Basic / **Advanced** / Expert

Application note

Email, SMS and FTP in spaceLYnk

Email notification, SMS management and USB flash memory with FTP access



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.

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.

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.

© 2014 Schneider Electric. All rights reserved

Table of Contents

1 Introduction 7

1.1 Email 7

1.2 SMS 7

1.3 FTP 7

2 Sending Email 9

2.1 Email scripts 9

2.2 Configure Email 9

2.3 Example of usage Email service 12

3 SMS 13

3.1 Scripts 13

3.2 Configuring SMS 14

3.3 Syntax 18

3.3.1 Example of usage - Sending request 18

3.3.2 Examples of usage – Sending commands 18

3.4 Troubleshooting 19

4 USB flash drive 20

4.1 Scripts 20

4.2 Configuring USB 21

4.3 Examples 22

4.3.1 Example of usage - How to write values 22

4.3.2 Example of usage - How to read values 23

5 Providing USB flash drive via FTP 24
5.1 Activating FTP in spaceLYnk 24
5.1.1 Verify FTP connection 24
5.2 USB via FTP 25
5.2.1 FTP Read only 25
5.2.2 FTP Read/Write 25
6 Multiple USB devices 26
6.1 Multiple USB flash drive 26
6.1.1 USB Partitions script 26
6.1.2 USB Mount partition script 27
6.2 Multiple USB device script 27
7 Appendix 28
7.1 Glossary 28

1 Introduction

This application note describes the configuration processes of spaceLYnk for the purpose of SMS management, email notification and USB flash memory with FTP access.

1.1 Email

Email notification is available in spaceLYnk, when spaceLYnk has connection to internet or email server. Notification alerts can be sent via email.

- Emails from spaceLYnk can be useful for sending daily status reports of system, energy consumption and some non-urgent alerts – email is less flexible than SMS

1.2 SMS

SMS (Short Message Service) – This service is available for spaceLYnk, when you have a 3G modem with activated SIM card connected. You can send request on status of objects in spaceLYnk or spaceLYnk can send you SMS notification – alert.

SMS can be used for very quick remote management of spaceLYnk.

- SMS can be automatically sent after important alert gets active (Fire, Flooding, Break-in, Overheating, Security) or after some time period as your weekly energy consumption values
- You can request actual room temperature wherever you are
- You can control heating in your living room before your arrival to prevent entry into cold house

1.3 FTP

FTP (File Transfer Protocol) – spaceLYnk contains an FTP server, so you need only an FTP client. Data can be transferred through this protocol. In case of this application note we combine usage of FTP and USB flash memory, so we can distribute data saved on USB flash via FTP.

- KNX project from ETS can be archived in the FTP root folder or USB folder for later usage.
- In this application note you can find setting of USB flash (Read, Write), which is necessary for distribution of USB via FTP.

Competencies

There are several necessary competencies required in order to follow the procedures mentioned in this application note. It is mandatory to have knowledge of basic spaceLYnk configuration described in the User Guide and basics in LUA programming.

System prerequisites

Software / Product name	Version
spaceLYnk (FTP server)	2.1
Total commander (FTP client)	8.01

Table 1: System prerequisites

2 Sending Email

Email client in spaceLYnk serves for sending alerts and notification. For availability of this service your spaceLYnk must be connected to the internet. As an example **we used Gmail** for our email account. Any email provider can be used, these have option to provide information about connection to their email server (SMTP).

With spaceLYnk you can send emails, however it is not possible to receive emails via spaceLYnk.

2.1 Email scripts

All needed scripts are attached in “scripting-backup-email-package.tar.gz”

They are deployed in the following script tabs:

- Event-based (script “Send Email”)
- User libraries (script “user.Email”)

2.2 Configure Email

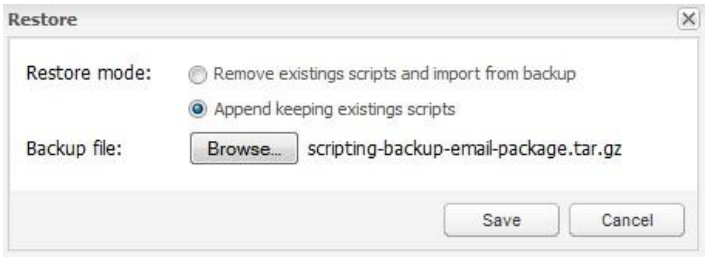
Email configuration consists of four steps:

1) Restoring scripts
2) Setting of User libraries script
3) Editing “User libraries” script
4) Editing “Send Email” script

To verify connection to internet you can use *Configurator ›› System ›› Status ›› Network utilities ›› Ping*

And you can try to ping to reliable web server as for example www.google.com

Step 1: Restoring scripts



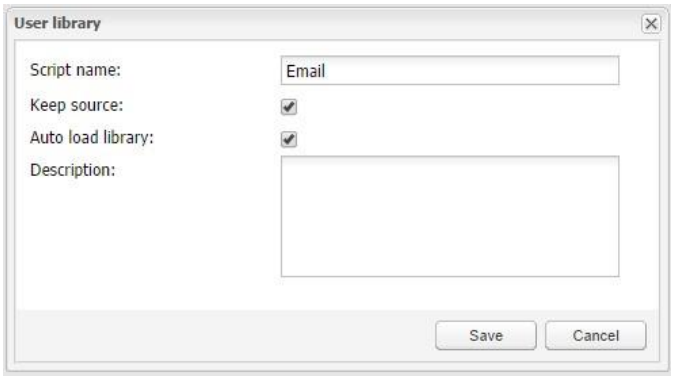
a) Open your web browser
b) Type IP address of your spaceLYnk
c) Click **Configurator**
d) Click **Scripting**
e) Click **Tools**
f) Select **Restore scripts**
g) Choose "scripting-backup-email-package.tar.gz"
   - Choose "Append keeping existing scripts" to preserve existing scripts in your spaceLYnk
h) Click Save
   - Your spaceLYnk is rebooting now, wait approximately 1 minute for spaceLYnk startup

Scripts are now restored and can be found in:

- Event-based (script "Send Email")
- User libraries (script "user.Email")

Step 2: Setting of user libraries script

a) Click **Scripting** (*Configurator ›› Scripting)*
b) Click **User libraries**
c) Click **user.Email** script
   - **User library** pop up window opens
d) Check if "**Auto load library**" function is enabled



Note: If you are using an older version of spaceLYnk than 1.2.0, where auto load feature is not available, use command "require(user.Email)" in each script where you need email function.

Step 3: Editing of user libraries script

a) Click the **Editor** icon of your email script (user.Email) to open script in script editor window. (*Configurator ›› Scripting ›› User libraries)*

b) Modify parameters in User editable area

- From – Specify sender email. Can be the same like user
- User – User name for email account for this service
- Password – Password for email account for this service
- Server – SMTP server address (smtp.gmail.com)
- Port – SMTP server port (465)
- Secure – Type of secure connection (sslv23)

c) Click **Save and Close**

- Mail function of your spaceLYnk is now successfully set-up.

```
----------BEGINNING OF USER EDITABLE AREA----------
local settings = {
  -- "from" field, only e-mail must be specified here
  from = 'solutiongtc@gmail.com',
  -- smtp username
  user = 'solutiongtc@gmail.com',
  -- smtp password
  password = 'xxxxxxxx',
  -- smtp server
  server = 'smtp.gmail.com',
  -- smtp server port
  port = 465,
  -- enable ssl, required for gmail smtp
  secure = 'sslv23',
}
----------END OF USER EDITABLE AREA----------
```

Step 4: Editing of "Send email" script

a) Click **Event-based** (*Configurator ›› Scripting*)

b) Click **Editor** icon of your email script (Send Email) for open script in script editor window

c) Modify script

- Destination mail – Mail address of receiver
- Subject – Subject of message
- Text message – Text of message

```
Event-based: Send Email
1 email("destination@mail.com", "subject", "text")
```

d) Click **Save and Close**

- This function can be used in resident and scheduled scripts too

Syntax: email(<destination mail>, <subject>, <text>)

Your spaceLYnk is now ready to send emails.

2.3 Examples of usage Email service

Example 1: Set-up a program which will automatically send the new IP address from the spaceLYnk once it has changed.

```
require('json')
data = io.readproc('if-json')
data = json.decode(data)
ip = data.eth0.inetaddr

 ip_old = storage.get('ip_old')
if ip_old == nil then
  storage.set('ip_old', ip)
end

if ip ~= ip_old then
   subject = 'spaceLYnk IP changed'
   message = 'The new IP is: ' .. ip
   email('user@example.com', subject, message)
   storage.set('ip_old', ip)
end
```

Example 2: Task of this example is send email on KNX object 1/2/2 status change with current value. This example is suitable for home alarm or unexpected situation.

```
email("destination@mail.com", "Home alarm", "KNX object 1/2/2 value is: " ..
tostring(event.getvalue()))
```

3 SMS

SMS service in spaceLYnk is suitable for sending commands to change KNX values and also sending requests for read actual value of desired group address.

spaceLYnk can act on requests received through SMS (Send requests to turn off all sockets or turn on heating) or can actively provide feedback via alerts (Glass sensor is tripped, bathroom is flooded) or notifications (Main door are open for more than 5 minutes, Home alarm has been armed).

Sending an SMS from spaceLYnk is charged in the same price as the standard SMS from your mobile service provider.

To obtain better GSM signal please use an USB extension (up to 5m) for leading modem out from cabinet with installed spaceLYnk (metal cabinets decrease reliability of wireless communication).

Compatibility is guaranteed with Huawei modem E3131 with activated SIM card.

3.1 Scripts

All needed scripts are attached in “SMS package scripting backup”.

They are deployed in following script tabs:

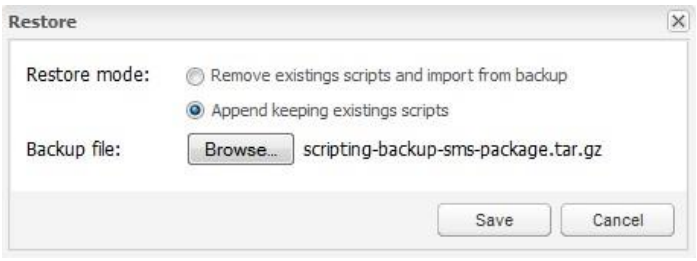
- Event-based (script “Send SMS”)
- Resident (script “SMS Handler”)
- User libraries (script “user.SMS”)

3.2 Configuring SMS

SMS configuration consists of six steps:

1) Restoring scripts
2) Setting of user libraries script
3) Setting of resident script
4) Editing of resident script
5) Editing of “Send SMS” script
6) Insert 3G modem to USB and reboot HL

Step 1: Restoring scripts



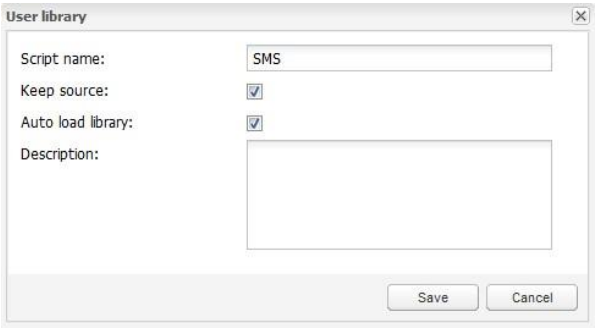
a) Open your web browser
b) Type IP address of your spaceLYnk
c) Click **Configurator**
d) Click **Scripting**
e) Click **Tools**
f) Select **Restore scripts**
g) Choose “scripting-backup-sms-package.tar.gz”
    - Choose “Append keeping existing scripts” to preserve existing scripts in your spaceLYnk
h) Click Save
    - Your spaceLYnk is rebooting now, wait approximately 1 minute for spaceLYnk startup

Scripts are now restored and can be found in:

- Event-based (script “Send Email”)
- Resident (script “SMS Handler”)
- User libraries (script “user.Email”)

Step 2: Setting of user libraries script

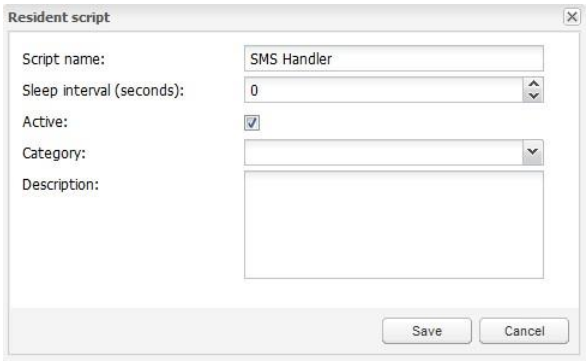
a) Click **Scripting** (*Configurator ›› Scripting)*
b) Click **User libraries**
c) Click **user.SMS** script
   - **User library** pop up window opens
d) Check if "**Auto load library**" function is enabled



Note: If you are using an older version of spaceLYnk than 1.2.0, where auto load feature is not available, use command "require(user.SMS)" in each script where you need email function.

Step 3: Setting of user libraries script

a) Click **Scripting** (*Configurator ›› Scripting)*
b) Click **Resident**
c) Click **SMS Handler** script
   - **User library** pop up window opens
d) Check if "Sleep interval" is set to 0
e) Click **Save**



Step 4: Editing of resident script

1. Click **Editor** icon of your SMS script (SMS Handler) for open script in script editor window. (*Configurator ›› Scripting ›› Resident)*
2. Modify parameters in User editable area
    - Numbers - Numbers have to be filled in international format without **00 or +** before number. No limit to amount of numbers in phone number. spaceLYnk will **accept requests only from numbers in this format "**420123456789", for more about number format see chapter 3.4

      Syntax: numbers = {<phone number 1>, <phone number 2>, <phone number n>}

    - Pin code - Set pin code when needed, limited to 4 digits. If no PIN is needed fill 000 or leave pin code empty
3. Click **Save and Close**

```
---------------------------------BEGINNING OF USER EDITABLE AREA--------------------------------------
    -- allowed numbers, SMS message from other number will be ignored
    numbers = {"420123456789", "420987654321"}
    -- replace 0000 with SIM pin number, or remove the line below if PIN check is disabled
    pincode = "0000"
-----------------------------------END OF USER EDITABLE AREA----------------------------------------
```

Step 5: Editing of "Send SMS" script

a) Click **Event-based** (*Configurator ›› Scripting*)
b) Click **Editor** icon of your SMS script (Send SMS) for open script in script editor window
c) Modify script
   - Destination telephone number in international format – Please use 00 instead of + before number
   - Text message – Each 160 characters is charged as one SMS

```
Event-based: Send SMS
1 sendsms('0042012345678', 'Text message')
```

d) Click **Save and Close**
   - This function can be used in resident and scheduled scripts too

Syntax: sendsms(<phone number>, <text message>)

Step 6: Insert 3G modem to USB and reboot HL

a) Insert 3G modem to USB
b) Click **Utilities** (*Configurator*)
c) Click **System**
d) Move mouse to **System**
e) Select **Reboot**
f) Click **OK**
   - Your spaceLYnk is rebooting now, wait approximately 1 minute for spaceLYnk startup

Your spaceLYnk is now ready to send/receive SMS.

3.3 Syntax

Read from bus

- R␣ALIAS
- On read request, script will reply with SMS message containing current value of selected object

Write to bus

- W␣ALIAS␣VALUE

ALIAS can be

- Group address (e.g. 1/1/1)
- Name (e.g. Obj1). If name contains spaces then it must be in inverted commas (e.g. “Room Temperature”)

Commands W/R are not key sensitive, so can be lower or upper case.

3.3.1 Example of usage - Sending request

For successful response of your requests, you need to add your telephone number to “white-list in resident script”. On read request, script will reply with SMS message containing current value of selected object.

- Read (send the following SMS to read the security panel value):
  R 2/1/1

3.3.2 Examples of usage – Sending commands

- Binary write (send the following SMS to switch kitchen lights on):
  W 1/1/1 true
- Scaling write (send the following SMS to set value 67% for red LED):
  W LED1Red 67
- Temperature write (send the following SMS to create setpoint in the living room at 22.5°C:
  W “Room Setpoint” 22.5

3.4 Troubleshooting

In each country there is a different system of phone numbers, 3G modems and operators. You can encounter a situation, that your HL will not send an SMS, because your country implemented different requirements for the GSM network.

Tested configuration:

Modem: Huawei E3131

Country: Czech Republic

Format for sending from spaceLYnk: 00420123456789

Format for receiving to spaceLYnk: 420123456789

00 – Replaces the standart prefix (+)

420 – Area code for Czech Republic

123456789 – Phone number

4 USB flash drive

USB interface can be used for increasing memory capacity of spaceLYnk via USB flash drive. This additional memory can be used only for data out coming from scripting (for example logging of variables). Data (as complete KNX project) could be uploaded to spaceLYnk USB drive even directly from any PC via FTP– more in chapter 5.

USB flash drive supports FAT, FAT32 and NTFS file system. Maximum size of Flash drive is 32GB.

Warning:

- **NTFS files created and modified from HL cannot be read/write in MS Windows environment (and vice versa), spaceLYnk is Linux based! NTFS system can be used only when you will manage data through spaceLYnk - FTP**
- **Before removing USB flash drive from spaceLYnk, the spaceLYnk has to be shut down!**

4.1 Scripts

- USB Start-up (init) script

```
-- create usb mount directory
os.execute('mkdir -p /mnt/usb')

-- find first matching usb storage device
dev = io.readproc('ls /dev/sd*1 2>/dev/null'):match('/dev/sd%l1')
-- found it, mount
if dev then
    os.execute('mount ' .. dev .. ' /mnt/usb')
    alert('[usb-mount] mounted %s', dev)
-- nothing found, local flash will be used
else
    alert('[usb-mount] no device found')
end
```

- USB Write script

```
-- write string „String Data" to the end of log file preserving all previous data
file = io.open('/mnt/usb/log.txt', 'a+')
file:write("String Data\r\n")
file:close()
```

- USB Read script

```
for line in io.lines('/mnt/usb/config.txt') do
  -- process line
  log(line)
end
```

4.2 Configuring USB

SMS configuration consists of six steps:

1) Create USB Start-up (init) script
2) Create USB write script
3) Create USB read script
4) Insert USB flash drive to HL and reboot

Step 1: Create USB Start-up (init) script

a) Open your web browser
b) Type IP address of your spaceLYnk
c) Click **Configurator**
d) Click **Scripting**
e) Click **Start-up (init) script**
f) Copy and paste code from chapter 4.1 - USB Start-up (init) script
g) Click Save and close

Start-up (init) script

```
-- init script
-- create usb mount directory
os.execute('mkdir -p /mnt/usb')

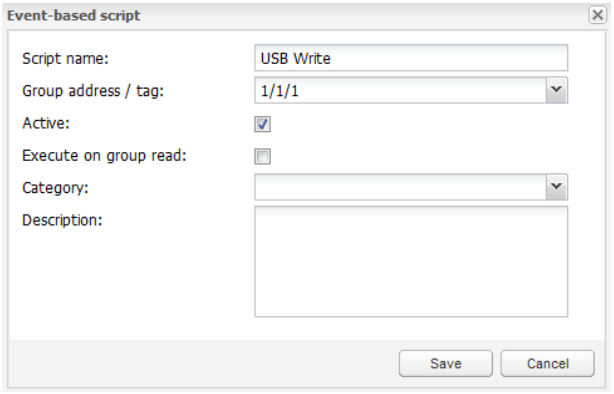
-- find first matching usb storage device
dev = io.readproc('ls /dev/sd*1 2>/dev/null'):match('/dev/sd%l1')

-- found it, mount
if dev then
    os.execute('mount ' .. dev .. ' /mnt/usb')
    alert('[usb-mount] mounted %s', dev)
-- nothing found, local flash will be used
else
    alert('[usb-mount] no device found')
end
```

Step 2: Create USB write script

Ignore this step, if you do not want to write to USB

a) Click to arrow under **Event-based** (*Configurator ›› Scripting*)
b) Select Add new script
    a. Type Script Name, Group Address and click checkbox “Active”
c) Click **Save**
d) Click **Editor** icon of your USB script (USB Write) for open script in script editor window
e) Copy and paste code from chapter 4.1 – USB Write script
f) Modify for your needs
g) Click **Save and close**



- After activation of this script spaceLYnk open/create log.txt saved on USB flash drive

- Write to this file on new line String “String Data” – Can be replaced for example by variable for your needs.

Step 3: Create USB read script

Ignore this step, if you do not want to read from USB

a) Click to arrow under **Event-based** (*Configurator ›› Scripting*)
b) Select Add new script
    a. Type Script Name, Group Address and click checkbox “Active”
c) Click **Save**
d) Click **Editor** icon of your USB Read script for open script in script editor window
e) Copy and paste code from chapter 4.1 – USB Read script
f) Modify for your needs
g) Click **Save and close**

- After activation of this script spaceLYnk read already created config log.txt saved on USB flash drive
- Read each line of this txt and enter it in log of spaceLYnk

Step 4: Insert USB flash drive to HL and reboot

a) Insert USB flash drive to spaceLYnk
b) Click **Utilities** (*Configurator*)
c) Click **System**
d) Move mouse to **System**
e) Select **Reboot**
f) Click **OK**
    - Your spaceLYnk is rebooting now, wait approximately 1 minute for spaceLYnk startup

Your spaceLYnk is now ready to read/write to USB flash drive

4.3 Examples

4.3.1 Example of usage - How to write values

Script for writing to USB is in our case event-based. With every change of value spaceLYnk will write the current time value to file on flash drive.

Useful for logging temperature or energy consumption less frequently on USB – data can be processed later, for example in Excel.

```
value = knxdatatype.decode(event.datahex, dt.bool)
data = string.format('%s value is %s', os.date('%c'), tostring(value))

-- write to the end of log file preserving all previous data
file = io.open('/mnt/usb/log.txt', 'a+')
file:write(data .. '\r\n')
file:close()
```

4.3.2 Example of usage - How to read values

Script for reading configuration file from USB – config.txt, which contains configuration data, which can be processed. Two words are split by equal symbol, for example:

- KNX = true
- IP = 192.168.0.10
- Etc.

This script can be modified for reading values or other data.

```
for line in io.lines('/mnt/usb/config.txt') do
  -- split line by '=' sing
  items = line:split('=')
  -- two items, line seems to be valid
  if #items == 2 then
    key = items[ 1 ]:trim()
    value = items[ 2 ]:trim()
    alert('[config] %s = %s', key, value)
  end
end
```

5 Providing USB flash drive via FTP

Data stored on flash drive connected to spaceLYnk can be read by two ways:

- spaceLYnk has to be shut down for safety remove of flash drive. Then flash drive with data can be inserted to PC.
- USB flash drive can be read via FTP client.

5.1 Activating FTP in spaceLYnk

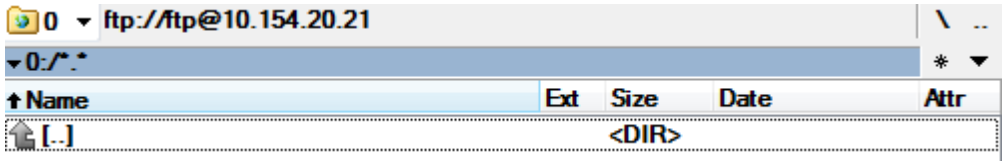
1. Click FTP server (Configurator ›› System ›› Services ›› FTP server)
2. Switch server status to **Enabled**
3. Type password for FTP access
4. Click **OK**

5.1.1 Verify FTP connection

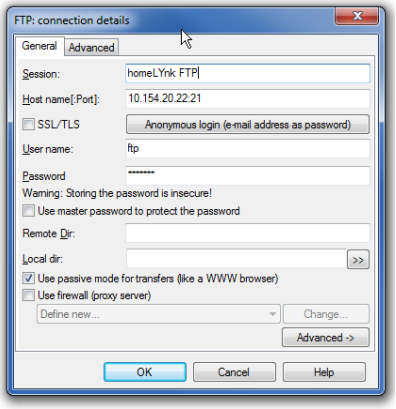
Process of verification of FTP connection in this chapter is via **Total commander.**

1. Open Total commander
2. Select “FTP New connection…” in Net tab
3. Type user name “ftp”
4. Type password for FTP access

If you haven’t any error message and you can see this empty folder, FTP server on spaceLYnk runs correctly.



Note: FTP connection can be edited in “FTP connect” in Net tab. Settings can look like this.



5.2 USB via FTP

There are 2 options how to realize this data transfer, FTP READ only and FTP READ/WRITE.

5.2.1 FTP Read only

1. Format your flash drive in FAT32 – in “My computer” by right click and format
2. Update USB start up script to:

```
-- create usb mount directory
os.execute('mkdir -p /mnt/usb')
 -- find first matching usb storage device
dev = io.readproc('ls /dev/sd*1 2>/dev/null'):match('/dev/sd%l1')
 -- found it, mount
if dev then
    os.execute('mount ' .. dev .. ' /mnt/usb')
    os.execute('mkdir -p /home/ftp/usb')
    os.execute('mount /mnt/usb /home/ftp/usb')
    alert('[usb-mount] mounted %s', dev)
-- nothing found, local flash will be used
else
    alert('[usb-mount] no device found')
end
```

3. Reboot spaceLYnk

USB flash drive is ready for read via FTP client.

5.2.2 FTP Read/Write

1. Format your flash drive in NTFS – in “My computer” by right click and format
2. Update USB start up script to:

```
-- create usb mount directory
os.execute('mkdir -p /mnt/usb')
 -- find first matching usb storage device
dev = io.readproc('ls /dev/sd*1 2>/dev/null'):match('/dev/sd%l1')
 -- found it, mount
if dev then
    os.execute('mount ' .. dev .. ' /mnt/usb')
    os.execute('mkdir -p /home/ftp/usb')
    os.execute('mount /mnt/usb /home/ftp/usb')
  os.execute('chown ftp:ftp /home/ftp/usb')
    alert('[usb-mount] mounted %s', dev)
-- nothing found, local flash will be used
else
    alert('[usb-mount] no device found')
end
```

3. Reboot spaceLYnk

USB flash drive is ready for read/write by FTP client.

Note: In case you use this option, it is impossible to read data on the flash drive inserted to PC, data on this flash drive can be read/write only by spaceLYnk!

6 Multiple USB devices

spaceLYnk can operate with USB 3G modem and USB flash drive at the same time. Both devices have to be connected to one USB hub.

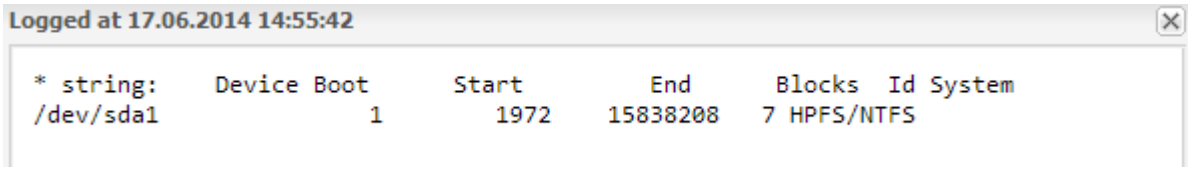
Issues can arise when you want to connect more than these two devices. If you want to connect more USB flash drive or 3G modem there is need to modify scripts for them. In next chapters the basics of Linux is requested – mounting of devices and advance knowledge in LUA scripting.

6.1 Multiple USB flash drive

This function will help you to select which partition or which device you want to mount. After performing the log this function can see all connected USB flash drives. It is then important to modify mounting start up script according to your needs.

6.1.1 USB Partitions script

```
function exploreDevices()
    local proc, res, line
    res = {}
    proc = io.popen("fdisk -l | egrep -B 1 '^/dev/s'", "r")
    for line in proc:lines() do
        table.insert(res, line)
    end
    proc:close()
    return table.concat(res, "\n")
end
```



6.1.2 USB Mount partition script

This function will help you to select which partition or which device you want to mount. After performing the log this function can see all connected USB flash drives. It is then important to modify mounting start up script according to your needs.

```
function mount(dev, dir)
  local cmd, res

  if not io.exists(dev) then
    return nil,  'device not found'
  end

  if not io.exists(dir) then
    cmd = string.format('mkdir -p %q', dir)
    os.execute(cmd)
  end

  if not io.exists(dir) then
    return nil, 'directory not found'
  end

  cmd = string.format('mount %q %q 2>&1', dev, dir)
  res = io.readproc(cmd)
  if res:find('failed') then
    return nil, res
  end

  return true
end
```

6.2 Multiple USB device script

This script is useful for print connected USB devices.

```
function findttyusb()
  local proc, chunks, tty, port, res

  res = {}
  proc = io.popen('ls -l /sys/class/tty', 'r')
  for line in proc:lines() do
    if line:find('ttyUSB') then
      chunks = line:split('/')
      tty = chunks[ #chunks ]
      port = chunks[ #chunks - 3 ]

      res[ port ] = tty
    end
  end
  proc:close()

  return res
end
```

7 Appendix

7.1 Glossary

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

Abbreviation	Description
SMS	Short Message Service
FTP	File Transfer Protocol
USB	Universal Serial BUS
SMTP	Simple Mail Transfer Protocol

Schneider Electric Industries SAS

Head Office
35, rue Joseph Monier
92506 Rueil-Malmaison Cedex
FRANCE

www.schneider-electric.com