Operation with keys

Protecting system prohibiting the start of any programs unauthorized by the operator is realized in the device. For this purpose the algorithm DSA (digital signature)with the key length equal to 2048 bits is used, as well as manufacturer keys, operator key and the key for controlling the device.

Manufacturer key

- This key is used for checking the digital signature of the operator key. The secret part of this key is kept by the manufacturer.

Operator key

- This key is owned by the operator. The secret part of this key is preserved by the operator and used for signing the program started by the Bootstrap. This can be the core of bootstrap.
- This key is also used for signing the image broadcast in a multicast group and it is used by bootstrap to update the file system located in the device. The public part of the key is installed in the device through the bootstrap menu or by assigning “oppubKEY” to the variable of the bootloader. The public part of the key must be signed using the manufacturer key. This digital signature is subject to checking before using the operator key.

Key for controlling the device

- This key is owned by the operator. The key is used for signing commands sent to the device. Operator must place the public part of this key in the device. Operator commands for controlling the device on the server are signed with the secret part of this key and are sent to the device. The digital signature is checked on the device using the public part of the key. If the digital signature is correct, the command shall be performed. Other variants of using this key are available on the discretion of the operator. This key is not used in the process of loading the basic program. Utilities dsign, mcsend and mcrec, are supplied with the device basic program and allow realizing this algorithm and adapt other algorithms if necessary.

Info

To obtain a digital signature for “operator key” at the factory - it is necessary to provide next information:

- Public part of the key.
- Additional information in text form:
  - Name of organization;
  - E-mail address;

- Contact phone number.

Useful Links:

http://www.gnupg.org/GnuPG - project official page

GnuPG - Wikipedia

Operator Guide

Software releases, documentation ...

Make firmware image of STB MAG-200/250

Preparing logo

Operator key preparation

In the process of the operator key preparation proceed as follows:

- Create the key

run:

```
gpg --gen-key
```

Select:

```
(4) RSA (sign only)
What keysize do you want? (2048) 2048
Key is valid for? (0) 0
Is this correct? (y/N) y
Real name:
Email address:
Enter passphrase:
```

Where:

Real name: - Organization name which you specified earlier during the application process;

Email address: - E-mail address which you specified earlier during the application process.

- Export of the key to the file:

```
gpg -o oppubbin.KEY --export "key ID"
```

Where:

oppubbin.KEY - file name, in which public part of the key will be exported;

“key ID” - **Real name**, which was described while key prepearing.

After performing this command the public part of the key shall be contained in the file oppubbin.KEY;

- Send the file oppubbin.KEY to the manufacturer for creating a digital signature of this key with the use of the manufacturer key.
- Install the resulting file in the device.

Making firmware image which signed by operator key

Bootstrap_clean - take from release.

- **Bootstrap** - making and signing by secret part of your operator key:

```
export MAG200_OP_KEY=ID
./bootstrap_sign_250.sh
```

Where:

ID - your key ID

Bootstrap_250 will be created in directory **images** - this is your **signed Bootstrap**

- **Kernel** - signing by secret part of your operator key:

```
./kernel_sign_250.sh
```

Signed kernel will be generated - **uImage_mag250**

- Specify in file img_make.profile.mag250:

```
export MAG200_OP_KEY=ID
```

Where: **ID** - your key ID

- Specify next variable in file **env.txt**:

```
loppubKEY=yes
```

loppubKEY - Key reset blocking from **Bootloader** (Attention! **loppubKEY** doesn't work in U-But versions which is under **07**)

- Making of your firmware image:

Make firmware image STB MAG-200/250 General instruction about firmware image making, Attention! In this instruction the default key is used.

Installation of the key in STB and firmware image update

For installation of the key in STB, which is signed by manufacturer, it is necessary:

- Th configure DHCP server with specified of necessary options for installing of the key and firmware updating;
- Enter the **Bootloader menu** and choose **Upgrade Tools → Set LOGO&Key** - STB will download and install key & Bootloader logo (if present).
- Choose **Exit&Save** in Bootloader menu. STB will reboot and the update firmware process will be start.

Attention! After updating you can't install another firmware image (which isn't signed by your key) to STB!

Operator Guide. About DHCP configuration: pages 11, 17

DHCP server configuration. Example:

In example specified options of key installation from the custom URL and firmware image update with checking version number.

```
option space Infomir;
option Infomir.autostart          code 1 = text;
option Infomir.bootargs           code 2 = text;
option Infomir.mcip               code 3 = ip-address;
option Infomir.mcport             code 4 = integer 16;
option Infomir.oppubfile          code 9 = text;
option Infomir.mcip_img           code 10 = ip-address;
option Infomir.mcport_img         code 11 = integer 16;
option Infomir.mcip_mng           code 12 = ip-address;
option Infomir.mcport_mng         code 13 = integer 16;
option Infomir.ip_log             code 14 = ip-address;
option Infomir.port_log           code 15 = integer 16;
option Infomir.logo_x             code 16 = integer 16;
option Infomir.logo_y             code 17 = integer 16;
option Infomir.bg_color           code 18 = integer 32;
option Infomir.fg_color           code 19 = integer 32;
option Infomir.VerNumber          code 20 = text;
option Infomir.DateTime           code 21 = text;
option Infomir.portal_dhcp        code 22 = text;
option Infomir.timezone           code 23 = text;
option Infomir.update_url         code 24 = text;
option Infomir.update_sboot       code 25 = text;
option Infomir.update_ver         code 26 = text;
option Infomir.update_mode        code 27 = text;
option Infomir.update_sboot_ver   code 28 = text;
```

```
class "MAG250_boot" {
match if (( option vendor-class-identifier="InfomirMAG250boot"));
filename "mag250/uImage_mag250_opkey";
next-server 10.1.0.1;
option root-path "10.1.0.1:/srv/mag250";
option ntp-servers 10.1.0.1;
vendor-option-space Infomir;
}


class "MAG250_upglogo" {
match if (( option vendor-class-identifier="InfomirMAG250upglogo"));
filename "mag250/logo.bmp.gz";
next-server 10.1.0.1;
option ntp-servers 10.1.0.1;
vendor-option-space Infomir;
option Infomir.logo_x 0;
option Infomir.logo_y 0;
option Infomir.bg_color 0x00000000;
option Infomir.fg_color 0x00ffffff;
option Infomir.oppubfile "mag250/OP.KEY";
}


class "MAG250_vendor" {
match if (( option vendor-class-identifier="InfomirMAG250"));
next-server 10.1.0.1;
option ntp-servers 10.1.0.1;
vendor-option-space Infomir;
option Infomir.update_url "tftp://10.1.0.1/mag250/imageupdate_250_211_opkey";
option Infomir.update_ver "211";
option Infomir.update_mode "tftp://10.1.0.1/mag250/Bootstrap_250_211_opkey";
option Infomir.portal_dhcp "http://10.1.0.1/stalker_portal/c/index.html";
}
```

Where:

```
filename "mag250/uImage_mag250_opkey"; - Kernel, which is signed by your key.
option root-path "10.1.0.1:/srv/mag250"; - Path to rootfs in the case if the
STB should load form NFS.
option Infomir.update_ver "211"; - Image version number.
option Infomir.update_url "tftp://10.1.0.1/mag250/imageupdate_250_211_opkey";
- URL of the image, which is signed by your key, on wich STB should updated.
option Infomir.update_mode "tftp://10.1.0.1/mag250/Bootstrap_250_211_opkey";
- Bootstrap, which is signed by your key.
option Infomir.oppubfile "mag250/OP.KEY"; - Public part of the key, which is
signed by manufacturer in the text form (takes from file, which was send by
manufacturer)
```

Notes concerning gpg program operation

The program **gpg** is used for working with keys and creating the digital signature of images.

To transfer the key from one device to another you may use the following commands:

for preserving the information of the key in the file and

```
gpg -o opsecbin.KEY --export-secret-keys "key ID"
```

for adding this key to gpg and

```
gpg --import opsecbin.KEY
```

for viewing currently available keys

```
gpg --list-keys
```

The utilities creating the image for upgrade and signing the core and bootstrap use the operator key according to “key ID ”.