![Schneider Electric logo]

★ ★ ★

Basic / **Advanced** / Expert

Application note

Modbus Power meters and spaceLYnk

Reading basic and complete Modbus parameters from Acti9 Power meters via LUA scripts

![Schneider Electric logo]

Safety Information

Important Information

Read these instructions carefully before trying to install, configure, or operate this software. The following special messages may appear throughout this bulletin or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of either symbol to a "Danger" or "Warning" safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠DANGER
DANGER indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.

⚠WARNING
WARNING indicates a potentially hazardous situation which, if not avoided, can result in death or serious injury.

⚠CAUTION
CAUTION indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

NOTICE
NOTICE is used to address practices not related to physical injury. The safety alert symbol shall not be used with this signal word.

Schneider Electric

Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction, installation, and operation of electrical equipment and has received safety training to recognize and avoid the hazards involved.

Safety Precautions

⚠WARNING
HAZARD OF INCORRECT INFORMATION • Do not incorrectly configure the software, as this can lead to incorrect reports and/or data results. • Do not base your maintenance or service actions solely on messages and information displayed by the software. • Do not rely solely on software messages and reports to determine if the system is functioning correctly or meeting all applicable standards and requirements. • Consider the implications of unanticipated transmission delays or failures of communications links. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Schneider Electric

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information that is contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

Schneider Electric

Table of Contents

1 Introduction

This application note describes Lua scripts which can read parameters from different Acti9 Modbus power meters via spaceLYnk controller. Provided scripts can be used for easy configuration of power metering in new or existing installations with one or more power meters.

Customer value proposition

The customer value propositions correspond to real use cases to connect spaceLYnk and Acti9 power meters. The power meters are used to measure electrical parameters of whole installation or just a part of it.

Power meters provide:

- Installation monitoring
- Alarming and consumption drifts
- Consumption monitoring
- Evaluation of energy items (cost, accounting, etc.)
- Logging of historical values of consumption
- Identifying harmonic disturbances

These electrical parameters are stored in registers and can be read or modified by spaceLYnk. After reading from an appropriate register, this value will be converted to standard KNX data point type.

In this context here are some identified possible use cases which can be implemented with this application note.

- Use case 1: One power meter is connected to spaceLYnk
    - o How to restore spaceLYnk with list of KNX objects to read basic and complete registers
- Use case 2: One power meter is connected to spaceLYnk
    - o How to restore scripts for reading basic or complete registers without list of knx objects
- Use case 3: More than one power meter is connected to spaceLYnk
    - o How to use scripts for reading registers from different power meters
- Use case 4: Customization of scripts via optional parameters
    - o How to customize provided script to achieve better optimization of data performance to preserve KNX network bandwith

Schneider Electric

Competencies

There are few necessary competencies to follow procedures mentioned in this application note. It is mandatory to have knowledge of basic spaceLYnk configuration and knowledge of Lua scripting language in the content described in the Product manual. You is also need to have knowledge of Modbus power meters described in user manual of each single power meter.

System prerequisities

Procedures contained in this application note were tested on the spaceLYnk with firmware version 1.0

Product name	version
spaceLYnk	1.0

Table 1: System prerequisities

2 Design

There are six Modbus power meters (Acti9 range) installed in architecture (see Table 2) and connected to spaceLYnk. There is no problem to use only one power meter or more different power meters in real architecture. Different usage is described based on use cases in chapter 1.

Commercial number	Type	Description
A9MEM3150	iEM3150	3 – phase energy meter
A9MEM3155	iEM3155	3 – phase energy meter
A9MEM3250	iEM3250	3 – phase energy meter
A9MEM3255	iEM3255	3 – phase energy meter
METSEPM3250	PM3250	3 – phase energy meter
METSEPM3255	PM3255	3 – phase energy meter

Table 2: Installed power meters in architecture

Picture 1: Power meters connected to spaceLYnk via Modbus

3 Configuration

Power meters are connected to spaceLYnk via Modbus connection. SpaceLYnk is able to read all parameters of Modbus power meters and process them. Provided Lua scripts read parameters and update objects determined KNX group addresses.

For using the provided scripts it is necessary to follow the next steps described in the following chapters.

1) **Power meters configuration**

Firstly it is necessary to configure power meters according to the procedures described in instructions, such as communication, speed, wiring parameters etc.

2) **SpaceLYnk configuration**

Basic configuration of spaceLYnk according to the product manual. Setting of IP address, date and time, user access etc.

Schneider Electric

3) **Restore of power meters and configuration options**

Procedures to restore scripts and customization of scripts according to user needs like modbus connection settings, Modbus registers settings etc. This third step is described in deep in the following chapters.

Lua scripts for reading Modbus parameters

Provided Lua scripts are divided into two types. The first type of Lua script is designed for reading and processing basic set of Modbus parameters from each power meter. The second type of Lua scripts is designed for reading and processing complete (all) Modbus parameters from each power meter.

Provided Lua scripts are located in a *.zip file with the following structure

- registers_basic
  - backup
  - script
- registers_complete
  - backup
  - script

There are Lua scripts for specified power meter with defined function (reading of complete set of registers or just basic registers) in the backup and script folder.

The backup folder is used for complete spaceLYnk project restore (script and list of KNX objects). The script folder is used for scripts restore without list of KNX objects.

Commercial number	Type	Script files for basic registers	Script files for complete registers
A9MEM3150	iEM3150	iem3150_basic registers.tar.gz scripting-iem3150_basic registers.tar.gz	iem3150_complete registers.tar.gz scripting-iem3150_complete registers.tar.gz
A9MEM3155	iEM3155	iem3155_basic registers.tar.gz scripting-iem3155_basic registers.tar.gz	iem3155_complete registers.tar.gz scripting-iem3155_complete registers.tar.gz
A9MEM3250	iEM3250	iem3250_basic registers.tar.gz scripting-iem3250_basic registers.tar.gz	iem3250_complete registers.tar.gz scripting-iem3250_complete registers.tar.gz
A9MEM3255	iEM3255	iem3255_basic registers.tar.gz scripting-iem3255_basic registers.tar.gz	iem3255_complete registers.tar.gz scripting-iem3255_complete registers.tar.gz
METSEPM3250	PM3250	pm3250_basic registers.tar.gz scripting-pm3250_basic registers.tar.gz	pm3250_complete registers.tar.gz scripting-pm3250_complete registers.tar.gz

Schneider Electric

| METSEPM3255 | PM3255 | pm3255_basic registers.tar.gz | pm3255_complete registers.tar.gz |
| | | scripting-pm3255_basic registers.tar.gz | scripting-pm3255_complete registers.tar.gz |

Table 3: list of scripts for reading Modbus registers from Acti9 power meters

3.1 How to work with provided Lua scripts

The provided scripts for reading Modbus parameters are unique according to power meter type. Each power meter has it's own Lua script suitable for spaceLYnk which must be restored and configured. In following chapters will be described what type of scripts is suitable for specific situation and how user can restore and configure scripts to achieve best results.

There are differences in application of basic and complete scripts. Scripts for basic registers automatically create KNX addresses based on user option while scripts for complete registers use a fixed list of KNX addresses.

Note: The provided scripts read Modbus parameters and make internal update of KNX objects by default. For write Modbus parameter to KNX network please refer to chapter 3.1.5.

3.1.1 Restore basic registers script

Scripts for reading basic registers support datapoint formats UTF8, Int16, Int32, Int64, Float32, Bitmap, 4QFPPF. For details about datapoint format please refer to instruction of specified power meter.

Date/Time format is not supported.

Basic registers scripts contains the ability to read the most used Modbus parameters like voltage, current, power etc. User can restore whole spaceLYnk (scripts and KNX objects) or only scripts and build own list of KNX objects.
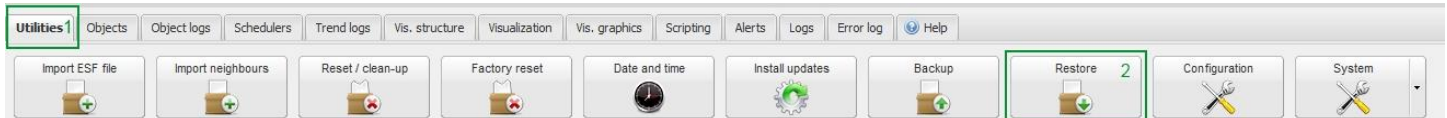
Steps how to restore spaceLYnk configuration and to use basic registers scripts:

1) Open web browser and log on spaceLYnk controller
2) Restore spaceLYnk project *Configurator ›› Utilities ›› Restore*

In spaceLYnk Configurator page

- Left click on Utilities [1]

Schneider Electric

- Left click on Restore

Picture 2: Restore of spaceLYnk project

Restore spaceLYnk with suitable script. Scripts are located in *register_basic ›› backup* folder.

Note: It is necessary to make refresh the webpage after scripts restore.

3) Initial Modbus configuration via restored Lua script

There are a few configuration steps after spaceLYnk restore which must be done before activating of script.

In spaceLYnk Scripting page under Resident scripts
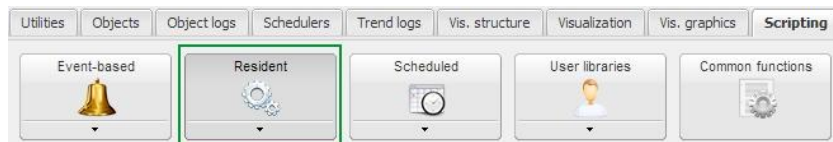
Picture 3: Resident scripts

- Open restored script via edit icon

Configuration parameters must be modified against to default settings.

```lua
local pwm = MbPwmRTU{
    address = "/dev/RS485",
    speed = 19200,
    parity = "E",
    databits = 8,
    stopbits = 1,
    duplex = "H",
    slaveAddress = 1,
    knxStartAddr = "1/1/",
}
```

Configuration parameters which are mentioned above are in default and must be modified according to the configuration which is set in specified power meter. For more instructions on how to configure power meter see instructions.

Note: knxStartAddr could be modified should it be required need especially in case of use in existing installations. Default format is main/middle/group. Group address is "1" by default and is automatically incremented +1 according to amount of Modbus registers.

AN003_v1.3: Modbus Power meters and spaceLYnk

KNX address format for automatic generation of KNX adress with increment of group address:

1) knxStartAddr = "1/2/"

   Generation of KNX adresses will start on group address 1 and main/middle part of knx address is open to modification according KNX rules.

2) knxStartAddr = "1/2/10"

   Generation of KNX adresses will start on group address 10 and main/middle part of knx address is open to modification according KNX rules.

3) knxStartAddr = "1/2"

   Generation of knx adresses will start on group address 1 and main/middle part of knx address is free to modify according KNX rules. No need to use „/" character.

There is also option to modify list of registers in case of user needs. Just delete or add registers which are suitable to use.

```lua
local regTable = pwm:allocRegisters {
        { addr = 3028, type = 'Float32'}, -- Voltage L1-N
        { addr = 3030, type = 'Float32'}, -- Voltage L2-N
        { addr = 3032, type = 'Float32'}, -- Voltage L3-N
        { addr = 3000, type = 'Float32'}, -- I1: phase 1 current
        { addr = 3002, type = 'Float32'}, -- I2: phase 2 current
        { addr = 3004, type = 'Float32'}, -- I3: phase 3 current
        { addr = 3110, type = 'Float32'}, -- Frequency
        { addr = 3084, type = 'Float32'}, -- Power Factor Total
        { addr = 3060, type = 'Float32'}, -- Active Power Total
        { addr = 3204, type = 'Int64'},  -- Total Active Energy Import
    }
```

4) Activate resident script for related power meter

After configuration of settings mentioned in previous steps the script can by clicking on the activation icon ⬤ . The script is running when Activation icon change to ⬤ .

5) Check relevant information is the depicted in list of objects

When script for reading Modbus parameters is running the parameters can be checked in Object tab.

Schneider Electric

Picture 4: list of KNX objects

Note: Be careful to not overload KNX bus with writing too many values to KNX objects. Information how to optimize data performance is described in chapter 3.1.5

3.1.2   Restore complete registers scripts

Provided scripts for reading complete Modbus parameters are based on scripts for reading basic registers. Scripts for reading complete registers support datapoint formats UTF8, Int16, Int32, Int64, Float32, Bitmap, 4QFPPF. For details about datapoint format refer to instruction of specified power meter.

Date/Time format is not supported.

Complete registers scripts contains ability to read all Modbus parameters according Modbus registers table listed in instructions for specified power meter. User can restore whole spaceLYnk (scripts and KNX objects) or only scripts and build their own list of KNX objects.

Steps how to restore spaceLYnk configuration and to use complete registers scripts:

1) Open web browser and log on spaceLYnk controller
2) Restore spaceLYnk project *Configurator ›› Utilities ›› Restore*

In spaceLYnk Configurator page

- Left click on Utilities [3]
- Left click on Restore [4]

Picture 5: Restore of spaceLYnk project

Restore spaceLYnk with script. Scripts are located in *register_complete ›› backup* folder.

Note: It is necessary to make refresh of webpage after scripts restore.

3) Initial Modbus configuration via restored Lua script

There are a few configuration steps after spaceLYnk restore which must be done before activating of script.

In spaceLYnk Scripting page under Resident scripts
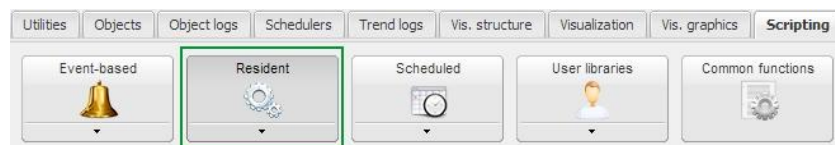
Picture 6: Resident scripts

- Open restored script via edit icon

Configuration parameters must be modified against to default settings.

```lua
local pwm = MbPwmRTU{
    address = "/dev/RS485",
    speed = 19200,
    parity = "E",
    databits = 8,
    stopbits = 1,
    duplex = "H",
    slaveAddress = 1,
}
```

Configuration parameters which are mentioned above are in default and must be modified according configuration which was set in specified power meter. For further instructions how to configure power meter see instructions.

Schneider Electric

Note: **There is missing knxStartAddr against initial configuration of scripts for reading basic registers because this KNX address is set in registers table.**

There is also the option to modify list of registers in case user needs. Just delete registers which are not necessary to use.

```lua
local regTable = {
    -- Date / Time
    { addr=1845, type="UInt16", knx="1/1/1"},          -- Year
    { addr=1846, type="UInt16", knx="1/1/2"},          -- Month & Day
    { addr=1847, type="UInt16", knx="1/1/3"},          -- Hour & Minute
    { addr=1848, type="UInt16", knx="1/1/4"},          -- Milisecond
    -- Wiring Setup
    { addr=2016, type="UInt16", knx="1/1/5"},          -- Power System
    { addr=2017, type="UInt16", knx="1/1/6"},          -- Nominal Frequency
    --Energy Pulse Output Setup
    { addr=2129, type="UInt16", knx="1/1/7"},          -- Pulse Duration
    { addr=2132, type="Float32", knx="1/1/8"},         -- Pulse Constant
    --Meter Data
    { addr=3000, type="Float32", knx="1/1/9"},         -- I1: phase 1 current
```

Note: All registers must be specified together with KNX address.

4) Activate resident script for related power meter

After configuration of settings mentioned in previous steps the script can by clicking on the activation icon ⬤ . The script is running when Activation icon change to ⬤ .

5) Check relevant information is the depicted in the list of objects

When script for reading Modbus parameters is running the parameters can be checked in Object tab (see chapter 3.1.1 for picture.

Note: **Be careful to not overload KNX bus with writing too many values to KNX objects. Information how to optimize data performance is described in chapter 3.1.5**

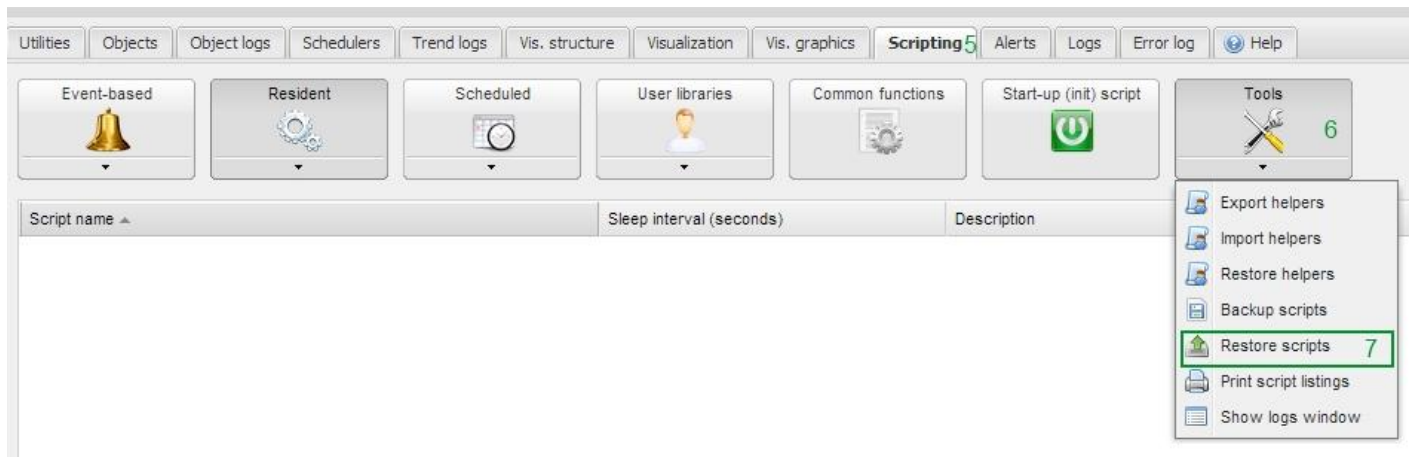### 3.1.3 Restore scripts without list of KNX objects for reading basic and complete Modbus registers

There is also possibility to restore Lua scripts for reading basic and complete registers without preset list of KNX objects. This can be useful for existing projects.

Schneider Electric

Steps how to restore spaceLYnk configuration and to use complete registers scripts:

1) Open web browser and log on spaceLYnk controller
2) Restore spaceLYnk project *Configurator ›› Utilities ›› Scripting ›› Tools ›› Restore scripts*

In spaceLYnk Configurator page

- Left click on Scripting [5]
- Left click on Tools [6]
- Left click on Restore scripts [7]

Picture 7: restore of scripts without presset KNX objects

3) Initial Modbus configuration via restored Lua script

Initial Modbus configuration procedure is the same as for restore of spaceLYnk project. Please refer to previous chapters to find more details. Initial configuration depends on the choice of script for reading basic or complete registers.

4) Create KNX objects

For visualization of read Modbus parameters it is necessary to build KNX object lists. This can be done via Object tab Add new object.

Click on icon Add new object open window to create KNX object.

Picture 8: Create KNX object

After filling these fields in the KNX object is created and can visualise value from Modbus register. This procedure must be followed for all Modbus registers listed in register table in script.

5) Activate resident script for related power meter

After configuration of settings mentioned in previous steps the script can by clicking on the activation icon ⬤ . The script is running when Activation icon change to ⬤ .

6) Check relevant information is shown on list of objects

When script for reading Modbus parameters is running the parameters can be checked in Object tab.

Note: Be careful to not overload KNX bus with writing too many values to KNX objects. Information how to optimize data performance is described in chapter 3.1.5

Note: Scripts for reading basic and complete registers are located in script folder in *.zip file.

3.1.4 Restore scripts for more power meters in installation

SpaceLYnk can read data from maximum 32 Modbus power meters located in installation. In this use case we want read parameters from 2 different power meters iEM3150 and iEM3155. In the case should be used restoration of scripts should be used instead of whole project. KNX objects must be built manually.
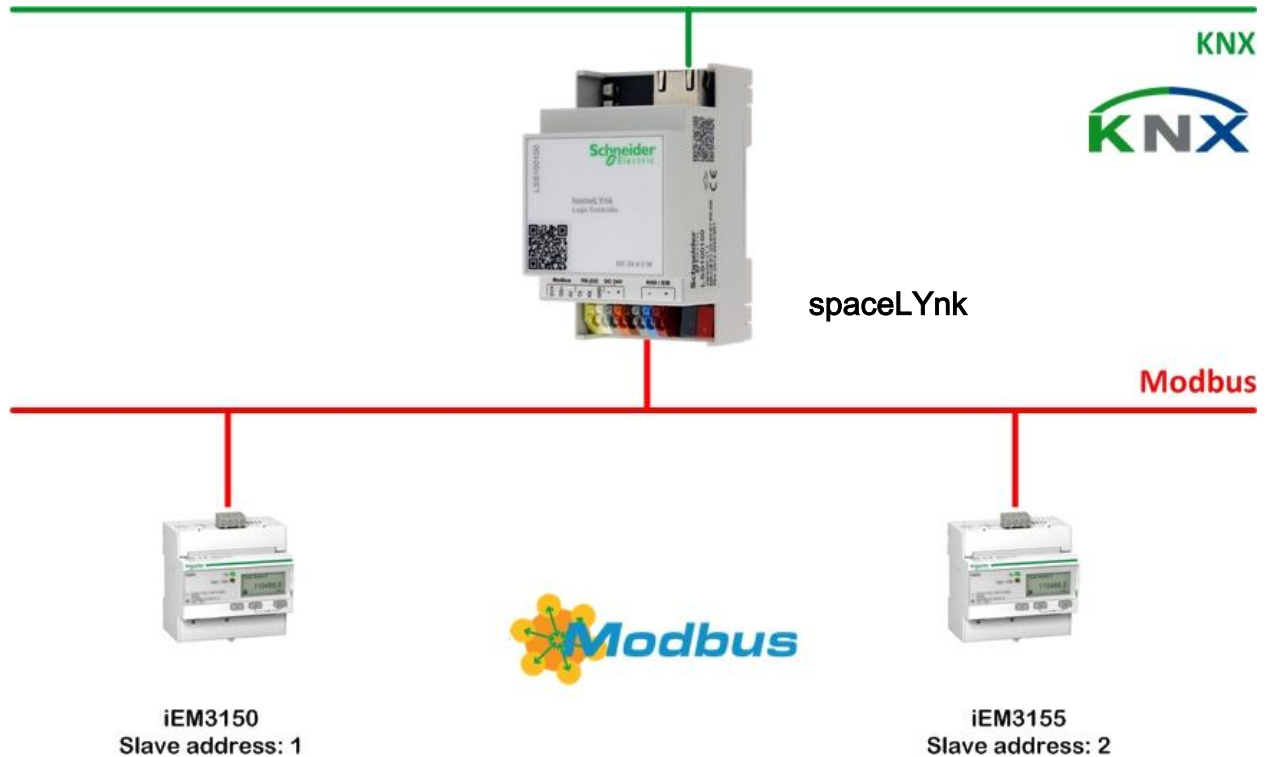
Picture 9: Architecture for reading registers from 2 power meters

Steps how to restore spaceLYnk scripts for reading basic registers.

1) Open web browser and log on spaceLYnk controller
2) Restore spaceLYnk project *Configurator ›› Utilities ›› Scripting ›› Tools ›› Restore scripts*

In spaceLYnk Configurator page

- Left click on Scripting [8]
- Left click on Tools [9]
- Left click on Restore scripts [10]

Scripts are located in *register_basic ›› scripts* folder

Schneider Electric
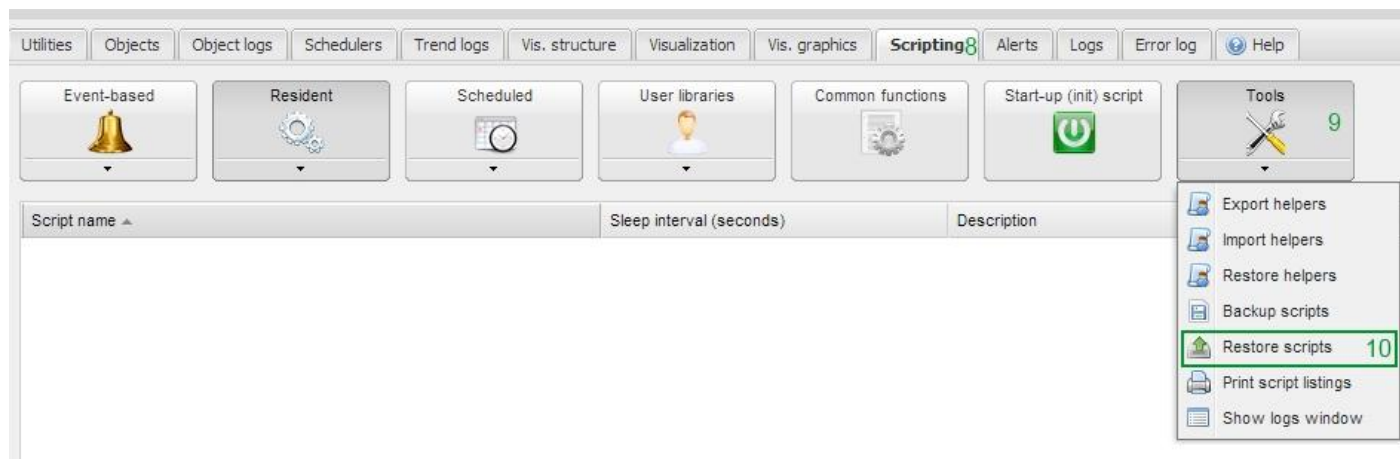
Picture 10: restore of scripts without presset KNX objects

Note: For first power meter use Restore mode (Remove existing scripts and import from backup). For other power meters change option to "Append keeping existing scripts".
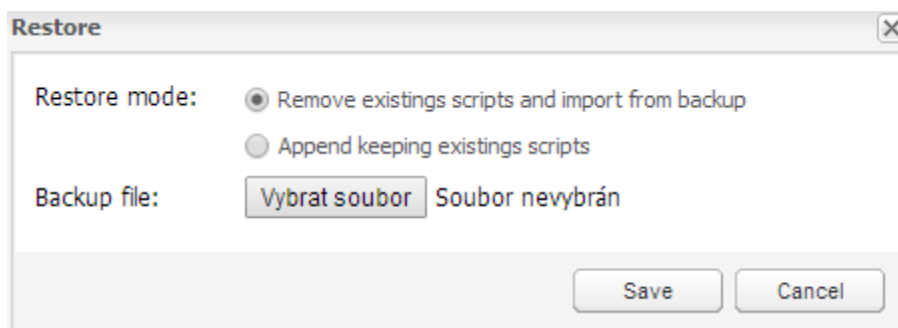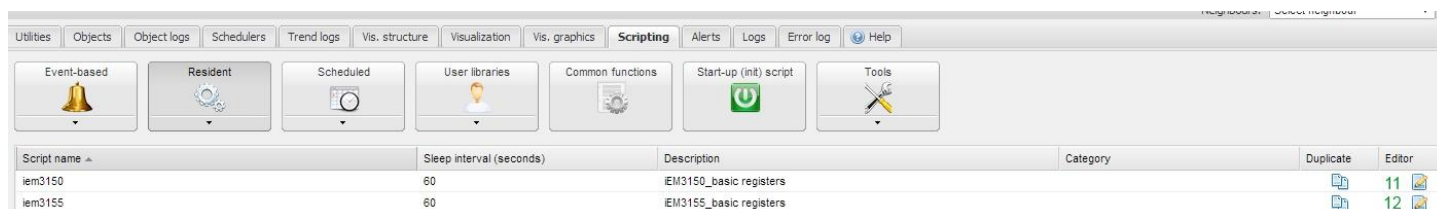

Picture 11: Restore options

Two script for power meters iEM3150 and iEM3155 are now available.

3) Modify scripts on Scripting page

- Open Editor for first power meter iEM3150[11]
  - o Set configuration parameters and list of registers

Picture 12: List of scripts for power meters

- Open Editor for second power meter iEM3155[12]

Schneider Electric

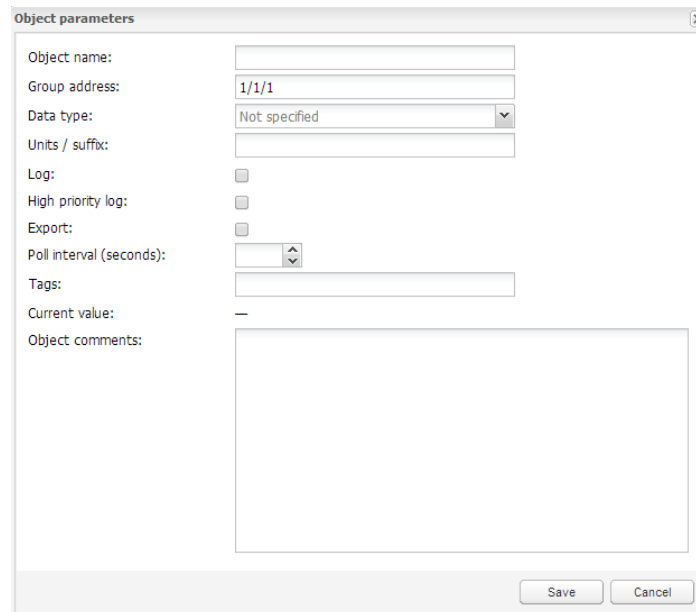o Set configuration parameters and eventually list of registers

Note: Configuration is described in chapter 3.1.1 Restore basic registers script. Configuration parameters depends on settings of specific power meters.

4) Create KNX objects

For visualization of read Modbus parameters it is necessary to build KNX object lists. This can be done via Object tab [Add new object].

Click on icon Add new object, open window to create KNX object.

Picture 13: Create KNX object
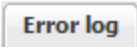
After filling these fields the KNX object is created and can visualise values from the Modbus register. This procedure must be done for all Modbus registers listed in register table in scripts.

5) Activate resident scripts for related power meter

After configuration of settings mentioned in previous steps the scripts can be activated by clicking on the activation icon ⬤. The script is running when Activation icon changes to ⬤.

After this step reading of Modbus registers from power meters is running.

Note: If reading from more than one power meter use semaphore functionality. One power meter locks the semaphore and other power meters must wait for unlock. In this case the script can start reading after a delay.

Schneider Electric

Semaphore lock issue is visible on tab Error log Error log . In case semaphore is locked, wait for automatic unlock.

The scripts for reading complete registers can also be used. The difference is in the part where scripts are modified according to specific power meters. The KNX address in this case is fixed so there is no need to set knxStartAddr.

3.1.5 How to customize provided scripts to achieve better optimization of data performance

Each power meter has many Modbus registers where data are stored. Reading of all data is not necessary in most use cases. The user could add parameters in script to preserve KNX network bandwith.

1) Set sleep interval of each script

   It is not necessary to read data from power meter every second. Sleep interval is set to 60 seconds by default. Sleep interval settings depends on number of power meters in installation and number of registers.

2) Change of value

   For each Modbus register with Float32 or Int datapoint type the parameter covDelta can be set. The Float32 datapoint type has the parameter covDelta (change of value Delta). This parameter has an affect on the intensity of update or write Modbus parameter to KNX objects. Script calculate difference between previous and actual value and if the difference is over covDelta parameter then the value is updated or written to KNX. Default value for covDelta is 0.1. In case of need just change the covDelta value.

   ```
   { addr = 3060, type = 'Float32', covDelta = 0.3}, -- Active Power Total
   ```

   The Int datapoint type can also use the covDelta parameter. There is no default value for covDelta for Int datapoint type.
   Note: There is a possibility to skip covDelta parameter with force parameter:

   ```
   { addr = 3060, type = 'Float32', covDelta = 0.3, force = true},
   ```

3) Write to KNX bus

   To preserve KNX network bandwith all KNX objects in spaceLYnk are updated by default. For write value to KNX network parameter flags can be used.

   ```
   { addr = 3060, type = 'Float32',flags="w" }, -- Active Power Total
   ```

Schneider Electric

This parameter has an impact on speed of KNX network and should be used where appropriate.

Example of modification behavior for register 3060 – Active power total which is write to KNX network only when difference between old and new value is more than 0.3.

```
{ addr = 3060, type = 'Float32', covDelta = 0.3, flag="w" }, -- Active Power Total
```

Schneider Electric

4 Conclusion

Procedures contained in this application note describe usage of Lua scripts for reading of Modbus registers from different Acti9 power meters. The provided scripts are based on use cases mentioned in Introduction chapter and could be use in different scenarios allow to read basic or complete set of Modbus registers depending on project request.

5 Appendix

5.1 Glossary

The following table describes the acronyms and defines the specific terms used in this document.

Abbreviation	Description
Lua	Lightweight multi-paradigm programming language designed as a scripting language with extensible semantics
Modbus	Serial communications protocol for use with its programmable logic controllers (PLCs)

Schneider Electric

Schneider Electric Industries SAS

Head Office

35, rue Joseph Monier

92506 Rueil-Malmaison Cedex

FRANCE

www.schneider-electric.com