HiNTEGRO Script Editor v 2.1

HiNTEGRO ha la possibilità di gestire Logiche evolute grazie a questo strumento. Script Editor mette a disposizione un linguaggio di scripting (standard ECMA-262), con cui scrivere procedure personalizzate.

Oggetti Globali

Proprietà Valori

- NaN
- Infinity
- undefined

Proprietà Funzioni

- eval(x)
- parseInt(string, radix)
- parseFloat(string)
- isNaN(number)
- isFinite(number)
- decodeURI(encodedURI)
- decodeURIComponent(encodedURIComponent)
- encodeURI(uri)
- encodeURIComponent(uriComponent)

Proprietà Costruttori

- Object
- Function
- Array
- String
- Boolean
- Number
- Date
- RegExp
- Error
- EvalError
- RangeError
- ReferenceError
- SyntaxError
- TypeError
- URIError

Altre Proprietà

- Math
- JSON

Oggetti Standard

Proprietà Funzioni

- getPrototypeOf(O)
- getOwnPropertyDescriptor(O, P)
- getOwnPropertyNames(O)
- create(O [, Properties])
- defineProperty(O, P, Attributes)
- defineProperties(O, Properties)
- keys(O)

Proprietà Oggetti

Proprietà Funzioni

- toString()
- toLocaleString()
- valueOf()
- hasOwnProperty(V)
- isPrototypeOf(V)
- propertyIsEnumerable(V)

Function Objects

Function Prototype Object

Proprietà Funzioni

- toString()
- apply(thisArg, argArray)
- call(thisArg [, arg1 [, arg2, ...]])

Oggetti Array

Costruttore Array

Proprietà Funzioni

- isArray(arg)

Prototipo Array

Proprietà Funzioni

- toString()
- toLocaleString()
- concat([item1 [, item2 [, ...]]])
- join(separator)
- pop()
- push([item1 [, item2 [, ...]]])
- reverse()
- shift()
- slice(start, end)
- sort(comparefn)
- splice(start, deleteCount[, item1 [, item2 [, ...]]])
- unshift([item1 [, item2 [, ...]]])
- indexOf(searchElement [, fromIndex])
- lastIndexOf(searchElement [, fromIndex])
- every(callbackfn [, thisArg])
- some(callbackfn [, thisArg])
- forEach(callbackfn [, thisArg])
- map(callbackfn [, thisArg])
- filter(callbackfn [, thisArg])
- reduce(callbackfn [, initialValue])
- reduceRight(callbackfn [, initialValue])

Oggetti String

Costruttore String

Proprietà Funzioni

- fromCharCode([char0 [, char1 [,...]]])

Prototipo String

Proprietà Funzioni

- toString()
- valueOf()
- charAt(pos)
- charCodeAt(pos)
- concat([string1 [, string2 [, ...]]])
- indexOf(searchString ,position)
- lastIndexOf(searchString, position)
- localeCompare(that)
- match(regexp)
- replace(searchValue, replaceValue)
- search(regexp)
- slice(start, end)
- split(separator, limit)
- substring(start, end)
- toLowerCase()
- toLocaleLowerCase()
- toUpperCase()
- toLocaleUpperCase()
- trim()

Boolean Objects

Boolean Prototype Object

Proprietà Funzioni

- toString()
- valueOf()
- Number Objects
- Number Prototype Object

Proprietà Funzioni

- toString(radix)
- toLocaleString()
- toFixed(fractionDigits)
- toExponential(fractionDigits)
- toPrecision(precision)

Oggetti Math

Proprietà Valori

- E
- LN10
- LN2
- LOG2E
- LOG10E
- PI
- SQRT1_2
- SQRT2

Proprietà Funzioni

- abs(x)
- acos(x)
- asin(x)
- atan(x)
- atan2(y, x)
- ceil(x)
- cos(x)
- exp(x)
- floor(x)
- log(x)
- max([value1 [, value2 [, ...]]])
- min([value1 [, value2 [, ...]]])
- pow(x, y)
- random()
- round(x)
- sin(x)
- sqrt(x)
- tan(x)

Oggetti Date

Costruttore Date

Proprietà Funzioni

- now()
- parse(string)
- UTC(year, month [, date [, hours [, minutes [, seconds [, ms]]]]])

Prototipo Date

Proprietà Funzioni

- toString()
- toDateString()
- toTimeString()
- toLocaleString()
- toLocaleDateString()
- toLocaleTimeString()
- valueOf()
- getTime()
- getFullYear()
- getUTCFullYear()
- getMonth()
- getUTCMonth()
- getDate()
- getUTCDate()
- getDay()
- getUTCDay()
- getHours()
- getUTCHours()
- getMinutes()
- getUTCMinutes()
- getSeconds()
- getUTCSeconds()
- getMilliseconds()
- getUTCMilliseconds()
- getTimeZoneOffset()
- setTime(time)
- setMilliseconds(ms)
- setUTCMilliseconds(ms)
- setSeconds(sec [, ms])
- setUTCSeconds(sec [, ms])
- setMinutes(min [, sec [, ms]])
- setUTCMinutes(min [, sec [, ms]])
- setHours(hour [, min [, sec [, ms]]])
- setUTCHours(hour [, min [, sec [, ms]]])
- setDate(date)
- setUTCDate(date)
- setMonth(month [, date])
- setUTCMonth(month [, date])
- setFullYear(year [, month [, date]])
- setUTCFullYear(year [, month [, date]])
- toUTCString()
- toISOString()

- toJSON()

Oggetto RegExp

Prototipo RegExp

Proprietà Funzioni

- exec(string)
- test(string)
- toString()

Error Objects

Error Prototype Object

Proprietà Valori

- name
- message

Proprietà Funzioni

- toString()

Oggetto JSON

Proprietà Funzioni

- parse(text [, reviver])
- stringify(value [, replacer [, space]])

Esempi di utilizzo dei metodi e delle funzioni disponibili

```
eval(x)

        scen.log(level.INFO, "eval(10 * 20) = " + eval(10 * 20));

parseInt(string, radix)

        scen.log(level.INFO, "parseInt('10', 10) = " + parseInt('10', 10));
```

```
    scen.log(level.INFO, "parseInt('010', 10) = " + parseInt('010', 10));
        scen.log(level.INFO, "parseInt('10', 8) = " + parseInt('10', 8));
        scen.log(level.INFO, "parseInt('0x10') = " + parseInt('0x10'));
        scen.log(level.INFO, "parseInt('10', 16) = " + parseInt('10', 16));


isNaN(number)

        scen.log(level.INFO, "isNaN(10) = " + isNaN(10));
        scen.log(level.INFO, "isNaN(NaN) = " + isNaN(NaN));

JSON

        var text = '{ "employees" : [' +
                '{ "firstName":"John" , "lastName":"Doe" },' +
                '{ "firstName":"Anna" , "lastName":"Smith" },' +
                '{ "firstName":"Peter" , "lastName":"Jones" } ]}';
        var obj = JSON.parse(text);
        scen.log(level.INFO, "obj.employees[1].firstName +
obj.employees[1].lastName = " + obj.employees[1].firstName + " " +
obj.employees[1].lastName);


Date

        scen.log(level.INFO, "Date.now() milliseconds from midnight = " +
Date.now());

        var d = new Date();
    var n = d.getFullYear();
        var m = d.getMonth();
        var g = d.getDate()

        scen.log(level.INFO, "getFullYear() = " + n);
        scen.log(level.INFO, "getMonth() = " + m);
        scen.log(level.INFO, "getDate() = " + g);
```

Logger

Livelli

- level.DEBUG
- level.INFO
- level.WARNING
- level.ERROR
- level.SYSTEM

Utilizzo

scen.log(level,message);

```
// Esempio di utilizzo della funzione Logger per stampare un ERRORE
scen.log(level.ERROR,"Questo é un errore");

// Esempio di utilizzo della funzione Logger per stampare un messaggio di SISTEMA
scen.log(level.SYSTEM,"Questo é un messaggio di sistema");

// Esempio di utilizzo della funzione Logger per stampare un AVVISO
scen.log(level.WARNING,"Questo é un messaggio di avviso");

// Esempio di utilizzo della funzione Logger per stampare una INFORMAZIONE
scen.log(level.INFO,"Questo é un messaggio di informazione");

// Esempio di utilizzo della funzione Logger per stampare un messaggio di
CONTROLLO
scen.log(level.DEBUG,"Questo é un messaggio di debug");
```

Esecuzione Comandi

Script Editor può eseguire delle azioni sui Gruppi Funzionali definiti, attraverso la funzione setAction.

setAction

scen.setAction(action);

action verrà sostituito con l'azione da eseguire nei gruppi funzionali.

action sarà composto di 3 parti divise da un punto (.): Esempio LUX.1.ON

- "LUX" gruppo funzionale Luci
- "1" identificativo univoco gruppo funzionale Luci
- "ON" azione da Eseguire

```
// Esempio di esecuzione comando di accensione gruppo funzionale luce 1
//scen.setAction("LUX.1.ON");
```

N.B. nel caso di luci/automazioni/relè se viene inserito l'id = 0 l'azione verrà attuata per tutte le luci/automazioni/rele (Esempio LUX.0.OFF spegne tutte le luci)

Di seguito le possibili azioni da inviare:

N.B. id verrà sostituito con l'identificatore univoco del gruppo funzionale.

Luci

- LUX.id.ON Accensione luce
- LUX.id.OFF Spegnimento luce
- LUX.id.TM.XXX Accensione luce con timer di spegnimento dopo XXX millisecondi

- LUX.id.DIM.XXX Accensione luce al livello dimmer XXX espresso in % (0 -100)
- LUX.id.UP Accensione luce salendo di un livello di dimming
- LUX.id.DWN Spegnimento luce scendendo di un livello di dimming
- LUX.id.TGL Toggle luce. Se accesa viene spenta o contrario

Tapparelle

- AUT.id.UP Comando salita/apertura tapparella
- AUT.id.DWN Comando discesa/chiusura tapparella
- AUT.id.STP Comando stop
- AUT.id.LVL.XXX Imposta la percentuale di apertura 0 chiuso - 100 aperto

Rele

- RLE.id.ON Accensione RELE
- RLE.id.OFF Spegnimento RELE
- RLE.id.BK.XXX Lampeggio RELE XXX espresso in ms (1000 = 1 secondo)
- RLE.id.TM.XXX Accensione Temporizzata RELE espresso in ms (1000 = 1 secondo)

HVAC Clima / Termoregolazione

- THM.id.ON Accensione
- THM.id.OFF Spegnimento
- THM.id.MODE.EST Cambio Modalità Estate
- THM.id.MODE.INV Cambio Modalità Inverno
- THM.id.MODE.DRY Cambio Modalità Dry
- THM.id.MODE.FAN Cambio Modalità Fan

Attenzione le modalità Dry e Fan devono essere supportate dall'Hardware usato altrimenti i comandi non verranno presi in considerazione

- THM.id.SETP.XXX Impostare Setpoint XXX livello decimale es. 25.5
- THM.id.UMI.XXX Umidita' relativa livello decimale es. 40.5
- THM.id.FSP.XXX Fanspeed velocita percentuale XXX (0 Auto 33 - 66 - 99 corrispondente al numero delle velocita)

Messaggi / Notifiche

Attenzione che la separazione usa il carattere | per evitare che punti o altro possano essere mal interpretati. Nella sezioni impostazioni devono essere inseriti correttamente i parametri per smtp server, telegram api e id chat e infime per gli SMS deve essere aggiunto e configurato un modem GSM con SIM installata e con credito disponibile.

- MAIL|TO|OBJECT|MESSAGE Mail Message

- BOT|MESSAGE Telegram Message
- BOP|URL Telegram Photo
- SMS|TO|MESSAGE SMS Message

Irrigazione / Sprinkler

- SPR.ONT.XXX Aggiunge una zona di irrigazione per XXX minuti
- SPR.SEQ.ON Attiva Sequenza
- SPR.SEQ.OFF Ferma Sequenza

Video

- VID.id.PLAY Visualizza lo streaming dell'oggetto video corrispondente all'id
- VID.id.OFF Interrompe lo streaming e ritorna alla schermata di base. "0" è l'id per il generale

```
// Esempio di esecuzione comando RELE ON
//scen.setAction("RLE.1.ON");

// Esempio di esecuzione comando RELE OFF
//scen.setAction("RLE.1.OFF");

// Esempio di esecuzione comando RELE LAMPEGGIO 500 ms
//scen.setAction("RLE.1.BK.500");

// Esempio di esecuzione comando RELE TEMPORIZZATO 1 s
//scen.setAction("RLE.1.TM.1000");

// Esempio di esecuzione comando Automazione salita
//scen.setAction("AUT.1.UP");

// Esempio di esecuzione comando Automazione discesa
//scen.setAction("AUT.1.DWN");

// Esempio di esecuzione comando Automazione stop
//scen.setAction("AUT.1.STP");

// Esempio di invio Mail con Oggetto: JS SCRIPT, Testo: Start dello script JS e
test Mail Sender.
//scen.setAction("MAIL|destinatario@dominio.com|JS SCRIPT|Start dello script JS e
test Mail Sender");

// Esempio di invio Telegram
//scen.setAction("BOT|Start dello script JS e test Telegram");

// Esempio di invio SMS
//scen.setAction("SMS|3480000000|Start dello script JS e test Telegram")
```

callScenario

scen.callScenario(idScenario);

idScenario verrà sostituito con l'id corrispondente allo scenario che si vuole attivare

```
// Esempio di esecuzione dello scenario con id = 1
//scen.setAction(1);
```

CallBack Gruppi Funzionali

HiNTEGRO per ogni Gruppo Funzionale invia l'evento alla CallBack di riferimento. In questo modo, quando si accende una luce, si apre o si chiude una tapparella etc. verrà notificato lo Script Editor richiamando la CallBack di riferimento.

Luci

event_onLight

```
function event_onLight(_type, _vobj, _status, _level) {
}
```

Questa funzione viene richiamata per ogni evento che accade per il gruppo funzionale Luci.

- _type = Tipo Oggetto
  - (LUX)
- _vobj = Identificatore Univoco Gruppo funzionale
- _status = Stato Gruppo funzionale
  - (ON | OFF)
- _level = Valore per livello dimmer
  - (0 - 100)

Automazioni

event_onAutomation

```
function event_onAutomation(_type, _vobj, _status, _level) {
}
```

Questa funzione viene richiamata per ogni evento che accade per il gruppo funzionale Automazioni.

- _type = Tipo Oggetto
  - (AUT)
- _vobj = Identificatore Univoco Gruppo funzionale

- _status = Stato Gruppo funzionale
  - (UP | DOWN | STOP)
- _level = Valore per livello tapparelle con ritorno di stato del posizionamento
  - (0 - 100)

Rele

event_onRele

```
function event_onRele(_type, _vobj, _status, _level) {
}
```

Questa callback viene richiamata per ogni evento che accade per il gruppo funzionale Rele.

- _type = Tipo Oggetto
  - (RLE)
- _vobj = Identificatore univoco Gruppo Funzionale
- _status = Stato Gruppo Funzionale
  - (ON | OFF | BK | TM)
- _level = valore riferito allo stato
  - (Bk = Blink _level = ms lampeggio | TM = Temporizzato _level = ms temporizzato)

Ingressi

event_onInput

Tipi

- SHP Pressione Breve
- SEP Inizio Pressione Lunga
- VEP Continua Pressione Lunga
- EEP Fine Pressione Lunga

```
function event_onInput(_type, _vobj, _evt, _tick) {
}
```

Questa funzione viene richiamata per ogni evento che accade per il gruppo funzionale Ingressi.

- _type = Tipo Oggetto
  - (INP)
- _vobj = Identificatore Univoco Gruppo funzionale

- _evt = Stato Gruppo funzionale
  - (SHP | SEP | VEP | EEP)
- _tick = Valore di ritorno pressione continua

Eventi

event_onEvent

```
function event_onEvent(_type, _vobj, _evt) {
}
```

Questa funzione viene richiamata per ogni evento che accade per il gruppo funzionale Eventi.

- _type = Tipo Oggetto
  - (EVT)
- _vobj = Identificatore Univoco Gruppo funzionale
- _evt = Stato Gruppo funzionale
  - (SHP | SEP | VEP | EEP)

HVAC Climatizzazione / Termoregolazione

event_onThermo

```
function event_onThermo(_type, _vobj, _status, _mode, _setpoint, _ambient,
_fanspeed, _umidity)
{
}
```

Stato

- ON
- OFF

Modalità

- 0 Condizionamento COOL
- 1 Riscaldamento HEAT
- 2 Deumidificatore DRY
- 3 Ventilazione FAN

Fanspeed

- 0 Automatica
- 1 Minima
- 2 Media
- 3 Massima

Questa funzione viene richiamata per ogni evento che accade per il gruppo funzionale Termoregolazione.

- _type = Tipo Oggetto
  - (THM)
- _vobj = Identificatore Univoco Gruppo funzionale
- _status = Stato
  - 0 OFF
  - 1 ON
- _mode = Modalità funzionamento
- 0 COOL
- 1 HEAT
- 2 DRY
- 3 FAN
- _setpoint = Valore Temperatura Set Point impostata
- _ambient = Valore Temperatura Ambiente rilevata
- _umidity = Valore umidità rilevata

Orologio Astronomico

event_onAstronomic

Tipi

- SUN.RISE Sorgere del sole
- SUN.SET Tramonto del sole
- SUN.ELE Elevazione in gradi del sole
- SUN.TRANSIT Mezzogiorno solare

```
function event_onAstronomic(_type, hour, minute )
{
}
```

Questa funzione viene richiamata per ogni evento astronomico legato al Sole.

- _type = Tipo evento Astronomico

- (SUN.RISE | SUN.SET | SUN.ELE | SUN.TRANSIT)
- hour = ore (0 - 24)
- minute = minuti (0 - 59)

N.B. Per l'evento SUN.ELE nella variabile hour viene riportato il grado di Elevazione

Messaggistica

event_onMessage

```
function event_onMessage(_message)
{
}
```

Questa callback viene richiamata per ogni messaggio che arriva per SMS e Telegram. Può essere usata per impartire comandi ad HiNTEGRO attraverso i BOT Telegram.

Eventi Generici

event_onGenericMessage

```
function event_onGenericMessage(_message)
{
}
```

La CallBack viene richiamata per tutti gli eventi mappati come generici. Al momento sono disponibili questi eventi

MASER

```
// MSR.W00.ON   Preallarme minimo livello
// MSR.W00.OFF  Ripristino Preallarme minimo livello
// MSR.W01.ON   Allarme minimo livello
// MSR.W01.OFF  Ripristino Allarme minimo livello
// MSR.W02.ON   Preallarme max livello
// MSR.W02.OFF  Ripristino Preallarme max livello
// MSR.W03.ON   Allarme max livello
// MSR.W03.OFF  Ripristino Allarme max livello
// MSR.W04.ON   Carico cisterna in corso
// MSR.W04.OFF  Ripristino Carico cisterna in corso
// MSR.W05.ON   Fine carico cisterna
// MSR.W05.OFF  Ripristino Fine carico cisterna
// MSR.W06.ON   Allarme Acqua
// MSR.W06.OFF  Ripristino Allarme Acqua
// MSR.W07.ON   Contante Cassaforte
// MSR.W07.OFF  Ripristino Contante Cassaforte
// MSR.W08.ON   Automatico
```

```
// MSR.W08.OFF  Manuale
// MSR.W09.ON   Apertura Cassaforte
// MSR.W09.OFF  Ripristino Apertura Cassaforte
// MSR.W10.ON   Input Programmazione
// MSR.W10.OFF  Ripristino Input Programmazione
// MSR.W11.ON   Cambio Prezzi
// MSR.W11.OFF  Ripristino Cambio Prezzi
// MSR.W12.ON   Apertura anti-tamper
// MSR.W12.OFF  Ripristino Apertura anti-tamper

// MSR.A00.ON   Errore singola sonda
// MSR.A00.OFF  Ripristino Errore singola sonda
// MSR.A01.ON   Offline singola sonda
// MSR.A01.OFF  Ripristino Offline singola sonda
// MSR.A02.ON   Offline tutte le sonde
// MSR.A02.OFF  Ripristino Offline tutte le sonde
// MSR.A03.ON   Lettore Banconote Fuori Servizio
// MSR.A03.OFF  Ripristino Lettore Banconote Fuori Servizio
// MSR.A04.ON   Esaurimento carta Stampante
// MSR.A04.OFF  Ripristino Esaurimento carta Stampante
// MSR.A05.ON   Stampante fuori servizio
// MSR.A05.OFF  Ripristino Stampante fuori servizio
// MSR.A06.ON   Erogatore fuori servizio
// MSR.A06.OFF  Ripristino Erogatore fuori servizio
// MSR.A07.ON   Piazzale fuori servizio
// MSR.A07.OFF  Ripristino Piazzale fuori servizio
// MSR.A08.ON   Ripristino Piazzale fuori servizio
// MSR.A08.OFF  Ripristino Memoria Backup piena
// MSR.A09.ON   Minimo livello
// MSR.A09.OFF  Ripristino Minimo livello
// MSR.A10.ON   Caduta tensione
// MSR.A10.OFF  Ripristino Caduta tensione
// MSR.A11.ON   Errore Display
// MSR.A11.OFF  Ripristino Errore Display
// MSR.A12.ON   Errore Lettroe Tessere
// MSR.A12.OFF  Ripristino Lettore Tessere

// MSR.F00.ON   Uscita inattesa carburante
// MSR.F00.OFF  Ripristino inattesa carburante
// MSR.F16.ON   Mancanza connessione TCP/IP
// MSR.F16.OFF  Ripristino connessione TCP/IP
```

Legrand

```
LGR.AUX.VAR1.VAR2.VAR3

VAR1 = id web server
VAR2 = numero AUX
VAR3 = valore AUX
```

```
LGR.ALR.VAR1.VAR2.VAR3

VAR1 = id web server
```

```
VAR2 = zona
VAR  = evento INTRUSION | TAMPER | PANIC
```

zona: - 0 Centrale - n Zona

evento: Zona 0 - MAINTENANCE - ACTIVE - BATTERYFAULT - BATTERYOK - NETWORKKO - NETWORKOK - ARM - DISARM - BATTERYKO Zona n - INCLUDED - EXCLUDED - INTRUSION - TAMPER - PANIC

```
LGR.CIT.VAR1
```

VAR1 = id web server

Chiamata video citofonica

Evento CUSTOM richiamato da SCENARIO

```
LGR.OWN.VAR1.VAR2
VAR1 = id web server
VAR2 = frame
```

VAR1 = id webserver corrispondete all'ID nella pagina gateway Legrand webserver VAR2 = frame inviata

Evento richiamato da SCENARIO

JAV.XXX

JAV = chiave che identifica un'azione personalizzata da eseguire in javascript XXX = stringa che verrà passata al javascript per eseguire l'azione personalizzata

Es. JAV.PROVA all'interno di onGenericMessage si può parsare il seguente messaggio e chiedere che se la stringa contiene PROVA esegua una determinata azione (es. accensione di una luce)

```
function event_onGenericMessage(_message) {
        var msg = _message.split(".");
        if (msg[0] == "JAV")
        {
                if(msg[1] == "PROVA")
                        {
                                scen.log(level.DEBUG, "Messaggio di prova");
                        }
        }
}
```

Energia

```
function event_onEnergy(_type, _vobj, _mesure, _direction, _value, _delta) {
```

```
}
```

- _type = ENR
- _vobj = ID Oggetto Virtuale
- _mesure = Tipologia misurazione effettuata
  - powernow (potenza istantanea)
- direction
  - in (Prodotta)
  - out (Consumata)
- _value = Valore misurazione
- _delta = Differenza tra Energia Prodotta e Consumata precalolato

Google Home WebHook

```
function event_onGoogleMessage(_message)
{
}
```

Questa callback viene richiamata per ogni JSON inviato dal servizio dialogflow Google

_message = JSON inviato da Google

KNX

E' possibile scrivere direttamente sul bus KNX con la funzione KnxWrite.

```
scen.KnxWrite(a, l, g, value, datapoint);
```

- a = int Area
- l = int Linea
- g = int Gruppo
- value = valore formattato esempio: datapoint 1 (1 bit true | false) datapoint 5 (8 bit unsigned value 0-255)
- datapoint = datapoint knx

datapoint sono predefiniti e vanno scelti tra i seguenti:

knxdtp.Type

Type:

```
# 1.xxx
- SWITCH        ( 1.001 ON          | OFF         )
- BOOLEAN       ( 1.002 TRUE        | FALSE       )
```

```
 - ENABLE         ( 1.003 ENABLE      | DISABLE    )
 - RAMP           ( 1.004 RAMP        | NORAMP     )
 - ALARM          ( 1.005 ALARM       | NOALARM    )
 - BINARY         ( 1.006 HIGH        | LOW        )
 - STEP           ( 1.007 INCREASE    | DECREASE   )
 - UPDOWN         ( 1.008 DOWN        | UP         )
 - OPENCLOSE      ( 1.009 CLOSE       | OPEN       )
 - STARTSTOP      ( 1.010 START       | STOP       )
 - STATE          ( 1.011 ACTIVE      | INACTIVE   )
 - INVERT         ( 1.012 INVERTED    | NOTINVERTED)
 - CYCLIC         ( 1.013 CYCLIC      | STARTSTOP  )
 - CALCULATED     ( 1.014 CALCULATED  | FIXED      )
 - RESET          ( 1.015 RESET       | NOACTION   )
 - ACKNOWLEDGE    ( 1.016 ACKNOWLEDGE| NOACTION   )
 - TRIGGERED      ( 1.017 TRIGGERED   | NOTRIGGERED)
 - OCCUPANCY      ( 1.018 OCCUPIED    | NOTOCCUPIED)
 - OPEN           ( 1.019 OPEN        | CLOSED     )
 - LOGICAL        ( 1.021 AND         | OR         )
 - SCENE          ( 1.022 SCENEB      | SCENEA     )
 - MOVEUP         ( 1.022 MOVEUPDOWN  | UPDOWNSTOP )
 - COOLHEAT       ( 1.100 HEATING     | COOLING    )

 # 5.xxx
 - PERC100        ( 5.001 PERCENTAGE      0 - 100 )
 - ANGLE          ( 5.003 DEGREES         0 - 255 )
 - PERC255        ( 5.004 PERCENTAGE      0 - 255 )
 - RATIO          ( 5.005 RATIO           0 - 255 )
 - TARIFF         ( 5.006 TARIFF          0 - 255 )
 - COUNTER        ( 5.010 PULSE           0 - 255 )


 # 9.xxx
 - TEMPERATUREC   ( 9.001 TEMPERATURE C )
 - TEMPERATUREDIFF       ( 9.002 KELVIN/HOUR )
 - LUX    ( 9.004 LUX INTENSITY )
 - WINDSPEEDMS    ( 9.005 WIND SPEED M/S )
 - PRESSURE       ( 9.006 PRESSURE Pa )
 - HUMIDITY       ( 9.007 HUMIDITY % )
 - PARTMILLION    ( 9.008 PARTS/MILLION Ppa )
 - AIRFLOW        ( 9.009 AIR FLOW M3/H )
 - TIMESEC        ( 9.010 TIME SECONDS )
 - TIMEMS         ( 9.011 TIME MILLISECONDS )
 - VOLTAGEMV      ( 9.020 VOLATAGE MILLIVOLTS )
 - CURRENTMA      ( 9.021 CURRENT MILLIAMPERE )
 -       POWERWMQ        ( 9.022 POWER WATT/M2 )
 - KELVINPERC     ( 9.023 KELVIN/PERCENT )
 - POWERKW        ( 9.024 KW/H )
 - FLOWLH         ( 9.025 VOLUME FLOW L/H )
 - RAINLMQ        ( 9.026 RAIN AMOUNT L/M2 )
 - TEMPERATUREF   ( 9.027 TEMEPERATURE F )
 - WINDSPEEDKMH   ( 9.028 WIND SPEED KM/H)
```

17.xxx

```
- SCENE (17.001 SCENE NUMBER)
```

esempio di invio datapoint 1.001 su a/l/g 0/0/1

```
scen.knxWrite(0,0,1,"OFF",knxdtp.SWITCH)
```

Questa callback viene richiamata per ogni datagramma che transita nel bus KNX

```
function event_onKnxDatagram(_phy_a, _phy_b, phy_c, a, l, g, value)
{
}
```

- _phy_a = Physycal address A
- _phy_b = Physycal address B
- _phy_c = Physycal address C
- _a = Area
- _l = Line
- _g = Group
- value = Valore non tipizzato (da convertire con il datapoint corretto) e separato da |

Inoltre e' possibile convertire il valore ricevuto indicando il datapoint corretto con la funzione knxConvertValue

```
var test = scen.knxConvertValue(knxdtp.BOOLEAN,value);
var test2 = scen.knxConvertValue(knxdtp.SWITCH,value);
var test3 = scen.knxConvertValue(knxdtp.ENABLE,value);
var test4 = scen.knxConvertValue(knxdtp.WINDSPEEDKMH,value);
```

Nell'esempio precedente avremo i seguenti risultati in formato stringa

test = "TRUE" oppure "FALSE"

test2 = "ON" oppure "OFF"

test3 = "ENABLE" oppure "DISABLE"

METEO

Questa callback viene richiamata per ogni evento inviato dalle stazioni meteo e inverter integrati con stazioni meteo

```
function event_onMeteoMessage(_type,_value)
```

```
{
}
```

_type = tipo di misurazione _value = valore ricevuto

- value
  - WNS.XXX = WIND SPEED
  - MWN.XXX = MAX WIND SPEED
    - IRR.XXX = IRRADIANCE

    XXX valore ricevuto con riferimento all'unita' di misura usata dalla centralina

TIMER

Questa callback viene richiamata per ogni evento crono impostato

function event_onTimerMessage(_id,_value) { }

_id = id crono _value = valore ricevuto

- value
  - ON Inizio crono
  - OFF Fine crono

HiNTEGRO mette a disposizione anche timer con id per gestire eventi multipli

Di seguito un esempio per creare un timer ripetitivo

var idTimer = scen.setInterval(expression, delay_time);

IdTimer = idTimer creato

expression = espressione javascript da Eseguire

scen.clearInterval(idTimer);

funzione per terminare un timer ed eliminarlo dalla memoria. Una volta richiamata questa funzione il timer non esiste più.

```
// Esempio di timer con funzione a scadenza

function main() {
        var statusOn = true;
        var id = scen.setInterval(function() {
                if (statusOn) {
                        scen.setAction("RLE.1.OFF");
                } else {
                        scen.setAction("RLE.1.ON");
                }
                statusOn = !statusOn;
```

```
        }, 2000);
}
```

Di seguito un esempio per creare un timer single shot

var idTimer = scen.setTimeout(expression, delay_time);

IdTimer = idTimer creato

expression = espressione javascript da Eseguire

scen.clearTimeout(idTimer);

funzione per terminare un timer ed eliminarlo dalla memoria. Una volta richiamata questa funzione il timer non esiste più.

```
// Esempio di timer con funzione a scadenza

function main() {
        var statusOn = true;
        var id = scen.setTimeout(function() {
                if (statusOn) {
                        scen.setAction("RLE.1.OFF");
                } else {
                        scen.setAction("RLE.1.ON");
                }
                statusOn = !statusOn;
        }, 2000);
}
```

CONTROLLO ACCESSI

Questa callback viene richiamata per ogni evento inviato dal controllo accessi

```
function event_onAccessEvent(_gate,_type,_code,_result)
{

}
```

_gate = numero del varco a cui è avvenuto l'evento _type = tipo di evento _code = codice numerico o alfanumerico della tessera o del pin inserito _result = risultato dell'accesso

- type
  - KYP = KEYPAD (varco con tasti numerici)
  - RFD = varco con rfid
- result

- OK = l'autenticazione per accedere ha avuto successo
- KO = l'autenticazione per accedere non ha avuto successo

MAIN

HiNTEGRO carica lo script e lo esegue partendo dalla funzione main. Quindi per poter personalizzare HiNTEGRO attraverso lo script editor deve essere usata questa funzione. Dal main possono essere usati tutti i tipi di oggetti e variabili e funzioni elencati sopra.

```
function main() {
}
```

Librerie

Uart

Questa libreria permette di leggere e scrivere su una seriale collegata ad HiNTEGRO

Per creare la seriale

```
serial = new Uart();
```

Per settare i parametri usare le seguenti variabili:

FlowControl:

```
flowcontrol.noflowcontrol
flowcontrol.hardwarecontrol
flowcontrol.softwarecontrol
flowcontrol.unknownflowcontrol
```

BaudRate:

```
1200
2400
4800
9600
19200
38400
57600
115200
```

dichiarare poi tutte le callback necessarie per la ricezione degli eventi:

```
serial.onConnect.connect(_onConnect);
serial.onDisconnect.connect(_onDisconnect);
serial.onData.connect(_onData);
serial.onError.connect(_onError);

function _onConnect(_id) {}
function _onDisconnect(_id) {}
function _onData(_id,_data) {}
function _onError(_id, _error) {}
```

Per scrivere sulla seriale si puo' usare:

scrittura di una stringa formato testo

writeData(String);

oppure scrittura di un array e specificando la linghezza dell'array da scrivere

writeData(Bytearray, length);

```
serial.writeData(stringa);

var bytearray[] = {0x00,0x02,0x03};
serial.writeData(bytearray,3);
```

esempio script:

```
var serial;

function main() {


        serial.setID("serial1");
        serial.setBaudRate(9600);
        serial.setFlowControl(flowcontrol.noflowcontrol);

        serial.onConnect.connect(_onConnect);
        serial.onDisconnect.connect(_onDisconnect);
        serial.onData.connect(_onData);
        serial.onError.connect(_onError);

        serial.openSerialPort("/dev/ttyUSB0");

}

function _onConnect(_id) {
        scen.log(level.WARNING,"connesso");
        serial.writeData("invio dati \n");
}
```

```
function _onDisconnect(_id) {
        scen.log(level.WARNING,"Disconnesso");
}

function _onData(_id,_data) {
        scen.log(level.WARNING,_data);
}

function _onError(_id, _error) {

}
```

HiNTEGRO mette a disposizione delle librerie per la gestione diretta dei bus di campo direttamente da Javascript.

Libreria Modbus Client

Questa libreria permette di creare Modbus Client.

Per creare il client basta richiamare il metodo new e settare ip e porta di connessione.

```
client = new Modbus();
client.connectTo("192.168.1.1",502);
```

dichiarare poi tutte le callback necessarie per la ricezione degli eventi:

```
client.onWriteSuccess.connect(_onWriteSuccess);
client.onWriteError.connect(_onWriteError);
client.onReadError.connect(_onReadError);
client.onReadRegister.connect(_onReadRegister);
client.onReadRegisters.connect(_onReadRegisters);
client.onReadCoil.connect(_onReadCoil);
client.onReadCoils.connect(_onReadCoils);
client.onReadInputStatus.connect(_onReadInputStatus);
client.onReadInputStatuses.connect(_onReadInputStatuses);
client.onReadInputRegister.connect(_onReadInputRegister);
client.onReadInputRegisters.connect(_onReadInputRegisters);


function _onWriteSuccess(station,address){}
function _onWriteError(station,address,value){}
function _onReadError(station,address,value){}
function _onReadRegister(station,address,value){}
function _onReadCoil(station,address,value){}
function _onReadRegisters(station,address,value){}
function _onReadCoils(station,address,value){}
function _onReadInputStatus(station,address,value){}
function _onReadInputStatuses(station,address,value){}
function _onReadInputRegister(station,address,value){}
function _onReadInputRegisters(station,address,value){}
```

Nel file _hintegro.js esistono e non devono essere toccate le prototipazioni delle CallBack!!!

per poter leggere e scrivere i registri modbus usare metodi seguenti:

```
var ret = [];
        ret = client.r_registers(station,address,n registri);
        ret = client.r_register(station,address);

        ret = client.r_coil(station,address);
        ret = client.r_coils(station,address,n registri);

        ret = client.r_input_status(station,address);
        ret = client.r_input_statuses(station,address,n registri);

        ret = client.r_input_register(station,address);
        ret = client.r_input_registers(station,address,n registri);


        ret = client.w_registers(station,address,[value,value,value]);
        ret = client.w_register(station,address,value);
        ret = client.w_coils(station,address,[value,value,value]);
        ret = client.w_coil(station,address,value);
```

Di seguito viene riportato un esempio completo di un programma javascript per la gestione completa del modbus client:

```
var id;

function main() {

        this.id = 1;

        var client = new Modbus();
        client.connectTo("192.168.2.111",502);

        client.onWriteSuccess.connect(_onWriteSuccess);
        client.onWriteError.connect(_onWriteError);
        client.onReadError.connect(_onReadError);
        client.onReadRegister.connect(_onReadRegister);
        client.onReadRegisters.connect(_onReadRegisters);
        client.onReadCoil.connect(_onReadCoil);
        client.onReadCoils.connect(_onReadCoils);
        client.onReadInputStatus.connect(_onReadInputStatus);
        client.onReadInputStatuses.connect(_onReadInputStatuses);
        client.onReadInputRegister.connect(_onReadInputRegister);
        client.onReadInputRegisters.connect(_onReadInputRegisters);

        var ret = [];
        ret = client.r_registers(1,1,4);
        ret = client.r_register(1,100);

        ret = client.r_coil(1,3);
        ret = client.r_coils(1,1,11);

        ret = client.r_input_status(1,11);
```

```
        ret = client.r_input_statuses(1,1,11);

        ret = client.r_input_register(1,11);
        ret = client.r_input_registers(1,1,11);


        ret = client.w_registers(1,1,[1,2,3,4]);
        ret = client.w_register(1,1,23);
        ret = client.w_coils(1,1,[1,0,1,0,1,1,1,1,1]);
        ret = client.w_coil(1,1,0);


}

function _onWriteSuccess(station,address)
{
        scen.log(level.DEBUG, "Write sucessfully on station: " + station + "
address: " + address);
}

function _onWriteError(station,address,value)
{

}

function _onReadError(station,address,value)
{

}

function _onReadRegister(station,address,value)
{
        scen.log(level.DEBUG, "Read register on station: " + station + " address:
" + address + " value: " + value);
}

function _onReadCoil(station,address,value)
{
        scen.log(level.DEBUG, "Read coil on station: " + station + " address: " +
address + " value: " + value);
}

function _onReadRegisters(station,address,value)
{
        value.forEach(function(element) {
                scen.log(level.INFO,"Registres Read from station: " + station + "
from address: " + address + " registers value: " + element);
                address++;
        });
}

function _onReadCoils(station,address,value)
{
        value.forEach(function(element) {
                scen.log(level.INFO,"Coils Read from station: " + station + " from
address: " + address + " registers value: " + element);
                address++;
        });
```

```
}

function _onReadInputStatus(station,address,value)
{
        scen.log(level.DEBUG, "Read input status on station: " + station + "
address: " + address + " value: " + value);
}


function _onReadInputStatuses(station,address,value)
{
        value.forEach(function(element) {
                scen.log(level.INFO,"Input status Read from station: " + station +
" from address: " + address + " registers value: " + element);
                address++;
        });
}

function _onReadInputRegister(station,address,value)
{
        scen.log(level.DEBUG, "Read input register on station: " + station + "
address: " + address + " value: " + value);
}


function _onReadInputRegisters(station,address,value)
{
        value.forEach(function(element) {
                scen.log(level.INFO,"Input register Read from station: " + station
+ " from address: " + address + " registers value: " + element);
                address++;
        });
}
```

Libreria Modbus Server

Questa libreria permette di creare Modbus Server anche con istanze multiple, basta assegnare un riferimento univoco in fase di creazione.

Le callback degli eventi avranno sempre il riferimento univoco riportato nei messaggi in modo di capire in qualsiasi momento da quale istanza arriva il messaggio.

Per creare il server basta richiamare il metodo new e settare id univoco e porta TCP/IP:

```
testModbusServer = new modbusServer();

testModbusServer.setId(id);
testModbusServer.setPort(502);
```

dichiarare poi tutte le callback necessarie per la ricezione degli eventi:

```
testModbusServer.logMessage.connect(_logMessage);
testModbusServer.onNewConnection.connect(_onNewConnection);
testModbusServer.onError.connect(_onError);
testModbusServer.onWriteMultipleRegistersJS.connect(_onWriteMultipleRegisters);
testModbusServer.onWriteMultipleCoilsJS.connect(_onWriteMultipleCoils);
testModbusServer.onWriteSingleRegister.connect(_onWriteSingleRegister);
testModbusServer.onWriteSingleCoil.connect(_onWriteSingleCoil);

function _onWriteSingleCoil(srv, station, address, data){}
function _onWriteSingleRegister(srv, station, address, data){}
function _onWriteMultipleCoils(srv, station, address,count, data){}
function _onWriteMultipleRegisters(srv, station, address,count, data){}
function _onNewConnection(srv){}
function _onError(srv,message){}
function _logMessage(srv,message){}
```

e per ultimo far partire il modbusserver:

```
testModbusServer.startServer();
```

La libreria mette a disposizione la possibilità di scrivere e leggere i registri direttamente da javascript i metodi seguenti:

```
- testModbusServer.setCoilRegister(serverId, station, address, value);
        - value boolean

- testModbusServer.setDiscretInputRegister(serverId, station, address, value);
        -value boolean

- testModbusServer.setHoldingRegister(serverId, station, address, value);
        -value integer

- testModbusServer.setInputRegister(serverId, station, address, value);
        - value integer


- testModbusServer.setCoilRegister(serverId, station, address, value);
        - value boolean

- testModbusServer.setDiscretInputRegister(serverId, station, address, value);
        - value boolean

- testModbusServer.setHoldingRegister(serverId, station, address, value);
        - value integer


- testModbusServer.getInputRegister(serverId, station, address, value);
        - value integer
```

Assolutamente il metodo startServer() deve essere richiamato solo dopo aver dichiarato le callback precedentemente

elencate. Nel file _hintegro.js esistono e non devono essere toccate le prototipazioni delle CallBack!!!

Di seguito viene riportato un esempio completo di un programma javascript per la gestione completa del modbus server:

```
var testModbusServer;
var id;

function main() {

        this.id = 1;

        scen.log(level.INFO,"Start Modbus Server with id: " + this.id);

        testModbusServer = new modbusServer();

        testModbusServer.setId(id);

        testModbusServer.setPort(502);

        testModbusServer.logMessage.connect(_logMessage);
        testModbusServer.onNewConnection.connect(_onNewConnection);
        testModbusServer.onError.connect(_onError);

testModbusServer.onWriteMultipleRegistersJS.connect(_onWriteMultipleRegisters);
        testModbusServer.onWriteMultipleCoilsJS.connect(_onWriteMultipleCoils);
        testModbusServer.onWriteSingleRegister.connect(_onWriteSingleRegister);
        testModbusServer.onWriteSingleCoil.connect(_onWriteSingleCoil);

        testModbusServer.setHoldingRegister(1,1,150,56);


        testModbusServer.startServer();


}

function _onWriteSingleCoil(srv, station, address, data)
{

}

function _onWriteSingleRegister(srv, station, address, data)
{

}


function _onWriteMultipleCoils(srv, station, address,count, data)
{

}

function _onWriteMultipleRegisters(srv, station, address,count, data)
{

                data.forEach(function(element) {
                scen.log(level.INFO,"Multiple Holding Register serverId: " + srv +
```

```
" Values Write from station: " + station + " from address: " + address + " for: "+
count + " registers value: " + element);
        });data
        .forEach(function(element) {

                scen.log(level.INFO,"Multiple Holding Register serverId: " + srv +
" Values Write from station: " + station + " from address: " + address + " for: "+
count + " registers value: " + element);
        });

}

function _onNewConnection(srv)
{
        scen.log(level.DEBUG,"New Connection incoming on srv " + srv);
}

function _onError(srv,message)
{
        scen.log(level.ERROR,"error on srv " + srv + " : " + message);
}

function _logMessage(srv,message)
{
   scen.log(level.DEBUG,"message on srv " + srv + " : " + message);
}
```