value to an empty string will automatically revert to the previous page
 - when used with attr=lock, will lock or unlock the navigation of all clients.
 - when used with attr=pageback, and value=<pageid>|<popupid> will force all clients that are currently displaying page <pageid> back to the previous page, or clients showing the pop-up <popupid> to close that pop-up
 - note that the pageback function can also be used with id=<projectid>, not only for all (*) projects, to control a specific project
- new user.user(String name, String param) command method, triggers userCommand() callback and USER events
- new dataLoggerInit(), dataLoggerRefresh(), dataLoggerUpdate(), dataLoggerOptions(), dataLoggerSave() and dataLoggerClear() user command methods for data charting
- new static byte[] modbusWriteMultipleRegisters(String name, int unit, int address, byte[] mask, byte[] bytes) method

- new static byte[] modbusReadCoils(String name, int unit, int address, int quantity) method
- new static byte[] modbusReadDiscreteInputs(String name, int unit, int address, int quantity) method
- new static byte[] writeSingleCoil(String name, int unit, int address, boolean value)
- new static byte[] writeSingleRegister(String name, int unit, int address, byte[] bytes)
- new static void closeComm(String portName) method. Closes a serial port connection when the serial port is defined as 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
- the LocationEvent() callback is called with the IP parameter set to null if the associated device is not connected to the HSYCO Web interface
- haActiveState() replaces haActiveState(int channel)

index.hsm / GUI:

- classicbig skin renamed to blue
- orientation mode (#size <landscape size>[; <portrait size>])
- popups are centered when they don't fit in the page
- compact mode for interfaces with a width less than 500:
 - icons behave like previous small skins
 - icons with no text
 - bottom bar: location, power
- new (sliderbutton) object
- www/admin/index.hsm is now embedded in the hsyco.jar distribution file
- live video from cameras is now delivered as MJPEG streams on Safari, Firefox and Chrome browsers. The ?nocamstream URL option disables MJPEG streaming
- when the HTML5 persistent cache is enabled, the cache is reloaded in the background before reloading the page
- support for camera recordings video files download
- new (temp) and (tempmini) objects for zone temperature control
- new (#locked true) directive, to disable any user command from the GUI of a project
- the (background) object can be assigned an id, and you can change the img attribute dynamically
- chart object have an attributes parameter for initialization ([name]=[value]; ...)
- removed the status message field parameter from the (user), (usermini), (usermicro), (userimage), (userrgb) objects

- removed the status message field parameter from the (ir), (irmini), (irmicro) objects
- (charts) object support for empty values (ex. values=1,2,,4)
- (charts) object new "origin" parameter, moves the origin to a value different from 0 (ex. origin=3.2)
- new (buttonimage) object
- new (weather) object
- if the (submit!id) object's id starts with \$, then the server will automatically set variables named \$<id>.<input id> for each input field to the values in the input fields

Bug-fix:

- EVENTS: actions like IO <id1> = IO <id2> did not work
- fixed PIN/PUK keypad problem with Android clients
- the Scheduler Server could hang in a time-out state when the system clock is changed
- chart fix
- devices added to the systemtopo.txt files are sorted alphabetically
- public www files are not listed in the manifest file and are not checked for www files changes

2.10.4

Bug-fix:

- Serial ports driver: bug fixes for the TCP/IP driver
- fixed bug in project editor: address for music objects was not initialized
- My Home driver fix for temperature control with multiple bus gateways and extended addressing
- in systemtopo.txt, the backslash character, single and double quotes are properly escaped to prevent client-side errors
- Tecnoalarm driver: in the high-availability configuration, the not-active server should not connect to the Tecnoalarm units

2.10.3

Bug-fix:

- fixed bug in project editor: address for music objects was not initialized

2.10.2

Bug-fix:

- fixed a UI problem with pop-ups on Microsoft Internet Explorer

2.10.1

Core:

- HID I/O Driver:
added support for event 2021: Granted Access - Extended Time, with event data-point <servername>.access.granted.extended

2.10.0

Core:

- support for BTicino My Home “Advanced Addressing”. You can set the `openServersAddressMode = advanced` parameter in `hsyco.ini` to enable this feature. When using multiple buses the 4 least significant digits of an address will represent the AAPP device address, the second bus will have addresses starting at 10000, the third start with 20000 etc.
- HSYCO can optionally serve live images of its cameras or grids through a password protected HTTP request, like `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. This feature is enabled setting the `Camera.<id>.RemoteRequestPassword` parameters in `hsyco.ini`.
To enable the same feature on grids, use “grid<N>” as the <cameraid> in the above URLs, and set the `CameraGrid.<N>.RemoteRequestPassword` parameters in `hsyco.ini`. It is also possible to have different passwords for clients within the trusted range of IP addresses and outside.
Use the `Camera.<cameraid>.TrustedRequestPassword` and `CameraGrid.<N>.TrustedRequestPassword` parameters, setting passwords that must be different from `RemoteRequestPassword`, or will be ignored. Note that requests with the `TrustedRequestPassword` value will be also served via HTTP, without SSL.
- support camera frames acquisition using HTTPS with self-signed certificates
- support camera frames acquisition through MJPEG streams
- support for SAMSUNG IP cameras, including motion detection, I/O, video analysis and PTZ for some models. Two new parameters in `hsyco.ini` are used to define the camera model type, `Camera.<camid>.Type`, and to optionally

enable I/O and VA, Camera.<camid>.IO=enabled. If you want to enable I/O only, set it to “enabled:io”, or to “enabled:va” to only enable VA

- support for HTML 5 off-line cache storage with Google Chrome 8.x browsers
- improved start-up time of the HSYCO server. Now the HTTPS server starts almost immediately. The motion markers rebuilding process was redesigned as a thread and runs in parallel at start-up. All threads start immediately, while cameras recording is disabled until the markers rebuild is completed
- new HTTPServerPublicDirectory in hsyco.ini, 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
- a missing systemtopo.txt file is not considered an error any more, and will not make the Web client fail
- you can now create an img/ sub-directory under the project's main directory. Files in this subdirectory have precedence over files in the root /img directory
- support for the Elsner P03/3 Modbus weather station
- updated ParadoxEVO driver

EVENTS:

- New events for temperature control central unit status:
 - TEMP <unit> = <WINTER | SUMMER>
 - TEMP <unit> = <OFF | PROT | MAN | STD 1 | STD 2 | STD 3 | HOLIDAY>
- MAIL action: 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
- you can now check if a variable is defined: the event condition \$<varname> = "" is true if the variable is not defined or if has been explicitly set to ""
- new DATALOGGER action:
 - support for COUNTER and RANGE data modes, e.g.:
 - DATALOGGER <name> [<name>] = INIT COUNTER [<upper limit> [<decimals> [<hour interval>]]]
 - DATALOGGER <name> [<name>] = INIT RANGE [<decimals> [<hour interval>]]
 - support for CSV statistic and log files:
 - DATALOGGER <name> [<name>] = FILE LOG <filename> [TIMESTAMP] - appends most recent data to a log file, with optional timestamp
 - DATALOGGER <name> [<name>] = FILE STAT <filename> - creates a new file with all statistic data (same data as current chart)
 - DataLoggerCsvSeparator = <tab | comma | semicolon> parameter in hsyco.ini

Java API:

- new ping(String host, int timeout) utility method. Returns true if the host is reachable, false otherwise. host is the hostname or IP address; timeout is the timeout in milliseconds
- sendMail() method: 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
- adds user.java callback method WebRootRequestEvent(InetAddress addr, boolean secure, String useragent). Called when the Web server receives a root URL request, can return a string to redirect the browser to a valid URL, or null to prevent redirection
- new static void user(String name, String param) command method. Triggers a USER event and the userCommand(String name, String param) Java method. Can be used as a calling mechanism between Java and EVENTS.
- new scheduler processor, 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:
 - user.SchedulerEvent(String groupname, String schedulename) callback blocking method
 - user.schedulerRegister(String groupname, String schedulename, int interval) command method, to register a new scheduled callback; interval is defined in milliseconds
 - user.schedulerRemove(String groupname, String schedulename) method to delete a schedule
- MODBUS I/O methods readHoldingRegisters(), readInputRegisters() and writeMultipleRegisters() can now 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. This makes possible to call these methods without defining an I/O server for each gateway
- new DATALOGGER API:
 - static boolean dataLoggerInit(String type, String name, double counterUpperLimit, int decimals, int hourInterval)
 - static boolean dataLoggerRefresh(String name)
 - static boolean dataLoggerUpdate(String name, double value)
 - static boolean dataLoggerClear(String name)
 - static boolean dataLoggerSave(String type, String[] names, String path, boolean timestamp)
 - DataLoggerCsvSeparator = <tab | comma | semicolon> parameter in hsyco.ini

index.hsm / GUI:

- access control: an administrator cannot change its own access rights
- support for AVI video files generation from recorded images
- when a camera frame is displayed, if the camera feed is standing-by and the last available frame is old, a blank frame is returned until a new frame is retrieved from the camera. It is also returned if the camera is offline
- (link), (linkmini), (linkmicro), (imagelink): you can 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
- new (userimagelid file; pos; width; height; name; param; text) tag
- new (video src; pos; width; height [, mode]) tag for playback of HTML5 supported video formats. This tag has the identified version, and allows dynamic control of position, visibility, play/pause and mode
- (securitylog) has a new format and is now the same for all skins. The new format is (securitylog <id>; <pos>)
- (securitydisplay) now has a transparent background, rather than the blue, display-like background of previous versions. Draw a panel behind the (securitydisplay) object to replicate the original graphics
- (securitylink) and (securitypage) have been removed. Use a normal page instead, with (securitylog) and (securitydisplay), even on fixed size skins
- (templink), (tempdlink), (temppage), (musicpage), (timerlink), (timerdlink), (timerpage) objects were deprecated in version 2.9.0 and have been removed
- new object (sliderlid address; pos; label). Values set with uiSet(<id>, "value", <value>): 0% - 100%; 0.00 - 1.00; 0/<Max> - <Max>/<Max>, on, off (case insensitive)

Bug-fix:

- several stability and performance fixes in the camera frames acquisition processes
- frame acquisition now works also for cameras returning single jpeg frames without the content length header
- fixes to the systemtopo.txt parser
- a bug in the MODBUS driver could prevent reading/writing data for devices with MODBUS address greater than 127
- PROGRAMTIMER events were disabled when running HSYCO with the VIDEO license
- fixed PIN/PUK keypad problem with Android clients
- the HSYCO Server process automatically quits and restarts in case of "too many open files" exceptions

2.9.0

User Interface:

- changes and new features for displaying cameras, including a new drop-down selection list of motion events. It is also possible to freely display cameras in any page
- pop-up pages in the variable size skin are now positioned above or below the link button if there isn't enough space to the left or right
- improved response time of PIN/PUK keypads for devices using iPhone OS 3.x
- if the PIN appears because the browser is connected to a not trusted IP and the session is already authenticated but expired, if the IP changes to a trusted IP the PIN panel is canceled and the session resumed automatically
- if the authentication request fails after entering the PIN or PUK, you can press C to return to the PIN entry dialog at any time, or automatically after 30 seconds
- on loading a page from the cache, if the connection to the server cannot be established, a dialog box appears with a button to force a page reload
- during cameras playback, the "Auto" button is replaced by an Email button. Pressing this button pauses the playback and shows a pop-up where you can type an email address to send a mail with the frame attached in its native resolution. You can also right click or click and hold on the frame when paused to save the image locally
- obsolete skin objects: (musiclink), (musicpage)

Core:

- new format for the systemtopo.txt file and automatic generation of the devices and temp sections, based on auto-detection of existing devices; to enable the automatic discovery and generation of the systemtopo.txt file, set openServersDiscovery = true in hsyco.ini
- support for the Ekahau RTLS Position Engine and location tags, including buzzer and display control for the T301BD tag
- performance and stability improvements in cameras image processing
- in the trustedNet parameter in the hsyco.ini file, you can now enter multiple, comma-separated IP ranges or individual IP addresses
- add-on class Tecnoalarm (TECNO OUT driver) is now included in the HSYCO distributions
- add-on class Bentel (KYO320 driver) is now included in the HSYCO distributions
- the ParadoxEVO, Bentel and Tecnoalarm drivers are now fully integrated in HSYCO. If the configuration files are found at start-up, the classes are

automatically instantiated and there is no need to add any Java code in user.java to make them work

- when HSYCO stops automatically after one of the files defined in AutoKillFiles changes, all camera recording markers are saved and quickly restored from the database when HSYCO restarts. If it restarts after a forced quit, markers are rebuilt based on the actual frames stored in the motion directory
- when HSYCO starts, the SSL certificate is checked and, if the name set in the ServerName parameter in hsyco.ini doesn't match the certificate name, a new self-signed certificate is automatically generated to replace the old one
- support for "HSYCO Mini" license rights. A new "limits" attribute in license.txt is used to enable the Mini restrictions. At this time the supported keywords are 1CAM, 4CAM and 1DMX
- we had hard thresholds of 6GB and 3GB of free disk space; below these limits, HSYCO would start to drop older frames, or stop recording new frames. Now if the motion directory has its own dedicated partition, the thresholds are 2GB and 1GB, while if the partition is the same as the HSYCO root directory, they are set to 15% and 10% of total disk space for that partition

User Interface Objects:

- (camera!id <camera id>; <position>; <width>; <height>; <destination panel id>) live display, with click to camera page. The <destination panel id> is optional; if defined, clicking on the camera image changes the camera displayed on the destination camera panel; the <camera id> is either numeric or the camera name or grid name
- (camerapanel!id <camera id>; <position>; <width>; <height>; <camera list>) live display within panel and in-place ptz and playback controls. The <camera list> parameter is optional, and can be used to restrict the list of cameras to show in this panel; the camera id is either numeric or the camera name or grid name
- (camerainage!id <camera id>; <position>; <width>; <height>) live display, no click, no controls; the camera id is either numeric or the camera name or grid name
- the <camera id> parameter in the (cameralink) object can be numeric or the literal camera name or grid name
- the (cameragridlink) object is obsolete and will be removed in HSYCO 3.0. Use (cameralink) instead
- the (cameralink) object has the optional parameter <destination panel id>: (cameralink <num_cam>; <position>; <color>; <text>; <destination panel id>). If defined, clicking on the button changes the camera displayed on the destination camera panel

- new (container <position>) and (endofcontainer) convenience objects. These objects are used to group other objects. The objects in a container have the position relative to the container position. Containers can be nested. With identified containers you can also control the visibility and position of the containers and all objects inside
- new (chart) object
- new optional CSS style attribute for the text object: (text <position>; <text>; <css>)
- individual parameters of objects in index.hsm can be nested with () parentheses to allow for semicolon inside a parameter
- new objects for forms support:
 - (input!id pos) and (input!id pos; css) input fields; uiSet() can be used to set the text and visibility of input fields
 - (submit!id pos; color; label) for the form submit button. When the submit button is pressed, all input fields and their ids are passed to userButton() or virtualRemote(); with userButton() you can control a page change on submit. (submitmini) and (submitmicro) objects are also available for small and very small buttons
 - new optional attribute color for the panel object: (panellid pos; width; height; color); supported color values: g,r,b,gr,y
 - new object (autombig), same syntax of (autom), shows a large button with areas for up/open, stop and down/close
 - new object (marqueelid <pos>; <width>; <height>; <direction>; <speed>; <text> [; css]). It is just like a text object, but with scrolling
 - the new directive (#deviceimage disable) can be used to disable the pop-up with the image or camera associated to device icons; works with all skins
 - the new directive (#scale f) allows you to define a fixed scaling factor for classicbig skins. f is a decimal number greater than 0
 - the new directive (#kiosk), only available in the classicbig skin, allows you to present a page without the HSYCO menu bar and borders. (#kiosk lock) also blocks users from changing page
 - (cameralist), (cameragridlist) and (cameraoverlay) are now meta directives: (#cameralist), (#cameragridlist) and (#cameraoverlay). The old format is supported for backward compatibility but is deprecated, and will be dropped in the next major release

Java Programming:

- new callback method SunPositionEvent(int azimuth, int elevation)
- new callback method userCommand(String rem, String id). This method will replace virtualRemote(), and allows you to return a page name, forcing the GUI to change page in response to a user button's click

- new callback method `programTimerEvent(String name)` called on program timers execution
- new methods `modbusReadHoldingRegisters(String name, int unit, int address, int quantity)`, `modbusReadInputRegisters(String name, int unit, int address, int quantity)` and `modbusWriteMultipleRegisters(String name, int unit, int address, byte[] bytes)` for reading and writing MODBUS registers on MODBUS/TCP I/O servers
- new methods `varGet(String name)` and `varSet(String name, String value)` to read/write program variables. Variables names ending with `!` are considered persistent and their values are preserved when HSYCO restarts; normal variables are deleted when HSYCO starts
- new methods `programTimerSet(String name, int seconds)`, `programTimerClear(String name)`, `programTimerReset(String name, int seconds)` and `programTimerRepeat(String name, int seconds)` to manage program timers
- the `uiSet(id, attr, value)` now supports the ability to force a page change from user code. You have to set the menu name (the name of the directory containing the `index.hsm` file) as `id`. Setting `attr` to `"page"` and `value` to the page id changes to that page; setting to an empty string changes to the previous page. Setting `attr` to `"lock"` and `value` to `"true"` blocks the user's navigation; set `value` to `"false"` to unlock
- `uiSet()` can be used to set the camera id (the attribute name is `camera`, and should be set to the camera number or name), visibility and position of `(camera)`, `(camerapanel)` and `(cameraimage)` objects
- `readComm()` and `writeComm()` now write the full trace of received and sent bytes to the log file only if both `verboseLog = true` and `userLog = true`
- a new method `public static int sendMail(String to, String from, String subject, Vector<String> body)` overloads the original `sendMail()` method, allowing you to send multi-part emails, with mixed text and camera images. To send an ordinary text, just fill the vector with the desired text string. 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. You can send multiple images in the same message, and mix text and images
- new methods `getBentelPlugIn(String id)`, `getParadoxEVOPlugIn(String id)` and `getTecnoalarmPlugIn(String id)` to retrieve the instance objects in `user.java` and use the specific API methods

Events Programming:

- new events interpreter, with support for complex events and scalar values comparison
- new events and action verbs
- improved performance

Bug-fix:

- improved reliability of the sendMail() method
- recording when the Rovio mobile camera moves doesn't stop any more while the camera is still moving
- fixed readComm() method to improve reliability when waiting for input data
- fixed a bug that could affect the reliability of the event monitor of OpenWebNet when using multiple servers; also improved the error messages of the OpenWebNet Monitor engine
- fixed a bug in LocationEvent() callback method that caused an incorrect long representation of the MAC parameter

2.8.4

Core:

- bidirectional serial port support for the RS232 port of IRTans
- support of CCF hexadecimal strings for IRTans
- new method public static int deviceFunctionGet(int device)
- new method public static int sendMail(String to, String from, String subject, String body)
- add-on classes Netstreams and ParadoxEVO are now included in the HSYCO distributions
- support for the HWg Damocles line of products, with input lines counters

Bug-fix:

- in some occasions, a bug in the Open Web Net monitor thread could cause issues when using multiple gateways
- the IO id = FLIP action in events.txt could generate a parse error and cause the event to be ignored at start-up
- the TOGGLE action of openCommand() didn't work properly
- powerSet(int power) now works even if no OpenWebNet server is defined in hsyco.ini
- using the classicbig skin, the camera page didn't display camera images after a rotation and subsequent switch between landscape and portrait page formats
- improved stability of PIN/PUK authentication
- because of a bug in the Syslog service parser, call events generated by the Grandstream GXE502x IPPBX didn't work

- in the TimeEvent(long time) user callback, the time parameter was not correctly aligned on the current time
- a bug in the WebServer's page parser caused parse errors when running the HSYCO Server on Microsoft Windows

2.8.3

GUI:

- (menu#portrait) (menu#landscape) (page#portrait) (page#landscape) .hsc files to support automatic page reformatting based on orientation --- for variable size skins only. On orientation change, look for the corresponding format, then the base format and finally the other orientation format
- (link!id), (linkmini!id), (linkmicro!id), (dlink!id), (image!id), (imagelink!id) support for dynamic attributes

Core:

- fully integrated support of the serial port in HWg I/O Controller
- improved reliability of files synchronization process in High Availability configurations
- the status loop in the GUI core has been optimized
- changes to the off-line cache manifest naming, for faster reloading of changes

2.8.2

API:

- new setPower() utility method
- the CameraCommandEvent() callback method is now called for all cameras with a PTZ driver defined in hsyco.ini, not only for PTZ=user. If the method returns -1, the PTZ driver command is skipped

events.txt:

- new CAMERACOMMAND event, called for all cameras with a PTZ driver defined in hsyco.ini. If the event matches, the PTZ driver command is skipped

GUI:

- new (lighticon) and (automicon) skin objects, with solid gray or glass semi-transparent effect
- new (background) object
- new status icons in camera pages for the Rovio camera

Bug-fix:

- because of a bug in the GUI code, cascaded pop-up windows didn't work properly. You could not use a (link) object inside a pop-up to open another pop-up
- multiple (text!id) or (user!id) objects with the same id were not handled correctly
- device timers (the off timers, usually set with `TIMER d = SET s, deviceOffTimerSet(int device, int seconds)` etc.) were not disabled in High Availability configurations when the server is not active. Now the execution of the off command is disabled if the HA state is not active
- minor GUI fixes:
 - in the classicbig skin, the micro pop-up that shows the image associated with a device were not handled correctly on Safari Mobile. Now they work properly, disappearing after a few seconds from the key press. They also have the X corner to close manually
 - Rovio status icons moved up a few bits in templates, classic and modern skins, to avoid overlapping with the command message
 - Rovio status icons temporarily showing on other cameras
 - pop-up GUI items not closing on session time-out events remained visible in the log-in window

2.8.1

Core:

- performance optimization for I/O devices state machine and events propagation

Security:

- `KeysTrustedValidityHours` and `KeysNotTrustedValidityHours` parameters in `hsyco.ini` also accept time in hh:mm format
- new inactivity session expiration logic, configurable with `KeysInactivityHours` and `KeysInactivityMode` in `hsyco.ini`

events.txt:

- multiple identical events can be declared in `events.txt`

GUI:

- Web interface network settings functions can be disabled setting `WebAdminNetConfigLock=true` in `hsyco.ini`

2.8.0

Core:

- High Availability master-slave configuration support
- support for HTML 5 off-line cache storage, dramatically improving page loading performance when accessing HSYCO using slow connections. Works with iPhone 3, Safari 4 and Firefox 3.5. the OffLineCache parameter in hsyco.ini can be used to enable/disable this feature
- support for DMX512 universes (using the KISS-BOX Ethernet Gateway)
- userLog parameter in hsyco.ini to enable/disable logging of user.java and events.txt command calls
- support for HWg I/O controllers and monitoring interfaces

Network cameras:

- several changes in the cameras frame acquisition engine, to optimize performance, improve frame rate and reduce CPU load
- when available disk space drops below 6GB, approximately 20% of the oldest recorded frames for each camera are automatically deleted from disk. If available space drops below 3GB recording of new frames is temporarily disabled
- new Cameras.DroppedFrame.<camera id> parameter in hsyco.ini allows to set how many frames to drop for each camera during recording
- new CameraGrid.<id>.Resolution parameter to limit the maximum size of each grid
- new Camera.<id>.URL.Small parameter lets you specify a different URL to fetch frames at a lower resolution, in order to optimize acquisition performance
- support for PANASONIC PTZ cameras
- support for WowWee Rovio, including status, recording when moving, and update of home position
- semi-transparent standard command overlay on PTZ cameras, customizable with the (cameraoverlay) object
- pan, tilt and zoom movement is now continuous; the center quadrant of the image is used to stop the pan and tilt movement
- specific support for virtual PTZ cameras (Axis and Mobotix): top-left corner in the picture area zooms in about 60%; bottom-left corner zooms out to full-field (home) view
- camera rotation, both with the AUTO key and manual, is now constrained to the cameras contained in the grid. Direct access to a camera prevents rotation

API:

- new DMX512 control APIs
- new High Availability APIs

- readComm(String portName, int len) and writeComm(String portName, String data) for RS232 local communication ports support
- deviceOffTimerSet(), deviceOffTimerClear(), deviceOffTimerPreset() and deviceOffTimerReset() now also work for automation devices
- new cameraRecMode(String cameraName, boolean enabled) method to enable/disable camera recording
- new cameraMode(String cameraName, boolean enabled) method to turn a camera on/off
- cameraRecTrigger(String cameraName, String source, int seconds) now accepts seconds=0 to stop an active recording
- new cameraRecTriggerFull(String cameraName, String source, int seconds) method to record at full frame rate, ignoring the DroppedFrames option in hsyco.ini
- CameraCommandEvent() now supports move/stop event
- new sleep(long millis) method
- new temperature control system APIs: tempZoneGetTemp(), tempZoneGetSetpoint(), tempZoneGetOffset(), tempZoneGetMode()
- new uiSet() and uiGet() methods to dynamically change text, visibility, color and img attributes of (text!id), (user!id), (usermini!id) and (usermicro!id) objects
- new ioSet(), ioGet(), IOStartupEvent(), IOEvent() methods for I/O interfaces control
- new OpenEventFrame() callback method

events.txt:

- new UISET action to dynamically change text, visibility, color and img attributes of (text!id), (user!id), (usermini!id) and (usermicro!id) objects
- improved checks on zone number for TEMP commands (AUTO, STD, HOLIDAY)
- CAMERA name = sec action changed to CAMERAREC name = sec
- new CAMERARECFULL action to record at full frame rate
- the LIGHT d = FLIP, LIGHTGROUP d = FLIP, AUTOM d = FLIP = FLIP, AUTOMGROUP d = FLIP = FLIP actions now works also with multiple devices, with all devices following the state of the first device in the list
- new IOSTART and IO events, and IO action for I/O interfaces control

GUI:

- new fixed size skins: classic and modern
- new variable size skin: classicbig
- new skin objects:
 - linkmini linkmicro
 - userrgb usermini usermicro
 - irmini irmicro
 - hbar, vbar

- image, imagelink
- text
- panel
- dimmer - controls SCS dimmer devices with a graphic object similar to (dmx)
- (cameraoverlay) - defines standard and custom overlays for PTZ cameras
- (text!id) (user!id) (usermini!id) (usermicro!id) - named objects with dynamic attributes that can be changed using UISET and UISet()
- popup (for classicbig only)
- musicsync (for classicbig only)
- securitydisplay (for classicbig only)
- obsolete skin objects:
 - templink, tempdlink, timerlink, timerdlink, temppage, endoftemppage, timerpage, endoftimerpage
- standard alarm log messages in the security system Web page can be replaced with customized messages defined in the (security) section of the systemtopo.txt file
- new administration pages to change OS's password and network settings
- improved handling of communication errors:
- if a communication error persists for about 30 seconds, the pop-up message has been removed and the page is automatically reloaded, but only immediately after loading; if a communication error occurs later, the send icon is raised and you can click on it to reload the web page, but there is no automatic reload
- new (#include <filename>) directive
- new (#size <page size>) and (#camerasize <width>x<height>) directives for classicbig
- multiple references to (light), (autom), (scene), (group) and (aux) objects are now possible without having to add the .x notation after the device id. The .x notation is still supported for backward compatibility

Bug-fix:

- fixed a bug that prevented access when using http://127.0.0.1/<pagename>/pin. from a loopback client (used to allow web clients running on the HSYCO server to use HTTP and omit the URLKEY)
- fixed a bug that prevented access when the URL had a query string parameter defined, like ?nopic or ?click
- if (security) was not defined in systemtopo.txt, the security log display failed on the Web page and errors could occur during state updates
- added support for serial devices with device names other than ttyS* on Linux (like USB-based serial ports)

- a bug in the OpenWebNet parser prevented the correct parsing of CEN frames on local SCS buses
- fixed a GUI bug during the PIN authentication phase, that could cause an unintentional login failure if roaming between untrusted and trusted client addresses occurred while the PIN page was displayed
- a bug in the index.hsm page parser caused a parse error for macro objects with empty last parameter, like: (page pagename;)
- a bug in the index.hsm page parser caused a parse error for files having a comment as last line
- the SCS startup procedure was changed to prevent a condition that, if the status of a device was changing state during the startup procedure, could cause a misalignment of that device's status
- SCS group commands were not handled properly by the monitor thread
- a bug in the events.txt parser prevented the correct match of IR events
- a bug in the events.txt parser prevented the correct match of events where the device id or channel had leading zeroes in the device number
- fixed a bug related to favicon.ico that could cause delays or hangups when accessing the HSYCO Web interface using some browsers

2.7.1

Core:

- location services: added support for Funkwerk GmbH access points

API:

- UserTimerEvent(String name, boolean active) is now also called once upon timer termination. The active parameter is set to true on activation calls and false on termination call. WARNING: this API change could affect user code compatibility running with previous releases of HSYCO core

GUI:

- the HSYCO web page is now automatically reloaded when any file changes in the www subdirectory
- on iPhone/iPod 3.0 or later the web page is not reloaded any more after a certain amount of camera frames have been loaded; the freeze issue has been solved on Safari 4.0 for iPhone
- improved handling of communication errors:
- if a communication error persists for about 30 seconds, a pop-up message asks to reload the page to try fix the issue (this is also needed when the page is loaded from the off-line cache, but the website certificate is not yet allowed - there is no other way to force the browser to ask permission to use a self-signed certificate)

- when the send icon is visible, you can click on it to reload the web page

Bug-fix:

- changed the text for manual zone temp setting in the Italian default skin
- multiple references to the same aux, group, scenario and cen object in index.hsm files now works properly, just like light and autom objects
- fixed a bug that could cause irregular or repeated calls to TimeEvent()
- openCommand (int function, int action, int device) now correctly supports AUX address commands
- improved support for high latency connections, including a 60 sec. idle timeout on HTTP server connections
- cameraCommandEvent() renamed CameraCommandEvent()
- OpenEventScene() and OpenEventAux() are now executed whenever a scene or aux frame is detected, not only on state changes
- OpenEventLight() and openEventAutom() are now executed only on state changes, not every time a frame is detected by the Open monitor thread

2.7.0

GUI:

- create, update and delete users
- define/change PIN and PUK
- grant administration rights to users
- selection of authorized Web pages for each user
- enable/disable users
- clicking on a dimmer button, when the status is on but the dimmer bar is not displayed, will bring the dimmer bar onscreen rather than turn off the light. This allows a smooth readjustment of the dimmer level, without having to turn the light off and back on again

Network cameras:

- redesigned navigation; server-based grid GUI (grid definition in hsyco.ini: CameraGrid.1 = c1 c2, c3 c4)
- cameras GUI: PTZ control for Axis cameras and user customizable actions
- pre-recording buffer: the Camera.id.MotionBuffer = <frames> parameter in hsyco.ini can be used to set the number of frames to be recored in advance of any recording event. Default is to have the pre-recording buffer disabled
- image rotation: the Camera.id.Rotate = <degrees> parameter in hsyco.ini can be used to rotate camera images. Images are eventually cropped and filled with black side bars, depending on the original and final image size
- additional optimizations in image resize for variable, client-controlled frame size

- if images from a camera are not available when first polled from HSYCO startup, then a black empty frame is returned to the browser request, instead of a 404 Not Found error
- when a button with associated camera is pressed, the camera live thumbnail remains visible in the display area for 8 seconds, and can be clicked to go directly to the corresponding camera page
- licensing code to allow for distinct license keys for HSYCO and HSYCO VIDEO

API:

- void user.powerEvent() becomes int user.powerEvent() - if the method returns a not-negative integer, then this integer is set as current power state
- user.powerEvent() is now called for every power reading, even if the value is unchanged from previous reading
- new callback method user.OpenEventTempCentralUnit(int serverIndex, int mode, int value)
- new callback method boolean PBXCallEvent(String host, String caller, String called)
- new callback method public static int cameraCommandEvent(String function, String action, String camera) to process custom PTZ requests
- new method user.tempCentralUnitGetMode(int serverIndex)
- new method user.tempCentralUnitGetSettings(int serverIndex)
- new method openFrameCommand(int serverIndex, String frame) method to send generic frames to the SCS gateways
- new method public static int cameraCommand(String function, String action, String camera) to send PTZ commands to cameras
- new method public static int wakeOnLan(String broadcast, String address)

Security:

- access.ini flag to temporarily disable PINs (a # character at the beginning of the encrypted PIN)
- each individual user can now be restricted to a specific list of www directories when accessing the Web-based GUI

Core:

- full support of burglar alarm system, including a new security.log log file where all status changes and alarms are logged
- support for CEN commands
- ability to choose a specific template/skin in index.hsm: (#skin <skin directory>) -- default is templates; pic subdirectory moved under templates
- language directory in index.hsm (#language <language>) for the selection of text.txt file -- default is "it"; relocate text_<lang>.txt file in skin subdirectory
- relocate pin.html in skin subdirectory (also rename to pin.hsc)

- the embedded Web Server properly supports HTTP 1.1 HEAD requests
- implemented powerAdj “auto” value to automatically set power adjustment factor
- ability to define different port numbers for each OpenWebNet server, overriding the generic openServersPort setting in hsyco.ini
- the SqueezeCenter status monitor thread is now less verbose in the logs. When an error occurs, like when a device is off, no log is written unless verboseLog is set to true in hsyco.ini
- setting verboseLog=silent in hsyco.ini disables all log messages but the most serious errors and security messages
- when sending IR commands, there is an initial delay of 500ms before sending each command. Now it is possible to avoid this delay using the @0 directive in the command sequence

Bug-fix:

- fixed a bug that affected monitoring of GROUP AUTOM commands
- fixed a bug sending commands to devices with address A=0
- camera page header not properly refreshed when transitioning between first camera and first camera grid
- improved handling of nonexistent pages, now shows an alert message for debugging purposes, and also fixes a bug when reloading the GUI and the last current page has been removed
- hsyco.ini parameters values are now trimmed of leading and trailing spaces
- fixed a bug that could corrupt the current state information, causing the GUI to display a blank page, when parsing scenario configuration frames
- if powerAdj is not defined there should be no error message in the log file at startup
- OpenWebNet parser fix to avoid restart on lighting frames that break current frame format definition

2.6.1

Bug-fix

- when transitioning from a media page to the cameras page, the SCREEN: ON/OFF status flag could remain visible if a network error occurred while loading camera frames - the fix affects the JavaScript client code only

2.6.0

Core:

- new software license key based on a unique hardware id. HSYCO Server will run with most monitor threads disabled and command execution disabled if license.txt is missing or invalid
- log files and folders, as well as camera recorded files and folders are now created with read/write permissions for all users, to simplify remote management from not-root users
- new HTML page template system and improved multi-language support
- httproot folder renamed www
- www files are now searched in the hsyco.jar package file if not found inside the www folder. All standard web client files, like templates, images, and JavaScript files are now distributed inside the hsyco.jar package
- added void user.StartupEvent() callback, called once at startup, before main threads are started
- added void OpenWebNetMonitorStartupEvent(int serverIndex) callback, called every time the Open Web Net monitor thread starts, after initialization, when monitoring is active
- added void user.TimeEvent(long time) callback, called every minute, at 00 seconds
- added measured temperature event callback function void OpenEventTemp(int zone, int temp), zone is the conventional zone number and temp is a positive number between 0 (0 C) and 500 (50.0 C)
- zones' set-point status temperatures are now updated much quicker than before
- the single thread used to fetch frames from the cameras has now been split into one distinct thread per camera, so that cameras' performance don't affect each other
- motion events markers implemented at core level
- HTTP Server performance improvements: content compression and HTTP/1.1 persistent connections fully implemented in the embedded HTTP server
- automatic Web page reload after HSYCO server restart
- user class now extends userBase skeleton class, and a number of stub methods and constants have been defined inside userBase to be used within user.java
- user call DaylightEvent() now not called anymore at HSYCO server startup
- CertCommonName parameter in hsyco.ini renamed ServerName; CertCommonName lookup performed for backward compatibility if ServerName not found

Access control:

- support multiple URLKEYs: the urlkey parameter in access.ini can be set as a list of comma separated URLKEYs; leading and trailing spaces for each key are ignored

- security fix: removed access key text and PIN from log files in case of access errors, leaving the first 6 characters of keys and the first and last digit of PINs
- by default, serving pages through the HTTP server is now disabled, except for localhost requests; use `HTTPServerLowSecurityEnabled = true` to fully enable the HTTP server, otherwise only requests for motion triggering will be processed (`/x/camerarec`)
- URLKEY is not required for localhost requests, a URL in the form `/<path>/pin.` is considered valid

GUI:

- dimmer level setting slider added for dimmer units
- overhauled << < > >> camera playback navigation buttons. Navigation is now event-based, forward and rewind buttons now allow for an accurate navigation through the individual events, while a sliding cursor shows the frame position within the event. The cursor can be clicked to jump frames within the same event with moving
- after a page reload, the GUI restores its state, to the sub-page and mode before reloading; GUI state is retained for 60 minutes
- workaround for iPhone 2.x Safari Mobile refusing to load images after a while. The client page is reloaded after a preset number of camera images have been loaded
- the keyboard can now be used to type digits when keypad is displayed (login page, timers and temperature control); allowed keys are 0-9, Backspace and Enter

Bug-fix:

- fixed PIN timeout page display clean-up and return to current page after successful login
- numeric keypad input now working properly; spacebar replaced C as alternative backspace key
- power meter also available in temperature control pages

2.5.4

Core:

- improved SCS monitor thread initialization - the monitor thread now requests device status three times, then continues instead of exiting. If the SCS server doesn't accept open access, but requires authentication, an appropriate error message is logged and the monitor thread quits

Access control:

- the authentication system based on two distinct pages, the pin.<urlkey> and index.html home page has been overhauled. Now the only client side url is pin.<urlkey>. When the client requests the pin page, if the key cookie is valid, the server directly serves the index.html content, skipping the redirection and subsequent loading of index.html. If the key cookie is not valid, the actual pin.html page is served as usual. Also, when the key expires, the client code directly jumps to the pin request panel, without reloading the pin.html page. Thanks to this, there is a significant time saving during the revalidation process after the key has expired

Bug-fix:

- a check on time validity has been added for timer set panels

2.5.3

Core:

- added support for multiple temperature control central units in multi-SCS environments
- added support for multiple power load control central units in multi-SCS environments
- electric loads control unit detection during Open Web Net monitor initialization, log of measured V, A, P
- server forced quit on critical files changes (autoKillFiles parameter in hsyco.ini defining the list of monitored files)
- improved location services - clients' location table is now checked and eventually updated during state update calls; MAC addresses have 10 hours expiration time
- improved update of temperature control's zones status after central unit commands
- self-generation of SSL certificates with appropriate common name based on server name if certificate not found
- changed default HTTP Server threads from 32 to 128
- support for zone parameter passed with camera triggers (camera=cameraName;zone=zoneName), also made available to user.CameraMotionEvent()
- automatic recording of motion events based on camera triggers can be disabled setting CamerasRecordingMotionTriggerSeconds = 0
- added public static void deviceOffTimerReset(int device, int seconds) method

Access Control:

- removed login/current trusted/untrusted network mismatch check and subsequent key removal

GUI:

- keys can be set as not-clickable using the NOCLICK option in systemtopo.txt
- keys can have a live camera associated in systemtopo.txt; hovering over the key or clicking it the associated live camera feed appears in the display area replacing the logo
- resized (smaller) camera images are used when the iPhone's orientation is vertical
- enhanced compatibility with iPhone 2.0:
- larger sensitive areas for logout, back, menu and location elements in the main display

Bug-fix:

- de-association location messages from access points should not overlap more recent location information. set location to unknown only if the de-association message comes from the same access point of current location
- camera frames time-stamp cookie should be set based on frame download time-stamp, not on current time
- changed OpenWebNet read timeout in hsyco.openDeviceStatus() to 10s
- user.CameraMotionEvent() now called once per trigger event, not every time a frame is saved
- reduced time needed to show location information
- improved responsiveness of camera playback << < > >> navigation buttons
- added remote IP Address in "POST method not supported" log message
- clicking a key with associated camera didn't work properly on iPhone, failing to execute command

2.5.2

Cameras:

- change to motion frames grabbing architecture: frames to be grabbed from normal live streams based on external HTTP trigger or API available to user callback code: cameraRecMode() to enable/disable camera feed recording; cameraRecTrigger() to trigger recording
- streams polling based on actual per-camera requests (live display or motion-triggered)
- remove camera images from motion DB after configurable expiration time (general default and camera-specific)
- display frame date/time during playback and current server date/time with live video

core:

- message and error messages logged to logs/yyyy/mmdd-message.log files instead of console

GUI:

- navigation history stack and standard back button on all pages
- AUX commands support (buttons with status and command execution)
- GUI container now centered inside browser's window
- favicon added

Bug-fix:

- GUI/camera page: when in PLAY mode, clicking on menu occasionally (and especially on slow connections) failed to make the last camera frame hidden, causing last frame to become visible on top of the menu page
- fixed bug in `SystemState.deviceOffTimerSet()` : calls to this method replaced the timer even when a timer was already present; now if a timer is already set, this call has no effect
- fixed navigation bug on music and temperature zone pages that prevented going back to the zones pages

2.5.1

Temperature control:

- support for central unit firmware v.3.0 (fix number of away days returned in monitor session)

Bug-fix:

- `hsyco.openDeviceStatus()` is now called for TEMP status initialization only if `tempZones` variable is initialized in `hsyco.ini` and server index is 0 for `OpenWebNetMonitor` thread
- GUI fix of -1 temperature display during status initialization

2.5.0

Access control:

- `keys.ini` file renamed `keys.data`

Core:

- timers/alarms management and GUI

Startup scripts:

- StartupItem script changed to log messages to console.log file in Hsyco home directory

2.4.3

Core:

- sun rise/set time (reset night sensor group at civil sun rise time, with adjustable offset)
- new GUI graphics

2.4.2

Location services:

- syslog server and parser for Apple Extreme and NETGEAR APs, with user event callback and configurable zone page

Access control:

- automatic reload of access.ini file based on file's last modified time
- keys generated in a network range (trusted or not trusted) are only accepted within that same network range

Bug-fix:

- cameras playback GUI fixes

2.4.1

Core:

- Open Web Net auxiliary commands support
- process camera upload via HTTP, save images and manage system state information, with user actions callback
- motion video browser

Bug-fix:

- various fixes to improve compatibility with Symbian S60 browsers and when working in slow mode

2.4.0

Core:

- Multiple Open Web Net servers support

2.3.0

Access control:

- HTTPS server implementation

Bug-fix:

- SlimServer getState error when library not available

2.2.2

Music:

- new music interface, with zones selection and synch page
- added shuffle and repeat status
- new home theater interface
- new user callback for slim status change events

Bug-fix:

- synching a player with itself was allowed and created a problem with SlimServer. Now a self-synch command from the client is ignored and not sent to SlimServer
- text selection on double-click is now disabled
- IE7.0 links to menu and zones now work properly
- fixed tap highlight with iPhone

2.2.1

Security:

- extended key cookie hash
- 10 sec. delay on PIN/PUK authentication, synchronous block
- security logging of PIN, PIN/PUK logins, login failures and not matching key cookies

2.2.0

Security:

- Remote access support and user authentication

2.1.7

GUI:

- clicking on camera picture shows next camera
- fixed add button bug in music pages
- temperature controller fully implemented (central unit control and zones manual setting)

Bug-fix:

- fixed temp zones monitor of automatic/manual state

2.1.6

Core:

- Open Web Net monitor thread restarts if unable to load status at start-up
- fixed Emergency page link

GUI:

- state update timeout reduced from 15 to 12 seconds
- iPhone on-rotation scrolling below display when landscape
- fixed bug with display's button text clean-up timer
- temperature: central unit operation mode monitor

Music:

- Slimserver players controller implementation, with sync/unsync
- user events on power and volume change
- players status monitor logic completely redesigned

IRTrans:

- fixed closed connection bug when sending multiple commands

2.1.5

- fixed width issue on iPhone
- added dimmer status support, and on/off command
- added IRTrans support (with TCP monitor and UDP command connections)
- added Squeezebox display and control interface
- detect iPhone rotation event and adjust viewport for proper page width
- pages navigation
- stateDisplay(), looping through JSONState items rather than JSONTopo
- it is now possible to have a GUI with multiple buttons for the same OpenWebNet device

2.0

Cameras:

- camera integration with proxy features and image processing (2.0.3b)

GUI:

- added click, clickall, nopics flags
- full compatibility with IE7, Firefox, Safari, PS3, Nokia N80, iPod touch
- fixed bug with off timer, causing repeated off commands sent in some cases

HSYCO and Home Systems Consulting are registered trademarks of Home Systems Consulting SpA. Mac, iPod touch, iPhone are registered trademarks of Apple Inc. Java and JavaScript are registered trademarks of Sun Microsystems Inc. Linux is a registered trademark of Linux Mark Institute. Windows is a registered trademark of Microsoft Corporation. IRTrans is a registered trademark of IRTrans GmbH. SlimServer, SqueezeCenter, Squeezebox, Transporter are registered trademarks of Logitech Inc. NuVo is a registered trademark of NuVo Technologies LLC. AXIS is a registered trademark of AXIS Communications. KNX is a registered trademark of Konnex Association. DALI trademarks are registered for ZVEI - Zentralverband Elektrotechnik- und Elektronikindustrie e.V. DOMINO and CONTATTO are registered trademarks of Duemmegi srl. My Home is a registered trademark of BTicino Spa. Panasonic is a registered trademark of Matsushita Electric Industrial Co.Ltd. Mobotix is a registered trademark of Mobotix AG. Rovio is a registered trademark of WowWee Group Limited. HWg is a registered trademark of HW group s.r.o. Kissbox is a registered trademark of Kiss-Box B.V. Tecnoalarm is a registered trademark of Tecnoalarm Srl. Paradox is a registered trademark of Paradox. Bentel is a registered trademark of Bentel Security Srl. Ekahau is a registered trademark of Ekahau Inc. 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.

Home Systems Consulting SpA

www.homesystemsconsulting.com

