vBulletin GERMANY

Abfrage & Steuerung von Viessmann Heizungen

Druckbare Version

Seite 4 von 8 Erste ... 2 3 4 5 6 ... Letzte

40 Beiträge dieses Themas auf einer Seite anzeigen

bastianstrauss

28.10.13, 18:27

Hier meine XML's
Anhang 23245
Für 20A5, 20AB, 2048

Liste der Anhänge anzeigen (Anzahl: 1)

Es handelt sich dabei nur um die Daten die auch wirklich funktionieren und Werte liefern!
Ach ja wenn Werte ins Negative rutschen gibt es ein Problem mit der Version von Coyote. Hier mal ne Lösung für das Außentemperaturproblem:

PHP-Code:

```
        case "Temperature10Minus":
             $lowByte = ord( $Data[0] );
             $highByte = ord( $Data[1] );
             if ($highByte == 255) {
              return ((65536-(( $highByte * 256 ) + $lowByte ))/10)*-1;
               } else {
             return (( $highByte * 256 ) + $lowByte ) / 10;
             }
         break;
```

Schablone

30.10.13, 10:53

Besten Dank. Ich schau mir die Liste heute Abend an!!

Flobo

07.11.13, 12:47

Huhu,

Liste der Anhänge anzeigen (Anzahl: 1)

ich nutze auch das Script von Coyote. Vielen dank für die Mühe.
Ich lese momentan eine Vitotronic 200 KW2 damit aus (ID 2098) und erstelle mir momentan die XML Datei dazu.

Jetzt meine Frage, wie lasst ihr die Abfragen über den Seriellen Port laufen. Ich habe gemerkt wenn ich alle Kommandos in einem Script durchlaufen lasse kommt recht schnell das PHP timeout.
Im moment Kaskadiere ich die Abfragen eine nach der anderen durch, heißt ich deaktiviere das eigene Ereignis und aktiviere das nächste. Ist recht mühsam aber scheint soweit zu funktionieren.

PHP-Code:

```
<?

  // Variablen Includieren
  include( "ViessmannDeviceTools.inc.php" );

    // Event IDs ermitteln
$eventidselfarray = IPS_GetChildrenIDs($IPS_SELF); // Eigenes Event 15 Minuten
$eventidself = $eventidselfarray[0]; // ID aus Array ziehen

  // Verbindung zur Heizung öffnen
  ViessmannOpen();

  // Wert Abfragen

  ViessmannSetVariableByCommand( "Sammelstoerung",25540 /*[Vitotronic 200 KW2\Heizungsdaten\Sammelstörung]*/ );

  // Verbindung schliessen
  ViessmannClose();

IPS_SetEventActive($eventidself, false); // Eigener Timer 15 Minuten deaktivieren
IPS_SetEventActive(32314 /*[Vitotronic 200 KW2\Heizungsdaten\Brennerleistung\Refresh\]*/ , true);

?>
```

Anhang 23371

kronos

07.11.13, 16:27

Ich habe mich hier inspirieren lassen und lese einen Parameter nach dem anderen aus.

Flobo

07.11.13, 17:08

Ui ... das ist natürlich auch eine Idee. Coole Sache danke dir. Werd mich da gleich mal ranmachen heute abend.

Gesendet von meinem GT-I9300 mit Tapatalk

Flobo 07.11.13, 20:20

Und da kommt auch gleich meine nächste Frage zu dem Thema.
Ich habe mir nu mal eine weile lang die werte aufgezeichnet und mit Highcharts ausgeplottet, und ich habe gemerkt das es sehr oft zu Fehlwerten kommt.
zb. die Speichertemperatur springt ab und zu mal auf 128 Grad oder eine Sammelstörung kommt mal sporadisch.

Liste der Anhänge anzeigen (Anzahl: 1)

An was könnte das liegen ?

Anhang 23381

bastianstrauss 07.11.13, 22:51

Hallo
bei mir lag es daran, dass nur ein Teil übertragen wird, aber das Script dann aus dem halben Teil + dem Antwort Byte einen falschen Wert ermittelt. Ich habe die Abfrage dahin gehend geändert, dass die Werte vor Auswertung im Script geprüft wird, ob ein Antwortbyte von der Heizungsanlage mit dabei ist. Wenn das der Fall ist wird die Anfrage verworfen. Ich hänge morgen früh mal die PHP bei!

Flobo 07.11.13, 23:41

Ah okay,

danke dir ich freu mich drauf. Werd mich in der Zwischenzeit mal um das o.g. Script kümmern. :)
.. IPS macht nich Süchtig nein xD

Thomas 08.11.13, 18:33

Ähnliche Probleme hatte ich auch.
Habe meine Scripts vom Post nochmal erweitert.
Nun wird auch das Timeout richtig ausgewertet. Damit geht die Abfrage im Gut-Fall schneller. Bei mir ist ca. innerhalb von 3 Sekunden die Antwort da. Ich frage 7 Werte ab, das dauert ca. 20 Sekunden und liegt innerhalb des PHP-Timeouts.
Ebenso kann man nun statt der VariablenID auch 0 übergeben, damit wird der Wert in der Funktion zurückgegeben und man kann eine Fehlerbehandlung machen (siehe Viessmann_Send.ips.php). Die Funktion von Bastian für negative Temperaturen hab ich mit eingebaut, auch wenn ich das momentan nicht brauche.
Ebenso gibt es nun einen Fehlerzähler, der bei jedem erkannten Fehler hochgezählt wird. Den kann man auswerten. Ich habe ungefähr 10 Fehler pro Tag, was bei 3360 Abfragen pro Tag ok ist.

Gruß Thomas

Flobo 09.11.13, 14:56

Hiho,

danke dir, habe deine Scripte gerade übernommen. Mal schauen ob sich das beruhigt, werd das mal beobachten.

Zum Abfragen nutze ich jetzt das Script :

PHP-Code:

```
<?

include( "ViessmannDeviceTools.inc.php" );
$step = GetValue(28063 /*[Vitotronic 200 KW2\Links\Abfrage\Abfrageintervall]*/);

if(($IPS_SENDER <> "TimerEvent")and($step == 0)){
    SetValue(28063 /*[Vitotronic 200 KW2\Links\Abfrage\Abfrageintervall]*/, 1);
    IPS_SetScriptTimer($IPS_SELF, 10);
        ViessmannOpen();
        ViessmannSetVariableByCommand( "Aussentemperatur",42389 /*[Vitotronic 200 KW2\Mitte\IST - Temperaturen\Aussentemperatur]*/ );
        ViessmannClose();
    return;
}
if(($IPS_SENDER == "TimerEvent")and($step == 1)){
    SetValue(28063 /*[Vitotronic 200 KW2\Links\Abfrage\Abfrageintervall]*/, 2);
    IPS_SetScriptTimer($IPS_SELF, 10);
        ViessmannOpen();
        ViessmannSetVariableByCommand( "Kesseltemperatur",12612 /*[Vitotronic 200 KW2\Mitte\IST - Temperaturen\Kesseltemperatur]*/ );
        ViessmannClose();
    return;
}
if(($IPS_SENDER == "TimerEvent")and($step == 2)){
    SetValue(28063 /*[Vitotronic 200 KW2\Links\Abfrage\Abfrageintervall]*/, 3);
    IPS_SetScriptTimer($IPS_SELF, 10);
        ViessmannOpen();
        ViessmannSetVariableByCommand( "Speichertemperatur",11245 /*[Vitotronic 200 KW2\Mitte\IST - Temperaturen\Warmwasserspeicher]*/ );
        ViessmannClose();
    return;
}
if(($IPS_SENDER == "TimerEvent")and($step == 3)){
    SetValue(28063 /*[Vitotronic 200 KW2\Links\Abfrage\Abfrageintervall]*/, 4);
```

```
        IPS_SetScriptTimer($IPS_SELF, 10);
            ViessmannOpen();
            ViessmannSetVariableByCommand( "Vorlauftemperatur",58221 /*[Vitotronic 200 KW2\Mitte\IST - Temperaturen\Vorlauftemperatur]*/ );
            ViessmannClose();
        return;
    }
    if(($IPS_SENDER == "TimerEvent")and($step == 4)){
        SetValue(28063 /*[Vitotronic 200 KW2\Links\Abfrage\Abfrageintervall]*/, 5);
        IPS_SetScriptTimer($IPS_SELF, 10);
            ViessmannOpen();
            ViessmannSetVariableByCommand( "Sammelstoerung",25540 /*[Vitotronic 200 KW2\Rechts\Wartung & Störung\Sammelstörung]*/);
            ViessmannClose();
        return;
    }
    if(($IPS_SENDER == "TimerEvent")and($step == 5)){
        SetValue(28063 /*[Vitotronic 200 KW2\Links\Abfrage\Abfrageintervall]*/, 6);
        IPS_SetScriptTimer($IPS_SELF, 10);
            ViessmannOpen();
            ViessmannSetVariableByCommand( "Brennerstoerung",27009 /*[Vitotronic 200 KW2\Rechts\Wartung & Störung\Brennerstörung]*/);
            ViessmannClose();
       return;
    }
    if(($IPS_SENDER == "TimerEvent")and($step == 6)){
        SetValue(28063 /*[Vitotronic 200 KW2\Links\Abfrage\Abfrageintervall]*/, 7);
        IPS_SetScriptTimer($IPS_SELF, 10);
            ViessmannOpen();
            ViessmannSetVariableByCommand( "Speicherladepumpe",19704 /*[Vitotronic 200 KW2\Rechts\Anlagen\Speicherladepumpe]*/);
            ViessmannClose();
       return;
    }
    if(($IPS_SENDER == "TimerEvent")and($step == 7)){
        SetValue(28063 /*[Vitotronic 200 KW2\Links\Abfrage\Abfrageintervall]*/, 0);
        IPS_SetScriptTimer($IPS_SELF, 10);
            ViessmannOpen();
            ViessmannSetVariableByCommand( "Zirkulationspumpe",33397 /*[Vitotronic 200 KW2\Rechts\Anlagen\Zirkulationspumpe]*/);
            ViessmannClose();
       return;
    }


    ?>
```

Das rufe ich alle 2 Minuten durch einen extra Timer auf und wird dann abgearbeitet durch die Step Variable.

Flobo

10.11.13, 20:11

So,

Liste der Anhänge anzeigen (Anzahl: 1)

das Script von Thomas hat wirkung gezeigt, vielen Lieben dank nochmal dafür.
Nun hat sich das ganze beruhigt.

Anhang 23425

Nun mal eine ganz doofe Frage .. ich schäm mich schon fast dafür.
Kann mir jemand Quick & Dirty erklären wie aus einer **F3 01** 49,9 °C werden ? Hab den ganzen Nachmittag gegoogelt wie ein Irrer .. ich komm einfach nicht drauf. Ich hab mir Hex to Decimal Converter angeschaut .. wenn ich den Hex Wert F3 01 eingebe dann kommt da in Dezimal 62209 raus. Ich kapiers einfach net :) auch das Script leuchtet mir noch net ganz ein.

Flobo

10.11.13, 21:14

AHHHHHH ... jetzt hats Klick gemacht.
Ich versuchs mal zu erklären.

Das ist der Teil an dem der Hex bzw. ASCII zu nem Temperaturwert wird :

PHP-Code:

```
            case "Temperature10":
                $lowByte = ord( $Data[0] );
                $highByte = ord( $Data[1] );
                return (( $highByte * 256 ) + $lowByte ) / 10;
                break;
```

Im Debugger steht zb. **CA 02** auf ASCII **Ê<STX>** (wobei <STX> kein Zeichen ist sondern ein Steuerzeichen.)

CA = LowByte
02 = HighByte

In Dezimal

CA = 202
02 = 2

Die Formel sieht dann so aus :
((2 * 256) + 202) / 10 = 71,4

Das heißt die Kesseltemperatur beträg **71,4 °C**

Habe ich das soweit richtig kapiert ? :) .. boah ich liebe sowas xD

bastianstrauss

11.11.13, 19:17

Hi Flobo,

ich habe auch lange wie ein Irrer an den HexWerten gewerkelt, daher auch mein Script für negative Werte. Mit 2 Bytes (2x 8 Bit = 16 Bit) ist es eigentlich schlecht negative Werte darzustellen, daher schickt die Viessmann Therme also den größten Wert raus. Ein Wort = 16 Bits / man kann auch sagen Integer 65535 (2x256).

Also FF FF

F: 15 · 1 = 15
F: 15 · 16 = 240
F: 15 · 256 = 3840
F: 15 · 4096 = 61440
———————
65535

Sind es also draußen -5°C dann sendet die Viessmann CD FF

Also (65.535 - (255 * 256 + 205))/10 *-1 =

Hier die entsprechende Zeile:

PHP-Code:

```
return ((65536-(( $highByte * 256 ) + $lowByte ))/10)*-1;
```

Und so sieht die komplette Berechnung aus:

PHP-Code:

```
case "Temperature10Minus":
              $lowByte = ord( $Data[0] );
              $highByte = ord( $Data[1] );
              if ($highByte == 255) {
               return ((65536-(( $highByte * 256 ) + $lowByte ))/10)*-1;
                } else {
              return (( $highByte * 256 ) + $lowByte ) / 10;
              }
          break;
```

Viel Spass beim rechnen! :-D

P.S. Allerdings funktioniert diese Funktion nur bis -25,5°C, da ich das highByte mit FF Abfrage, wird es kälter sinkt dieser HighByte auf FE, das ist dann die Temperatur von -25,6°C bis -51°C usw.

Evtl müsste die If Abfrage so lauten:

PHP-Code:

```
if ($highByte >= 245)
```

Damit wäre diese Funktion immer erfüllt selbst bei 0 Kelvin ;-) Dann würde die Antwort der Heizung so aussehen (haben dann eher Supraleiter) F5 54 => (65536 - ((245*265)+84))/10 *-1 = 273,1°C oder 0° Kelvin

aghira

11.12.13, 22:56

Partybetrieb

Guten Abend,

ich kämpfe seit geraumer Zeit mit Schreiben Richtung Anlage ... :confused:

Versuche einfach den Partybetrieb über IP-Symcon einzuleiten. Abgekürzt so darstellbar:

PHP-Code:

```
RegVar_SendText($id_com, chr(0x04));
RegVar_SendText($id_com, chr(0x16).chr(0x00).chr(0x00));

$IPS_Party_SET= chr(0x41).chr(0x06).chr(0x00).chr(0x02).chr(0x23).chr(0x03).chr(0x01).chr(0x01);

RegVar_SendText($id_com, $IPS_Party_SET);
```

Hat jemand eine Idee ob ich auf den richtigen Weg bin? Hat das schon jemand gemacht über Protokoll 300?

Herzlichen Dank!

Flobo

23.05.14, 22:24

Hiho,

ich versuche momentan das Script für das KW Protokoll umzuschreiben, bin aber auf ein Problem mit den Integerwerten gestoßen.

Ich bekomme die Hex Antwort der Heizung : **33 D5 01 00**
Der Integerwert sollte **120115** sein.

hat jemand ne Ahnung wie genau das zum Umrechnen ist ? Kleine Kniffelaufgabe xD .. ich bin schon gescheitert, habe auch nicht aus dem Script rauslesen können wie genau die Mathematische Formel dafür ist.

spooniester

23.05.14, 23:45

Hi, hab da eine Idee aber um festzustellen ob die nicht kompletter Blödsinn ist benötige ich ein zweites Beispiel, also Hex und Integerwert!

Flobo

23.05.14, 23:54

Huhu,

ich hätte da noch die Brennerbetriebsstunden.

B3 D2 0B 06
.. sagt IPS, ich weiß das man diesen Wert durch 3600 teilen muss um auf die Stunden zu kommen.

Vitosoft sagt 28177,26 Stunden

Gleiche Zeit der Abfrage.

Thomas

24.05.14, 00:17

Hallo zusammen,
die Integer und Hexzahlen sind doch komplett identisch.
120115dez entspricht 01D533 in hex
und 060BD2B3 ist 101437200 Sekunden und damit 28177,26 Stunden.
Ich versteh jetzt das Problem nicht?

Gruß Thomas.

Flobo

24.05.14, 09:59

Hiho,

warum von Hinten nach Vorne ? :) .. sorry ich kapier das mit diesem Hex Integer kram nur so halb, bis jetzt is der Groschen noch nicht gefallen^^

Der erste Wert sind die Brennerbetriebsstunden, dort funktioniert der Trick mit den Zahlen von Hinten nach vorne nicht.

Ich habe :

Hex : 00 01 D5 38
Dec (von Hinten nach vorne) : 0 1 213 56
Tatsächlicher Wert (Aus der Viessmann Software Vitosoft) : 120120

passt also irgendwie net zusammen *grübel*

muckel

27.06.14, 15:11

Zitat:

> *Zitat von **Flobo***
>
> *Hiho,*
>
> *warum von Hinten nach Vorne ? :) .. sorry ich kapier das mit diesem Hex Integer kram nur so halb, bis jetzt is der Groschen noch nicht gefallen^^*
>
> *Der erste Wert sind die Brennerbetriebsstunden, dort funktioniert der Trick mit den Zahlen von Hinten nach vorne nicht.*
>
> *Ich habe :*
>
> *Hex : 00 01 D5 38*
> *Dec (von Hinten nach vorne) : 0 1 213 56*
> *Tatsächlicher Wert (Aus der Viessmann Software Vitosoft) : 120120*
>
> *passt also irgendwie net zusammen *grübel**

Mit welcher Addresse fragst du ab ?

Wenn du mit BetriebsstundenStufe 1 08A7 abfragst ... versuch mal mit 0886 , stimmt bei mir genau ;) wenn man es durch 3600 teilt

Fraunhofer

04.07.14, 11:27

Moin Moin,

ich habe es mit eurer Anleitung nun ebenfalls geschaft die Wert aus der Heizung abzufragen, nun hab ich zwar noch das Problem das meine Raum Soll Temp bei 6500° liegt aber ich denke da könnt ihr mit helfen.

Die Anleitung aus Coyotes Post habe ich genutzt um die
ViessmannDeviceOutputHandlingKW
ViessmannDeviceTools.inc
ViessmannVariables.inc

von Thomas genutzt weil ich dachte die Umrechnung der Fehlerhaften werte ist damit automatisch gegeben.

Wie bekomme ich mit der Umrechnung hin , oder hab ich nur was übersehen ?

Fraunhofer

25.07.14, 09:38

Moin Moin,

auf die Frage gibts wohl keine Antwort mehr , dann stell ich mal eine neue :)

im V-Control 125 , kann man umschalten zwischen :

Abschaltbetrieb
Nur WW
WW - Heizen
Immer Tagesbetrieb
immer Absenkbetrieb

, wie geht das mit IPS ?

andreasypsilon

25.07.14, 20:16

Hi,

schade, dass es hier nicht so richtig weiter geht... Ich komme mit meiner Heizung auch nicht so richtig zu Rande. Viele Werte bekomme ich einfach nicht ausgelesen. Zum Beispiel hätte ich vor einigen Wochen die Störung gerne vorher gewusst, bevor meine Frau unter der Dusche plötzlich kaltes Wasser hatte... Kein Druck in der Heizung = Notbetrieb :mad:

Bezgl. Deiner Frage: Betriebsarten und Partymodus schalte ich so:

PHP-Code:

```
<?
//$zugriff = GetValue (30593 /*[Objekt #30593 existiert nicht]*/ );
if($IPS_SENDER == "WebFront")

{
    include( "ViessmannDeviceTools.inc.php" );

    switch($IPS_VALUE)
    {

          case 0:
                //if ($zugriff == 0)
                {
                     ViessmannOpen();
                           ViessmannSetData( "BetriebsartA1M1", chr(0x00) );
                     IPS_Sleep(200);
                     ViessmannClose();
                        SetValue(38479 /*[Allgemein\Heizung\Heizung Ausgabe\Betriesartschalter\BetriebsString]*/ , "aus");
                        SetValue($IPS_VARIABLE, $IPS_VALUE);
                }

        break;

                case 1:
                //if ($zugriff == 0)
                {
                     ViessmannOpen();
                     ViessmannSetData( "BetriebsartA1M1", chr(0x01) );
                     IPS_Sleep(200);
                     ViessmannClose();
                     SetValue(38479 /*[Allgemein\Heizung\Heizung Ausgabe\Betriesartschalter\BetriebsString]*/ , "WW");
                     SetValue($IPS_VARIABLE, $IPS_VALUE);
                 }
            break;

            case 2:
                 // if ($zugriff == 0)
               {
                     ViessmannOpen();
                     ViessmannSetData( "BetriebsartA1M1", chr(0x02) );
                     ViessmannClose();
                     SetValue(38479 /*[Allgemein\Heizung\Heizung Ausgabe\Betriesartschalter\BetriebsString]*/ , "Hz+WW");
                     SetValue($IPS_VARIABLE, $IPS_VALUE);
                 }
            break;

            case 3:
                       //if ($zugriff == 0)
               {
                     ViessmannOpen();
                     ViessmannSetData( "BetriebsartA1M1", chr(0x03) );
                     ViessmannClose();
                     SetValue(38479 /*[Allgemein\Heizung\Heizung Ausgabe\Betriesartschalter\BetriebsString]*/ , "reduziert");
                     SetValue($IPS_VARIABLE, $IPS_VALUE);
                 }
            break;

            case 4:
                       //if ($zugriff == 0)
               {
                     ViessmannOpen();
                     ViessmannSetData( "BetriebsartA1M1", chr(0x04) );
                     ViessmannClose();
                     SetValue(38479 /*[Allgemein\Heizung\Heizung Ausgabe\Betriesartschalter\BetriebsString]*/ , "normal");
                     SetValue($IPS_VARIABLE, $IPS_VALUE);
                 }
            break;
```

```
}}
?>
```

Du musst dieses Skript einem zu erstellenden Schalter zuweisen. Ich glaube, dazu musst Du eine Variable anlegen (Integer) ein neues Profil erstellen mit der Anzahl der Optionen, die du im Skript als Case definierst... Und als Aktion obiges Skript zuweisen...

Das kann ein PHP Profi bestimmt besser erklären. Bin nach über zwei Jahren IPS immer noch Anfänger ;-)

Sei vorsichtig mit den ViessmannSetData Schreibzugriffen auf die Heizung! Alles auf eigene Gefahr. Ich hab keine Ahnung, was passiert wenn man die falschen Datenpunkte erwischt!?!

Hoffe, etwas geholfen zu haben...

Fraunhofer

29.07.14, 09:24

Moin Moin ,

leider echt tote Hose hier , ich bin leider 100km von der Heizung entfernt und lasse jetzt jemanden hinfahren der gucken soll ob die Signale vom IPS an der Anlage gezeigt werden.

im Webfront bekomme ich einen Fehler angezeigt , aber nur das Kreuz und keine Info dazu .Im Normalfall sollte das ja kein Hexenwerk sein mit dem Senden.

Wenn es fertig ist , würde ich meinen das ich mein Projekt mal mit dem Exporter von Raketenschnecke hier einstelle und so haben alle was davon .

Fraunhofer

30.07.14, 13:03

Moin Moin,

Die Steuerung funktioniert .Im verlauf findet ihr den Export von meinem Projekt , das euch den Seriellen Port mit allen Variablen anlegt und das Heizungssteuerungs Script , das im zweiten Teil ist , installiert wird.

Export zum Anlegen des Seriellen Ports , dort muss dann einfach nur noch euer Com Port eingetragen werden

PHP-Code:

```
<?
/*****************************************************************************************************************
*  this Install-Script was automated generated by RS IPS Project-Exporter                                        *
*  © by Raketenschnecke 2012-2014, mail: raketenschnecke@gmx.de                                                  *
*  // Projekt-HomePage: http://www.raketenschnecke.net/rs-projekte/rs-ips-project-exporter/                       *
*****************************************************************************************************************/

/*****************************************************************************************************************
    Project: Viessmann Heizung Serial Port (Quell-ID: 53482), generated on 30.07.2014, 12:56 Uhr
*                                                                                                                *
*    Dieses Script beinhaltet ein automatisch in einem Script zusammengeführtes IPS-Projekt (Quell-Projekt)      *
*    Zur Installation des (Quell-) Projekts auf dem lokalen System (Zielsystem) bitte dieses Script an einer beliebigen *
*    Stelle im Objektbaum platzieren und einmalig manuell starten (Konfiguration siehe Block "Konfig" unten)     *
*  Es werden alle notwendigen Strukturen (Kategorien, Variablen, Scripte etc) aus dem Quell-Projekt im Ziel-System  *
*  angelegt.                                                                                                       *
*    Dies ist !!! KEINE !!! vollständige Projekt-Installation! D.h.: eine Konfiguration, Verlinkung von Objekten etc. *
*    muss durch Scripts des eigentlichen Projekts oder manuell durch den Anwender erfolgen.                      *
*  Bei Fragen zum installierten Projekt bitte den Projektautor fragen!                                             *
*                                                                                                                *
*****************************************************************************************************************/

##############################  Konfig ###########################################################################

// Ziel-WFC angeben
   $WFC_TargetID                = 0;     // bei falscher WFC-ID wird keine Installation von WFE-Komponenten vorgenommen
   $WFC_existItemoverwrite    = 1;  // 0=> bereits im Zielsystem bestehende WFC-Items werden NICHT überschrieben,
                                          // 1=> existierende Objekte werden überschrieben (default)

// Copy-Parameter (1: Objekte werden im Zielsystem installiert; 0: Objekte werden nicht installiert) +++++++++++++++++
    $VarProfile                = 1;    // 1: überschreibt vorhandene Profile im Zielsystem, 0: installiert nur neue Profile
    # !!! die folgende Option mit äußerster Vorsicht nutzen (Default-Wert 0)!!! #
    $existsScriptsoverwrite    = 0;    // 1: im Zielsystem existierende Scripte (=benamste Scriptfiles) werden überschrieben


############################## Main Area ###########################################################################

   ##### Project Exporter Comment: Projekt-Export Viessmann Heizung Serial Port (Quell-ID: 53482) vom 30.07.2014 12:56 #####

// IPS Version detektieren
   $IPS_VERSION        = IPS_GetKernelVersion();

// Inventorys laden
    $MessageProkoll   = array();
    $Inv                    = ObjectInventory();
    $VProfInv            = VarProfileInventory();
    if(function_exists('loadFileExportInventory'))
    {
        $ExpFileInv        = loadFileExportInventory();
        if($ExpFileInv[0] != NULL)
        createExportFiles($ExpFileInv);
    }

// Parameters-Array erzeugen
   $Parameters    = array('VarProfile'=>$VarProfile, 'ScrOverwrite'=>$existsScriptsoverwrite, 'WFCItems'=>$WFC_existItemoverwrite,  'Update'=>0

// letztes Logfile finden & Updatemodus setzen
    $lastLog_ID    = findlastLogfile();
    if($lastLog_ID > 0)
```

```
        {
            $Parameters['Update'] = 1;
            include $lastLog_ID.'.ips.php';

            // last Inventory laden
            $lastInv     = ObjectInventoryProtocol();

            // Meldungsausgabe
            if(isset($lastInv[0]))
            {
                $MessageProkoll['Info'][] = "#1004 letztes Inventory aus Protokollfile geladen";
            }

            // Last Inventory initialisieren
            $InvData            = explode(';', substr(preg_replace("/\r|\n/s", "", $lastInv[1]),0, -1));
           $InvObjID          = (int)explode(',', $InvData[0])[1];
           $lastInvObj        = createlastInventoryObjects($InvObjID);

            // Meldungsausgabe
            if(IPS_ObjectExists((int)$lastInvObj[0]['newObjectID']))
            {
               $MessageProkoll['Info'][] = "#1005 bestehendes Zielprojekt für Update gefunden: ".$lastInvObj[0]['ObjectName']." ID#".$lastInvObj[0]
            }
            else
            {
               $MessageProkoll['Failure'][] = "#1025 Fehler: kein bestehendes Zielprojekt gefunden";
               exit("Scriptabbruch: kein bestehendes Zielprojekt gefunden! Nur Neu-Installation möglich,\nbitte alle Child-Files unterhalb dieses S
            }
        }
        $MessageProkoll['InstallParameters']  = $Parameters;


    // Funktionsaufrufe
        // Variablenprofile aus Profile-Inventory holen und Profil-Baum generieren
        $VarProfile    = createVarProfileInventory();

        // Variablenprofile anlegen (abhängig von Parameters)
        createVarProfile($VarProfile);

        // Projekt Root-Element aus Inventory generieren und anlegen
        $ObjectTree = createInventoryObjects();

        // Child-Objekte vom Root aus Inventory generieren und anlegen
        $ObjInstallTree = createObjectsTree($ObjectTree);

        // Funktion zum Verlinken aller neu instalierten Elemente
        linknewObjects($ObjInstallTree);

        // Funktion zum Installieren der WFC-Items
        installWFC_Items($WFC_TargetID);

        // Install-Script ins Config-Verzeichnis verschieben
        moveInstallScripttoConfig();

        // Installationsprotokoll erzeugen
        $InstProtocol        = createObjInventoryProtocol($ObjInstallTree);

        // Script self-renaming
                IPS_SetName($_IPS['SELF'], 'Viessmann Heizung Serial Port (Quell-ID: 53482) 2014_07_30-12_56'.', Ziel-ID: '.$ObjInstallTree[0]['new


        // Installationsprotokoll schreiben
        CreateProtocolFile($InstProtocol);

    // ++++++++++++++ Script-Core ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


    // prüft, ob Logfiles unterhalb des Export-Scripts hängen und gibt die ID des letzten Logfiles aus +++++++++++++++++
    function findlastLogfile()
    {
       $Childs    = IPS_GetChildrenIDs($_IPS['SELF']);
       $ScrTS   = 0;

       if(isset($Childs[0]))
       {
          $ScrTS   = 0;
          $ScrID   = 0;
          for($i=0;$i<count($Childs);$i++)
          {
             // 1. Zeile aus jedem Protokollfile auslesen und Installationsdatum ermitteln
             $script      = IPS_GetKernelDir()."scripts\\".$Childs[$i].'.ips.php';
              $fh            = fopen($script, "r");
              $firstLine  = mb_strlen(fgets($fh));

                fseek($fh, $firstLine);
              $zeile        = explode(' ', fgets($fh));

              if(isset($zeile[4]))
              {
                  $TS       = strtotime($zeile[4].$zeile[5]);
                }
                else
               {
                   $MessageProkoll['Failure'][] = "#1024 Fehler: letztes Inventory aus Protokollfile NICHT geladen";
                   exit("Scriptabbruch: kein gültiges Protokollfile gefunden! Nur Installation möglich,\nbitte alle Child-Files unterhalb dieses Sc
               }

              // Child mit größtem Timestamp ermitteln
              if(($ScrTS < $TS))
              {
                 $ScrTS   = $TS;
                 $ScrID   = $Childs[$i];
                }
             fclose($fh);
          }
```

```
    }

     // alte Logfiles löschen
    for($i=0;$i<count($Childs);$i++)
    {
       if($ScrID != $Childs[$i]) IPS_DeleteScript($Childs[$i], true);
    }
    return @$ScrID;
}


// Last Install Einzelobjekte-Array (aus Inv-Protokoll) bauen
function createlastInventoryObjects($ID)
{
    global $lastInv;

    for($a=1;$a<count($lastInv);$a++)
    {
        // Einzel-Objekt zusammenstellen
        $InvData     = explode(';', substr(preg_replace("/\r|\n/s", "", $lastInv[$a]),0, -1));
        unset($InvData[0]);  // ersten Datensatz entfernen
        $InvData    = array_slice($InvData, 0);
        for($i=0;$i<count($InvData);$i++)
        {
           $KeyValue         = explode(',', $InvData[$i]);
            $lastInvPlain[$a][$KeyValue[0]]  = $KeyValue[1];
        }
    }
    $lastInvPlain  = array_slice($lastInvPlain, 0);

    return $lastInvPlain;
}

// Funktion installiert externe Scripte (sofern vorhanden)
function createExportFiles($Array)
{
   global                 $MessageProkoll;
    $PicList          = array('png', 'jpg', 'jpeg', 'gif');
    $Cnt                 = count($Array);
    for($i=0;$i<$Cnt;$i++)
    {

       $File                = explode('\\', $Array[$i]);
       $FileName           = trim(array_pop($File));
       $FilePath           = IPS_GetKernelDir().substr(str_replace($FileName, '', $Array[$i]), 0, -1);
       $FileExtension    = str_replace(".", '', substr($FileName, -4));
       $isPic            = in_array($FileExtension, $PicList);

        // Filecontent aus Funktion holen
        $FuncName           = 'ExtFile_'.preg_replace('/[^a-zA-Z0-9]/', '', $Array[$i]);
        $ScrContent       = $FuncName();
        if($isPic)
        {
           $ScrContent        = base64_decode($ScrContent);

        }
        else
        {
           $ScrContent        = str_replace("@$@", "\\", $ScrContent);
            $ScrContent        = str_replace("\\'", "'", $ScrContent);
        }

        // Ordner anlegen, wenn nicht vorhanden
            if (!is_dir($FilePath))
            mkdir ($FilePath  , 0777, true);

        // schreiben der Content-Datei
            $file            = $FilePath.'\\'.$FileName;

            $handle             = fopen($file, "w");
            $result            = fwrite($handle, $ScrContent);
            if($result)
            {
               $MessageProkoll['extFiles']['OK'][]         = $file;
            }
            else
            {
               $MessageProkoll['extFiles']['Failure'][]    = $file;
            }
            fclose($handle);
    }
}

// Array Variablenprofile-Inventory erstellen
function createVarProfileInventory()
{
   global $VProfInv;

   if(@$VProfInv[0] != NULL)
    {
       for($i=0;$i<count($VProfInv);$i++)
       {
          $SubSet[] = explode('|', substr(preg_replace("/\r\n/s", "", $VProfInv[$i]),0, -1));
        }

        for($p=0;$p<count($SubSet);$p++)
        {
           $SubSet[$p][0] = substr($SubSet[$p][0], 0, -1);
           $SubSet[$p][1] = substr(@$SubSet[$p][1], 0, -1);

           $Profiles[$p]    = explode(';', $SubSet[$p][0]);
           for($i=0;$i<count($Profiles[$p]);$i++)
           {
               $Key          = explode(',', $Profiles[$p][$i])[0];
                $Value          = explode(',', $Profiles[$p][$i])[1];
```

```
                $ProfileObj[$Key]     = $Value;
            }
            $VarProfile[$p]     = $ProfileObj;

            // Associations hinzufügen
            if(@$SubSet[$p][1] != NULL)
            {
                $AssRaw    = explode('§', $SubSet[$p][1]);
               for($i=0;$i<count($AssRaw);$i++)
               {
                    $AssBlock    = explode(';', $AssRaw[$i]);
                    for($v=0;$v<4;$v++)
                    {
                       if($AssBlock[$v] != NULL)
                        {
                           $Key        = explode(',',($AssBlock[$v]))[0];
                           $Value    = explode(',',($AssBlock[$v]))[1];
                           $ValuePair[$Key]  = $Value;
                        }
                    }
                    $newAssBlock[] = $ValuePair;
                    unset($ValuePair);
                }
             $VarProfile[$p]['Associations'] = $newAssBlock;
             unset($newAssBlock);
                unset($AssBlock[$p]);
            }
        }
        $MessageProkoll['Info'][] = "#1010 VarProfil-Array erstell";
        return $VarProfile;
    }
}

// Function Variablenprofil anlegen
function createVarProfile($VarProfile)
{
    global $Parameters;

    if(isset($VarProfile[0]))
    {
    // Profile-Array durchgehen
        for($i=0;$i<count($VarProfile);$i++)
        {
           $Profilname = $VarProfile[$i]['ProfileName'];
           $isSystemProfile  = strpos($Profilname, '~');
           if(($Parameters['VarProfile'] == 1) && (IPS_VariableProfileExists($Profilname)) && ($isSystemProfile === false))
           {
              $MessageProkoll['Install']['OK'] = "#2011 Profil $Profilname wird gelöscht";
              if(IPS_VariableProfileExists($Profilname)) IPS_DeleteVariableProfile($Profilname);
            }
            else
            {
               $MessageProkoll['Install']['User'] = "#2010 bestehendes Variablen-Profil $Profilname wird nicht überschrieben (Systemprofil oder

            }

            // Variablen-Profil anlegen (nur Neuanlage)
           if((!IPS_VariableProfileExists($Profilname))&& ($isSystemProfile === false))
            {
               IPS_CreateVariableProfile($Profilname, (int)$VarProfile[$i]['ProfileType']);
              IPS_SetVariableProfileText($Profilname, $VarProfile[$i]['Prefix'], $VarProfile[$i]['Suffix']);
              IPS_SetVariableProfileValues($Profilname, (float)$VarProfile[$i]['MinValue'], (float)$VarProfile[$i]['MaxValue'], (float)$VarProf
             IPS_SetVariableProfileDigits($Profilname, (int)$VarProfile[$i]['Digits']);
                IPS_SetVariableProfileIcon($Profilname, $VarProfile[$i]['Icon']);
                $MessageProkoll['Install']['OK'] = "#2012 Profil $Profilname angelegt";

                // Associations setzen
                if(isset($VarProfile[$i]['Associations']))
                {
                    $Ass = $VarProfile[$i]['Associations'];
                    for($a=0;$a<count($Ass);$a++)
                    {
                       // Falls Assoziation "Name und Icon" leer sind, wird Leerzeichen bei Name eingefügt (sonst Fehlermeldung
                       // bei Assoziations-Zuweisung
                       if(($Ass[$a]['Name'] == NULL) && ($Ass[$a]['Icon'] == NULL)) $Ass[$a]['Name'] = ' ';
                        IPS_SetVariableProfileAssociation($Profilname, (float)$Ass[$a]['Value'], $Ass[$a]['Name'], $Ass[$a]['Icon'], (int)$Ass[
                    }
                    $MessageProkoll['Install']['OK'] = "#2013 Profilassociations für Profil $Profilname angelegt";
                }
            }
        }
        return;
    }
}

// Objekt-Tree generieren und Childs holen
function createInventoryObjects()
{
    global $Inv;
    global $IObjects;
   $DataSet     = explode(';', substr(preg_replace("/\r|\n/s", "", $Inv[1]),0, -1));
   $ObjID          = (int)explode(',', $DataSet[0])[1];
   $Objects[0]    = createObject($ObjID);
   if(isset($Objects[0]['ChildrenIDs']) != NULL)
   $Objects[0]['ChildrenIDs']    = createChildObjects($Objects[0]['ChildrenIDs']);
    return $Objects;
}

// rekursive Objektanlage aller Objekte im Ziel-Projekt
function createObjectsTree($Objects)
{
//global $Objects;
    for($i=0;$i<count($Objects);$i++)
    {
        // Objekt anlegen
        $Objects[$i] = Objekteanlegen($Objects[$i]);
```

```
        // nach Childs durchsuchen
        if(isset($Objects[$i]['ChildrenIDs']))
        {
           for($ch=0;$ch<count($Objects[$i]['ChildrenIDs']);$ch++)
           {
              $Objects[$i]['ChildrenIDs'][$ch]['newParentID']    = $Objects[$i]['newObjectID'];
           }
           // nächste Rekursion:
           $newObjects = $Objects[$i]['ChildrenIDs'];
           $Objects[$i]['ChildrenIDs']  = createObjectsTree($newObjects);
        }
    }
    return $Objects;
}

// Inventory rekursiv abarbeiten und Einzelobjekte zur Anlage übergeben
function createChildObjects($Childs)
{
    for($i=0;$i<count($Childs);$i++)
    {
        $Childs[$i]    = createObject($Childs[$i]);
        if(isset($Childs[$i]['ChildrenIDs']))
        {
           $newChilds  = $Childs[$i]['ChildrenIDs'];
            $Childs[$i]['ChildrenIDs']  = createChildObjects($newChilds);
        }
    }
    return $Childs;
}


// Einzel-Objekt aus Inventory generieren
function createObject($ObjID)
{
   global $Inv;
   global $Parameters;
    global $lastInvObj;

    // Inventory nach Objekt-ID durchsuchen und Array-ID zurückgeben
   for($s=0;$s<count($Inv);$s++)
   {
      if(strstr($Inv[$s], 'ObjectID,'.$ObjID)) $Key = $s;
   }
      $DataSet = explode(';', substr(preg_replace("/\r|\n/s", "", $Inv[$Key]),0, -1));

    // Objekt Key-Value Zuordnung
   for($p=0;$p<count($DataSet);$p++)
   {
       $Key          = explode(',', $DataSet[$p])[0];
        $Value          = explode(',', $DataSet[$p])[1];
        $Object[$Key]    = $Value;
    }

    // newObjectID aus lastInventory (Inst-Protokoll) ankleben, wenn Update = 1
    if($Parameters['Update'] == 1)
    {
       for($l=0;$l<count($lastInvObj);$l++)
       {
          if($Object['ObjectID'] == $lastInvObj[$l]['ObjectID'])
          $Object['newObjectID'] = @$lastInvObj[$l]['newObjectID'];
       }
    }

    // Child-IDs ankleben
   for($i=0;$i<count($Inv);$i++)
   {
        $DataSet = explode(';', substr(preg_replace("/\r|\n/s", "", $Inv[$i]),0, -1));
       for($p=0;$p<count($DataSet);$p++)
       {
           $Key          = explode(',', $DataSet[$p])[0];
            $Value          = explode(',', $DataSet[$p])[1];
            $Subject[$Key]     = $Value;
        }
        if(@$Subject['ParentID'] == $Object['ObjectID']) $Object['ChildrenIDs'][]  = $Subject['ObjectID'];
    }

    return $Object;
}

// +++ Function ZielProjekt-Objekte anlegen ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
function Objekteanlegen($Object)
{
    global $Parameters;
    global $MessageProkoll;
    global $IPS_VERSION;

   $MessageProkoll['Install']['OK'][] = "#3011 Objekt ".$Object['ObjectName']."., altID #".$Object['ObjectID']." zur Anlage übernommen";

    // ObjectInfo Zeilenumbruch zurückwandeln
    $Object['ObjectInfo'] = str_replace("§&§", "\r\n", $Object['ObjectInfo']);
    $Object['ObjectInfo'] = str_replace("§&&§", ",", $Object['ObjectInfo']);
    $Object['ObjectInfo'] = str_replace("§&&&§", ";", $Object['ObjectInfo']);

   switch ($Object['ObjectType'])
    {
        case 0: // Kategorien
                if((@$Object['newObjectID'] > 0) && ($Object['isConfigTree'] == 0))
                {
                $Cat_ID        = (int)$Object['newObjectID'];
                // prüfen, ob Kategorie tats. noch existiert (wenn nicht: anlegen)
                if(!IPS_ObjectExists($Cat_ID)) $Cat_ID = IPS_CreateCategory();
                IPS_SetName($Cat_ID, $Object['ObjectName']);
                    IPS_SetHidden($Cat_ID, (bool)$Object['ObjectIsHidden']);
                    IPS_SetPosition($Cat_ID, (int)$Object['ObjectPosition']);
```

```
                IPS_SetInfo($Cat_ID, $Object['ObjectInfo']);
                IPS_SetIcon($Cat_ID, $Object['ObjectIcon']);
                if(isset($Cat_ID, $Object['newParentID'])) IPS_SetParent($Cat_ID, (int)$Object['newParentID']);
            if($Cat_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3023 Kategorie-Objekt ".$Object['ObjectName']."., ID #".$Cat_ID." existiert (Update e
                $MessageProkoll['Install']['OK'][] = "#3025 Kategorie-Objekt ".$Object['ObjectName']."., ID #".$Cat_ID." neu konfiguriert";
            }
            elseif((@$Object['isConfigTree'] == 0) || (@$Object['newObjectID'] == 0) || (!IPS_ObjectExists((int)$Object['newObjectID'])))
            {
                $Cat_ID        = IPS_CreateCategory();
                $Object['newObjectID']  = $Cat_ID;
                if($Cat_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3022 Kategorie-Objekt ".$Object['ObjectName']."., neuID #".$Cat_ID." angelegt";
                IPS_SetName($Cat_ID, $Object['ObjectName']);
                IPS_SetHidden($Cat_ID, (bool)$Object['ObjectIsHidden']);
                IPS_SetPosition($Cat_ID, (int)$Object['ObjectPosition']);
                IPS_SetInfo($Cat_ID, $Object['ObjectInfo']);
                IPS_SetIcon($Cat_ID, $Object['ObjectIcon']);
                if(isset($Cat_ID, $Object['newParentID'])) IPS_SetParent($Cat_ID, (int)$Object['newParentID']);
                $MessageProkoll['Install']['OK'][] = "#3025 Kategorie-Objekt ".$Object['ObjectName']."., ID #".$Cat_ID." neu konfiguriert";

                if($Cat_ID == 0)
                $MessageProkoll['Install']['Failure'][] = "#3021 Kategorie-Objekt ".$Object['ObjectName']."., , altID #".$Object['ObjectID']
            }
            else
            {
               $MessageProkoll['Install']['OK'][] = "#3024 Kategorie-Objekt ".$Object['ObjectName']." nicht neu konfiguriert (Objekt im Cor
            }
        return $Object;
    break;

    case 1:  // Instanzen
            if((@$Object['newObjectID'] > 0) && ($Object['isConfigTree'] == 0))
            {
            $Inst_ID        = (int)$Object['newObjectID'];
            // prüfen, ob Kategorie tats. noch existiert (wenn nicht: anlegen)
            if(!IPS_ObjectExists($Inst_ID)) $Inst_ID = IPS_CreateInstance($Object['ModuleID']);
            if($Inst_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3033 Kategorie-Objekt ".$Object['ObjectName']."., ID #".$Inst_ID." bereits vorhanden
            }
            elseif(($Object['isConfigTree'] == 0) || (@$Object['newObjectID'] == 0) || (!IPS_ObjectExists((int)$Object['newObjectID'])))
            {
                $Inst_ID        = IPS_CreateInstance($Object['ModuleID']);
                if($Inst_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3032 Instanz-Objekt ".$Object['ObjectName']."., neuID #".$Inst_ID." angelegt";
                if($Inst_ID == 0)
                $MessageProkoll['Install']['Failure'][] = "#3031 Instanz-Objekt ".$Object['ObjectName']."., , altID #".$Object['ObjectID']."
            }
            else
            {
               $MessageProkoll['Install']['OK'][] = "#3034 Instanz-Objekt ".$Object['ObjectName']." nicht neu konfiguriert (Objekt im Confi
            }

           IPS_SetName($Inst_ID, $Object['ObjectName']);
           if(isset($Inst_ID, $Object['newParentID'])) IPS_SetParent($Inst_ID, (int)$Object['newParentID']);
           IPS_SetHidden($Inst_ID, (bool)$Object['ObjectIsHidden']);
           IPS_SetPosition($Inst_ID, (int)$Object['ObjectPosition']);
            IPS_SetInfo($Inst_ID, $Object['ObjectInfo']);
            IPS_SetIcon($Inst_ID, $Object['ObjectIcon']);
           IPS_ApplyChanges($Inst_ID);
           $Object['newObjectID']  = $Inst_ID;

           if($Inst_ID > 0)
            $MessageProkoll['Install']['OK'][] = "#3035 Instanz-Objekt ".$Object['ObjectName']."., neuID #".$Inst_ID." neu konfiguriert";

        return $Object;
    break;

    case 2:  // Variablen, nur anlegen, wenn sie nicht mit einer Instanz verküpft sind ($Object['ObjectIsReadOnly'] == 0)
        if((int)$Object['ObjectIsReadOnly'] == 0)
       {
          $ah_ID = IPS_GetInstanceListByModuleID('{43192F0B-135B-4CE7-A0A7-1475603F3060}')[0]; // Archive Handler
            if((@$Object['newObjectID'] > 0) && ($Object['isConfigTree'] == 0))
            {
            $Var_ID        = (int)$Object['newObjectID'];
            // prüfen, ob Var tats. noch existiert (wenn nicht: anlegen)
            if(!IPS_ObjectExists($Var_ID)) $Var_ID = IPS_CreateVariable((int)$Object['ValueType']);
            if($Var_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3043 Variablen-Objekt ".$Object['ObjectName']."., ID #".$Var_ID." bereits vorhanden (
            }
            elseif((@$Object['isConfigTree'] == 0) || (@$Object['newObjectID'] == 0)|| (!IPS_ObjectExists((int)$Object['newObjectID'])))
            {
                $Var_ID          = IPS_CreateVariable((int)$Object['ValueType']);
                if($Var_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3042 Variablen-Objekt ".$Object['ObjectName']."., neuID #".$Var_ID." angelegt";
                if($Var_ID == 0)
                $MessageProkoll['Install']['Failure'][] = "#3041 Variablen-Objekt ".$Object['ObjectName']."., , altID #".$Object['ObjectID']
            }
            else
            {
               $MessageProkoll['Install']['OK'][] = "#3044 Variablen-Objekt ".$Object['ObjectName']." nicht neu konfiguriert (Objekt im Cor
            }
            //echo "Variablen-Konfiguration für Var ID $Var_ID startet:\n";
           IPS_SetName($Var_ID, $Object['ObjectName']);
           if(isset($Var_ID, $Object['newParentID'])) IPS_SetParent($Var_ID, (int)$Object['newParentID']);
           IPS_SetHidden($Var_ID, (bool)$Object['ObjectIsHidden']);
           IPS_SetPosition($Var_ID, (int)$Object['ObjectPosition']);
            IPS_SetInfo($Var_ID, $Object['ObjectInfo']);
            IPS_SetIcon($Var_ID, $Object['ObjectIcon']);
            //if($Object['VariableCustomAction'] > 0)
            //IPS_SetVariableCustomAction($Var_ID, $Object['VariableCustomAction']);
            IPS_SetVariableCustomProfile($Var_ID, $Object['VariableCustomProfile']);
            AC_SetLoggingStatus($ah_ID, $Var_ID, (bool)$Object['LoggingStatus']);
            AC_SetAggregationType($ah_ID, $Var_ID, (int)$Object['AggregationType']);
```

```
            // Variablenwert schreiben (nur bei Neuinstallation)
            if($Parameters['Update'] = 0)
            {
                switch ((int)$Object['ValueType'])
                {
                   //  Variablenwert je nach Variablentyp schreiben (0= Boolean, 1= Integer, 2= Float; 3= String)
                   case 0:
                      SetValue($Var_ID, (bool)$Object['ValueBoolean']);
                   break;

                   case 1:
                      SetValue($Var_ID, (int)$Object['ValueInteger']);
                   break;

                   case 2:
                      SetValue($Var_ID, (float)$Object['ValueFloat']);
                   break;

                    case 3:
                      SetValue($Var_ID, $Object['ValueString']);
                   break;
                }
            }

            // Ab IPS V 3.0:
            if(IPS_GetKernelVersion() > 2.9)
            {
                AC_SetGraphStatus($ah_ID, $Var_ID, (bool)$Object['GraphStatus']);
            }
        IPS_ApplyChanges($ah_ID);

           $Object['newObjectID']  = $Var_ID;
           if($Var_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3045 Variablen-Objekt ".$Object['ObjectName']."., neuID #".$Var_ID." neu konfiguriert

        }
        else
        {
            $MessageProkoll['Install']['User'][] = "#3044 Variablen-Objekt ".$Object['ObjectName']."., altID #".$Object['ObjectID']." nicht
        }
        return $Object;
    break;

    case 3: // Scripte
         // bei ScriptOverwrite = 0: Prüfen, ob manuell benanntes Scriptfile im System vorhanden (Scriptabbuch, wenn ja)
        if(($Parameters['Update'] == 0) && ($Parameters['ScrOverwrite'] == 0))
        {
                if(explode('.', $Object['ScriptFile'])[0] != $Object['ScriptID'])
                {
                $FileCheck      = file_exists(IPS_GetKernelDir()."scripts\\".$Object['ScriptFile']);

            if($FileCheck)    exit('#3058 Scriptabbruch: Script '.$Object['ScriptFile'].' bereits im Script-Ordner vorhanden,
                                                        bereits installierte Zielprojekt-Objekte müssen manuell gelöscht werden');
                }
        }

           $Scr_Name             = $Object['ObjectName'];
           $Scr_oldID             = @IPS_GetScriptIDByFile($Object['ScriptFile']);
           $ScrFunctionName    = preg_replace('/[^a-zA-Z]/', '',$Scr_Name).$Object['ObjectID'];
          $Scr_Content      = $ScrFunctionName();
          $Scr_Content      = stripslashes($ScrFunctionName());
          $Scr_Content      = str_replace('§\'', "'", $Scr_Content);
          $Scr_Content      = str_replace("@§@", "\\",$Scr_Content);

          // Vorabcheck Script exist
          if(isset($Object['newObjectID']) && (@$Object['newObjectID'] >0))
            {
                $ScrExist     = true;
                $Scr_ID        = (int)$Object['newObjectID'];
                if(!IPS_ObjectExists((int)$Object['newObjectID']))
                {
                    $ScrExist     = false;
                    $Scr_ID        = NULL;
              }
            }
            else
            {
                $ScrExist = false;
            }

          // Script-Neuanlage
           if(($ScrExist == false))
           {
               $Scr_ID                 = IPS_CreateScript(0);
               IPS_SetName($Scr_ID, $Scr_Name);

                if(isset($Scr_ID, $Object['newParentID'])) IPS_SetParent($Scr_ID, (int)$Object['newParentID']);
                $fh                                                 = fopen(IPS_GetKernelDir()."scripts\\".$Scr_ID.'.ips.php', 'w') or die ("ca
                fwrite($fh, $Scr_Content);
                fclose($fh);
                IPS_SetScriptFile($Scr_ID, $Scr_ID.'.ips.php');
                IPS_SetHidden($Scr_ID, (bool)$Object['ObjectIsHidden']);
               IPS_SetPosition($Scr_ID, (int)$Object['ObjectPosition']);
                IPS_SetInfo($Scr_ID, $Object['ObjectInfo']);
                IPS_SetIcon($Scr_ID, $Object['ObjectIcon']);
                $Scr_IPS_ID  = (int)explode('.',$Object['ScriptFile'])[0];
                $Object['newObjectID']  = $Scr_ID;

                //Scriptfile umbenenen, wenn anderer Name als IPS-ID vergeben
              $ScrFilePrefix = explode('.', $Object['ScriptFile'])[0];
              if($ScrFilePrefix != $Object['ObjectID'])
              {
                    rename($Scr_ID.".ips.php", $Object['ScriptFile']);
                     $result    = IPS_SetScriptFile($Scr_ID, $Object['ScriptFile']);
                     if($result)
```

```
                    $MessageProkoll['Install']['OK'][] = "#3053 Script-Objekt ".$Object['ObjectName']."", File in ".$Object['ScriptFile']."
                }
                else
                {
                if($Scr_ID > 0)
                    $MessageProkoll['Install']['OK'][] = "#3052 Script-Objekt ".$Object['ObjectName']."", neuID #".$Scr_ID." neu angelegt";
                }
            }
            elseif(($ScrExist == true) && ($Object['isConfigTree'] != 1))
            {
               // bei ScriptOverwrite = 0: Prüfen, ob Scriptfile im System vorhanden (Scriptabbuch, wenn ja)
            if(($Parameters['Update'] == 1) && ($Parameters['ScrOverwrite'] == 0))
            {
                if(explode('.', $Object['ScriptFile'])[0] != $Object['ScriptID'])
                    {
                    $FileCheck      = file_exists(IPS_GetKernelDir()."scripts\\".$Object['ScriptFile']);

                if($FileCheck)    exit('#3059 Scriptabbruch: Script '.$Object['ScriptFile'].' bereits im Script-Ordner vorhanden,
                                                        bereits installierte Zielprojekt-Objekte müssen manuell gelöscht werden');
                    }
            }

               // Update (nur, wenn ConfigKategorie-Flag nicht gesetzt)
              if($ScrExist)
              {
                 // Filename mit ID oder individuell benannt
                 $ScrFilePrefix = explode('.', $Object['ScriptFile'])[0];
                 if($ScrFilePrefix == $Object['ObjectID'])
                 {
                        $ScrFileName   = $Object['newObjectID'].'.ips.php';
                        $ScrOverwrite  = 1;
                    }
                    else
                    {
                       $ScrFileName   = $Object['ScriptFile'];
                       if($Parameters['ScrOverwrite'] == 1) {$ScrOverwrite  = 1;}else{$ScrOverwrite  = 0;}
                    }
                }
                else
                {
                   $ScrFileName   = $Object['ScriptFile'];
                   $ScrOverwrite  = 1;
                }

                // Script-File löschen (wenn  vorhanden und ScriptOverwrite = 1) und neu schreiben
                $scrReturn = 0;
                if($ScrExist && $ScrOverwrite)
                {
                  unlink(IPS_GetKernelDir()."scripts\\".$ScrFileName);
                    $fh                                         = fopen(IPS_GetKernelDir()."scripts\\".$ScrFileName, 'w') or die ("can'
                    $MessageProkoll['Install']['OK'][]    = "#3055 Script-File ".$Object['ObjectName']."", ".$ScrFileName.", gelöscht";
                    $scrReturn                                  = fwrite($fh, $Scr_Content);
                    fclose($fh);
               IPS_SetHidden($Scr_ID, (bool)$Object['ObjectIsHidden']);
                   IPS_SetPosition($Scr_ID, (int)$Object['ObjectPosition']);
                    IPS_SetInfo($Scr_ID, $Object['ObjectInfo']);
                    IPS_SetIcon($Scr_ID, $Object['ObjectIcon']);
                }

                // Meldungen für Inst-Protokoll
                if($scrReturn)
                {
                    $MessageProkoll['Install']['OK'][] = "#3051 Script-File ".$Object['ObjectName']."", neu geschrieben";
                }
                elseif($ScrOverwrite == 0)
                {
                   $MessageProkoll['Install']['OK'][] = "#3054 Script-File ".$Object['ObjectName'].', nicht überschrieben (Option $existsSc
                }
                else
                {
                    $MessageProkoll['Install']['Failure'][] = "#3055 Script-File ".$Object['ObjectName']."", neu schreiben fehlgeschlagen";
                }
            }
        return $Object;
    break;

    case 4: // Events
           $Evnt_Name             = $Object['ObjectName'];

          if((@$Object['newObjectID'] > 0) && ($Object['isConfigTree'] == 0))
            {
            $Evnt_ID        = (int)$Object['newObjectID'];
            // prüfen, ob Var tats. noch existiert (wenn nicht: anlegen)
            if(!IPS_ObjectExists($Evnt_ID)) $Evnt_ID = IPS_CreateEvent((int)$Object['EventType']);
            if($Evnt_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3063 Event-Objekt ".$Object['ObjectName']."", ID #".$Evnt_ID." bereits vorhanden (Upc
            }
            elseif((@$Object['isConfigTree'] == 0) || (@$Object['newObjectID'] == 0)|| (!IPS_ObjectExists((int)$Object['newObjectID'])))
            {
                $Evnt_ID = IPS_CreateEvent((int)$Object['EventType']);
                if($Evnt_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3062 Event-Objekt ".$Object['ObjectName']."", neuID #".$Evnt_ID." angelegt";
                if($Evnt_ID == 0)
                $MessageProkoll['Install']['Failure'][] = "#3061 Event-Objekt ".$Object['ObjectName']."", , altID #".$Object['ObjectID']." N
            }
            else
            {
               $MessageProkoll['Install']['OK'][] = "#3064 Event-Objekt ".$Object['ObjectName']." nicht neu konfiguriert (Objekt im Config-
            }
          if(isset($Evnt_ID, $Object['newParentID'])) IPS_SetParent($Evnt_ID, (int)$Object['newParentID']);
          IPS_SetName($Evnt_ID, $Evnt_Name);
          IPS_SetPosition($Evnt_ID, (int)$Object['ObjectPosition']);
          IPS_SetHidden($Evnt_ID, (bool)$Object['ObjectIsHidden']);
           IPS_SetPosition($Evnt_ID, (int)$Object['ObjectPosition']);
            IPS_SetInfo($Evnt_ID, $Object['ObjectInfo']);
```

```
            IPS_SetIcon($Evnt_ID, $Object['ObjectIcon']);
            IPS_SetEventLimit($Evnt_ID, (int)$Object['EventLimit']);

            // getriggertes Element
            if((int)$Object['EventType'] == 0)
            {
                //!!! (int)$Oject['TriggerVariableID'] Auslöser ID derzeit nur ParentObject, da newTarget noch nicht bekannt.
                // Konfiguration von IPS_SetEventTrigger wird erst mit Neuverlinkung abgeschlossen !!
               @IPS_SetEventTrigger($Evnt_ID, (int)$Object['TriggerType'], 0);
               if((int)$Object['TriggerType'] == 4)
                IPS_SetEventTriggerValue($Evnt_ID, $Object['TriggerValue']);
                IPS_SetEventTriggerSubsequentExecution($Evnt_ID, (bool)$Object['TriggerSubsequentExecution']);
                IPS_SetEventActive($Evnt_ID, true);  // Ereignis nach Installation immer aktiv
                $MessageProkoll['Install']['OK'][] = "#3065 Event-Objekt ".$Object['ObjectName']."', neuID #".$Evnt_ID." neu konfiguriert ur
            }
            else
            {
                // zyklisches Element
                IPS_SetEventCyclic($Evnt_ID, (int)$Object['CyclicDateType'], (int)$Object['CyclicDateValue'], (int)$Object['CyclicDateDay']

                // Prüfung IPS-Version (ab 3.1 wurden die SetEventCyclic-Befehle geändert)
                if($IPS_VERSION < 3.1)
                {
                   // Wenn Zielsystem-Version < 3.1 ist
                    IPS_SetEventCyclicDateBounds($Evnt_ID, (float)$Object['CyclicDateFrom'], (float)$Object['CyclicDateTo']);
                    //if((int)$Object['CyclicTimeType'] == 1)
                    IPS_SetEventCyclicTimeBounds($Evnt_ID, (float)$Object['CyclicTimeFrom'], (float)$Object['CyclicTimeTo']);
                }
                else
                {
                   // Wenn Zielsystem >= 3.1 ist
                   if($Object['CyclicDateFrom'] > 0)
                   {
                       IPS_SetEventCyclicDateFrom($Evnt_ID, (int)date("d", $Object['CyclicDateFrom']), (int)date("m", $Object['CyclicDateFr
                     }
                     else
                     {
                        IPS_SetEventCyclicDateFrom($Evnt_ID, 0, 0, 0);
                     }

                     if($Object['CyclicDateTo'] > 0)
                     {
                        IPS_SetEventCyclicDateTo($Evnt_ID, (int)date("d", $Object['CyclicDateTo']), (int)date("m", $Object['CyclicDateTo'])
                    }
                    else
                    {
                       IPS_SetEventCyclicDateTo($Evnt_ID, 0, 0, 0);
                    }
                    IPS_SetEventCyclicTimeFrom($Evnt_ID, (int)date("H", $Object['CyclicTimeFrom']), (int)date("i", $Object['CyclicTimeFrom'
                    IPS_SetEventCyclicTimeTo($Evnt_ID, (int)date("H", $Object['CyclicTimeTo']), (int)date("i", $Object['CyclicTimeTo']), (i
                }

                IPS_SetEventActive($Evnt_ID, false);  // Ereignis nach Installation immer inaktiv
                $MessageProkoll['Install']['User'][] = "#3066 Event-Objekt ".$Object['ObjectName']."', neuID #".$Evnt_ID." neu konfiguriert,
            }

            // PHP-Code einschleusen, wenn vorhanden
            $FuncName   = 'Event'.$Object['ObjectID'];
            if(function_exists($FuncName))
            {
                $FuncContent                    = $FuncName();
                IPS_SetEventScript($Evnt_ID, $FuncContent);
                $MessageProkoll['Install']['OK'][] = "#3067 Event-Objekt ".$Object['ObjectName']."', neuID #".$Evnt_ID." PHP-Code eingebaut"
            }
           $Object['newObjectID']  = $Evnt_ID;
     return $Object;
    break;

    case 5: // Media-Objekte
           $MessageProkoll['Install']['Failure'][] = "#3061 Media-Objekt ".$Object['ObjectName']."', altID #".$Object['ObjectID']." nicht ar
           return 0;

        //return $Object;
    break;

    case 6: // Links

          $Lnk_Name             = $Object['ObjectName'];
          if((@$Object['newObjectID'] > 0) && ($Object['isConfigTree'] == 0))
            {
            $Lnk_ID        = (int)$Object['newObjectID'];
           // prüfen, ob Lnk tats. noch existiert (wenn nicht: anlegen)
            if(!IPS_ObjectExists($Lnk_ID)) $Lnk_ID = IPS_CreateLink();
            if($Lnk_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3063 Link-Objekt ".$Object['ObjectName']."', ID #".$Lnk_ID." bereits vorhanden (Updat
            }
            elseif((@$Object['isConfigTree'] == 0) || (@$Object['newObjectID'] == 0)|| (!IPS_ObjectExists((int)$Object['newObjectID'])))
            {
                $Lnk_ID = IPS_CreateLink();
                if($Lnk_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3072 Link-Objekt ".$Object['ObjectName']."', neuID #".$Lnk_ID." angelegt";
                if($Lnk_ID == 0)
                $MessageProkoll['Install']['Failure'][] = "#3071 Link-Objekt ".$Object['ObjectName']."', , altID #".$Object['ObjectID']." Ne
            }
            else
            {
               $MessageProkoll['Install']['OK'][] = "#3074 Link-Objekt ".$Object['ObjectName']." nicht neu konfiguriert (Objekt im Config-E
            }
          if(isset($Object['newParentID'])) IPS_SetParent($Lnk_ID, (int)$Object['newParentID']);
          IPS_SetName($Lnk_ID, $Lnk_Name);
          IPS_SetPosition($Lnk_ID, (int)$Object['ObjectPosition']);
          IPS_SetHidden($Lnk_ID, (bool)$Object['ObjectIsHidden']);
           IPS_SetPosition($Lnk_ID, (int)$Object['ObjectPosition']);
            IPS_SetInfo($Lnk_ID, $Object['ObjectInfo']);
            IPS_SetIcon($Lnk_ID, $Object['ObjectIcon']);
```

```
                //IPS_SetLinkChildID($LinkID, 12345 /*[Objekt #12345 existiert nicht]*/);
                //IPS_SetLinkTargetID($LinkID, 12345 /*[Objekt #12345 existiert nicht]*/);

                  $MessageProkoll['Install']['OK'][] = "#3071 Link-Objekt ".$Object['ObjectName']."., neuID #".$Lnk_ID." neu konfiguriert";
              $Object['newObjectID']  = $Lnk_ID;
              return $Object;

          break;
      }
  }

  // nachträgliche Verlinkung aller neu installierten Objekte
  function linknewObjects($ObjLinkTree)
  {
     global $ObjInstallTree;
     global $newTgtID;
     global $newObjID;
     global $MessageProkoll;

     for($i=0;$i<count($ObjLinkTree);$i++)
     {
        $newObjID        = 0;
        $newTgtID        = 0;
        // Inventory nach Variablen-, Script-, Event- und Link-Objekten durchsuchen und neue Target-ID finden
        // 2= Variable, 3= Script, 4= Event, 6 = Link
        if(($ObjLinkTree[$i]['ObjectType'] == 2) && ($ObjLinkTree[$i]['VariableCustomAction'] >0))    // Scripts nach austauschbaren IDs durchsuc
          {
           // neue Target-ID (Action-Script) für Variable finden
              $newTargetID = findNewTargetID($ObjInstallTree, $ObjLinkTree[$i]['VariableCustomAction']);
           IPS_SetVariableCustomAction((int)$ObjLinkTree[$i]['newObjectID'], (int)$newTargetID);
           if($newTargetID >0)
              $MessageProkoll['Link']['OK'][] = "#4011 Variablen-Objekt ".$ObjLinkTree[$i]['ObjectName']."., ID #".$ObjLinkTree[$i]['newObjectID']
           if($newTargetID == 0)
              $MessageProkoll['Link']['Failure'][] = "#4010 Variablen-Objekt ".$ObjLinkTree[$i]['ObjectName']."., ID #".$ObjLinkTree[$i]['newObjec

         }
        elseif($ObjLinkTree[$i]['ObjectType'] == 3)     // Scripts nach austauschbaren IDs durchsuchen
          {
           // aktuellen Scriptnamen finden
           $newObjID = 0;
              $Scr  = IPS_GetScript((int)$ObjLinkTree[$i]['newObjectID'])['ScriptFile'];
           changeObjectIDs_inScript($ObjInstallTree, $Scr);
              $MessageProkoll['Link']['OK'][] = "#4021 Script-Objekt ".$ObjLinkTree[$i]['ObjectName']."., ID #".$ObjLinkTree[$i]['newObjectID']."
         }
        elseif(($ObjLinkTree[$i]['ObjectType'] == 4) && ($ObjLinkTree[$i]['EventType'] == 0))    // Neuverlinkung nur für nichtzyklische Events
          {
             $newTgtID   =0;
              $newTargetID = findNewTargetID($ObjInstallTree, $ObjLinkTree[$i]['TriggerVariableID']);

              // neue Verlinkung setzen
              if($newTgtID !=0)
              {
                  IPS_SetEventTrigger((int)$ObjLinkTree[$i]['newObjectID'], (int)$ObjLinkTree[$i]['TriggerType'], $newTgtID);
                  $MessageProkoll['Link']['OK'][] = "#4031 Event-Objekt ".$ObjLinkTree[$i]['ObjectName']."., ID #".$ObjLinkTree[$i]['newObjectID']
              }
              else
              {
                  $MessageProkoll['Link']['User'][] = "#4030 Event-Objekt ".$ObjLinkTree[$i]['ObjectName']."., neue ID #".$ObjLinkTree[$i]['newObje
              }
         }
        elseif($ObjLinkTree[$i]['ObjectType'] == 6)
          {
              $newTgtID   =0;
              $newTgtID = findNewTargetID($ObjInstallTree, $ObjLinkTree[$i]['LinkChildID']);
              if($newTgtID !=0)
              {
                  IPS_SetLinkTargetID((int)$ObjLinkTree[$i]['newObjectID'], (int)$newTgtID);
                  $MessageProkoll['Link']['OK'][] = "#4041 Link-Objekt ".$ObjLinkTree[$i]['ObjectName']."., ID #".$ObjLinkTree[$i]['newObjectID'].
              }
              else
              {
                  $MessageProkoll['Link']['User'][] = "#4040 Link-Objekt ".$ObjLinkTree[$i]['ObjectName']."., ID #".$ObjLinkTree[$i]['newObjectID'
              }
         }

         // Objekt-Childs durchsuchen
         if(isset($ObjLinkTree[$i]['ChildrenIDs']))
         {
            $newLnkTree = $ObjLinkTree[$i]['ChildrenIDs'];
           linknewObjects($newLnkTree);
         }
     }
     unset($ObjLinkTree);
  }

  // sucht neue Target-ID für verlinkte Objekte
  function findNewTargetID($ObjLinkTree, $oldTargetID)
  {
     global $newTgtID;
     for($i=0;$i<count($ObjLinkTree);$i++)
     {
        if($ObjLinkTree[$i]['ObjectID'] == $oldTargetID)
        {
            $newTgtID   = $ObjLinkTree[$i]['newObjectID'];
              $i = 9999;  // Abbruch, wenn ID gefunden
        }
          // in Childs suchen (rekursiv)
        if((isset($ObjLinkTree[$i]['ChildrenIDs'])) && ($newTgtID == 0))
        {
              $newObjInstTree   = $ObjLinkTree[$i]['ChildrenIDs'];
           findNewTargetID($newObjInstTree, $oldTargetID);
           unset($newObjInstTree);
        }
     }
```

```
   return $newTgtID;
}

// Austausch der Objekt-IDs im Script
function changeObjectIDs_inScript($ObjLinkTree, $Scr)
{
   global $newObjID;
   global $MessageProkoll;

   $ScrContent = file(IPS_GetKernelDir()."scripts\\".$Scr);
    $file         = fopen(IPS_GetKernelDir()."scripts\\".$Scr, "w");
    $search         = '/[^0-9\/\{-]([1-5][0-9]{4})[^0-9]/'; // suche 5stellige Zahlen von 10.000 - 59.999
    $counter    = 0;

    foreach ($ScrContent as $z)
    {
       $newObjID = NULL;
       $newObjectID = NULL;
        preg_match($search, $z, $matches);
        if(isset($matches[1]))
        {
           $oldID           = $matches[1];
            $newObjectID   = findNewObjectID($ObjLinkTree, $oldID);
            if(($matches[1] == $oldID) && ($newObjectID >0))
            {
               $z = str_replace($oldID, $newObjectID, $z);
               $MessageProkoll['Script']['OK'][] = "#5011 Script ID#$Scr: Inhalt verändert: Zeile ".($counter + 1).", alte Target-ID #$oldID ge
            }
            else
            {
               $MessageProkoll['Script']['User'][] = "#5010 Script ID#$Scr: Inhalt NICHT verändert: Zeile ".($counter + 1).", alte Target-ID #$
            }
        }
        fwrite($file, $z);
        $counter ++;
    }
    unset($ScrContent);
    unset($ObjLinkTree);
    fclose($file);
    return;
}

// sucht neue Object-ID für Script-Variablen
function findNewObjectID($ObjTree, $oldObjID)
{
   global $newObjID;
   for($i=0;$i<count($ObjTree);$i++)
   {
      if($ObjTree[$i]['ObjectID'] == $oldObjID)
      {
          $newObjID   = $ObjTree[$i]['newObjectID'];
            $i = 9999;  // Abbruch, wenn ID gefunden
      }

       // in Childs suchen (rekursiv)
       if(isset($ObjTree[$i]['ChildrenIDs']))
       {
            $newObjTree   = $ObjTree[$i]['ChildrenIDs'];
          findNewObjectID($newObjTree, $oldObjID);
          unset($newObjTree);
       }
   }
   unset($ObjTree);
   return $newObjID;
}


############################## WFE-Items installieren ##########################################################

function installWFC_Items($WFC_TargetID)
{
    global $ObjInstallTree;
    global $newObjID;
    global $WFC_existItemoverwrite;
    global $MessageProkoll;
    $MessageProkoll['WFC']['OK'][] = "#6001 lade WFC-Objektbaum";

    // Objektbaum Ziel-ID (Root-ID des Projekts)
    $ZielID                 = $ObjInstallTree[0]['newObjectID'];

    // Inventory des Ziel-WFC laden
    if(@WFC_GetItems($WFC_TargetID))
    $WFC_Target_Inventory   = WFC_GetItems($WFC_TargetID);

    // Inventory des Quell-Projektes laden
    $WFC_Raw        = loadWFCInventory();

    if(isset($WFC_Target_Inventory[0]) && ($WFC_Raw[0] != NULL))
    {
        // Quell-Inventory durchgehen und Items installieren
        $MessageProkoll['WFC']['OK'][] = "#6002 durchsuche QuellWFC-Objektbaum";
        $Cnt          = count($WFC_Raw);

        for($i=0;$i<$Cnt;$i++)
        {
            // Inventory-String letzte 2 Zeichen abschneiden
            $str  = substr($WFC_Raw[$i], 0, -2);

            // Item-Zeilen in Key|Value-Sätze zerlegen (1.Stufe)
            $WFC_DataSet = explode(';', $str);

            // Key|Value-Sätze in Key und Value zerlegen (2. Stufe)
            for($e=0;$e<count($WFC_DataSet);$e++)
            {
               $WFC_Values[$e]     = explode('|', $WFC_DataSet[$e]);
```

```
                    }

                // Suche für alte TargetIDs neue Target-ID und tausche diese aus (nur Objekte mir Target-ID im Objektbaum)
                   $newTargetID           = NULL;
                   unset                        ($matches);
                $search                   = '/[1-5]\d\d\d\d/'; // suche 5stellige Zahlen von 10.000 - 59.999
                if($WFC_Values[1][1] == 'Category' || $WFC_Values[1][1] == 'InfoWidget' || $WFC_Values[1][1] == 'Graph' || $WFC_Values[1][1] == 'Cc
                    {
                    preg_match($search, $WFC_Values[2][1], $matches);
                        if(isset($matches[0])) $oldTargetID   = $matches[0];
                        if(isset($matches[0])) $newTargetID      = findNewObjectID($ObjInstallTree, $matches[0]);
                        if($newTargetID != NULL)
                        {
                            $WFC_Values[2][1]     = str_replace($oldTargetID, $newTargetID, $WFC_Values[2][1]);
                            $MessageProkoll['WFC']['OK'][] = "#6011 WFC altTargetID #$oldTargetID gegen newTargetID #$newTargetID getauscht";
                        }
                        else
                        {
                           $MessageProkoll['WFC']['User'][] = "#6012 WFC Item ".$WFC_Values[1][1].", ".$WFC_Values[3][1]." keine neue TargetID gefu
                        }
                    }
                    unset($newObjID);

                // Namen für neues WFC-Item ändern (Präfix "Cpy" hinzufügen)
                    if(strpos($WFC_Values[2][1], 'roottp') == 0)
                    {
                       $StartPos         = strpos($WFC_Values[2][1], '"name":')+8;
                       $EndPos           = strpos($WFC_Values[2][1], '","', $StartPos);
                       $OldItemName      = substr($WFC_Values[2][1], $StartPos, $EndPos - $StartPos);
                       $NewItemName      = substr($WFC_Values[2][1], $StartPos, $EndPos - $StartPos).'_ScrID'.$ZielID;
                        $WFC_Values[2][1] = str_replace($OldItemName, $NewItemName, $WFC_Values[2][1]);
                    }

                // Namen für ID (Item) ändern
                    if($WFC_Values[3][1] != 'roottp')
                    $WFC_Values[3][1] = str_replace($WFC_Values[3][1], $WFC_Values[3][1].'_ScrID'.$ZielID, $WFC_Values[3][1]);

                // Namen für ParentID (Item) ändern
                    if($WFC_Values[4][1] != 'roottp')
                    $WFC_Values[4][1] = str_replace($WFC_Values[4][1], $WFC_Values[4][1].'_ScrID'.$ZielID, $WFC_Values[4][1]);

                // Root-Item aus Quell-Projekt definieren (wird bei Updates nicht gelöscht)
                    if($i == 0) $WFC_RootItem  = $WFC_Values[3][1];

                // Ziel-Inventory nach vorhandenen Elementen durchsuchen
                    $Item_exist = 0;
                    foreach($WFC_Target_Inventory as $value)
                    {
                       // wenn WFE-Objekt gefunden (aktuelle Syntax 'ItemName_ScrID12345' bzw. alte Syntax 'CpyItemName')
                        if(($value['ID'] == $WFC_Values[3][1]) || ($value['ID'] == 'Cpy'.substr($WFC_Values[3][1], 0, -11)))
                        {
                           // wenn Überschreib-Option abgeschaltet und WFE Objekt nicht alter Syntax ('CpyItemName') entspricht
                            if(($WFC_existItemoverwrite == 0) && ($value['ID'] != 'Cpy'.substr($WFC_Values[3][1], 0, -11)))
                            {
                                $Item_exist = 1;
                                $MessageProkoll['WFC']['User'][] = "#6020 WFC-Item ".$WFC_Values[3][1]." nicht gelöscht (Option abgeschaltet)";
                                break;
                            }
                            elseif(($WFC_Values[3][1] != $WFC_RootItem) && ($value['ID'] != 'Cpy'.substr($WFC_Values[3][1], 0, -11)))
                            {
                               WFC_DeleteItem($WFC_TargetID, $WFC_Values[3][1]); // vorhandenes Item (neue Syntax) löschen
                               $MessageProkoll['WFC']['OK'][] = "#6021 WFC-Item ".$WFC_Values[3][1]." gelöscht (Option eingeschaltet)";
                            }
                            elseif($value['ID'] == 'Cpy'.substr($WFC_Values[3][1], 0, -11))
                            {
                               WFC_DeleteItem($WFC_TargetID, 'Cpy'.substr($WFC_Values[3][1], 0, -11)); // vorhandenes Item (alte Syntax) löschen
                               $MessageProkoll['WFC']['OK'][] = "#6022 WFC-Item ".$WFC_Values[3][1]." veraltetes WFE-Objekt ('CpyItemname') gelösch
                            }

                            if($WFC_Values[3][1] == $WFC_RootItem)
                            {
                               // wenn Root-Item veraltete Syntax hat
                               $Item_exist = 1;
                            }
                        }
                    }

                // Wenn Item im Ziel-WFC nicht vorhanden, dann installieren
                    if($Item_exist == 0)
                    {
                        if($WFC_Values[3][1] != 'roottp')
                        WFC_AddItem($WFC_TargetID, $WFC_Values[3][1], $WFC_Values[1][1], $WFC_Values[2][1], $WFC_Values[4][1]);
                        WFC_UpdatePosition($WFC_TargetID, $WFC_Values[3][1], (int)$WFC_Values[5][1]);
                        WFC_UpdateVisibility($WFC_TargetID, $WFC_Values[3][1], (bool)$WFC_Values[6][1]);
                        $MessageProkoll['WFC']['OK'][] = "#6031 WFC-Item ".$WFC_Values[3][1]." installiert";
                    }
                    else
                    {
                       $MessageProkoll['WFC']['OK'][] = "#6030 WFC-Item ".$WFC_Values[3][1]." nicht installiert, da bereits vorhanden";
                    }

                unset($WFC_Values);
                unset($WFC_DataSet);
            }
            IPS_ApplyChanges($WFC_TargetID);
        }
        else
        {
           $MessageProkoll['WFC']['User'][] = "#6000 WFCObjektbaum: keine Installation möglich (kein Quell-Inventory oder kein Ziel-WFE gefunden)";
        }
    }


    // Install-Script ins Projekt-Config Verzeichnis verschieben
    function moveInstallScripttoConfig()
```

```
{
    global $ObjInstallTree;
    global $MessageProkoll;

    $RootID     = (int)$ObjInstallTree[0]['newObjectID'];
    $search     = 'Config';
    $SrchResult = searchConfigDir($ObjInstallTree);
    if($SrchResult > 0)
    {
        IPS_SetParent($_IPS['SELF'], $SrchResult);
        $MessageProkoll['Info'][] = "#1011 Install-Script: Install-Script und -Protokoll in Config-Ordner verschoben";
    }
    else
    {
        $MessageProkoll['Info'][] = "#1010 Install-Script: Install-Script und -Protokoll NICHT verschoben (kein Config-Ordner im Zielprojekt ge
    }
    return;
}

// Funktion zum rekursiven Durchsuchen des Obj-Inventorys nach Config-Dir
function searchConfigDir($Array)
{
    global $SrchResult;
    $search    = 'Config';
   $Cnt     = count($Array);
   for($i=0;$i<$Cnt;$i++)
   {
      if($Array[$i]['ObjectName'] == $search)
      {
         $SrchResult    = (int)$Array[$i]['newObjectID'];
        }

        if((isset($Array[0]['ChildrenIDs'])) && (!isset($SrchResult)))
        {
          $newArray = $Array[0]['ChildrenIDs'];
          searchConfigDir($newArray);
        }
   }
    unset($Array);
    unset($newArray);
    return $SrchResult;
}

// Protokoll ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
function InstallProtocoll()
{
global $MessageProkoll;

// +++ Installations-Parameter +++++++++++++++++++++++++++
    $ps   ='// +++ Installationsprotokoll vom '.date("d.m.Y, H:i:s", time()).' ++++++++++++++++++++++++++++++++++++++++++++++'."\r\n\r\n";
    $ps   .="/** +++++++++++++++++ Installationsparameter ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\r\n";
    $ps   .="    Update:                                ".$MessageProkoll['InstallParameters']['Update']."\r\n";
    $ps   .="    Variablenprofile installieren:     ".$MessageProkoll['InstallParameters']['VarProfile']."\r\n";
    $ps   .="    benamste Scripte überschreiben:    ".$MessageProkoll['InstallParameters']['ScrOverwrite']."\r\n";
    $ps   .="    WFC-Items überschreiben:            ".$MessageProkoll['InstallParameters']['WFCItems']."\r\n";
    $ps   .="++++++++++++++++++++ Installationsparameter Ende ++++++++++++++++++++++++++++++++++++++++++++++++++++++\r\n\r\n\r\n";

// +++ Infos +++++++++++++++++++++++++++++++++++++
    $ps   .="++++++++++++++++++++ Installations-Infos ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\r\n";
    for($i=0;$i<count(@$MessageProkoll['Info']);$i++)
    {
        $ps   .="   ".@$MessageProkoll['Info'][$i]."\r\n";
    }
    $ps   .="++++++++++++++++++++ Installations-Infos Ende ++++++++++++++++++++++++++++++++++++++++++++++++++++++\r\n\r\n\r\n";

// +++ Installation ++++++++++++++++++++++++++++++++++++
    $ps   .="++++++++++++++++++++ Objekt-Installation ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\r\n";
    $ps   .="        --- OK-Meldungen ----------------------------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['Install']['OK']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['Install']['OK'][$i]."\r\n";
    }
    $ps   .="        --- OK-Meldungen Ende ----------------------------------------------------------------------\r\n\r\n\r\n";

    $ps   .="        --- Kontrolle durch User erforderlich ----------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['Install']['User']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['Install']['User'][$i]."\r\n";
    }
    $ps   .="        --- Kontrolle durch User erforderlich Ende----------------------------------------------\r\n\r\n\r\n";

    $ps   .="        --- Fehlermeldungen ----------------------------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['Install']['Failure']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['Install']['Failure'][$i]."\r\n";
    }
    $ps   .="        --- Fehlermeldungen Ende----------------------------------------------------------------------\r\n\r\n\r\n";

// +++ WFC-Objekte installieren und verlinken ++++++++++++++++++++++++++++++++++++
    $ps   .="    +++++++++++++++++ WFC-Objekte installieren und verlinken ++++++++++++++++++++++++++++++++++++++++\r\n";
    $ps   .="        --- OK-Meldungen ----------------------------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['WFC']['OK']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['WFC']['OK'][$i]."\r\n";
    }
    $ps   .="        --- OK-Meldungen Ende ----------------------------------------------------------------------\r\n\r\n\r\n";

    $ps   .="        --- Kontrolle durch User erforderlich ----------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['WFC']['User']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['WFC']['User'][$i]."\r\n";
    }
    $ps   .="        --- Kontrolle durch User erforderlich Ende----------------------------------------------\r\n\r\n\r\n";

    $ps   .="        --- Fehlermeldungen ----------------------------------------------------------------------\r\n";
```

```
    for($i=0;$i<count(@$MessageProkoll['WFC']['Failure']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['WFC']['Failure'][$i]."\r\n";
    }
    $ps   .="       --- Fehlermeldungen Ende----------------------------------------------------------------------\r\n\r\n";

// +++ Scripte nachbearbeiten ++++++++++++++++++++++++++++++++++++
    $ps   .="    ++++++++++++++++++ ID Austausch in Scripts ++++++++++++++++++++++++++++++++++++++++++++++++++++\r\n";
    $ps   .="       --- OK-Meldungen -----------------------------------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['Script']['OK']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['Script']['OK'][$i]."\r\n";
    }
    $ps   .="       --- OK-Meldungen Ende ---------------------------------------------------------------------\r\n\r\n";

    $ps   .="       --- Kontrolle durch User erforderlich -----------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['Script']['User']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['Script']['User'][$i]."\r\n";
    }
    $ps   .="       --- Kontrolle durch User erforderlich Ende------------------------------------------------\r\n\r\n";

    $ps   .="       --- Fehlermeldungen -----------------------------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['Script']['Failure']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['Script']['Failure'][$i]."\r\n";
    }
    $ps   .="       --- Fehlermeldungen Ende------------------------------------------------------------------\r\n\r\n";

// +++ Objekte neu verlinken ++++++++++++++++++++++++++++++++++++
    $ps   .="    ++++++++++++++++++ Objekte neu verlinken ++++++++++++++++++++++++++++++++++++++++++++++++++++\r\n";
    $ps   .="       --- OK-Meldungen -----------------------------------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['Link']['OK']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['Link']['OK'][$i]."\r\n";
    }
    $ps   .="       --- OK-Meldungen Ende ---------------------------------------------------------------------\r\n\r\n";

    $ps   .="       --- Kontrolle durch User erforderlich -----------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['Link']['User']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['Link']['User'][$i]."\r\n";
    }
    $ps   .="       --- Kontrolle durch User erforderlich Ende------------------------------------------------\r\n\r\n";

    $ps   .="       --- Fehlermeldungen -----------------------------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['Link']['Failure']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['Link']['Failure'][$i]."\r\n";
    }
    $ps   .="       --- Fehlermeldungen Ende------------------------------------------------------------------\r\n\r\n";

    // +++ externe Files installiert ++++++++++++++++++++++++++++++++++++
    $ps   .="    ++++++++++++++++++ Installation externe Files ++++++++++++++++++++++++++++++++++++++++++++++++\r\n";
    $ps   .="       --- OK-Meldungen -----------------------------------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['extFiles']['OK']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['extFiles']['OK'][$i]."\r\n";
    }
    $ps   .="       --- OK-Meldungen Ende ---------------------------------------------------------------------\r\n\r\n";

    $ps   .="       --- Fehlermeldungen -----------------------------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['extFiles']['Failure']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['extFiles']['Failure'][$i]."\r\n";
    }
    $ps   .="       --- Fehlermeldungen Ende------------------------------------------------------------------\r\n\r\n";

    $ps   .="**/\r\n";

echo $ps;
return $ps;
}


############################## Installations-Protokoll ##################################################
// +++ erstellt Installations-Protokoll Objekt-Inventory für Exportscript +++++++++++++++
function createObjInventoryProtocol($ObjInstInv)
{
    // Object-Inventory Function-Kopf generieren
    $s = "<?\r\n";
    $s .= InstallProtocoll();
    $s .= "// ### InstallationsProtokoll Ende ################################################\r\n\r\n\r\n\r\n";
    $s .= '// Function Object Inventory Install-Protocol'."\r\n";
    $s .= 'function ObjectInventoryProtocol()'."\r\n";
    $s .= '{'."\r\n";
    $s .= '$raw = \''."\r\n";

    // Array Copy-Parameters
    $s .= createCopyParametersString();

    // Daten der Einzelobjekte aus Objekt-Inventory an Export-String anfügen
    $ObjStringArray = ChildGenerator($ObjInstInv);
    //print_r($ObjInstInv);
    $s .= createInventoryString($ObjStringArray);

    // Function Footer
    $s .= "';\r\n";
    $s .= '$Inv    = explode("\n",substr(substr($raw, 1), 1, -1)); // ersten und letzten Zeilenumbruch entfernen'."\r\n";
    $s .= 'return $Inv;'."\r\n";
    $s .= "}\r\n\r\n";
    $s .= "?>";
    return $s;
}
```

```
// generiert rekursiv Protkoll-String für ggf. vorhandene Child-Arrays unter einem Objekt im Objekt-Inventory
function ChildGenerator($Array)
{
    for($i=0;$i<count($Array);$i++)
   {
      $obj[] = createObjectRow($Array[$i]);
        if((isset($Array[$i]['ChildrenIDs'])))
       {
          $newArray = $Array[$i]['ChildrenIDs'];
            $obj[]    = ChildGenerator($newArray);
        }
   }
   return $obj;
}

// Function Objekt Install Protocol: Datenstring für 1 Objekt generieren
function createObjectRow($Obj)
{
   // Objektstammdaten aus IPS_GetObject
   unset($Obj['ChildrenIDs']);

   // Steuerzeichen CR LF aus Object-Info filtern und ersetzen
    $Obj['ObjectInfo'] = str_replace("\r\n", "§&§", $Obj['ObjectInfo']);

    // Stringstart definieren, 1. Entrag muss ObjectID sein!
   $s   = 'ObjectID,'.$Obj['ObjectID'].';';

   // Inventory Export: Stringteil Objekt-Stammdaten vervollständigen
    for($i=0;$i<count($Obj);$i++)
    {
       $key        = key($Obj);
        $s .= key($Obj).','.$Obj[key($Obj)].';';
        next ($Obj);
    }
    $s .= "\r\n";

    return $s;
}

// +++ Export: geht Objekt-Inventory rekrusiv durch und fügt Objektdaten der Einzelobjekte an Protokoll-String an
function createInventoryString($Array)
{
   $str = '';
    for($i=0;$i<count($Array);$i++)
    {
       if(is_array($Array[$i]))
        {

           $newArray = $Array[$i];
           $str  .= createInventoryString($newArray);

        }
        else
        {
           $str .= $Array[$i];
       }
    }
    unset($Array);

    return $str;
}

// Copy-Parameters String für Install-Protocol generieren
function createCopyParametersString()
{
    global $Parameters;
    //$Array   = $_IPS['InstParameters'];
    $s = 'CopyParameters,;';
    for($i=0;$i<count($Parameters);$i++)
    {
       $s .=key($Parameters).','.current($Parameters).';';
       next($Parameters);
    }
    $s .= "\r\n";

    return $s;
}

// Protokoll-File schreiben
function CreateProtocolFile($InstProtocol)
{
    $ScrName    = IPS_GetName($_IPS['SELF'])." InstallProtocol";
    echo "Protokollname: ".$ScrName;
    $Script_ID = IPS_CreateScript(0);
    IPS_SetName($Script_ID, $ScrName);
    IPS_SetParent($Script_ID, $_IPS['SELF']);
    IPS_SetInfo($Script_ID, "This script was created by: RS IPS Exporter ID #".$_IPS['SELF']);
    $fh = fopen(IPS_GetKernelDir()."scripts\\".$Script_ID.".ips.php", 'w') or die("cant open file");
    fwrite($fh, $InstProtocol);
    fclose($fh);
    IPS_SetScriptFile($Script_ID, $Script_ID.'.ips.php');
    echo "Installationsprotokoll-Script \"$ScrName\" ID #$Script_ID angelegt\n";
    return $Script_ID;
}

################ Inventory-Daten ################################################################################

// Function Object Inventory
function ObjectInventory()
{
global $MessageProkoll;
$raw = '
CopyParameters,;VarProfile,1;ScrOverwrite,0;
ObjectID,53482 /*[Viessmann Heizung Serial Port]*/;HasChildren,1;ObjectIcon,;ObjectID,53482 /*[Viessmann Heizung Serial Port]*/;ObjectIdent,;Ob
ObjectID,56186 /*[Viessmann Heizung Serial Port\Viessmann Register Variable]*/;HasChildren,1;ObjectIcon,;ObjectID,56186 /*[Viessmann Heizung Se
```

```
ObjectID,35796 /*[Viessmann Heizung Serial Port\Viessmann Register Variable\ErrorCount]*/;HasChildren,;ObjectIcon,;ObjectID,35796 /*[Viessmann
ObjectID,42628 /*[Viessmann Heizung Serial Port\Viessmann Register Variable\Rückgabewert]*/;HasChildren,;ObjectIcon,;ObjectID,42628 /*[Viessmar
ObjectID,49172 /*[Viessmann Heizung Serial Port\Viessmann Register Variable\Verbindung ok]*/;HasChildren,;ObjectIcon,;ObjectID,49172 /*[Viessma
ObjectID,57385 /*[Viessmann Heizung Serial Port\Viessmann Register Variable\Steuerungstyp]*/;HasChildren,;ObjectIcon,;ObjectID,57385 /*[Viessma
ObjectID,58616 /*[Viessmann Heizung Serial Port\Viessmann Register Variable\Kommando]*/;HasChildren,;ObjectIcon,;ObjectID,58616 /*[Viessmann He
';
$echo = "#1001 Quell-ObjektInventory geladen";
$MessageProkoll['Info'][]  = $echo;
$Inv    = explode("\n",substr(substr($raw, 1), 1, -1)); // ersten und letzten Zeilenumbruch entfernen
return $Inv;
}

// Variablen-Profile Inventory
function VarProfileInventory()
{
global $MessageProkoll;
$raw = '
ProfileName,~String;Digits,0;Icon,;IsReadOnly,1;MaxValue,0;MinValue,0;Prefix,;ProfileType,3;StepSize,0;Suffix,;|
';
$echo = "#1002 VariablenProfile-ObjektInventory geladen";
$MessageProkoll['Info'][]  = $echo;
$VarProfiles    = explode("\n",substr(substr($raw, 2), 0, -1)); // ersten und letzten Zeilenumbruch entfernen
return $VarProfiles;
}

// Function WFC Inventory
function loadWFCInventory()
{
global $MessageProkoll;
$raw = '
';
$echo = "#1003 WFC-ObjektInventory geladen";
$MessageProkoll['Info'][]  = $echo;
$WFC_Inventory    = explode("\n",substr(substr($raw, 2), 0, -1)); // ersten und letzten Zeilenumbruch entfernen
return $WFC_Inventory;
}

// #### Script-Exports ##################################################################################################################

?>
```

Die Heizungssteuerung

PHP-Code:

```
<?
/*****************************************************************************************************************
*  this Install-Script was automated generated by RS IPS Project-Exporter                                        *
*  © by Raketenschnecke 2012-2014, mail: raketenschnecke@gmx.de                                                  *
*  // Projekt-HomePage: http://www.raketenschnecke.net/rs-projekte/rs-ips-project-exporter/                       *
*****************************************************************************************************************/

/*****************************************************************************************************************
    Project: Heizung (Quell-ID: 24978), generated on 30.07.2014, 12:55 Uhr
*                                                                                                                 *
*    Dieses Script beinhaltet ein automatisch in einem Script zusammengeführtes IPS-Projekt (Quell-Projekt)        *
*    Zur Installation des (Quell-) Projekts auf dem lokalen System (Zielsystem) bitte dieses Script an einer beliebigen *
*    Stelle im Objektbaum platzieren und einmalig manuell starten (Konfiguration siehe Block "Konfig" unten)      *
*  Es werden alle notwendigen Strukturen (Kategorien, Variablen, Scripte etc) aus dem Quell-Projekt im Ziel-System    *
*  angelegt.                                                                                                        *
*    Dies ist !!! KEINE !!! vollständige Projekt-Installation! D.h.: eine Konfiguration, Verlinkung von Objekten etc.   *
*    muss durch Scripts des eigentlichen Projekts oder manuell durch den Anwender erfolgen.                       *
*  Bei Fragen zum installierten Projekt bitte den Projektautor fragen!                                                *
*                                                                                                                 *
*****************************************************************************************************************/

##############################  Konfig ###########################################################################

// Ziel-WFC angeben
   $WFC_TargetID                = 0;     // bei falscher WFC-ID wird keine Installation von WFE-Komponenten vorgenommen
   $WFC_existItemoverwrite    = 1;  // 0=> bereits im Zielsystem bestehende WFC-Items werden NICHT überschrieben,
                                                // 1=> existierende Objekte werden überschrieben (default)

// Copy-Parameter (1: Objekte werden im Zielsystem installiert; 0: Objekte werden nicht installiert) +++++++++++++++++
    $VarProfile                 = 1;    // 1: überschreibt vorhandene Profile im Zielsystem, 0: installiert nur neue Profile
    # !!! die folgende Option mit äußerster Vorsicht nutzen (Default-Wert 0)!!! #
    $existsScriptsoverwrite    = 0;    // 1: im Zielsystem existierende Scripte (=benamste Scriptfiles) werden überschrieben


############################### Main Area ###########################################################################

   ##### Project Exporter Comment: Projekt-Export Heizung (Quell-ID: 24978) vom 30.07.2014 12:55 #####

// IPS Version detektieren
   $IPS_VERSION        = IPS_GetKernelVersion();

// Inventorys laden
    $MessageProkoll   = array();
    $Inv                    = ObjectInventory();
    $VProfInv             = VarProfileInventory();
    if(function_exists('loadFileExportInventory'))
    {
        $ExpFileInv        = loadFileExportInventory();
        if($ExpFileInv[0] != NULL)
        createExportFiles($ExpFileInv);
    }

// Parameters-Array erzeugen
   $Parameters    = array('VarProfile'=>$VarProfile, 'ScrOverwrite'=>$existsScriptsoverwrite, 'WFCItems'=>$WFC_existItemoverwrite,  'Update'=>0

// letztes Logfile finden & Updatemodus setzen
    $lastLog_ID    = findlastLogfile();
    if($lastLog_ID > 0)
```

```
        {
            $Parameters['Update'] = 1;
            include $lastLog_ID.'.ips.php';

            // last Inventory laden
            $lastInv     = ObjectInventoryProtocol();

            // Meldungsausgabe
            if(isset($lastInv[0]))
            {
                $MessageProkoll['Info'][] = "#1004 letztes Inventory aus Protokollfile geladen";
            }

            // Last Inventory initialisieren
            $InvData            = explode(';', substr(preg_replace("/\r|\n/s", "", $lastInv[1]),0, -1));
           $InvObjID          = (int)explode(',', $InvData[0])[1];
           $lastInvObj        = createlastInventoryObjects($InvObjID);

            // Meldungsausgabe
            if(IPS_ObjectExists((int)$lastInvObj[0]['newObjectID']))
            {
               $MessageProkoll['Info'][] = "#1005 bestehendes Zielprojekt für Update gefunden: ".$lastInvObj[0]['ObjectName']." ID#".$lastInvObj[0]
            }
            else
            {
               $MessageProkoll['Failure'][] = "#1025 Fehler: kein bestehendes Zielprojekt gefunden";
               exit("Scriptabbruch: kein bestehendes Zielprojekt gefunden! Nur Neu-Installation möglich,\nbitte alle Child-Files unterhalb dieses S
            }
        }
        $MessageProkoll['InstallParameters']  = $Parameters;


    // Funktionsaufrufe
        // Variablenprofile aus Profile-Inventory holen und Profil-Baum generieren
        $VarProfile    = createVarProfileInventory();

        // Variablenprofile anlegen (abhängig von Parameters)
        createVarProfile($VarProfile);

        // Projekt Root-Element aus Inventory generieren und anlegen
        $ObjectTree = createInventoryObjects();

        // Child-Objekte vom Root aus Inventory generieren und anlegen
        $ObjInstallTree = createObjectsTree($ObjectTree);

        // Funktion zum Verlinken aller neu instalierten Elemente
        linknewObjects($ObjInstallTree);

        // Funktion zum Installieren der WFC-Items
        installWFC_Items($WFC_TargetID);

        // Install-Script ins Config-Verzeichnis verschieben
        moveInstallScripttoConfig();

        // Installationsprotokoll erzeugen
        $InstProtocol        = createObjInventoryProtocol($ObjInstallTree);

        // Script self-renaming
                IPS_SetName($_IPS['SELF'], 'Heizung (Quell-ID: 24978) 2014_07_30-12_55'.', Ziel-ID: '.$ObjInstallTree[0]['newObjectID']);


        // Installationsprotokoll schreiben
        CreateProtocolFile($InstProtocol);

    // ++++++++++++++ Script-Core ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


    // prüft, ob Logfiles unterhalb des Export-Scripts hängen und gibt die ID des letzten Logfiles aus +++++++++++++++++
    function findlastLogfile()
    {
       $Childs    = IPS_GetChildrenIDs($_IPS['SELF']);
       $ScrTS   = 0;

       if(isset($Childs[0]))
       {
          $ScrTS   = 0;
          $ScrID   = 0;
          for($i=0;$i<count($Childs);$i++)
          {
             // 1. Zeile aus jedem Protokollfile auslesen und Installationsdatum ermitteln
             $script      = IPS_GetKernelDir()."scripts\\".$Childs[$i].'.ips.php';
              $fh            = fopen($script, "r");
              $firstLine  = mb_strlen(fgets($fh));

                fseek($fh, $firstLine);
              $zeile        = explode(' ', fgets($fh));

              if(isset($zeile[4]))
              {
                  $TS       = strtotime($zeile[4].$zeile[5]);
                }
                else
               {
                   $MessageProkoll['Failure'][] = "#1024 Fehler: letztes Inventory aus Protokollfile NICHT geladen";
                   exit("Scriptabbruch: kein gültiges Protokollfile gefunden! Nur Installation möglich,\nbitte alle Child-Files unterhalb dieses Sc
               }

              // Child mit größtem Timestamp ermitteln
              if(($ScrTS < $TS))
              {
                 $ScrTS   = $TS;
                 $ScrID   = $Childs[$i];
                }
             fclose($fh);
          }
```

```
    }

     // alte Logfiles löschen
    for($i=0;$i<count($Childs);$i++)
    {
       if($ScrID != $Childs[$i]) IPS_DeleteScript($Childs[$i], true);
    }
    return @$ScrID;
}


// Last Install Einzelobjekte-Array (aus Inv-Protokoll) bauen
function createlastInventoryObjects($ID)
{
    global $lastInv;

    for($a=1;$a<count($lastInv);$a++)
    {
        // Einzel-Objekt zusammenstellen
        $InvData     = explode(';', substr(preg_replace("/\r|\n/s", "", $lastInv[$a]),0, -1));
        unset($InvData[0]);  // ersten Datensatz entfernen
        $InvData    = array_slice($InvData, 0);
        for($i=0;$i<count($InvData);$i++)
        {
           $KeyValue         = explode(',', $InvData[$i]);
            $lastInvPlain[$a][$KeyValue[0]]  = $KeyValue[1];
        }
    }
    $lastInvPlain  = array_slice($lastInvPlain, 0);

    return $lastInvPlain;
}

// Funktion installiert externe Scripte (sofern vorhanden)
function createExportFiles($Array)
{
   global                 $MessageProkoll;
    $PicList          = array('png', 'jpg', 'jpeg', 'gif');
    $Cnt                 = count($Array);
    for($i=0;$i<$Cnt;$i++)
    {

       $File                = explode('\\', $Array[$i]);
       $FileName           = trim(array_pop($File));
       $FilePath           = IPS_GetKernelDir().substr(str_replace($FileName, '', $Array[$i]), 0, -1);
       $FileExtension    = str_replace(".", '', substr($FileName, -4));
       $isPic            = in_array($FileExtension, $PicList);

        // Filecontent aus Funktion holen
        $FuncName           = 'ExtFile_'.preg_replace('/[^a-zA-Z0-9]/', '', $Array[$i]);
        $ScrContent       = $FuncName();
        if($isPic)
        {
           $ScrContent        = base64_decode($ScrContent);

        }
        else
        {
           $ScrContent        = str_replace("@$@", "\\", $ScrContent);
            $ScrContent        = str_replace("\\'", "'", $ScrContent);
        }

        // Ordner anlegen, wenn nicht vorhanden
            if (!is_dir($FilePath))
            mkdir ($FilePath  , 0777, true);

        // schreiben der Content-Datei
            $file            = $FilePath.'\\'.$FileName;

            $handle             = fopen($file, "w");
            $result            = fwrite($handle, $ScrContent);
            if($result)
            {
               $MessageProkoll['extFiles']['OK'][]         = $file;
            }
            else
            {
               $MessageProkoll['extFiles']['Failure'][]    = $file;
            }
            fclose($handle);
    }
}

// Array Variablenprofile-Inventory erstellen
function createVarProfileInventory()
{
   global $VProfInv;

   if(@$VProfInv[0] != NULL)
    {
       for($i=0;$i<count($VProfInv);$i++)
       {
          $SubSet[] = explode('|', substr(preg_replace("/\r\n/s", "", $VProfInv[$i]),0, -1));
        }

        for($p=0;$p<count($SubSet);$p++)
        {
           $SubSet[$p][0] = substr($SubSet[$p][0], 0, -1);
           $SubSet[$p][1] = substr(@$SubSet[$p][1], 0, -1);

           $Profiles[$p]    = explode(';', $SubSet[$p][0]);
           for($i=0;$i<count($Profiles[$p]);$i++)
           {
               $Key          = explode(',', $Profiles[$p][$i])[0];
                $Value          = explode(',', $Profiles[$p][$i])[1];
```

```
                $ProfileObj[$Key]     = $Value;
            }
            $VarProfile[$p]     = $ProfileObj;

            // Associations hinzufügen
            if(@$SubSet[$p][1] != NULL)
            {
                $AssRaw    = explode('§', $SubSet[$p][1]);
               for($i=0;$i<count($AssRaw);$i++)
               {
                    $AssBlock    = explode(';', $AssRaw[$i]);
                    for($v=0;$v<4;$v++)
                    {
                       if($AssBlock[$v] != NULL)
                        {
                           $Key        = explode(',',($AssBlock[$v]))[0];
                           $Value    = explode(',',($AssBlock[$v]))[1];
                           $ValuePair[$Key]  = $Value;
                        }
                    }
                    $newAssBlock[] = $ValuePair;
                    unset($ValuePair);
                }
             $VarProfile[$p]['Associations'] = $newAssBlock;
             unset($newAssBlock);
                unset($AssBlock[$p]);
            }
        }
        $MessageProkoll['Info'][] = "#1010 VarProfil-Array erstell";
        return $VarProfile;
    }
}

// Function Variablenprofil anlegen
function createVarProfile($VarProfile)
{
    global $Parameters;

    if(isset($VarProfile[0]))
    {
    // Profile-Array durchgehen
        for($i=0;$i<count($VarProfile);$i++)
        {
           $Profilname = $VarProfile[$i]['ProfileName'];
           $isSystemProfile  = strpos($Profilname, '~');
           if(($Parameters['VarProfile'] == 1) && (IPS_VariableProfileExists($Profilname)) && ($isSystemProfile === false))
           {
              $MessageProkoll['Install']['OK'] = "#2011 Profil $Profilname wird gelöscht";
              if(IPS_VariableProfileExists($Profilname)) IPS_DeleteVariableProfile($Profilname);
            }
            else
            {
               $MessageProkoll['Install']['User'] = "#2010 bestehendes Variablen-Profil $Profilname wird nicht überschrieben (Systemprofil oder

            }

            // Variablen-Profil anlegen (nur Neuanlage)
           if((!IPS_VariableProfileExists($Profilname))&& ($isSystemProfile === false))
            {
               IPS_CreateVariableProfile($Profilname, (int)$VarProfile[$i]['ProfileType']);
              IPS_SetVariableProfileText($Profilname, $VarProfile[$i]['Prefix'], $VarProfile[$i]['Suffix']);
              IPS_SetVariableProfileValues($Profilname, (float)$VarProfile[$i]['MinValue'], (float)$VarProfile[$i]['MaxValue'], (float)$VarProf
             IPS_SetVariableProfileDigits($Profilname, (int)$VarProfile[$i]['Digits']);
                IPS_SetVariableProfileIcon($Profilname, $VarProfile[$i]['Icon']);
                $MessageProkoll['Install']['OK'] = "#2012 Profil $Profilname angelegt";

                // Associations setzen
                if(isset($VarProfile[$i]['Associations']))
                {
                    $Ass = $VarProfile[$i]['Associations'];
                    for($a=0;$a<count($Ass);$a++)
                    {
                       // Falls Assoziation "Name und Icon" leer sind, wird Leerzeichen bei Name eingefügt (sonst Fehlermeldung
                       // bei Assoziations-Zuweisung
                       if(($Ass[$a]['Name'] == NULL) && ($Ass[$a]['Icon'] == NULL)) $Ass[$a]['Name'] = ' ';
                        IPS_SetVariableProfileAssociation($Profilname, (float)$Ass[$a]['Value'], $Ass[$a]['Name'], $Ass[$a]['Icon'], (int)$Ass[
                    }
                    $MessageProkoll['Install']['OK'] = "#2013 Profilassociations für Profil $Profilname angelegt";
                }
            }
        }
        return;
    }
}

// Objekt-Tree generieren und Childs holen
function createInventoryObjects()
{
    global $Inv;
    global $IObjects;
   $DataSet     = explode(';', substr(preg_replace("/\r|\n/s", "", $Inv[1]),0, -1));
   $ObjID          = (int)explode(',', $DataSet[0])[1];
   $Objects[0]    = createObject($ObjID);
   if(isset($Objects[0]['ChildrenIDs']) != NULL)
   $Objects[0]['ChildrenIDs']    = createChildObjects($Objects[0]['ChildrenIDs']);
    return $Objects;
}

// rekursive Objektanlage aller Objekte im Ziel-Projekt
function createObjectsTree($Objects)
{
//global $Objects;
    for($i=0;$i<count($Objects);$i++)
    {
        // Objekt anlegen
        $Objects[$i] = Objekteanlegen($Objects[$i]);
```

```
        // nach Childs durchsuchen
        if(isset($Objects[$i]['ChildrenIDs']))
        {
           for($ch=0;$ch<count($Objects[$i]['ChildrenIDs']);$ch++)
           {
              $Objects[$i]['ChildrenIDs'][$ch]['newParentID']    = $Objects[$i]['newObjectID'];
           }
           // nächste Rekursion:
           $newObjects = $Objects[$i]['ChildrenIDs'];
           $Objects[$i]['ChildrenIDs']  = createObjectsTree($newObjects);
        }
    }
    return $Objects;
}

// Inventory rekursiv abarbeiten und Einzelobjekte zur Anlage übergeben
function createChildObjects($Childs)
{
    for($i=0;$i<count($Childs);$i++)
    {
        $Childs[$i]    = createObject($Childs[$i]);
        if(isset($Childs[$i]['ChildrenIDs']))
        {
           $newChilds  = $Childs[$i]['ChildrenIDs'];
            $Childs[$i]['ChildrenIDs']  = createChildObjects($newChilds);
        }
    }
    return $Childs;
}


// Einzel-Objekt aus Inventory generieren
function createObject($ObjID)
{
   global $Inv;
   global $Parameters;
    global $lastInvObj;

    // Inventory nach Objekt-ID durchsuchen und Array-ID zurückgeben
   for($s=0;$s<count($Inv);$s++)
   {
      if(strstr($Inv[$s], 'ObjectID,'.$ObjID)) $Key = $s;
   }
      $DataSet = explode(';', substr(preg_replace("/\r|\n/s", "", $Inv[$Key]),0, -1));

    // Objekt Key-Value Zuordnung
   for($p=0;$p<count($DataSet);$p++)
   {
       $Key          = explode(',', $DataSet[$p])[0];
        $Value          = explode(',', $DataSet[$p])[1];
        $Object[$Key]    = $Value;
    }

    // newObjectID aus lastInventory (Inst-Protokoll) ankleben, wenn Update = 1
    if($Parameters['Update'] == 1)
    {
       for($l=0;$l<count($lastInvObj);$l++)
       {
          if($Object['ObjectID'] == $lastInvObj[$l]['ObjectID'])
          $Object['newObjectID'] = @$lastInvObj[$l]['newObjectID'];
       }
    }

    // Child-IDs ankleben
   for($i=0;$i<count($Inv);$i++)
   {
        $DataSet = explode(';', substr(preg_replace("/\r|\n/s", "", $Inv[$i]),0, -1));
       for($p=0;$p<count($DataSet);$p++)
       {
           $Key          = explode(',', $DataSet[$p])[0];
            $Value          = explode(',', $DataSet[$p])[1];
            $Subject[$Key]     = $Value;
        }
        if(@$Subject['ParentID'] == $Object['ObjectID']) $Object['ChildrenIDs'][]  = $Subject['ObjectID'];
    }

    return $Object;
}

// +++ Function ZielProjekt-Objekte anlegen ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
function Objekteanlegen($Object)
{
    global $Parameters;
    global $MessageProkoll;
    global $IPS_VERSION;

   $MessageProkoll['Install']['OK'][] = "#3011 Objekt ".$Object['ObjectName']."., altID #".$Object['ObjectID']." zur Anlage übernommen";

    // ObjectInfo Zeilenumbruch zurückwandeln
    $Object['ObjectInfo'] = str_replace("§&§", "\r\n", $Object['ObjectInfo']);
    $Object['ObjectInfo'] = str_replace("§&&§", ",", $Object['ObjectInfo']);
    $Object['ObjectInfo'] = str_replace("§&&&§", ";", $Object['ObjectInfo']);

   switch ($Object['ObjectType'])
    {
        case 0: // Kategorien
                if((@$Object['newObjectID'] > 0) && ($Object['isConfigTree'] == 0))
                {
                $Cat_ID        = (int)$Object['newObjectID'];
                // prüfen, ob Kategorie tats. noch existiert (wenn nicht: anlegen)
                if(!IPS_ObjectExists($Cat_ID)) $Cat_ID = IPS_CreateCategory();
                IPS_SetName($Cat_ID, $Object['ObjectName']);
                    IPS_SetHidden($Cat_ID, (bool)$Object['ObjectIsHidden']);
                    IPS_SetPosition($Cat_ID, (int)$Object['ObjectPosition']);
```

```
                IPS_SetInfo($Cat_ID, $Object['ObjectInfo']);
                IPS_SetIcon($Cat_ID, $Object['ObjectIcon']);
                if(isset($Cat_ID, $Object['newParentID'])) IPS_SetParent($Cat_ID, (int)$Object['newParentID']);
            if($Cat_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3023 Kategorie-Objekt ".$Object['ObjectName']."", ID #".$Cat_ID." existiert (Update e
                $MessageProkoll['Install']['OK'][] = "#3025 Kategorie-Objekt ".$Object['ObjectName']."", ID #".$Cat_ID." neu konfiguriert";
            }
            elseif((@$Object['isConfigTree'] == 0) || (@$Object['newObjectID'] == 0) || (!IPS_ObjectExists((int)$Object['newObjectID'])))
            {
                $Cat_ID        = IPS_CreateCategory();
                $Object['newObjectID']  = $Cat_ID;
                if($Cat_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3022 Kategorie-Objekt ".$Object['ObjectName']."", neuID #".$Cat_ID." angelegt";
                IPS_SetName($Cat_ID, $Object['ObjectName']);
                IPS_SetHidden($Cat_ID, (bool)$Object['ObjectIsHidden']);
                IPS_SetPosition($Cat_ID, (int)$Object['ObjectPosition']);
                IPS_SetInfo($Cat_ID, $Object['ObjectInfo']);
                IPS_SetIcon($Cat_ID, $Object['ObjectIcon']);
                if(isset($Cat_ID, $Object['newParentID'])) IPS_SetParent($Cat_ID, (int)$Object['newParentID']);
                $MessageProkoll['Install']['OK'][] = "#3025 Kategorie-Objekt ".$Object['ObjectName']."", ID #".$Cat_ID." neu konfiguriert";

                if($Cat_ID == 0)
                $MessageProkoll['Install']['Failure'][] = "#3021 Kategorie-Objekt ".$Object['ObjectName']."", , altID #".$Object['ObjectID']
            }
            else
            {
               $MessageProkoll['Install']['OK'][] = "#3024 Kategorie-Objekt ".$Object['ObjectName']." nicht neu konfiguriert (Objekt im Cor
            }
        return $Object;
    break;

    case 1:  // Instanzen
            if((@$Object['newObjectID'] > 0) && ($Object['isConfigTree'] == 0))
            {
            $Inst_ID        = (int)$Object['newObjectID'];
            // prüfen, ob Kategorie tats. noch existiert (wenn nicht: anlegen)
            if(!IPS_ObjectExists($Inst_ID)) $Inst_ID = IPS_CreateInstance($Object['ModuleID']);
            if($Inst_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3033 Kategorie-Objekt ".$Object['ObjectName']."", ID #".$Inst_ID." bereits vorhanden
            }
            elseif(($Object['isConfigTree'] == 0) || (@$Object['newObjectID'] == 0) || (!IPS_ObjectExists((int)$Object['newObjectID'])))
            {
                $Inst_ID        = IPS_CreateInstance($Object['ModuleID']);
                if($Inst_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3032 Instanz-Objekt ".$Object['ObjectName']."", neuID #".$Inst_ID." angelegt";
                if($Inst_ID == 0)
                $MessageProkoll['Install']['Failure'][] = "#3031 Instanz-Objekt ".$Object['ObjectName']."", , altID #".$Object['ObjectID']."
            }
            else
            {
               $MessageProkoll['Install']['OK'][] = "#3034 Instanz-Objekt ".$Object['ObjectName']." nicht neu konfiguriert (Objekt im Confi
            }

           IPS_SetName($Inst_ID, $Object['ObjectName']);
           if(isset($Inst_ID, $Object['newParentID'])) IPS_SetParent($Inst_ID, (int)$Object['newParentID']);
           IPS_SetHidden($Inst_ID, (bool)$Object['ObjectIsHidden']);
           IPS_SetPosition($Inst_ID, (int)$Object['ObjectPosition']);
            IPS_SetInfo($Inst_ID, $Object['ObjectInfo']);
            IPS_SetIcon($Inst_ID, $Object['ObjectIcon']);
           IPS_ApplyChanges($Inst_ID);
           $Object['newObjectID']  = $Inst_ID;

           if($Inst_ID > 0)
            $MessageProkoll['Install']['OK'][] = "#3035 Instanz-Objekt ".$Object['ObjectName']."", neuID #".$Inst_ID." neu konfiguriert";

        return $Object;
    break;

    case 2:  // Variablen, nur anlegen, wenn sie nicht mit einer Instanz verküpft sind ($Object['ObjectIsReadOnly'] == 0)
        if((int)$Object['ObjectIsReadOnly'] == 0)
       {
          $ah_ID = IPS_GetInstanceListByModuleID('{43192F0B-135B-4CE7-A0A7-1475603F3060}')[0]; // Archive Handler
            if((@$Object['newObjectID'] > 0) && ($Object['isConfigTree'] == 0))
            {
            $Var_ID        = (int)$Object['newObjectID'];
            // prüfen, ob Var tats. noch existiert (wenn nicht: anlegen)
            if(!IPS_ObjectExists($Var_ID)) $Var_ID = IPS_CreateVariable((int)$Object['ValueType']);
            if($Var_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3043 Variablen-Objekt ".$Object['ObjectName']."", ID #".$Var_ID." bereits vorhanden (
            }
            elseif((@$Object['isConfigTree'] == 0) || (@$Object['newObjectID'] == 0)|| (!IPS_ObjectExists((int)$Object['newObjectID'])))
            {
                $Var_ID          = IPS_CreateVariable((int)$Object['ValueType']);
                if($Var_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3042 Variablen-Objekt ".$Object['ObjectName']."", neuID #".$Var_ID." angelegt";
                if($Var_ID == 0)
                $MessageProkoll['Install']['Failure'][] = "#3041 Variablen-Objekt ".$Object['ObjectName']."", , altID #".$Object['ObjectID']
            }
            else
            {
               $MessageProkoll['Install']['OK'][] = "#3044 Variablen-Objekt ".$Object['ObjectName']." nicht neu konfiguriert (Objekt im Cor
            }
            //echo "Variablen-Konfiguration für Var ID $Var_ID startet:\n";
           IPS_SetName($Var_ID, $Object['ObjectName']);
           if(isset($Var_ID, $Object['newParentID'])) IPS_SetParent($Var_ID, (int)$Object['newParentID']);
           IPS_SetHidden($Var_ID, (bool)$Object['ObjectIsHidden']);
           IPS_SetPosition($Var_ID, (int)$Object['ObjectPosition']);
            IPS_SetInfo($Var_ID, $Object['ObjectInfo']);
            IPS_SetIcon($Var_ID, $Object['ObjectIcon']);
            //if($Object['VariableCustomAction'] > 0)
            //IPS_SetVariableCustomAction($Var_ID, $Object['VariableCustomAction']);
            IPS_SetVariableCustomProfile($Var_ID, $Object['VariableCustomProfile']);
            AC_SetLoggingStatus($ah_ID, $Var_ID, (bool)$Object['LoggingStatus']);
            AC_SetAggregationType($ah_ID, $Var_ID, (int)$Object['AggregationType']);
```

```
            // Variablenwert schreiben (nur bei Neuinstallation)
            if($Parameters['Update'] = 0)
            {
                switch ((int)$Object['ValueType'])
                {
                   //  Variablenwert je nach Variablentyp schreiben (0= Boolean, 1= Integer, 2= Float; 3= String)
                   case 0:
                      SetValue($Var_ID, (bool)$Object['ValueBoolean']);
                   break;

                   case 1:
                      SetValue($Var_ID, (int)$Object['ValueInteger']);
                   break;

                   case 2:
                      SetValue($Var_ID, (float)$Object['ValueFloat']);
                   break;

                    case 3:
                      SetValue($Var_ID, $Object['ValueString']);
                   break;
                }
            }

            // Ab IPS V 3.0:
            if(IPS_GetKernelVersion() > 2.9)
            {
                AC_SetGraphStatus($ah_ID, $Var_ID, (bool)$Object['GraphStatus']);
            }
        IPS_ApplyChanges($ah_ID);

           $Object['newObjectID']  = $Var_ID;
           if($Var_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3045 Variablen-Objekt ".$Object['ObjectName']."., neuID #".$Var_ID." neu konfiguriert

        }
        else
        {
            $MessageProkoll['Install']['User'][] = "#3044 Variablen-Objekt ".$Object['ObjectName']."., altID #".$Object['ObjectID']." nicht
        }
        return $Object;
    break;

    case 3: // Scripte
         // bei ScriptOverwrite = 0: Prüfen, ob manuell benanntes Scriptfile im System vorhanden (Scriptabbuch, wenn ja)
        if(($Parameters['Update'] == 0) && ($Parameters['ScrOverwrite'] == 0))
        {
                if(explode('.', $Object['ScriptFile'])[0] != $Object['ScriptID'])
                {
                $FileCheck      = file_exists(IPS_GetKernelDir()."scripts\\".$Object['ScriptFile']);

            if($FileCheck)    exit('#3058 Scriptabbruch: Script '.$Object['ScriptFile'].' bereits im Script-Ordner vorhanden,
                                                        bereits installierte Zielprojekt-Objekte müssen manuell gelöscht werden');
                }
        }

           $Scr_Name             = $Object['ObjectName'];
           $Scr_oldID             = @IPS_GetScriptIDByFile($Object['ScriptFile']);
           $ScrFunctionName    = preg_replace('/[^a-zA-Z]/', '',$Scr_Name).$Object['ObjectID'];
          $Scr_Content      = $ScrFunctionName();
          $Scr_Content      = stripslashes($ScrFunctionName());
          $Scr_Content      = str_replace('§\'', "'", $Scr_Content);
          $Scr_Content      = str_replace("@§@", "\\",$Scr_Content);

          // Vorabcheck Script exist
          if(isset($Object['newObjectID']) && (@$Object['newObjectID'] >0))
            {
                $ScrExist     = true;
                $Scr_ID        = (int)$Object['newObjectID'];
                if(!IPS_ObjectExists((int)$Object['newObjectID']))
                {
                    $ScrExist     = false;
                    $Scr_ID        = NULL;
              }
            }
            else
            {
                $ScrExist = false;
            }

          // Script-Neuanlage
           if(($ScrExist == false))
           {
               $Scr_ID                = IPS_CreateScript(0);
               IPS_SetName($Scr_ID, $Scr_Name);

                if(isset($Scr_ID, $Object['newParentID'])) IPS_SetParent($Scr_ID, (int)$Object['newParentID']);
                $fh                                              = fopen(IPS_GetKernelDir()."scripts\\".$Scr_ID.'.ips.php', 'w') or die ("ca
                fwrite($fh, $Scr_Content);
                fclose($fh);
                IPS_SetScriptFile($Scr_ID, $Scr_ID.'.ips.php');
                IPS_SetHidden($Scr_ID, (bool)$Object['ObjectIsHidden']);
               IPS_SetPosition($Scr_ID, (int)$Object['ObjectPosition']);
                IPS_SetInfo($Scr_ID, $Object['ObjectInfo']);
                IPS_SetIcon($Scr_ID, $Object['ObjectIcon']);
                $Scr_IPS_ID  = (int)explode('.',$Object['ScriptFile'])[0];
                $Object['newObjectID']  = $Scr_ID;

                //Scriptfile umbenenen, wenn anderer Name als IPS-ID vergeben
              $ScrFilePrefix = explode('.', $Object['ScriptFile'])[0];
              if($ScrFilePrefix != $Object['ObjectID'])
              {
                    rename($Scr_ID.".ips.php", $Object['ScriptFile']);
                     $result    = IPS_SetScriptFile($Scr_ID, $Object['ScriptFile']);
                     if($result)
```

```
                    $MessageProkoll['Install']['OK'][] = "#3053 Script-Objekt ".$Object['ObjectName']."", File in ".$Object['ScriptFile']."
                }
                else
                {
                if($Scr_ID > 0)
                    $MessageProkoll['Install']['OK'][] = "#3052 Script-Objekt ".$Object['ObjectName']."", neuID #".$Scr_ID." neu angelegt";
                }
            }
            elseif(($ScrExist == true) && ($Object['isConfigTree'] != 1))
            {
               // bei ScriptOverwrite = 0: Prüfen, ob Scriptfile im System vorhanden (Scriptabbuch, wenn ja)
            if(($Parameters['Update'] == 1) && ($Parameters['ScrOverwrite'] == 0))
            {
                if(explode('.', $Object['ScriptFile'])[0] != $Object['ScriptID'])
                    {
                    $FileCheck      = file_exists(IPS_GetKernelDir()."scripts\\".$Object['ScriptFile']);

                if($FileCheck)    exit('#3059 Scriptabbruch: Script '.$Object['ScriptFile'].' bereits im Script-Ordner vorhanden,
                                                        bereits installierte Zielprojekt-Objekte müssen manuell gelöscht werden');
                    }
            }

               // Update (nur, wenn ConfigKategorie-Flag nicht gesetzt)
              if($ScrExist)
              {
                 // Filename mit ID oder individuell benannt
                 $ScrFilePrefix = explode('.', $Object['ScriptFile'])[0];
                 if($ScrFilePrefix == $Object['ObjectID'])
                 {
                        $ScrFileName   = $Object['newObjectID'].'.ips.php';
                        $ScrOverwrite  = 1;
                    }
                    else
                    {
                       $ScrFileName   = $Object['ScriptFile'];
                       if($Parameters['ScrOverwrite'] == 1) {$ScrOverwrite  = 1;}else{$ScrOverwrite  = 0;}
                    }
                }
                else
                {
                   $ScrFileName   = $Object['ScriptFile'];
                   $ScrOverwrite  = 1;
                }

                // Script-File löschen (wenn  vorhanden und ScriptOverwrite = 1) und neu schreiben
                $scrReturn = 0;
                if($ScrExist && $ScrOverwrite)
                {
                  unlink(IPS_GetKernelDir()."scripts\\".$ScrFileName);
                    $fh                                         = fopen(IPS_GetKernelDir()."scripts\\".$ScrFileName, 'w') or die ("can'
                    $MessageProkoll['Install']['OK'][]    = "#3055 Script-File ".$Object['ObjectName']."", ".$ScrFileName.", gelöscht";
                    $scrReturn                                  = fwrite($fh, $Scr_Content);
                    fclose($fh);
               IPS_SetHidden($Scr_ID, (bool)$Object['ObjectIsHidden']);
                   IPS_SetPosition($Scr_ID, (int)$Object['ObjectPosition']);
                    IPS_SetInfo($Scr_ID, $Object['ObjectInfo']);
                    IPS_SetIcon($Scr_ID, $Object['ObjectIcon']);
                }

                // Meldungen für Inst-Protokoll
                if($scrReturn)
                {
                    $MessageProkoll['Install']['OK'][] = "#3051 Script-File ".$Object['ObjectName']."", neu geschrieben";
                }
                elseif($ScrOverwrite == 0)
                {
                   $MessageProkoll['Install']['OK'][] = "#3054 Script-File ".$Object['ObjectName'].', nicht überschrieben (Option $existsSc
                }
                else
                {
                    $MessageProkoll['Install']['Failure'][] = "#3055 Script-File ".$Object['ObjectName']."", neu schreiben fehlgeschlagen";
                }
            }
        return $Object;
    break;

    case 4: // Events
           $Evnt_Name             = $Object['ObjectName'];

          if((@$Object['newObjectID'] > 0) && ($Object['isConfigTree'] == 0))
            {
            $Evnt_ID        = (int)$Object['newObjectID'];
            // prüfen, ob Var tats. noch existiert (wenn nicht: anlegen)
            if(!IPS_ObjectExists($Evnt_ID)) $Evnt_ID = IPS_CreateEvent((int)$Object['EventType']);
            if($Evnt_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3063 Event-Objekt ".$Object['ObjectName']."", ID #".$Evnt_ID." bereits vorhanden (Upc
            }
            elseif((@$Object['isConfigTree'] == 0) || (@$Object['newObjectID'] == 0)|| (!IPS_ObjectExists((int)$Object['newObjectID'])))
            {
                $Evnt_ID = IPS_CreateEvent((int)$Object['EventType']);
                if($Evnt_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3062 Event-Objekt ".$Object['ObjectName']."", neuID #".$Evnt_ID." angelegt";
                if($Evnt_ID == 0)
                $MessageProkoll['Install']['Failure'][] = "#3061 Event-Objekt ".$Object['ObjectName']."", , altID #".$Object['ObjectID']." N
            }
            else
            {
               $MessageProkoll['Install']['OK'][] = "#3064 Event-Objekt ".$Object['ObjectName']." nicht neu konfiguriert (Objekt im Config-
            }
          if(isset($Evnt_ID, $Object['newParentID'])) IPS_SetParent($Evnt_ID, (int)$Object['newParentID']);
          IPS_SetName($Evnt_ID, $Evnt_Name);
          IPS_SetPosition($Evnt_ID, (int)$Object['ObjectPosition']);
          IPS_SetHidden($Evnt_ID, (bool)$Object['ObjectIsHidden']);
           IPS_SetPosition($Evnt_ID, (int)$Object['ObjectPosition']);
            IPS_SetInfo($Evnt_ID, $Object['ObjectInfo']);
```

```
            IPS_SetIcon($Evnt_ID, $Object['ObjectIcon']);
            IPS_SetEventLimit($Evnt_ID, (int)$Object['EventLimit']);

            // getriggertes Element
            if((int)$Object['EventType'] == 0)
            {
                //!!! (int)$Oject['TriggerVariableID'] Auslöser ID derzeit nur ParentObject, da newTarget noch nicht bekannt.
                // Konfiguration von IPS_SetEventTrigger wird erst mit Neuverlinkung abgeschlossen !!
               @IPS_SetEventTrigger($Evnt_ID, (int)$Object['TriggerType'], 0);
               if((int)$Object['TriggerType'] == 4)
                IPS_SetEventTriggerValue($Evnt_ID, $Object['TriggerValue']);
                IPS_SetEventTriggerSubsequentExecution($Evnt_ID, (bool)$Object['TriggerSubsequentExecution']);
                IPS_SetEventActive($Evnt_ID, true);  // Ereignis nach Installation immer aktiv
                $MessageProkoll['Install']['OK'][] = "#3065 Event-Objekt ".$Object['ObjectName']."', neuID #".$Evnt_ID." neu konfiguriert ur
            }
            else
            {
                // zyklisches Element
                IPS_SetEventCyclic($Evnt_ID, (int)$Object['CyclicDateType'], (int)$Object['CyclicDateValue'], (int)$Object['CyclicDateDay']

                // Prüfung IPS-Version (ab 3.1 wurden die SetEventCyclic-Befehle geändert)
                if($IPS_VERSION < 3.1)
                {
                   // Wenn Zielsystem-Version < 3.1 ist
                    IPS_SetEventCyclicDateBounds($Evnt_ID, (float)$Object['CyclicDateFrom'], (float)$Object['CyclicDateTo']);
                    //if((int)$Object['CyclicTimeType'] == 1)
                    IPS_SetEventCyclicTimeBounds($Evnt_ID, (float)$Object['CyclicTimeFrom'], (float)$Object['CyclicTimeTo']);
                }
                else
                {
                   // Wenn Zielsystem >= 3.1 ist
                   if($Object['CyclicDateFrom'] > 0)
                   {
                       IPS_SetEventCyclicDateFrom($Evnt_ID, (int)date("d", $Object['CyclicDateFrom']), (int)date("m", $Object['CyclicDateFr
                     }
                     else
                     {
                        IPS_SetEventCyclicDateFrom($Evnt_ID, 0, 0, 0);
                     }

                     if($Object['CyclicDateTo'] > 0)
                     {
                        IPS_SetEventCyclicDateTo($Evnt_ID, (int)date("d", $Object['CyclicDateTo']), (int)date("m", $Object['CyclicDateTo'])
                    }
                    else
                    {
                       IPS_SetEventCyclicDateTo($Evnt_ID, 0, 0, 0);
                    }
                    IPS_SetEventCyclicTimeFrom($Evnt_ID, (int)date("H", $Object['CyclicTimeFrom']), (int)date("i", $Object['CyclicTimeFrom'
                    IPS_SetEventCyclicTimeTo($Evnt_ID, (int)date("H", $Object['CyclicTimeTo']), (int)date("i", $Object['CyclicTimeTo']), (i
                }

                IPS_SetEventActive($Evnt_ID, false);  // Ereignis nach Installation immer inaktiv
                $MessageProkoll['Install']['User'][] = "#3066 Event-Objekt ".$Object['ObjectName']."', neuID #".$Evnt_ID." neu konfiguriert,
            }

            // PHP-Code einschleusen, wenn vorhanden
            $FuncName   = 'Event'.$Object['ObjectID'];
            if(function_exists($FuncName))
            {
                $FuncContent                    = $FuncName();
                IPS_SetEventScript($Evnt_ID, $FuncContent);
                $MessageProkoll['Install']['OK'][] = "#3067 Event-Objekt ".$Object['ObjectName']."', neuID #".$Evnt_ID." PHP-Code eingebaut"
            }
           $Object['newObjectID']  = $Evnt_ID;
     return $Object;
    break;

    case 5: // Media-Objekte
           $MessageProkoll['Install']['Failure'][] = "#3061 Media-Objekt ".$Object['ObjectName']."', altID #".$Object['ObjectID']." nicht ar
           return 0;

        //return $Object;
    break;

    case 6: // Links

          $Lnk_Name             = $Object['ObjectName'];
          if((@$Object['newObjectID'] > 0) && ($Object['isConfigTree'] == 0))
            {
            $Lnk_ID        = (int)$Object['newObjectID'];
           // prüfen, ob Lnk tats. noch existiert (wenn nicht: anlegen)
            if(!IPS_ObjectExists($Lnk_ID)) $Lnk_ID = IPS_CreateLink();
            if($Lnk_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3063 Link-Objekt ".$Object['ObjectName']."', ID #".$Lnk_ID." bereits vorhanden (Updat
            }
            elseif((@$Object['isConfigTree'] == 0) || (@$Object['newObjectID'] == 0)|| (!IPS_ObjectExists((int)$Object['newObjectID'])))
            {
                $Lnk_ID = IPS_CreateLink();
                if($Lnk_ID > 0)
                $MessageProkoll['Install']['OK'][] = "#3072 Link-Objekt ".$Object['ObjectName']."', neuID #".$Lnk_ID." angelegt";
                if($Lnk_ID == 0)
                $MessageProkoll['Install']['Failure'][] = "#3071 Link-Objekt ".$Object['ObjectName']."', , altID #".$Object['ObjectID']." Ne
            }
            else
            {
               $MessageProkoll['Install']['OK'][] = "#3074 Link-Objekt ".$Object['ObjectName']." nicht neu konfiguriert (Objekt im Config-E
            }
          if(isset($Object['newParentID'])) IPS_SetParent($Lnk_ID, (int)$Object['newParentID']);
          IPS_SetName($Lnk_ID, $Lnk_Name);
          IPS_SetPosition($Lnk_ID, (int)$Object['ObjectPosition']);
          IPS_SetHidden($Lnk_ID, (bool)$Object['ObjectIsHidden']);
           IPS_SetPosition($Lnk_ID, (int)$Object['ObjectPosition']);
            IPS_SetInfo($Lnk_ID, $Object['ObjectInfo']);
            IPS_SetIcon($Lnk_ID, $Object['ObjectIcon']);
```

```
                //IPS_SetLinkChildID($LinkID, 12345 /*[Objekt #12345 existiert nicht]*/);
                //IPS_SetLinkTargetID($LinkID, 12345 /*[Objekt #12345 existiert nicht]*/);

                  $MessageProkoll['Install']['OK'][] = "#3071 Link-Objekt ".$Object['ObjectName']."., neuID #".$Lnk_ID." neu konfiguriert";
              $Object['newObjectID']  = $Lnk_ID;
              return $Object;

          break;
      }
}

// nachträgliche Verlinkung aller neu installierten Objekte
function linknewObjects($ObjLinkTree)
{
   global $ObjInstallTree;
   global $newTgtID;
   global $newObjID;
   global $MessageProkoll;

   for($i=0;$i<count($ObjLinkTree);$i++)
   {
      $newObjID        = 0;
      $newTgtID        = 0;
      // Inventory nach Variablen-, Script-, Event- und Link-Objekten durchsuchen und neue Target-ID finden
      // 2= Variable, 3= Script, 4= Event, 6 = Link
      if(($ObjLinkTree[$i]['ObjectType'] == 2) && ($ObjLinkTree[$i]['VariableCustomAction'] >0))    // Scripts nach austauschbaren IDs durchsuc
        {
         // neue Target-ID (Action-Script) für Variable finden
            $newTargetID = findNewTargetID($ObjInstallTree, $ObjLinkTree[$i]['VariableCustomAction']);
         IPS_SetVariableCustomAction((int)$ObjLinkTree[$i]['newObjectID'], (int)$newTargetID);
         if($newTargetID >0)
            $MessageProkoll['Link']['OK'][] = "#4011 Variablen-Objekt ".$ObjLinkTree[$i]['ObjectName']."., ID #".$ObjLinkTree[$i]['newObjectID']
         if($newTargetID == 0)
            $MessageProkoll['Link']['Failure'][] = "#4010 Variablen-Objekt ".$ObjLinkTree[$i]['ObjectName']."., ID #".$ObjLinkTree[$i]['newObjec

       }
      elseif($ObjLinkTree[$i]['ObjectType'] == 3)     // Scripts nach austauschbaren IDs durchsuchen
        {
         // aktuellen Scriptnamen finden
         $newObjID = 0;
            $Scr  = IPS_GetScript((int)$ObjLinkTree[$i]['newObjectID'])['ScriptFile'];
         changeObjectIDs_inScript($ObjInstallTree, $Scr);
            $MessageProkoll['Link']['OK'][] = "#4021 Script-Objekt ".$ObjLinkTree[$i]['ObjectName']."., ID #".$ObjLinkTree[$i]['newObjectID']."., 
       }
      elseif(($ObjLinkTree[$i]['ObjectType'] == 4) && ($ObjLinkTree[$i]['EventType'] == 0))    // Neuverlinkung nur für nichtzyklische Events
        {
           $newTgtID   =0;
            $newTargetID = findNewTargetID($ObjInstallTree, $ObjLinkTree[$i]['TriggerVariableID']);

            // neue Verlinkung setzen
            if($newTgtID !=0)
            {
                IPS_SetEventTrigger((int)$ObjLinkTree[$i]['newObjectID'], (int)$ObjLinkTree[$i]['TriggerType'], $newTgtID);
                $MessageProkoll['Link']['OK'][] = "#4031 Event-Objekt ".$ObjLinkTree[$i]['ObjectName']."., ID #".$ObjLinkTree[$i]['newObjectID']
            }
            else
            {
                $MessageProkoll['Link']['User'][] = "#4030 Event-Objekt ".$ObjLinkTree[$i]['ObjectName']."., neue ID #".$ObjLinkTree[$i]['newObje
            }
       }
      elseif($ObjLinkTree[$i]['ObjectType'] == 6)
        {
           $newTgtID   =0;
            $newTgtID = findNewTargetID($ObjInstallTree, $ObjLinkTree[$i]['LinkChildID']);
            if($newTgtID !=0)
            {
                IPS_SetLinkTargetID((int)$ObjLinkTree[$i]['newObjectID'], (int)$newTgtID);
                $MessageProkoll['Link']['OK'][] = "#4041 Link-Objekt ".$ObjLinkTree[$i]['ObjectName']."., ID #".$ObjLinkTree[$i]['newObjectID'].
            }
            else
            {
                $MessageProkoll['Link']['User'][] = "#4040 Link-Objekt ".$ObjLinkTree[$i]['ObjectName']."., ID #".$ObjLinkTree[$i]['newObjectID'
            }
       }

       // Objekt-Childs durchsuchen
       if(isset($ObjLinkTree[$i]['ChildrenIDs']))
       {
          $newLnkTree = $ObjLinkTree[$i]['ChildrenIDs'];
         linknewObjects($newLnkTree);
       }
   }
   unset($ObjLinkTree);
}

// sucht neue Target-ID für verlinkte Objekte
function findNewTargetID($ObjLinkTree, $oldTargetID)
{
   global $newTgtID;
   for($i=0;$i<count($ObjLinkTree);$i++)
   {
      if($ObjLinkTree[$i]['ObjectID'] == $oldTargetID)
      {
          $newTgtID   = $ObjLinkTree[$i]['newObjectID'];
            $i = 9999;  // Abbruch, wenn ID gefunden
      }
        // in Childs suchen (rekursiv)
      if((isset($ObjLinkTree[$i]['ChildrenIDs'])) && ($newTgtID == 0))
      {
            $newObjInstTree   = $ObjLinkTree[$i]['ChildrenIDs'];
         findNewTargetID($newObjInstTree, $oldTargetID);
         unset($newObjInstTree);
      }
   }
```

```
   return $newTgtID;
}

// Austausch der Objekt-IDs im Script
function changeObjectIDs_inScript($ObjLinkTree, $Scr)
{
   global $newObjID;
   global $MessageProkoll;

   $ScrContent = file(IPS_GetKernelDir()."scripts\\".$Scr);
    $file         = fopen(IPS_GetKernelDir()."scripts\\".$Scr, "w");
    $search         = '/[^0-9\/\{-]([1-5][0-9]{4})[^0-9]/'; // suche 5stellige Zahlen von 10.000 - 59.999
    $counter    = 0;

    foreach ($ScrContent as $z)
    {
       $newObjID = NULL;
       $newObjectID = NULL;
        preg_match($search, $z, $matches);
        if(isset($matches[1]))
        {
           $oldID           = $matches[1];
            $newObjectID   = findNewObjectID($ObjLinkTree, $oldID);
            if(($matches[1] == $oldID) && ($newObjectID >0))
            {
               $z = str_replace($oldID, $newObjectID, $z);
               $MessageProkoll['Script']['OK'][] = "#5011 Script ID#$Scr: Inhalt verändert: Zeile ".($counter + 1).", alte Target-ID #$oldID ge
            }
            else
            {
               $MessageProkoll['Script']['User'][] = "#5010 Script ID#$Scr: Inhalt NICHT verändert: Zeile ".($counter + 1).", alte Target-ID #$
            }
        }
        fwrite($file, $z);
        $counter ++;
    }
    unset($ScrContent);
    unset($ObjLinkTree);
    fclose($file);
    return;
}

// sucht neue Object-ID für Script-Variablen
function findNewObjectID($ObjTree, $oldObjID)
{
   global $newObjID;
   for($i=0;$i<count($ObjTree);$i++)
   {
      if($ObjTree[$i]['ObjectID'] == $oldObjID)
      {
          $newObjID   = $ObjTree[$i]['newObjectID'];
            $i = 9999;  // Abbruch, wenn ID gefunden
      }

       // in Childs suchen (rekursiv)
       if(isset($ObjTree[$i]['ChildrenIDs']))
       {
            $newObjTree   = $ObjTree[$i]['ChildrenIDs'];
          findNewObjectID($newObjTree, $oldObjID);
          unset($newObjTree);
       }
   }
   unset($ObjTree);
   return $newObjID;
}


############################## WFE-Items installieren ##########################################################

function installWFC_Items($WFC_TargetID)
{
    global $ObjInstallTree;
    global $newObjID;
    global $WFC_existItemoverwrite;
    global $MessageProkoll;
    $MessageProkoll['WFC']['OK'][] = "#6001 lade WFC-Objektbaum";

    // Objektbaum Ziel-ID (Root-ID des Projekts)
    $ZielID                 = $ObjInstallTree[0]['newObjectID'];

    // Inventory des Ziel-WFC laden
    if(@WFC_GetItems($WFC_TargetID))
    $WFC_Target_Inventory   = WFC_GetItems($WFC_TargetID);

    // Inventory des Quell-Projektes laden
    $WFC_Raw        = loadWFCInventory();

    if(isset($WFC_Target_Inventory[0]) && ($WFC_Raw[0] != NULL))
    {
        // Quell-Inventory durchgehen und Items installieren
        $MessageProkoll['WFC']['OK'][] = "#6002 durchsuche QuellWFC-Objektbaum";
        $Cnt          = count($WFC_Raw);

        for($i=0;$i<$Cnt;$i++)
        {
            // Inventory-String letzte 2 Zeichen abschneiden
            $str  = substr($WFC_Raw[$i], 0, -2);

            // Item-Zeilen in Key|Value-Sätze zerlegen (1.Stufe)
            $WFC_DataSet = explode(';', $str);

            // Key|Value-Sätze in Key und Value zerlegen (2. Stufe)
            for($e=0;$e<count($WFC_DataSet);$e++)
            {
               $WFC_Values[$e]     = explode('|', $WFC_DataSet[$e]);
```

```
                    }

                // Suche für alte TargetIDs neue Target-ID und tausche diese aus (nur Objekte mir Target-ID im Objektbaum)
                   $newTargetID           = NULL;
                   unset                        ($matches);
                $search                   = '/[1-5]\d\d\d\d/'; // suche 5stellige Zahlen von 10.000 - 59.999
                if($WFC_Values[1][1] == 'Category' || $WFC_Values[1][1] == 'InfoWidget' || $WFC_Values[1][1] == 'Graph' || $WFC_Values[1][1] == 'Cc
                    {
                    preg_match($search, $WFC_Values[2][1], $matches);
                        if(isset($matches[0])) $oldTargetID   = $matches[0];
                        if(isset($matches[0])) $newTargetID      = findNewObjectID($ObjInstallTree, $matches[0]);
                        if($newTargetID != NULL)
                        {
                            $WFC_Values[2][1]     = str_replace($oldTargetID, $newTargetID, $WFC_Values[2][1]);
                            $MessageProkoll['WFC']['OK'][] = "#6011 WFC altTargetID #$oldTargetID gegen newTargetID #$newTargetID getauscht";
                        }
                        else
                        {
                           $MessageProkoll['WFC']['User'][] = "#6012 WFC Item ".$WFC_Values[1][1].", ".$WFC_Values[3][1]." keine neue TargetID gefu
                        }
                    }
                    unset($newObjID);

                // Namen für neues WFC-Item ändern (Präfix "Cpy" hinzufügen)
                    if(strpos($WFC_Values[2][1], 'roottp') == 0)
                    {
                       $StartPos         = strpos($WFC_Values[2][1], '"name":')+8;
                       $EndPos           = strpos($WFC_Values[2][1], '","', $StartPos);
                       $OldItemName      = substr($WFC_Values[2][1], $StartPos, $EndPos - $StartPos);
                       $NewItemName      = substr($WFC_Values[2][1], $StartPos, $EndPos - $StartPos).'_ScrID'.$ZielID;
                        $WFC_Values[2][1] = str_replace($OldItemName, $NewItemName, $WFC_Values[2][1]);
                    }

                // Namen für ID (Item) ändern
                    if($WFC_Values[3][1] != 'roottp')
                    $WFC_Values[3][1] = str_replace($WFC_Values[3][1], $WFC_Values[3][1].'_ScrID'.$ZielID, $WFC_Values[3][1]);

                // Namen für ParentID (Item) ändern
                    if($WFC_Values[4][1] != 'roottp')
                    $WFC_Values[4][1] = str_replace($WFC_Values[4][1], $WFC_Values[4][1].'_ScrID'.$ZielID, $WFC_Values[4][1]);

                // Root-Item aus Quell-Projekt definieren (wird bei Updates nicht gelöscht)
                    if($i == 0) $WFC_RootItem  = $WFC_Values[3][1];

                // Ziel-Inventory nach vorhandenen Elementen durchsuchen
                    $Item_exist = 0;
                    foreach($WFC_Target_Inventory as $value)
                    {
                       // wenn WFE-Objekt gefunden (aktuelle Syntax 'ItemName_ScrID12345' bzw. alte Syntax 'CpyItemName')
                        if(($value['ID'] == $WFC_Values[3][1]) || ($value['ID'] == 'Cpy'.substr($WFC_Values[3][1], 0, -11)))
                        {
                           // wenn Überschreib-Option abgeschaltet und WFE Objekt nicht alter Syntax ('CpyItemName') entspricht
                            if(($WFC_existItemoverwrite == 0) && ($value['ID'] != 'Cpy'.substr($WFC_Values[3][1], 0, -11)))
                            {
                                $Item_exist = 1;
                                $MessageProkoll['WFC']['User'][] = "#6020 WFC-Item ".$WFC_Values[3][1]." nicht gelöscht (Option abgeschaltet)";
                                break;
                            }
                            elseif(($WFC_Values[3][1] != $WFC_RootItem) && ($value['ID'] != 'Cpy'.substr($WFC_Values[3][1], 0, -11)))
                            {
                               WFC_DeleteItem($WFC_TargetID, $WFC_Values[3][1]); // vorhandenes Item (neue Syntax) löschen
                               $MessageProkoll['WFC']['OK'][] = "#6021 WFC-Item ".$WFC_Values[3][1]." gelöscht (Option eingeschaltet)";
                            }
                            elseif($value['ID'] == 'Cpy'.substr($WFC_Values[3][1], 0, -11))
                            {
                               WFC_DeleteItem($WFC_TargetID, 'Cpy'.substr($WFC_Values[3][1], 0, -11)); // vorhandenes Item (alte Syntax) löschen
                               $MessageProkoll['WFC']['OK'][] = "#6022 WFC-Item ".$WFC_Values[3][1]." veraltetes WFE-Objekt ('CpyItemname') gelösch
                            }

                            if($WFC_Values[3][1] == $WFC_RootItem)
                            {
                               // wenn Root-Item veraltete Syntax hat
                               $Item_exist = 1;
                            }
                        }
                    }

                // Wenn Item im Ziel-WFC nicht vorhanden, dann installieren
                    if($Item_exist == 0)
                    {
                        if($WFC_Values[3][1] != 'roottp')
                        WFC_AddItem($WFC_TargetID, $WFC_Values[3][1], $WFC_Values[1][1], $WFC_Values[2][1], $WFC_Values[4][1]);
                        WFC_UpdatePosition($WFC_TargetID, $WFC_Values[3][1], (int)$WFC_Values[5][1]);
                        WFC_UpdateVisibility($WFC_TargetID, $WFC_Values[3][1], (bool)$WFC_Values[6][1]);
                        $MessageProkoll['WFC']['OK'][] = "#6031 WFC-Item ".$WFC_Values[3][1]." installiert";
                    }
                    else
                    {
                       $MessageProkoll['WFC']['OK'][] = "#6030 WFC-Item ".$WFC_Values[3][1]." nicht installiert, da bereits vorhanden";
                    }

                unset($WFC_Values);
                unset($WFC_DataSet);
            }
            IPS_ApplyChanges($WFC_TargetID);
        }
        else
        {
           $MessageProkoll['WFC']['User'][] = "#6000 WFCObjektbaum: keine Installation möglich (kein Quell-Inventory oder kein Ziel-WFE gefunden)";
        }
    }


    // Install-Script ins Projekt-Config Verzeichnis verschieben
    function moveInstallScripttoConfig()
```

```
{
    global $ObjInstallTree;
    global $MessageProkoll;

    $RootID     = (int)$ObjInstallTree[0]['newObjectID'];
    $search     = 'Config';
    $SrchResult = searchConfigDir($ObjInstallTree);
    if($SrchResult > 0)
    {
        IPS_SetParent($_IPS['SELF'], $SrchResult);
        $MessageProkoll['Info'][] = "#1011 Install-Script: Install-Script und -Protokoll in Config-Ordner verschoben";
    }
    else
    {
        $MessageProkoll['Info'][] = "#1010 Install-Script: Install-Script und -Protokoll NICHT verschoben (kein Config-Ordner im Zielprojekt ge
    }
    return;
}

// Funktion zum rekursiven Durchsuchen des Obj-Inventorys nach Config-Dir
function searchConfigDir($Array)
{
    global $SrchResult;
    $search    = 'Config';
   $Cnt     = count($Array);
   for($i=0;$i<$Cnt;$i++)
   {
      if($Array[$i]['ObjectName'] == $search)
      {
         $SrchResult    = (int)$Array[$i]['newObjectID'];
        }

        if((isset($Array[0]['ChildrenIDs'])) && (!isset($SrchResult)))
        {
          $newArray = $Array[0]['ChildrenIDs'];
          searchConfigDir($newArray);
        }
   }
    unset($Array);
    unset($newArray);
    return $SrchResult;
}

// Protokoll ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
function InstallProtocoll()
{
global $MessageProkoll;

// +++ Installations-Parameter +++++++++++++++++++++++++++
    $ps   ='// +++ Installationsprotokoll vom '.date("d.m.Y, H:i:s", time()).' ++++++++++++++++++++++++++++++++++++++++++++++'."\r\n\r\n";
    $ps   .="/** +++++++++++++++++ Installationsparameter ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\r\n";
    $ps   .="    Update:                                ".$MessageProkoll['InstallParameters']['Update']."\r\n";
    $ps   .="    Variablenprofile installieren:     ".$MessageProkoll['InstallParameters']['VarProfile']."\r\n";
    $ps   .="    benamste Scripte überschreiben:    ".$MessageProkoll['InstallParameters']['ScrOverwrite']."\r\n";
    $ps   .="    WFC-Items überschreiben:            ".$MessageProkoll['InstallParameters']['WFCItems']."\r\n";
    $ps   .="++++++++++++++++++++ Installationsparameter Ende ++++++++++++++++++++++++++++++++++++++++++++++++++++++\r\n\r\n\r\n";

// +++ Infos +++++++++++++++++++++++++++++++++++++
    $ps   .="++++++++++++++++++++ Installations-Infos ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\r\n";
    for($i=0;$i<count(@$MessageProkoll['Info']);$i++)
    {
        $ps   .="   ".@$MessageProkoll['Info'][$i]."\r\n";
    }
    $ps   .="++++++++++++++++++++ Installations-Infos Ende ++++++++++++++++++++++++++++++++++++++++++++++++++++++\r\n\r\n\r\n";

// +++ Installation ++++++++++++++++++++++++++++++++++++
    $ps   .="++++++++++++++++++++ Objekt-Installation ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\r\n";
    $ps   .="        --- OK-Meldungen ----------------------------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['Install']['OK']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['Install']['OK'][$i]."\r\n";
    }
    $ps   .="        --- OK-Meldungen Ende ----------------------------------------------------------------------\r\n\r\n\r\n";

    $ps   .="        --- Kontrolle durch User erforderlich ----------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['Install']['User']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['Install']['User'][$i]."\r\n";
    }
    $ps   .="        --- Kontrolle durch User erforderlich Ende----------------------------------------------\r\n\r\n\r\n";

    $ps   .="        --- Fehlermeldungen ----------------------------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['Install']['Failure']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['Install']['Failure'][$i]."\r\n";
    }
    $ps   .="        --- Fehlermeldungen Ende----------------------------------------------------------------------\r\n\r\n\r\n";

// +++ WFC-Objekte installieren und verlinken ++++++++++++++++++++++++++++++++++++
    $ps   .="    ++++++++++++++++++ WFC-Objekte installieren und verlinken ++++++++++++++++++++++++++++++++++++++++\r\n";
    $ps   .="        --- OK-Meldungen ----------------------------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['WFC']['OK']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['WFC']['OK'][$i]."\r\n";
    }
    $ps   .="        --- OK-Meldungen Ende ----------------------------------------------------------------------\r\n\r\n\r\n";

    $ps   .="        --- Kontrolle durch User erforderlich ----------------------------------------------------\r\n";
    for($i=0;$i<count(@$MessageProkoll['WFC']['User']);$i++)
    {
        $ps   .="       ".@$MessageProkoll['WFC']['User'][$i]."\r\n";
    }
    $ps   .="        --- Kontrolle durch User erforderlich Ende----------------------------------------------\r\n\r\n\r\n";

    $ps   .="        --- Fehlermeldungen ----------------------------------------------------------------------\r\n";
```

```
        for($i=0;$i<count(@$MessageProkoll['WFC']['Failure']);$i++)
        {
            $ps   .="       ".@$MessageProkoll['WFC']['Failure'][$i]."\r\n";
        }
        $ps   .="       --- Fehlermeldungen Ende------------------------------------------------------------------------\r\n\r\n";

    // +++ Scripte nachbearbeiten ++++++++++++++++++++++++++++++++++++
        $ps   .="    ++++++++++++++++++ ID Austausch in Scripts ++++++++++++++++++++++++++++++++++++++++++++++++++++++\r\n";
        $ps   .="       --- OK-Meldungen ------------------------------------------------------------------------------\r\n";
        for($i=0;$i<count(@$MessageProkoll['Script']['OK']);$i++)
        {
            $ps   .="       ".@$MessageProkoll['Script']['OK'][$i]."\r\n";
        }
        $ps   .="       --- OK-Meldungen Ende ----------------------------------------------------------------------\r\n\r\n";

        $ps   .="       --- Kontrolle durch User erforderlich -------------------------------------------------------\r\n";
        for($i=0;$i<count(@$MessageProkoll['Script']['User']);$i++)
        {
            $ps   .="       ".@$MessageProkoll['Script']['User'][$i]."\r\n";
        }
        $ps   .="       --- Kontrolle durch User erforderlich Ende---------------------------------------------------\r\n\r\n";

        $ps   .="       --- Fehlermeldungen -------------------------------------------------------------------------\r\n";
        for($i=0;$i<count(@$MessageProkoll['Script']['Failure']);$i++)
        {
            $ps   .="       ".@$MessageProkoll['Script']['Failure'][$i]."\r\n";
        }
        $ps   .="       --- Fehlermeldungen Ende---------------------------------------------------------------------\r\n\r\n";

    // +++ Objekte neu verlinken ++++++++++++++++++++++++++++++++++++
        $ps   .="    ++++++++++++++++++ Objekte neu verlinken ++++++++++++++++++++++++++++++++++++++++++++++++++++++\r\n";
        $ps   .="       --- OK-Meldungen ------------------------------------------------------------------------------\r\n";
        for($i=0;$i<count(@$MessageProkoll['Link']['OK']);$i++)
        {
            $ps   .="       ".@$MessageProkoll['Link']['OK'][$i]."\r\n";
        }
        $ps   .="       --- OK-Meldungen Ende ----------------------------------------------------------------------\r\n\r\n";

        $ps   .="       --- Kontrolle durch User erforderlich -------------------------------------------------------\r\n";
        for($i=0;$i<count(@$MessageProkoll['Link']['User']);$i++)
        {
            $ps   .="       ".@$MessageProkoll['Link']['User'][$i]."\r\n";
        }
        $ps   .="       --- Kontrolle durch User erforderlich Ende---------------------------------------------------\r\n\r\n";

        $ps   .="       --- Fehlermeldungen -------------------------------------------------------------------------\r\n";
        for($i=0;$i<count(@$MessageProkoll['Link']['Failure']);$i++)
        {
            $ps   .="       ".@$MessageProkoll['Link']['Failure'][$i]."\r\n";
        }
        $ps   .="       --- Fehlermeldungen Ende---------------------------------------------------------------------\r\n\r\n";

        // +++ externe Files installiert ++++++++++++++++++++++++++++++++++++
        $ps   .="    ++++++++++++++++++ Installation externe Files ++++++++++++++++++++++++++++++++++++++++++++++++++++\r\n";
        $ps   .="       --- OK-Meldungen ------------------------------------------------------------------------------\r\n";
        for($i=0;$i<count(@$MessageProkoll['extFiles']['OK']);$i++)
        {
            $ps   .="       ".@$MessageProkoll['extFiles']['OK'][$i]."\r\n";
        }
        $ps   .="       --- OK-Meldungen Ende ----------------------------------------------------------------------\r\n\r\n";

        $ps   .="       --- Fehlermeldungen -------------------------------------------------------------------------\r\n";
        for($i=0;$i<count(@$MessageProkoll['extFiles']['Failure']);$i++)
        {
            $ps   .="       ".@$MessageProkoll['extFiles']['Failure'][$i]."\r\n";
        }
        $ps   .="       --- Fehlermeldungen Ende---------------------------------------------------------------------\r\n\r\n";

        $ps   .="**/\r\n";

    echo $ps;
    return $ps;
    }


    ############################## Installations-Protokoll ##################################################
    // +++ erstellt Installations-Protokoll Objekt-Inventory für Exportscript +++++++++++++++
    function createObjInventoryProtocol($ObjInstInv)
    {
        // Object-Inventory Function-Kopf generieren
        $s = "<?\r\n";
        $s .= InstallProtocoll();
        $s .= "// ### InstallationsProtokoll Ende ################################################\r\n\r\n\r\n\r\n";
        $s .= '// Function Object Inventory Install-Protocol'."\r\n";
        $s .= 'function ObjectInventoryProtocol()'."\r\n";
        $s .= '{'."\r\n";
        $s .= '$raw = \''."\r\n";

        // Array Copy-Parameters
        $s .= createCopyParametersString();

        // Daten der Einzelobjekte aus Objekt-Inventory an Export-String anfügen
        $ObjStringArray = ChildGenerator($ObjInstInv);
        //print_r($ObjInstInv);
        $s .= createInventoryString($ObjStringArray);

        // Function Footer
        $s .= "';\r\n";
        $s .= '$Inv    = explode("\n",substr(substr($raw, 1), 1, -1)); // ersten und letzten Zeilenumbruch entfernen'."\r\n";
        $s .= 'return $Inv;'."\r\n";
        $s .= "}\r\n\r\n";
        $s .= "?>";
        return $s;
    }
```

```
// generiert rekursiv Protkoll-String für ggf. vorhandene Child-Arrays unter einem Objekt im Objekt-Inventory
function ChildGenerator($Array)
{
    for($i=0;$i<count($Array);$i++)
   {
      $obj[] = createObjectRow($Array[$i]);
        if((isset($Array[$i]['ChildrenIDs'])))
       {
          $newArray = $Array[$i]['ChildrenIDs'];
            $obj[]    = ChildGenerator($newArray);
        }
   }
   return $obj;
}

// Function Objekt Install Protocol: Datenstring für 1 Objekt generieren
function createObjectRow($Obj)
{
   // Objektstammdaten aus IPS_GetObject
   unset($Obj['ChildrenIDs']);

   // Steuerzeichen CR LF aus Object-Info filtern und ersetzen
    $Obj['ObjectInfo'] = str_replace("\r\n", "§&§", $Obj['ObjectInfo']);

    // Stringstart definieren, 1. Entrag muss ObjectID sein!
   $s   = 'ObjectID,'.$Obj['ObjectID'].';';

   // Inventory Export: Stringteil Objekt-Stammdaten vervollständigen
    for($i=0;$i<count($Obj);$i++)
    {
       $key        = key($Obj);
        $s .= key($Obj).','.$Obj[key($Obj)].';';
        next ($Obj);
    }
    $s .= "\r\n";

    return $s;
}

// +++ Export: geht Objekt-Inventory rekrusiv durch und fügt Objektdaten der Einzelobjekte an Protokoll-String an
function createInventoryString($Array)
{
   $str = '';
    for($i=0;$i<count($Array);$i++)
    {
       if(is_array($Array[$i]))
        {

           $newArray = $Array[$i];
           $str  .= createInventoryString($newArray);

        }
        else
        {
           $str .= $Array[$i];
       }
    }
    unset($Array);

    return $str;
}

// Copy-Parameters String für Install-Protocol generieren
function createCopyParametersString()
{
    global $Parameters;
    //$Array   = $_IPS['InstParameters'];
    $s = 'CopyParameters,;';
    for($i=0;$i<count($Parameters);$i++)
    {
       $s .=key($Parameters).','.current($Parameters).';';
       next($Parameters);
    }
    $s .= "\r\n";

    return $s;
}

// Protokoll-File schreiben
function CreateProtocolFile($InstProtocol)
{
    $ScrName    = IPS_GetName($_IPS['SELF'])." InstallProtocol";
    echo "Protokollname: ".$ScrName;
    $Script_ID = IPS_CreateScript(0);
    IPS_SetName($Script_ID, $ScrName);
    IPS_SetParent($Script_ID, $_IPS['SELF']);
    IPS_SetInfo($Script_ID, "This script was created by: RS IPS Exporter ID #".$_IPS['SELF']);
    $fh = fopen(IPS_GetKernelDir()."scripts\\".$Script_ID.".ips.php", 'w') or die("cant open file");
    fwrite($fh, $InstProtocol);
    fclose($fh);
    IPS_SetScriptFile($Script_ID, $Script_ID.'.ips.php');
    echo "Installationsprotokoll-Script \"$ScrName\" ID #$Script_ID angelegt\n";
    return $Script_ID;
}

################ Inventory-Daten ################################################################################

// Function Object Inventory
function ObjectInventory()
{
global $MessageProkoll;
$raw = '
CopyParameters,;VarProfile,1;ScrOverwrite,0;
ObjectID,24978 /*[Funktionen\Heizung]*/;HasChildren,1;ObjectIcon,;ObjectID,24978 /*[Funktionen\Heizung]*/;ObjectIdent,;ObjectInfo,;ObjectIsHidc
ObjectID,20977 /*[Funktionen\Heizung\Vorlauftemperatur]*/;HasChildren,;ObjectIcon,;ObjectID,20977 /*[Funktionen\Heizung\Vorlauftemperatur]*/;Ob
```

```
ObjectID,23090 /*[Funktionen\Heizung\Aktueller Modus]*/;HasChildren,;ObjectIcon,;ObjectID,23090 /*[Funktionen\Heizung\Aktueller Modus]*/;Object
ObjectID,27399 /*[Funktionen\Heizung\Viessmann Steuerung]*/;HasChildren,1;ObjectIcon,;ObjectID,27399 /*[Funktionen\Heizung\Viessmann Steuerung]
ObjectID,11051 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe]*/;HasChildren,1;ObjectIcon,;ObjectID,11051 /*[Funktionen\Heizung\Viessmann St
ObjectID,20154 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Raum Soll Temperatur]*/;HasChildren,;ObjectIcon,;ObjectID,20154 /*[Funktionen\
ObjectID,22474 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Vorlauftemperatur]*/;HasChildren,;ObjectIcon,;ObjectID,22474 /*[Funktionen\Hei
ObjectID,23092 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Uhrzeit]*/;HasChildren,;ObjectIcon,;ObjectID,23092 /*[Funktionen\Heizung\Viess
ObjectID,28759 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\AktuelleBetriebsartA1M]*/;HasChildren,;ObjectIcon,;ObjectID,28759 /*[Funktione
ObjectID,28911 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Kesseltemperatur]*/;HasChildren,;ObjectIcon,;ObjectID,28911 /*[Funktionen\Heiz
ObjectID,30781 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Heizkreispumpe A1M1]*/;HasChildren,;ObjectIcon,;ObjectID,30781 /*[Funktionen\H
ObjectID,31685 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Brennerstufe]*/;HasChildren,;ObjectIcon,;ObjectID,31685 /*[Funktionen\Heizung\
ObjectID,33016 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Sparbetrieb]*/;HasChildren,;ObjectIcon,;ObjectID,33016 /*[Funktionen\Heizung\V
ObjectID,34221 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Partybetrieb]*/;HasChildren,;ObjectIcon,;ObjectID,34221 /*[Funktionen\Heizung\
ObjectID,34515 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Kesselsolltemperatur]*/;HasChildren,;ObjectIcon,;ObjectID,34515 /*[Funktionen\
ObjectID,36927 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Ruecklauftemperatur17A]*/;HasChildren,;ObjectIcon,;ObjectID,36927 /*[Funktione
ObjectID,38089 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Zirkulationspumpe]*/;HasChildren,;ObjectIcon,;ObjectID,38089 /*[Funktionen\Hei
ObjectID,41873 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Brennerstörung]*/;HasChildren,;ObjectIcon,;ObjectID,41873 /*[Funktionen\Heizur
ObjectID,45011 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\AussentemperaturGedaempft]*/;HasChildren,;ObjectIcon,;ObjectID,45011 /*[Funkti
ObjectID,55063 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\AnlagenIstLeistung]*/;HasChildren,;ObjectIcon,;ObjectID,55063 /*[Funktionen\He
ObjectID,55242 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Speicherladepumpe]*/;HasChildren,;ObjectIcon,;ObjectID,55242 /*[Funktionen\Hei
ObjectID,55953 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Sammelstoerung]*/;HasChildren,;ObjectIcon,;ObjectID,55953 /*[Funktionen\Heizur
ObjectID,26263 /*[Funktionen\Heizung\Viessmann Steuerung\ViessmannDeviceTools.inc.php]*/;HasChildren,;ObjectIcon,;ObjectID,26263 /*[Funktionen\
ObjectID,28921 /*[Funktionen\Heizung\Viessmann Steuerung\Viessmann Com Port an ]*/;HasChildren,;ObjectIcon,;ObjectID,28921 /*[Funktionen\Heizur
ObjectID,30059 /*[Funktionen\Heizung\Viessmann Steuerung\ViessmannVariables.inc.php]*/;HasChildren,;ObjectIcon,;ObjectID,30059 /*[Funktionen\He
ObjectID,31851 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen ]*/;HasChildren,1;ObjectIcon,;ObjectID,31851 /*[Funktionen\Heizung\Viessmann
ObjectID,13694 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Senden an Viessmann]*/;HasChildren,;ObjectIcon,;ObjectID,13694 /*[Funktioner
ObjectID,14385 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Modus setzen]*/;HasChildren,;ObjectIcon,;ObjectID,14385 /*[Funktionen\Heizur
ObjectID,20684 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Empfang Test 2]*/;HasChildren,;ObjectIcon,;ObjectID,20684 /*[Funktionen\Heiz
ObjectID,23210 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \setvalue test]*/;HasChildren,;ObjectIcon,;ObjectID,23210 /*[Funktionen\Heizu
ObjectID,42841 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Abfrage der Anlage ]*/;HasChildren,;ObjectIcon,;ObjectID,42841 /*[Funktioner
ObjectID,44685 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Heizungsmodus]*/;HasChildren,;ObjectIcon,;ObjectID,44685 /*[Funktionen\Heizu
ObjectID,45941 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Abfrage intervall]*/;HasChildren,;ObjectIcon,;ObjectID,45941 /*[Funktionen\H
ObjectID,54179 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Modus abfragen]*/;HasChildren,;ObjectIcon,;ObjectID,54179 /*[Funktionen\Heiz
ObjectID,59240 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Abfrage Viessmann per Com5]*/;HasChildren,1;ObjectIcon,;ObjectID,59240 /*[Fu
ObjectID,27506 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Abfrage Viessmann per Com5\]*/;HasChildren,;ObjectIcon,;ObjectID,27506 /*[Fu
ObjectID,40564 /*[Funktionen\Heizung\Viessmann Steuerung\ViessmannDeviceOutputHandlingKW.php]*/;HasChildren,;ObjectIcon,;ObjectID,40564 /*[Funk
ObjectID,42690 /*[Funktionen\Heizung\Viessmann Steuerung\Viessmann_Send]*/;HasChildren,;ObjectIcon,;ObjectID,42690 /*[Funktionen\Heizung\Viessm
ObjectID,29979 /*[Funktionen\Heizung\Letze Meldung Heizstab Einschalten]*/;HasChildren,1;ObjectIcon,;ObjectID,29979 /*[Funktionen\Heizung\Letze
ObjectID,42874 /*[Funktionen\Heizung\Letze Meldung Heizstab Einschalten\letzte Einschaltempfehlung]*/;HasChildren,;ObjectIcon,;ObjectID,42874 /
ObjectID,58711 /*[Funktionen\Heizung\Letze Meldung Heizstab Einschalten\Letzte Einschaltempfehlung]*/;HasChildren,1;ObjectIcon,;ObjectID,58711
ObjectID,57147 /*[Funktionen\Heizung\Letze Meldung Heizstab Einschalten\Letzte Einschaltempfehlung\]*/;HasChildren,;ObjectIcon,;ObjectID,57147
ObjectID,31405 /*[Funktionen\Heizung\Einschaltempfehlung]*/;HasChildren,;ObjectIcon,;ObjectID,31405 /*[Funktionen\Heizung\Einschaltempfehlung]*
ObjectID,34519 /*[Funktionen\Heizung\Modus Test]*/;HasChildren,;ObjectIcon,;ObjectID,34519 /*[Funktionen\Heizung\Modus Test]*/;ObjectIdent,;Obj
ObjectID,35625 /*[Funktionen\Heizung\Kessel Temperatur]*/;HasChildren,;ObjectIcon,;ObjectID,35625 /*[Funktionen\Heizung\Kessel Temperatur]*/;Ob
ObjectID,37401 /*[Funktionen\Heizung\Heizstab]*/;HasChildren,1;ObjectIcon,;ObjectID,37401 /*[Funktionen\Heizung\Heizstab]*/;ObjectIdent,;Object
ObjectID,13174 /*[Funktionen\Heizung\Heizstab\WORKING]*/;HasChildren,;ObjectIcon,;ObjectID,13174 /*[Funktionen\Heizung\Heizstab\WORKING]*/;Obje
ObjectID,22752 /*[Funktionen\Heizung\Heizstab\INHIBIT]*/;HasChildren,;ObjectIcon,;ObjectID,22752 /*[Funktionen\Heizung\Heizstab\INHIBIT]*/;Obje
ObjectID,23617 /*[Funktionen\Heizung\Heizstab\STATE]*/;HasChildren,;ObjectIcon,;ObjectID,23617 /*[Funktionen\Heizung\Heizstab\STATE]*/;ObjectIc
ObjectID,49266 /*[Funktionen\Heizung\Heizstab\Heizstab PV Gesteuert Leistungsüberschuss  ]*/;HasChildren,1;ObjectIcon,;ObjectID,49266 /*[Funkti
ObjectID,59370 /*[Funktionen\Heizung\Heizstab\Heizstab PV Gesteuert Leistungsüberschuss  \]*/;HasChildren,;ObjectIcon,;ObjectID,59370 /*[Funkti
ObjectID,42890 /*[Funktionen\Heizung\Heizungsmodus]*/;HasChildren,;ObjectIcon,;ObjectID,42890 /*[Funktionen\Heizung\Heizungsmodus]*/;ObjectIder
ObjectID,47310 /*[Funktionen\Heizung\Zirkulationspumpe]*/;HasChildren,;ObjectIcon,;ObjectID,47310 /*[Funktionen\Heizung\Zirkulationspumpe]*/;Ob
';
$echo = "#1001 Quell-ObjektInventory geladen";
$MessageProkoll['Info'][]  = $echo;
$Inv    = explode("\n",substr(substr($raw, 1), 1, -1)); // ersten und letzten Zeilenumbruch entfernen
return $Inv;
}

// Variablen-Profile Inventory
function VarProfileInventory()
{
global $MessageProkoll;
$raw = '
ProfileName,~HTMLBox;Digits,0;Icon,;IsReadOnly,1;MaxValue,0;MinValue,0;Prefix,;ProfileType,3;StepSize,0;Suffix,;|
ProfileName,~Temperature;Digits,1;Icon,Temperature;IsReadOnly,1;MaxValue,70;MinValue,-30;Prefix,;ProfileType,2;StepSize,5;Suffix, °C;|
ProfileName,integer_dummy;Digits,0;Icon,;IsReadOnly,;MaxValue,0;MinValue,0;Prefix,;ProfileType,1;StepSize,1;Suffix,;|
ProfileName,~Switch;Digits,0;Icon,Power;IsReadOnly,1;MaxValue,1;MinValue,0;Prefix,;ProfileType,0;StepSize,0;Suffix,;|Color,-1;Icon,;Name,Aus;Va
ProfileName,~Alert;Digits,0;Icon,Warning;IsReadOnly,1;MaxValue,1;MinValue,0;Prefix,;ProfileType,0;StepSize,0;Suffix,;|Color,-1;Icon,;Name,OK;Va
ProfileName,~String;Digits,0;Icon,;IsReadOnly,1;MaxValue,0;MinValue,0;Prefix,;ProfileType,3;StepSize,0;Suffix,;|
ProfileName,Heizungsmodus;Digits,0;Icon,;IsReadOnly,;MaxValue,0;MinValue,0;Prefix,;ProfileType,2;StepSize,0;Suffix,;|Color,-1;Icon,;Name,Abscha
';
$echo = "#1002 VariablenProfile-ObjektInventory geladen";
$MessageProkoll['Info'][]  = $echo;
$VarProfiles    = explode("\n",substr(substr($raw, 2), 0, -1)); // ersten und letzten Zeilenumbruch entfernen
return $VarProfiles;
}

// Function WFC Inventory
function loadWFCInventory()
{
global $MessageProkoll;
$raw = '
';
$echo = "#1003 WFC-ObjektInventory geladen";
$MessageProkoll['Info'][]  = $echo;
$WFC_Inventory    = explode("\n",substr(substr($raw, 2), 0, -1)); // ersten und letzten Zeilenumbruch entfernen
return $WFC_Inventory;
}

// #### Script-Exports ##########################################################################################################

// Script "ViessmannDeviceTools.inc.php" ------------------------------------------------------------------------------------------
function ViessmannDeviceToolsincphp26263()
{
return
'<?
 ##### Project Exporter Comment: Script Version Stand 30.07.2014 12:55 #####

  // Constant definition for processing ----------------------------------------
  define( "VIESSMANN_GERAETEKENNUNG",   chr(0x00).chr(0xF8) );

  include( "ViessmannVariables.inc.php");

  // Get XML
  $ViessmannDeviceID = GetValueString( VIESSMANN_VARIABLE_DEVICE );
```

```
$DeviceXMLName = IPS_GetKernelDir()."scripts@§@@§@ViessmannDevice".$ViessmannDeviceID.".xml";
try
{
  $ViessmannDeviceXML = simplexml_load_file( $DeviceXMLName );
}
catch (Exception $e)
{
  $ViessmannDeviceXML = "";
}

function ViessmannGetCommand( $Command )
{
  $CommandData = array();

  global $ViessmannDeviceXML;
  if ( !isset( $ViessmannDeviceXML ) OR $ViessmannDeviceXML == "" )
    return false;

  $CommandData["deviceid"] = $ViessmannDeviceXML["id"];
  $CommandData["protocol"] = $ViessmannDeviceXML["protocol"];

  // Get Command details
  $ViessmannDeviceCommands = $ViessmannDeviceXML->commands[0];

  foreach($ViessmannDeviceXML->commands[0] as $a => $XMLCommand )
  {
    if ( $XMLCommand["name"] == $Command )
    {
      $CommandData["name"]        = $Command;
      $CommandData["action"]      = $XMLCommand["action"];
      $CommandData["address"]     = $XMLCommand->address[0];
      $CommandData["length"]      = $XMLCommand->length[0];
      $CommandData["description"] = $XMLCommand->description[0];
      $CommandData["format"]      = $XMLCommand->format[0];
      break; // for
    }
  }
  if ( isset( $CommandData["name"] ) )
    return $CommandData;
  else
    return false;
}

function ViessmannResultToHexDisplay( $Result )
{
  $ResultConv = "";
  for( $x=0; $x < strlen($Result); $x++)
  {
    if ( ord($Result[$x]) < 16 ) $ResultConv .= "0";
    $ResultConv .= strtoupper(dechex(ord($Result[$x])));
  }
  return $ResultConv;
}

function ViessmannConvertData( $Command, $Data )
{
    global $ViessmannDeviceID;
    $ViessmannCommand = ViessmannGetCommand( $Command, $ViessmannDeviceID );
    if ( $ViessmannCommand === false ) return false;
    $Format = $ViessmannCommand["format"];

    switch( $Format )
    {
         case "Boolean":
          if ( ord( $Data[0] ) == 0 )
             return false;
             else
                 return true;
            break;
        case "Temperature10":
            $lowByte = ord( $Data[0] );
            $highByte = ord( $Data[1] );
            return (( $highByte * 256 ) + $lowByte ) / 10;
            break;
    case "Temperature10Minus":
      $lowByte = ord( $Data[0] );
      $highByte = ord( $Data[1] );
      if ($highByte == 255) {
        return ((65536-(( $highByte * 256 ) + $lowByte ))/10)*-1;
        } else {
        return (( $highByte * 256 ) + $lowByte ) / 10;
      }
      break;
        case "Temperature100":
            $lowByte = ord( $Data[0] );
            $highByte = ord( $Data[1] );
            return (( $highByte * 256 ) + $lowByte ) / 100;
            break;
        case "Count" or "Mode":
            $Count = 0;
            $HexString = ViessmannResultToHexDisplay( $Data );
      for ( $x=0; $x < strlen( $HexString ); $x=$x+2 )
            {
               $hex = substr( $HexString, $x,2 );
               $dec = hexdec( $hex );
               switch ( $x )
               {
                  case 0:
                    $Count = $Count + $dec;
                    break;
                  case 2:
                    $Count = $Count + ( $dec * 256 );
                    break;
                  case 4:
                    $Count = $Count + ( $dec * 65536 );
```

```
                    break;
                  case 6:
                    $Count = $Count + ( $dec * 16777216 );
                    break;
                  default:
                    break;
                }
              }
              return $Count;
              break;
            case "BooleanPercent":
              return "BOOLEANPERCENT convertion not coded yet.";
              break;
            case "Triple":
              return "TRIPLE convertion not coded yet.";
              break;
            case "Mode":
              return hexdec(ord($Data[0]));
              break;
          case "ControlID":
            $Result = "";
          for( $x=0; $x < strlen($Data); $x++)
          {
            if ( ord($Data[$x]) < 16 ) $Result .= "0";
            $Result .= strtoupper(dechex(ord($Data[$x])));
          }
          return $Result;
          break;
            default:
              return "";
              break;
      }
      return $ConvertedResult;
}

function ViessmannGetData( $Command )
{
   $Run = 3; // Maximum time a recheck is done, if Ping (0x05 is returnvalue)
   // get Command Data
    $ViessmannCommand = ViessmannGetCommand( $Command );
   if ( $ViessmannCommand === false ) return false;

   // clear previous result
   do
   {
     SetValueString( VIESSMANN_VARIABLE_RESULT, "" );
     // create command string and send to COM Port
     $CommandString = "R:".$ViessmannCommand["protocol"].":".$ViessmannCommand["address"].":".$ViessmannCommand["length"].": ";
      $StartTime = microtime(true);
      $RunTime = 0;
      SetValueString( VIESSMANN_VARIABLE_COMMAND, $CommandString );
     // wait for execution
    $status = getvalue( VIESSMANN_VARIABLE_COMMAND );
    //IPS_LogMessage("ViessmannDT",$StartTime);
    while ($status != "" AND $RunTime < 10 )
      {
        $status = getvalue( VIESSMANN_VARIABLE_COMMAND );
        $RunTime = microtime(true) - $StartTime;
      }
    // get result
    $Result = GetValue( VIESSMANN_VARIABLE_RESULT );
    if ( $Result != chr(0x05) ) $Run = -1;
    $Run--;
   } while ( $Run > 0 );
 //IPS_LogMessage("ViessmannDT", $StartTime."-".bin2hex($Result)."-".$CommandString."-".$Run."-".$RunTime."-".$status);
 if (strlen($Result)==0) {
   IPS_LogMessage("ViessmannDT", "Timeout:".$RunTime."(".$Run.")");
   SetValue(VIESSMANN_VARIABLE_ERRORCOUNT,GetValue(VIESSMANN_VARIABLE_ERRORCOUNT)+1);
   $Result = false;
    ViessmannClose();
    ViessmannOpen();
  }
  else if ( $Result[0] == chr(0x05) ) {
   IPS_LogMessage("ViessmannDT", bin2hex($Result)."-".$CommandString."-".$Run);
   if (strlen($Result)==2) {
     if ($Result[1] == chr(0x05)) {
       IPS_LogMessage("ViessmannDT", "skipped");
       SetValue(VIESSMANN_VARIABLE_ERRORCOUNT,GetValue(VIESSMANN_VARIABLE_ERRORCOUNT)+1);
       $Result = false;
    }
   }
 }
 if (strlen($Result)==4) {
   if (($Result[2] == chr(0x05))&&($Result[3] == chr(0x05))) {
     IPS_LogMessage("ViessmannDT", "skipped4-".bin2hex($Result));
     SetValue(VIESSMANN_VARIABLE_ERRORCOUNT,GetValue(VIESSMANN_VARIABLE_ERRORCOUNT)+1);
     $Result = false;
   }
 }
   return $Result;
}

function ViessmannSetData( $Command, $Data )
{
   // get Command Data
    $ViessmannCommand = ViessmannGetCommand( $Command );
   if ( $ViessmannCommand === false ) return false;
   if ( $ViessmannCommand["action"] != "ReadWrite" ) return false;
   // clear previous result
  SetValueString( VIESSMANN_VARIABLE_RESULT, "" );
  // create command string and send to COM Port
     $CommandString = "W:".$ViessmannCommand["protocol"].":".$ViessmannCommand["address"].":".$ViessmannCommand["length"].":".$Data;
  SetValueString( VIESSMANN_VARIABLE_COMMAND, $CommandString );
  // wait for execution
  $status = getvalue( VIESSMANN_VARIABLE_COMMAND );
```

```
  while ($status != "") $status = getvalue( VIESSMANN_VARIABLE_COMMAND );
  // get result
  $Result = GetValue( VIESSMANN_VARIABLE_RESULT );
  return $Result;
}

function ViessmannOpen()
{
  // Open Com Port
  if ( COMPort_GetOpen( VIESSMANN_VARIABLE_COMPORT ) != true )
  {
     COMPort_SetOpen( VIESSMANN_VARIABLE_COMPORT, true );
     IPS_ApplyChanges( VIESSMANN_VARIABLE_COMPORT );
  }
}

function ViessmannClose()
{
  // Close Com Port
  if ( COMPort_GetOpen( VIESSMANN_VARIABLE_COMPORT ) != false )
  {
     COMPort_SetOpen( VIESSMANN_VARIABLE_COMPORT, false );
     IPS_ApplyChanges( VIESSMANN_VARIABLE_COMPORT );
  }
}

function ViessmannSetVariableByCommand( $Command, $Var )
{
  $Value = ViessmannGetData( $Command );
  if ( $Value === false )
   {
     SetValueBoolean( VIESSMANN_VARIABLE_CONNECTOK, false );
   }
  else
  {
    SetValueBoolean( VIESSMANN_VARIABLE_CONNECTOK, true );
   if ($Var) SetValue( $Var, ViessmannConvertData( $Command, $Value ) );
   else return (ViessmannConvertData( $Command, $Value ));
   }
}

?>
';
}

// Script "Viessmann Com Port an " ---------------------------------------------------------------------------------------------------------
function ViessmannComPortan28921()
{
return
'<?
 ##### Project Exporter Comment: Script Version Stand 30.07.2014 12:55 #####

IPS_SetProperty(53482 /*[Viessmann Heizung Serial Port]*/, §\'Open§\', true);
IPS_ApplyChanges(53482 /*[Viessmann Heizung Serial Port]*/ );
// COMPort_SetDTR(53482 /*[Viessmann Heizung Serial Port]*/, true);
IPS_Sleep(5000);
IPS_SetProperty(53482 /*[Viessmann Heizung Serial Port]*/, §\'Open§\', false);
IPS_ApplyChanges(53482 /*[Viessmann Heizung Serial Port]*/ );
?>
';
}

// Script "ViessmannVariables.inc.php" -----------------------------------------------------------------------------------------------------
function ViessmannVariablesincphp30059()
{
return
'<?
 ##### Project Exporter Comment: Script Version Stand 30.07.2014 12:55 #####


  // IPS Variables ------------------------------------------------------------
  define( "VIESSMANN_VARIABLE_DEVICE"      , 57385 /*[Viessmann Heizung Serial Port\Viessmann Register Variable\Steuerungstyp]*/ );
  define( "VIESSMANN_VARIABLE_COMMAND"     , 58616 /*[Viessmann Heizung Serial Port\Viessmann Register Variable\Kommando]*/ );
  define( "VIESSMANN_VARIABLE_RESULT"      , 42628 /*[Viessmann Heizung Serial Port\Viessmann Register Variable\Rückgabewert]*/ );
  define( "VIESSMANN_VARIABLE_REGISTER"    , 56186 /*[Viessmann Heizung Serial Port\Viessmann Register Variable]*/ );
  define( "VIESSMANN_VARIABLE_COMPORT"     , 53482 /*[Viessmann Heizung Serial Port]*/ );
  define( "VIESSMANN_VARIABLE_CONNECTOK"   , 49172 /*[Viessmann Heizung Serial Port\Viessmann Register Variable\Verbindung ok]*/ );
    define( "VIESSMANN_VARIABLE_ERRORCOUNT"   , 35796 /*[Viessmann Heizung Serial Port\Viessmann Register Variable\ErrorCount]*/);
?>
';
}

// Script "Senden an Viessmann" ------------------------------------------------------------------------------------------------------------
function SendenanViessmann13694()
{
return
'<?
 ##### Project Exporter Comment: Script Version Stand 30.07.2014 12:55 #####

if($IPS_SENDER == "WebFront")

{
    include( "ViessmannDeviceTools.inc.php" );

    switch($IPS_VALUE)
    {

          case 0:

                {
                     ViessmannOpen();
                     ViessmannSetData( "BetriebsartM2", chr(0x00) );
                     //IPS_Sleep(2000);
                     ViessmannClose();
                     SetValue(44685 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Heizungsmodus]*/, $IPS_VALUE);
```

```
                    SetValue($IPS_VARIABLE, $IPS_VALUE);
                }

        break;

                case 1:

                {
                    ViessmannOpen();
                    ViessmannSetData( "BetriebsartM2", chr(0x01) );
                    //IPS_Sleep(2000);
                    ViessmannClose();
                    SetValue(44685 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Heizungsmodus]*/, $IPS_VALUE);
                    SetValue($IPS_VARIABLE, $IPS_VALUE);
                 }
            break;

            case 2:

               {
                    ViessmannOpen();
                    ViessmannSetData( "BetriebsartM2", chr(0x02) );
                    //IPS_Sleep(2000);
                    ViessmannClose();
                    SetValue(44685 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Heizungsmodus]*/ ,$IPS_VALUE);
                    SetValue($IPS_VARIABLE, $IPS_VALUE);
                 }
            break;

            case 3:

               {
                    ViessmannOpen();
                    ViessmannSetData( "BetriebsartM2", chr(0x03) );
                    //IPS_Sleep(2000);
                    ViessmannClose();
                    SetValue(44685 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Heizungsmodus]*/ , $IPS_VALUE);
                    SetValue($IPS_VARIABLE, $IPS_VALUE);
                 }
            break;

            case 4:

               {
                    ViessmannOpen();
                    ViessmannSetData( "BetriebsartM2", chr(0x04) );
                    //IPS_Sleep(2000);
                    ViessmannClose();
                    SetValue(44685 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Heizungsmodus]*/ , $IPS_VALUE);
                    SetValue($IPS_VARIABLE, $IPS_VALUE);

                 }
            break;

}}
?>
';
}

// Script "Modus setzen" ---------------------------------------------------------------------------------------------------------
function Modussetzen14385()
{
return
'<?
 ##### Project Exporter Comment: Script Version Stand 30.07.2014 12:55 #####


include( "ViessmannDeviceTools.inc.php" );
ViessmannOpen();
ViessmannSetData( "BetriebsartM2", chr(0x04) );// 00 aus 01 ww 02 WWheizen 03 reduziert 04 normal
ViessmannClose();

?>
';
}

// Script "Empfang Test 2" -------------------------------------------------------------------------------------------------------
function EmpfangTest20684()
{
return
'<?
 ##### Project Exporter Comment: Script Version Stand 30.07.2014 12:55 #####

// Variablen Includieren
include( "ViessmannDeviceTools.inc.php" );

// Verbindung zur Heizung öffnen
ViessmannOpen();

// Wert Abfragen
ViessmannSetVariableByCommand ( "BetriebsartM2",42890 /*[Funktionen\Heizung\Heizungsmodus]*/  );

// Verbindung schliessen
ViessmannClose();
?>
';
}

// Script "setvalue test" --------------------------------------------------------------------------------------------------------
function setvaluetest23210()
{
return
'<?
 ##### Project Exporter Comment: Script Version Stand 30.07.2014 12:55 #####
```

```
SetValuefloat(42890 /*[Funktionen\Heizung\Heizungsmodus]*/ , 5);

?>
';
}

// Script "Abfrage der Anlage " ---------------------------------------------------------------------------------------------------------
function AbfragederAnlage42841()
{
return
'<?
 ##### Project Exporter Comment: Script Version Stand 30.07.2014 12:55 #####


  // Variablen Includieren
  include( "ViessmannDeviceTools.inc.php" );

  // Verbindung zur Heizung öffnen
  ViessmannOpen();

  // Wert Abfragen

ViessmannSetVariableByCommand( "BetriebsartM2", 18705 /*[Objekt #18705 existiert nicht]*/ );
//ViessmannSetVariableByCommand( "Speichertemperatur", 58240 /*[Objekt #58240 existiert nicht]*/ );



  // Verbindung schliessen
  ViessmannClose();

?>
';
}

// Script "Modus abfragen" --------------------------------------------------------------------------------------------------------------
function Modusabfragen54179()
{
return
'<?
 ##### Project Exporter Comment: Script Version Stand 30.07.2014 12:55 #####


include( "ViessmannDeviceTools.inc.php" );
ViessmannOpen();
ViessmannSetVariableByCommand( "BetriebsartM2",42890 /*[Funktionen\Heizung\Heizungsmodus]*/ );
ViessmannClose();

?>
';
}

// Script "Abfrage Viessmann per Com5" --------------------------------------------------------------------------------------------------
function AbfrageViessmannperCom59240()
{
return
'<?
 ##### Project Exporter Comment: Script Version Stand 30.07.2014 12:55 #####

include( "ViessmannDeviceTools.inc.php" );
$step = GetValue(45941 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Abfrage intervall]*/ );

if(($IPS_SENDER == "TimerEvent")and($step == 0)){
    SetValue(45941 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Abfrage intervall]*/ , 1);
//IPS_SetScriptTimer($IPS_SELF, 10);
        ViessmannOpen();
        ViessmannSetVariableByCommand( "BetriebsartM2",42890 /*[Funktionen\Heizung\Heizungsmodus]*/ );
        ViessmannSetVariableByCommand( "Aussentemperatur",45011 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\AussentemperaturGedaempft]*/
        ViessmannSetVariableByCommand( "Vorlauftemperatur",22474 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Vorlauftemperatur]*/ );
        ViessmannSetVariableByCommand( "Sammelstoerung",55953 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Sammelstoerung]*/);
        ViessmannSetVariableByCommand( "Brennerstoerung",41873 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Brennerstörung]*/ );
        ViessmannSetVariableByCommand( "PartybetriebA1M1",34221 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Partybetrieb]*/);
          ViessmannSetVariableByCommand( "Ruecklauftemperatur17A",36927 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Ruecklauftemperatur17

        ViessmannClose();
    return;
}
if(($IPS_SENDER == "TimerEvent")and($step == 1)){
    SetValue(45941 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Abfrage intervall]*/ , 2);
//IPS_SetScriptTimer($IPS_SELF, 10);
        ViessmannOpen();
        ViessmannSetVariableByCommand( "Kesseltemperatur",28911 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Kesseltemperatur]*/ );
        ViessmannSetVariableByCommand( "Speicherladepumpe",55242 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Speicherladepumpe]*/ );
        ViessmannSetVariableByCommand( "Zirkulationspumpe",38089 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Zirkulationspumpe]*/);
        ViessmannSetVariableByCommand( "AktuelleBetriebsartA1M",28759 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\AktuelleBetriebsartA1M]
        ViessmannSetVariableByCommand( "SparbetriebM2",33016 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Sparbetrieb]*/);
        ViessmannSetVariableByCommand( "Kesselsolltemperatur",34515 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Kesselsolltemperatur]*/);
        ViessmannClose();
    return;
}

if(($IPS_SENDER == "TimerEvent")and($step == 2)){
    SetValue(45941 /*[Funktionen\Heizung\Viessmann Steuerung\Abfragen \Abfrage intervall]*/ , 0);
// IPS_SetScriptTimer($IPS_SELF, 10);
        ViessmannOpen();
        ViessmannSetVariableByCommand( "AnlagenIstLeistung",55063 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\AnlagenIstLeistung]*/ );
        ViessmannSetVariableByCommand( "RaumSolltemperatur",20154 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Raum Soll Temperatur]*/);
        ViessmannSetVariableByCommand( "AktuelleBetriebsartA1M1",28759 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\AktuelleBetriebsartA1M
        ViessmannSetVariableByCommand( "BetriebsartM2",42890 /*[Funktionen\Heizung\Heizungsmodus]*/ );
        ViessmannSetVariableByCommand( "Brennerstufe",31685 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Brennerstufe]*/ );
        ViessmannSetVariableByCommand( "VorlauftemperaturM2",22474 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Vorlauftemperatur]*/ );
        ViessmannSetVariableByCommand( "Uhrzeit-m",23092 /*[Funktionen\Heizung\Viessmann Steuerung\Ausgabe\Uhrzeit]*/);

        ViessmannClose();
   return;
```

```
}
?>
';
}

// Script "ViessmannDeviceOutputHandlingKW.php" ---------------------------------------------------------------------------------------------------
function ViessmannDeviceOutputHandlingKWphp40564()
{
return
'<?
 ##### Project Exporter Comment: Script Version Stand 30.07.2014 12:55 #####

  include( "ViessmannVariables.inc.php");
  // Constants
  define( "VIESSMANN_PING",             chr(0x05) );
  define( "VIESSMANN_KW_READ_PRAEFIX",  chr(0x01).chr(0xF7) );
  define( "VIESSMANN_KW_WRITE_PRAEFIX", chr(0x01).chr(0xF4) );

  // init variables
  $CurrentAction   = ""; $CurrentProtocol = ""; $CurrentCommand  = "";
  $CurrentLength   = ""; $CurrentSetValue = "";

  $ViessmannDeviceID = GetValueString( VIESSMANN_VARIABLE_DEVICE );

  // get current Command if available or determine Device ID automatically
  $Command = GetValueString( VIESSMANN_VARIABLE_COMMAND );
  if ( $ViessmannDeviceID == "" AND strlen( $Command ) == 0 )
  {
     $Command = "R:KW:00F8:2: "; // retrieve Device ID
     SetValueString( VIESSMANN_VARIABLE_COMMAND, $Command );
  }

  if ( $Command != "" )
  {
    $CommandArray = explode(":", $Command);
    $parts = count( $CommandArray );
    if ( $parts > 0 ) $CurrentAction   = $CommandArray[0];
    if ( $parts > 1 ) $CurrentProtocol = $CommandArray[1];
    if ( $parts > 2 ) $CurrentAddress  = $CommandArray[2];
    if ( $parts > 3 ) $CurrentLength   = $CommandArray[3];
    if ( $parts > 4 ) $CurrentSetValue = $CommandArray[4];
  }

  // Get remaining data in the com port via the buffer of the register variable
  $ComData = RegVar_GetBuffer( VIESSMANN_VARIABLE_REGISTER );
  // and merge it together with the passed data
  $ComData .= $IPS_VALUE;
//echo $CurrentAction."-";
//if ($ComData == VIESSMANN_PING ) echo "Ping"; else echo strlen( $ComData );
  // Process Com Data ------------------------------------------------------------
  if (   $ComData == VIESSMANN_PING AND
         ( $CurrentAction == "R" OR $CurrentAction == "W" ) )
  {
    // No data expected (Command available), so execute command if available

    // Check, if expected Action and Command capabilities fit

    // Prepare Command Address + Length
    $Address =  chr(hexdec(substr($CurrentAddress,0,2)));
    $Address .= chr(hexdec(substr($CurrentAddress,2,2)));
    switch ( $CurrentAction )
    {
      case "R": // Read
        $CommandString = VIESSMANN_KW_READ_PRAEFIX.$Address.chr($CurrentLength);
        $Command[0] = "-"; // Mark Command as executed
        break;
      case "W": // Write
        $CommandString = VIESSMANN_KW_WRITE_PRAEFIX.$Address.chr($CurrentLength).$CurrentSetValue;
        $Command = ""; // Clear command
        break;
      default:
        $CommandString = "";
        break;
    }
    if ( $CommandString != "" )
    {
      SetValueString( VIESSMANN_VARIABLE_COMMAND, $Command );
       // Clear current result
      SetValueString( VIESSMANN_VARIABLE_RESULT, "" );
        // Send command to Viessmann heating device via COM Port
        COMPort_SendText( VIESSMANN_VARIABLE_COMPORT, $CommandString );
    }
  }
  else
  {
      if ( $CurrentAction == "-" and strlen( $ComData ) >= $CurrentLength )
      {
          // Com Data is equal or longer as needed
          $Result = substr( $ComData, 0, $CurrentLength );
          // Set Result in IPS Variable
        SetValueString( VIESSMANN_VARIABLE_RESULT, $Result );

        // if Device ID is initial and command was to retrieve device ID
        if ( $ViessmannDeviceID == "" AND $Command == "-:KW:00F8:2: " )
        {
             // set device ID
             $DeviceID = "";
          for( $x=0; $x < strlen($Result); $x++)
          {
            if ( ord($Result[$x]) < 10 ) $DeviceID .= "0";
            $DeviceID .= strtoupper(dechex(ord($Result[$x])));
          }
          SetValueString( VIESSMANN_VARIABLE_DEVICE, $DeviceID );
        }
```

```
        // clear command and remaining buffer of register variable
            SetValueString( VIESSMANN_VARIABLE_COMMAND, "" );
        RegVar_SetBuffer( VIESSMANN_VARIABLE_REGISTER, "" );
      }
      else
      {
        // as a result is expected and the current length of data is not
        // sufficient, write the data back to the buffer of the register var
        if ( $Command != "" AND $CurrentAction == "-" )
          RegVar_SetBuffer( VIESSMANN_VARIABLE_REGISTER, $ComData);
          else
            RegVar_SetBuffer( VIESSMANN_VARIABLE_REGISTER, "");
    }

  }
?>
';
}

// Script "Viessmann_Send" ---------------------------------------------------------------------------------------------------
function ViessmannSend42690()
{
return
'<?
 ##### Project Exporter Comment: Script Version Stand 30.07.2014 12:55 #####

/*require IPS_GetKernelDir()."scripts@§@@§@ViessmannVariablen.ips.php";
include "globals.ips.php";
COMPort_SendText(Viessmann, IPS_Geraete_Kennung );
*/
// Variablen Includieren
include( "ViessmannDeviceTools.inc.php" );

  // Verbindung zur Heizung öffnen
//  ViessmannOpen();

  // Wert Abfragen
//ViessmannSetVariableByCommand( "Uhrzeit-m", 54552 /*[Objekt #54552 existiert nicht]*/ );
ViessmannSetVariableByCommand( "Kesseltemperatur", 40940 /*[Objekt #40940 existiert nicht]*/ );
$temp=ViessmannSetVariableByCommand( "Kesselsolltemperatur", 0 );
if ($temp<100) SetValue(18210 /*[Objekt #18210 existiert nicht]*/,$temp);
else {
  IPS_LogMessage("Viessmann_Send", "skipped Kesselsoll");
  SetValue(VIESSMANN_VARIABLE_ERRORCOUNT,GetValue(VIESSMANN_VARIABLE_ERRORCOUNT)+1);
}
//ViessmannSetVariableByCommand( "Aussentemperatur", 53852 /*[Objekt #53852 existiert nicht]*/ );
ViessmannSetVariableByCommand( "AbgastemperaturTiefpass", 42042 /*[Objekt #42042 existiert nicht]*/ );
//ViessmannSetVariableByCommand( "ZustandStufe1", 44201 /*[Objekt #44201 existiert nicht]*/ );
ViessmannSetVariableByCommand( "AnlagenIstLeistung", 23970 /*[Objekt #23970 existiert nicht]*/ );
//ViessmannSetVariableByCommand( "Brennerstoerung", 44754 /*[Objekt #44754 existiert nicht]*/ );
$temp=ViessmannSetVariableByCommand( "BetriebsstundenStufe1", 0 );
$temp2=GetValue(56842 /*[Objekt #56842 existiert nicht]*/);
if (($temp>=$temp2) &&($temp<$temp2+3600*24)) SetValue(56842 /*[Objekt #56842 existiert nicht]*/,$temp);
else {
  IPS_LogMessage("Viessmann_Send", "skipped StundenStufe1-".$temp."-".$temp2);
  SetValue(VIESSMANN_VARIABLE_ERRORCOUNT,GetValue(VIESSMANN_VARIABLE_ERRORCOUNT)+1);
}

ViessmannSetVariableByCommand( "BetriebsstundenStufe2", 12263 /*[Objekt #12263 existiert nicht]*/ );
ViessmannSetVariableByCommand( "Brennerstarts", 25035 /*[Objekt #25035 existiert nicht]*/ );

  // Verbindung schliessen
//  ViessmannClose();


?>
';
}

// Script "Letzte Einschaltempfehlung" ---------------------------------------------------------------------------------------------------
function LetzteEinschaltempfehlung58711()
{
return
'<?
 ##### Project Exporter Comment: Script Version Stand 30.07.2014 12:55 #####

$id_status = 31405 /*[Funktionen\Heizung\Einschaltempfehlung]*/;
$wert= getvalueboolean(31405 /*[Funktionen\Heizung\Einschaltempfehlung]*/);

echo $wert;
?>
';
}

// Script "Heizstab PV Gesteuert Leistungsüberschuss  " ---------------------------------------------------------------------------------------------------
function HeizstabPVGesteuertLeistungsberschuss49266()
{
return
'<?
 ##### Project Exporter Comment: Script Version Stand 30.07.2014 12:55 #####

$PV=getvalue(11369 /*[Funktionen\PV Anlage\AC Aktuell]*/); // Besser beiPV - Verbrauch zu schalten
// $PV=0; // Besser beiPV - Verbrauch zu schalten
$Verbrauchaktuell = getvalue(21094 /*[Funktionen\Verbrauch aktuell\*\Gesamt]*/);
$Delta=$PV - $Verbrauchaktuell;
if ($Delta < -1 )
{
echo "kleiner Null wir kaufen  "; // Grundlast sind immer 1-2 Kw also hier lieber nicht schalten da Heizstab 2-4 KW hat
echo "@§@n";
echo $Delta;
setvalue(31405 /*[Funktionen\Heizung\Einschaltempfehlung]*/ ,false);
}
if ($Delta > 1  )
{
echo "mehr als 1 Watt und Positiv "; // Grundlast sind immer 1-2 Kw also hier lieber nicht schalten da Heizstab 2-4 KW hat
```

```
echo "@§@n";
echo $Delta;
setvalue(31405 /*[Funktionen\Heizung\Einschaltempfehlung]*/ ,false);
}
if ($Delta > 1001 && $Delta <= 2000)
{
echo "1000 Watt Über"; // an Wochenenden und Feiertagen also Betriebsfreie zeit zu überlegen
echo "@§@n";
echo $Delta;
}
if ($Delta > 2000  && $Delta < 4000 )
{
echo "mehr als 2000 Watt über"; // Ab hier Heizstab Nutzbar
echo "@§@n";
echo $Delta;
setvalue(31405 /*[Funktionen\Heizung\Einschaltempfehlung]*/ ,true);
}
?>
';
}

?>
```

Ich hoffe ich kenn einigen damit eine menge Arbeit abnehmen . Ganz großen Dank aber an Raketenschnecke für diesen Exporter , der das Exportieren erst möglich gemacht hat.

ika

30.07.14, 17:21

Hi Marcel,

ich werde erst im Laufe der kommenden Wochen eine Viessmann-Heizung in Betrieb nehmen.
Trotzdem an dieser Stelle schonmal vielen Dank für deinen Export! Ich denke, das könnte mir deutlich schneller aufs Pferd helfen ;-)

Gruß,
Michael

Fraunhofer

05.08.14, 14:53

Partymodus setzen

Moin Moin Jungs,

hat es einer geschafft den Partymodus zu setzen ? Ich sende das an die Anlage aber bei Abfrage kommt immer Partymodus ist aus :(

kronos

05.08.14, 15:53

Funktioniert bei mir. Anlagenkennung ist bei mir die 2094. Allerdings können Spar- und Party-Modus nicht gleichzeitig aktiv sein.

Fraunhofer

05.08.14, 16:45

Hallo Kronos ,

die Anlagenkennung ist 52000 ,,, das ist ein Integer Wert ich hoffe das passt so , denn die Gerätekennung habe ich nicht abgefragt.

Kannst du damit was anfangen ?

kronos

05.08.14, 17:06

Zitat:

> *hat es einer geschafft den Partymodus zu setzen ? Ich sende das an die Anlage aber bei Abfrage kommt immer Partymodus ist aus*

Nur mal um die Frage zu klären. Du sendest eine Befehlssequenz für den Partymodus an die Anlage. Schaltet Sie um oder nicht? Man könnte Deine Frage so intepretieren, dass nur das Zurücklesen des Status das Problem darstellt.

Fraunhofer

05.08.14, 17:14

Hi Kronos ,

ich bin morgen auf der Baustelle und guck mir das mal vor Ort an , ich hatte versucht den Status zu setzen , kann aber dann nur per abfrage feststellen das er diesen nicht gesetzt hat , ich konnte auch mit dem Viessmann Tool 2.5 den Partymodus nicht setzen , daher muss ich mal gucken was passiert wenn ich vor ort den Status an der Anlage setze und dann abfrage . Dein Tip mit dem Sparmodus ist auf jeden Fall Gold wert denn das wird ja ein elementares Problem sein .

Das werde ich dir morgen berichten .

Vielen Dank für die schnelle Antwort

kronos 05.08.14, 17:18

Es dauert (zumindest bei meiner Anlage) auch immer eine gewisse Zeit (5-60 Sekunden) bis er die Änderungen an der Anlage anzeigt. :rolleyes:

Mir war es während der Staun-und-Lern-Phase bis das Skript lief irgendwann zu blöd mit dem Laptop im Heizungskeller zu sitzen bzw. dauernd die Treppen zu rennen. Ich hatte da dann temporär eine Webcam hin gestellt und konnte ganz bequem von der Küchentheke aus prüfen was die Viessmann da so treibt.

Fraunhofer 05.08.14, 21:40

Hi Kronos,

du willst wohl das mich meine Frau mich entmündigen lässt was :)

kronos 06.08.14, 08:03

Wenn Sie das schon wegen der Installation einer Webcam in Betracht zieht, hast Du ganz andere Probleme....:p

Wir sehen uns dann in der Klapse.

Fraunhofer 07.08.14, 08:33

Moin Moin Jungs,

ich war also gestern auf der Baustelle , ich kann mit der App am Apfel und an der Anlage den Partymodus einschalten mit der IPS Abfrage kommt allerdings ,auch nach 20 min. ,dieser Vorgang nicht an . auch mit dem Viessmann Tool sagt er mir das der Partymodus "Aus" ist . Ich habe dazu keine Idee mehr.

kronos 07.08.14, 09:15

Dann passt entweder die Befehlssequenz die Du an die Viessmann übermittelst nicht zu der Steuerung oder Dein Skript schickt nicht das was es senden sollte.

Nachdem aber das Tool auch nicht funktioniert tippe ich mal auf die Befehls-Sequenz. Bekommt Dein Skript denn den Party-Modus mit wenn Du es an der App/Heizung umstellst?

Fraunhofer 07.08.14, 09:55

Moin Kronos ,

das ist es ja , der bekommt das garnicht mit auch dieses Viesmann Tool v-control1_2_5 , zeigt mir an das der Partymodus "Aus" ist keine Ahnung was da abgeht. Aber ich denk mit dem Iphone kann man das auch mal schnell schalten. Das Viessmann App ist ja schon genial einfach für sowas. Nur eben schade das unser IPS das nicht macht.

PumpkinEater 13.01.15, 12:13

Hallo zusammen,
ich muss meine Heizung irgendwie remote an den entfernt stehenden IPS-Server anbinden. Dazu steht an der Heizung bereits ein "vernetzter" Raspberry. Macht es Sinn, anstelle von ser2net den vcontrold auf dem Raspberry laufen zu lassen? Dieser stellt ja bereits eine Socket-Schnittstelle zur Verfügung. Somit könnte ich die Skripte dann auf dem IPS deutlich vereinfachen, da die Device-abhängige Kodierung der Befehle bereits durch vcontrold gemacht wird.

Gruß
Peter

hcp 13.01.15, 20:33

Zitat:

> *Zitat von **PumpkinEater***
>
> *Hallo zusammen,*
> *ich muss meine Heizung irgendwie remote an den entfernt stehenden IPS-Server anbinden. Dazu steht an der Heizung bereits ein "vernetzter" Raspberry. Macht es Sinn, anstelle von ser2net den vcontrold auf dem Raspberry laufen zu lassen? Dieser stellt ja bereits eine Socket-Schnittstelle zur Verfügung. Somit könnte ich die Skripte dann auf dem IPS deutlich vereinfachen, da die Device-abhängige Kodierung der Befehle bereits durch vcontrold gemacht wird.*
>
> *Gruß*
> *Peter*

Genau so mache ich das seit längerer Zeit. Habe die IPS-Skripte trotz vieler Versuche nie zuverlässig zum Laufen gebracht. Dies insbesondere beim aktiven Schalten der Betriebszustände. Mit dem Raspberry und vcontrold lief es dann recht schnell recht rund :-)

Fraunhofer 14.01.15, 10:53

Moin Moin,

Du schaltest dann am IPS ,, der dann den RASPI bedient ?

40 Beiträge dieses Themas auf einer Seite anzeigen

Seite 4 von 8 Erste ... 2 3 4 5 6 ... Letzte