ISMtools v1.0.4

Automation tools for Fujitsu Software Infrastructure Manager (ISM) using REST API.

These are some scripts/examples to automate tasks regarding ISM. Of course there are much more things possible. But you can use this toolset as a platform for own extensions.

Please keep in mind, that the ISM advanced license is required to use the REST API [ISM_REST_API]!

The number of tools/scripts might increase over time …

Any feedback is appreciated.

Please note: This toolset is provided W/O ANY WARRANTY and is to be used at your own risk!

Table of Contents

History
Installation
- Requirements
- Setup

Quick overview
- General options
- List of commands

Commands
- Configfile
- Used ENV vars
- Commandline options
- Commands

Security concerns
Bibliography

History

Table 1. ChangeLog

Version	Date	Description	Author
1.0.4	2023-02-05	Added paragraph regarding security concerns	Jürgen Orth
1.0.3	2023-01-12	Added this file as PDF below dir doc	Jürgen Orth
1.0.2	2023-01-11	Initial version with a set of about 20 scripts	Jürgen Orth

Installation

Requirements

This toolset is intended to be used in Linux environments. Alternatively it can be used in Windows environments with activated WSL (Windows Subsystem Linux) and installed Linux like *Ubuntu*, *Debian*, *Fedora*, *Rocky* or *OpenSUSE* for example from *Microsoft Store*. You can also use CygWin (In this case WSL is not needed).

Following commands are required:

- Bash (including common tools like awk, sed, grep, host, openssl, …)
- cURL (for talking with iRMC)
- jq (for filtering/processing of JSON data - needed by some scripts)
- optional git and/or wget

If some of those packages are not installed then you normally can get them by using your OS' paket manager e.g. `yum`, `zypper` or `apt`.

Setup

To use this toolset run the following steps:

Clone the repository:

```
$ git clone https://github.com/fujitsu/ISMtools.git
$ cd ISMtools
$ export PATH=$PATH:$PWD/scripts
```

or download this ZIP file e.g. with `wget` or with your internet browser and unzip it accordingly.

```
$ wget https://github.com/fujitsu/ISMtools/archive/refs/heads
/master.zip # (1)
$ unzip master.zip
$ cd ISMtools-main
$ export PATH=$PATH:$PWD/scripts
```

(1) You can also use `curl -sLo master.zip https://github.com/fujitsu/ISMtools/archive/refs/heads/master.zip` instead of `wget`.

Quick overview

All tools below dir `scripts` (don't forget to add this dir to your PATH var) start with `ism_` in their script names. The second part of the name, e.g. `list` indicates the intended functionality and the optional third one some further description. So `ism_list_nodes` will list all registered nodes in formatted JSON format whereas `ism_show_racks` will display all *RackIds* and their corresponding *RackNames* in a formatted table view.

General options

All scripts have a basic set of options. Some might have more options as described later in this document.

- `-h` for usage help
- `-i` to define ISM VA (overrides default in `.ism_env`)
- `-u` to enter user credentials (overrides default in `.ism_env`)
- `-d` to set a debug level (overrides default in `.ism_env`)

Additional you can define ENV vars `ISM_VA` (ISM virtual appliance), `ISM_USER` or `DEBUG` like `export ISM_VA=myism.mydomain.net:25566`, which override defaults in `.ism_env`, too. Precedence is: Commandline option, ENV var, config file.

List of commands

- Configfile
  Configuration file `.ism_env` is mainly used to define common settings like IP address of ISM-VA, username, password, …
  There are also some helper functions included.
- `ism_show_env`
  Displays the effective settings depending on the configuration file, ENV vars or given options.
- `ism_chk_con`
  Displays output of `ism_show_env` and checks the connection to the ISM VA to see if you can communicate with the REST API and everything is fine (password or session id for example).
- `ism_login`
  Creates an ISM session. Should be used like `eval $(ism_login)`.
- `ism_logout`
  Ends an ISM session. Should be used like `eval $(ism_logout)`.
- `ism_cmd`
  Basic command to use the REST API. Used by all other commands. Output is native JSON.
- [ism_list_assets]
  List all asset information in formatted JSON format.
- `ism_list_firmware`
  List all firmware information in formatted JSON format.
- `ism_list_inventory`
  List all inventory information in formatted JSON format.
- `ism_list_events <nodename>|<nodeip>|<nodesn>`
  List all events for a given node in formatted JSON format.
- `ism_list_nodes [<filter>]`
  List all node information in formatted JSON format.
- `ism_list_traps <nodename>|<nodeip>|<nodesn>`
  List all traps for given node in formatted JSON format.
- `ism_j2c [NODE|EVENT|TRAP|FIRMWARE|ASSET|<ColumnSpec>]`
  Converts output of `ism_list_*` commands from JSON to CSV with the specified columns.

- `ism_get_nodeid <nodename>|<nodeip>|<nodesn>`
  Displays the *NodeId* of given node.
- `ism_get_rackid <RackName>`
  Displays the *RackId* of given rack name.
- `ism_get_sysrep [-o <outputfile>] <nodename>|<nodeip>|<nodesn>`
  Creates and download ZIP file with system event log and SystemReport (XML-Format).
- `ism_add_server [<inputfile>]`
  Registers servers listed in given input file.
- `ism_run_refreshnodes [<filter>]`
  Updates/refreshs the info and status of nodes.
- `ism_run_gfupdate [-s]`
  Updates the ISM internal repository from *GlobalFlash*. With option `-s` only firmware for registered components are downloaded.
- `ism_set_thresholds [<warn> [<critical> [<filter>]]]`
  Sets power thresholds for all/given nodes.
- `ism_show_racks`
  Display an overview of racks with *RackId* and *RackName*.
- `ism_show_isos`
  Display imported ISO files.
- `ism_del_iso [<id>]`
  Deletes given ISO. Without param the command runs in interactive mode.

Commands

Configfile

`.ism_env` contains defaults to make things more comfortable.

```
#!/bin/bash
# (c) Juergen Orth ;-)
# $Id: .ism_env 166 2023-01-10 11:23:42Z HMBJOrth $
# for documentation see https://github.com/fujitsu/ISMtools
#
# Settings and tools for ISMtools based on bash and curl

# IP, Name or FQDN of ISM VA with optional portnumber
ISM_VA_DEFAULT=ism.customer.net
```

```
# ISM VA default portnumber
ISM_PORT_DEFAULT=25566
# User and passwort. Format username:password
ISM_USER_DEFAULT=administrator:admin
# Debug settings: 0=none, 1=few, 2=more, 3=much more debug output
DEBUG_DEFAULT=0

# CERT file. Must not exist.
CACERT=${0%/*}/DCMA.crt
# Default options for cURL - --silent suppresses progress bar
CURLOPTIONS="--silent"
# LOGFILE: to see some log output of commands
LOGFILE=/tmp/ISMtools-$$.log
# OUTPUTFILE: to see some output of commands
OUTPUTFILE=/tmp/ISMtools-$$.zip
# TMPFILE: for temporary files
TMPFILE=/tmp/ISMtools-$$

############################################################
# Don't change lines below
############################################################

# Define vars PROG, DIR and expand PATH to find subcommands
... (truncated)
```

Used ENV vars

- `ISM_VA`: IP-address, name or FQDN of iRMC and optional port number like `ism.customer.net:4711`.
- `ISM_USER`: User credentials in format `user:password`
- `DEBUG`: If set (e.g. `export DEBUG=1`) the scripts will output debug information to *stderr*. As higher the number as more output will be produced.
- `ISM_session`: These var is set by command `eval $(ism_login)` and is used to handle ISM sessions. They should be unset with command `eval $(ism_logout)`.
- `WARNING`: If set a warning message appears when https data is not confirmed by certificate. Use it like `export WARNING=true`.

Commandline options

Generic options for all commands:

- `-h`

Gives a short overview for possible options of a command.

- `-i <ISMname>|<ISMip>|<ISMfqdn>[:<portnum>]`
  Overrides settings in `.ism_env` and ENV var `ISM_VA`.
- `-u <username>:<password>`
  Overrides settings in `.ism_env` and ENV var `ISM_USER`.
- `-d <debuglevel>`
  Overrides settings in `.ism_env` and ENV var `DEBUG`.

These general options are not described again below. Further options that are specific for some command will be explained at the corresponding command.

Commands

`ism_show_env`

Display the current environment that would be effective when running one of `irmc_xxx` scripts:

```
$ ism_show_env -i 10.172.125.109
2022-12-23 11:44:51 -- Effective settings:
                       ISM_VA:       10.172.125.109:25566
                       ISM_FQDN:     tvm-ism109.bupc-test.hmb.fsc.net.
                       ISM_IP:       10.172.125.109
                       ISM_USER:     administrator:admin
                       ISM_session:
                       CACERT:       /tmp/ism/DCMA.crt
                       JSON:         jq . (1)
```

(1) The `jq` tool is available which is needed for some scripts.

`ism_chk_con`

This command checks the connection. So you can see if you can use the REST API of ISM_VA. Additional the current settings from `ism_show_env` are displayed.

```
$ ism_chk_con -i 10.172.125.109
2022-12-23 11:58:51 -- Effective settings:
                       ISM_VA:       10.172.125.109:25566
                       ISM_FQDN:     tvm-ism109.bupc-test.hmb.fsc.net.
                       ISM_IP:       10.172.125.109
                       ISM_USER:     administrator:admin
                       ISM_session:
                       CACERT:       /tmp/ism/DCMA.crt
```

```
                            JSON:         jq .
        2022-12-23 11:58:52 -- Connection OK (1)

        $ ism_chk_con -i 10.172.125.109 -u administrator:wrongPW
        2022-12-23 12:03:38 -- Effective settings:
                            ISM_VA:       10.172.125.109:25566
                            ISM_FQDN:     tvm-ism109.bupc-test.hmb.fsc.net.
                            ISM_IP:       10.172.125.109
                            ISM_USER:     administrator:wrongPW
                            ISM_session:
                            CACERT:       /tmp/ism/DCMA.crt
                            JSON:         jq .
        2022-12-23 12:03:39 -- NO Connection (2)
```

(1) This connection is working

(2) This connection couldn't be established

`ism_login`

Used for initiating an ISM session and setting of the required ENV var `ISM_session`. Usage: `eval $(ism_login)`. With an established session there is no need for authentication overhead when doing several requests in a row. Please notice that sessions expire after some time of inactivity!

```
        $ eval $(ism_login -i 10.172.125.109)
        $ ism_show_env
        2022-12-23 12:15:12 -- Effective settings:
                            ISM_VA:       ism.customer.net:25566
                            ISM_FQDN:     ism.customer.net
                            ISM_IP:       169.254.254.254
                            ISM_USER:     administrator:admin
                            ISM_session:  d1b2533efc595f2ef535d97941d80e35
(1)
                            CACERT:       /tmp/ism/DCMA.crt
                            JSON:         jq .
```

(1) This session id is used for further requests.

`ism_logout`

Used for destroying an ISM session and unsetting the session related ENV var. Usage: `eval $(ism_logout)`

`ism_cmd`

Basic command to perfom REST API tasks: Usage: `ism_cmd get|post|patch|delete <endpoint> [other options ..]`. You can write the method in lower or upper case letters and use <endpoint> w/ or w/o leading "/".

Output is in formatted JSON format (one very long line). To beautify output and make it easier to read you can pipe the output to `jq .` or `python -m json.tool` for example.

Possible tool for formatting is displayed in output of [_ism_show_env] at entry *JSON*.

So if you have some documentation in [ISM_REST_API] like:

4.3.2 List Retrieval for Nodes

[Overview]

Retrieves information of all nodes under the management of the user group that the user belongs to.

Query parameters can be used to narrow down the nodes to be retrieved.

[Format]

```
GET /nodes
```

[Query Parameter]

Parameter	Type	Description
name	string	Narrowing down by Node Name

then you can use `ism_cmd` in the following manner:

Example:

```
        $ ism_cmd GET /nodes (1)
        {"MessageInfo":[],"SchemaType":"https://10.172.125.109:25566/ism/schema
/v2/Nodes/Nodes-GET-Out.0.0.1.json","IsmBody":{"Nodes":[{"AdditionalData":
{},"Fabric":[],"ParentFabricId":null,"DataCenterInfo":
{"DcId":null,"Name":null},"SlotNum":null,"UpdateDate":"2022-12-23T06:44:41.931Z
","ChildNodeList":[],"IpAddress":"10.172.124.85","Model":"PRIMERGY RX100
S8","Status":"Normal","Description":null,"AlarmStatus":"Warning","Type":"server
","NodeGroupId":8,"NodeTagList":[],"IpVersion":"V4"
        ... (truncated)
```

(1) `ism_cmd get nodes` or `ism_cmd get "nodes?name=mynodename"` would also be valid examples.

`ism_list_assets`

List all assets in formatted JSON format.

```
        List all inventory data in formatted JSON format.$ ism_list_assets
        {
          "MessageInfo": [],
          "SchemaType": "https://10.172.125.85:25566/ism/schema/v2/Nodes
/NodesInventory-GET-Out.0.0.1.json",
          "IsmBody": {
            "Nodes": [
              {
                "Manufacture": "FUJITSU",
                "MacAddress": "b0-ac-fa-a0-65-cf",
                "Wwnn": null,
                "VariableData": {
                  "Firmware": [
                    {
                      "Function": null,
                      "Slot": null,
                      "Type": "storage",
                      "Name": "ET203AU",
                      "Unified": null,
                      "Bus": null,
                      "Device": null,
                      "Model": "ET203AU",
                      "Segment": null,
                      "FirmwareVersion": "V10L90-3000"
                    }
                  ],
                  "Raid": [
                    {
                      "Status": "Available",
                      "Name": "EXCP0000",
                      "Level": "RAID0",
                      "Disks": 1,
                      "Number": 0,
                      "FreeCapacity": 0,
                      "TotalCapacity": 374528
                    },
        ... (truncated)
```

ism_list_firmware

List all firmware data in formatted JSON format. This is nearly the same as [ism_list_assets]. The difference is that only *Firmware* will be displayes in *VariableData*. So output size is much smaller.

```
        $ ism_list_firmware
        {
          "MessageInfo": [],
          "SchemaType": "https://10.172.125.85:25566/ism/schema/v2/Nodes
/NodesInventory-GET-Out.0.0.1.json",
          "IsmBody": {
            "Nodes": [
              {
                "Manufacture": "FUJITSU",
                "MacAddress": "b0-ac-fa-a0-65-cf",
                "Wwnn": null,
                "VariableData": {
                  "Firmware": [
                    {
                      "Function": null,
                      "Slot": null,
                      "Type": "storage",
                      "Name": "ET203AU",
                      "Unified": null,
                      "Bus": null,
                      "Device": null,
                      "Model": "ET203AU",
                      "Segment": null,
                      "FirmwareVersion": "V10L90-3000"
                    }
                  ]
                },
                "Name": "ET-DX200S3-C11",
                "HardwareLogTarget": 1,
                "SerialNumber": "4601547358",
                "ServerViewLogTarget": 0,
                "NodeId": 10115,
                "ProductName": "ETERNUSDXLS3 ET203AU",
                "UpdateDate": "2023-01-05T06:36:03.270Z",
                "Progress": "Complete",
                "RaidLogTarget": 0,
                "SoftwareLogTarget": 0
              },
        ... (truncated)
```

ism_list_inventory

List all inventory data in formatted JSON format.

```
        $ ism_list_inventory
```

```
        {
          "MessageInfo": [],
          "SchemaType": "https://10.172.125.85:25566/ism/schema/v2/Nodes
/NodesInventory-GET-Out.0.0.1.json",
          "IsmBody": {
            "Nodes": [
              {
                "Manufacture": "FUJITSU",
                "MacAddress": "b0-ac-fa-a0-65-cf",
                "Wwnn": null,
                "VariableData": {
                  "Firmware": [
                    {
                      "Function": null,
                      "Slot": null,
                      "Type": "storage",
                      "Name": "ET203AU",
                      "Unified": null,
                      "Bus": null,
                      "Device": null,
                      "Model": "ET203AU",
                      "Segment": null,
                      "FirmwareVersion": "V10L90-3000"
                    }
                  ],
                  "Raid": [
                    {
                      "Status": "Available",
                      "Name": "EXCP0000",
                      "Level": "RAID0",
                      "Disks": 1,
                      "Number": 0,
                      "FreeCapacity": 0,
                      "TotalCapacity": 374528
                    },
        ... (truncated)
```

ism_list_events <nodename>|<nodeip>|<nodesn>

List all events in formatted JSON format for given node.

```
        $ ism_list_events EWAL001056
        {
          "SchemaType": "https://10.172.125.85:25566/ism/schema/v2/Event
/EventHistoryEventShow-GET-Out.0.0.1.json",
          "MessageInfo": [],
```

```
          "IsmBody": {
            "Logs": [
              {
                "Id": "478966",
                "OccurrenceDate": "2023-01-05T09:31:15.547Z",
                "Type": "asynchronous operation complete",
                "Level": "info",
                "MessageId": "10020303",
                "Message": "Reacquisition of node information was completed.",
                "TargetInfo": {
                  "Name": "rx4770m6-4-112",
                  "ResourceIdType": "NodeId",
                  "ResourceId": 10180
                },
                "Operator": null
              },
        ... (truncated)
```

`ism_list_nodes [<filter>]`

List all node data (that is accessible for the user group the current user belongs to) in formatted JSON format. Output can be filtered with following filter keywords (that can be combined if nececssary):

Possible filter keywords are:

`name, type, model, ipaddress, rackid, floorid, dcid, nodegroupid, status, alarmstatus, nodetag, uniqinfo`

So if you want to output all data of nodes for a given *rack id* that are in status *Warning* then you could do it like this:

```
        $ ism_list_nodes "rackid=1&status=Warning" ❶
        {
          "MessageInfo": [],
          "SchemaType": "https://10.172.125.85:25566/ism/schema/v2/Nodes/Nodes-
GET-Out.0.0.1.json",
          "IsmBody": {
            "Nodes": [
              {
                "AdditionalData": {},
                "Fabric": [],
                "ParentFabricId": null,
                "DataCenterInfo": {
                  "DcId": 1,
                  "Name": "TEST DC FFM"
```

```
                },
                ... (truncated)
```

1. Please note that you have to use single or double quotes for the filter as the "&" character has a special meaning for the bash interpreter.

`ism_list_traps <nodename>|<nodeip>|<nodesn>`

List all traps in formatted JSON format for a given node.

```
        $ ism_list_traps EWAB001946 (1)
        {
          "MessageInfo": [],
          "SchemaType": "https://10.172.125.85:25566/ism/schema/v2/Event
/EventHistoryTrap-GET-Out.0.0.1.json",
          "IsmBody": {
            "TrapLogs": [
              {
                "TrapLogId": "3252753",
                "TrapMessage": "Received from 10.172.126.150. Authentication
failure: Unauthorized message received.",
                "ResourceType": "Node",
                "TimeStamp": "2023-01-05T08:28:27.989Z",
                "OID": ".1.3.6.1.6.3.1.1.5.5",
                "TrapType": "authenticationFailure",
                "ResourceId": 10145,
                "Severity": "Minor"
              },
        ... (truncated)
```

1. In this example serial number is used to define node.

`ism_j2c [NODE|EVENT|TRAP|FIRMWARE|ASSET|<ColumnSpec>]`

Converts JSON to CSV. JSON data is read from *STDIN* and written to *STDOUT*. You can only specify keys at level three of the JSON input. Parameters `NODE`, `EVENT` etc. define example `ColumnSpecs` for the corresponding ism_list_* command. If no parameter is given then `NODE` is assumed.

```
        $ ism_list_nodes "type=server&rackid=1" | ism_j2c
'["UniqInfo","IpAddress"]' (1)
        "sep=,"
        "UniqInfo","IpAddress"
        "MACK001036","10.172.124.101"
```

```
"EWAL001056","10.172.124.113"
"YLNS001039","10.172.124.125"
"YM6D024205","10.172.124.233"
"YLVT001989","10.172.124.87"
"YMSQ002118","10.172.124.225"
"YM6D009446","10.172.124.145"
"YLNV001022","10.172.124.203"
"YMTJ001026","10.172.124.221"
"YM6D024204","10.172.124.231"
```

1 Please note the quoting which is necessary!

`ism_get_nodeid <nodename>|<nodeip>|<nodesn>`

Extracts the NodeId for then specified node. If the name contains spaces or other special characters it has to be quoted.

```
$ ism_get_nodeid EWAL001056
10180
```

`ism_get_rackid <RackName>`

Extracts the RackId for a given Rackname. If the name contains spaces or other special characters it has to be quoted.

```
$ ism_get_rackid "HQ Server Rack #1"
9
```

`ism_get_sysrep [-o <outputfile>] <nodename>|<nodeip>|<nodesn>`

Creates and downloads a System-Report ZIP file which contains the system report and the system event log (SEL). If no outputfile is given then default value *OUTPUTFILE* defined in Configfile is used.

```
$ ism_get_sysrep EWAL001056
2023-01-05 10:16:22 -- Log in to ISM if necessary ...
2023-01-05 10:16:25 --    Session_Id=fc045d8db0565cb83f8e1f649202cab7
2023-01-05 10:16:26 -- Retrieving NodeId
2023-01-05 10:16:28 --    NodeId=10180 for EWAL001056
2023-01-05 10:16:28 -- Start Systemreport generation
2023-01-05 10:16:30 --    TaskId=396 - waiting for finishing  ...
2023-01-05 10:16:52 -- Complete Success
2023-01-05 10:16:53 -- Creating Systemreport
```

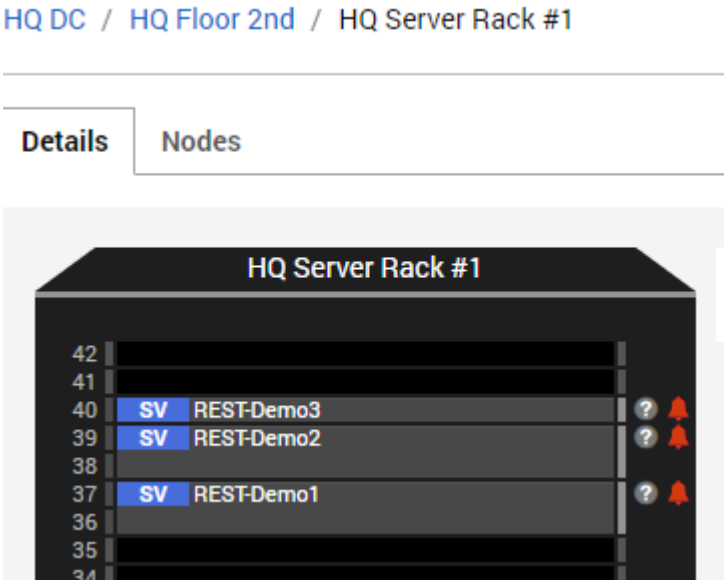
```
        2023-01-05 10:16:54 --    TaskId=397 - waiting for finishing
        2023-01-05 10:16:57 -- Complete Success
        2023-01-05 10:16:57 -- Create ZIP file
        2023-01-05 10:16:59 --    ZIP file=https://10.172.125.85:25566/ism/data
/export/Administrator/transfer/Archive/fc045d8db0565cb83f8e1f649202cab7
/download/archivedlog/397/ArchivedLog_20230105101654.zip
        2023-01-05 10:16:59 -- Download ZIP file to /tmp/ISMtools.out
        2023-01-05 10:17:04 -- Result file /tmp/ISMtools.out (Size=39K /
Type=ZIP)
        2023-01-05 10:17:04 -- Logging out
        2023-01-05 10:17:06 -- Finished
```

`ism_add_server [<inputfile>]`

Registers new servers to your ISM VA. Input data is read from *inputfile*. If it is omitted then default file `ism_nodes.csv` in the same directory as the `ism_add_server` command is taken. The syntax can be seen in example below. Empty lines and such with "#" at the beginning are ignored. If you do not like to enter mounting position enter `null` for the corresponding entry.

```
        $ cat ism_nodes.csv
        MODEL;NAME;DESC;SERVER;USER;PW;RACK;POS;HE;TAGS
        PRIMERGY RX2540 M6;REST-Demo1;Added by
script;10.172.124.223;admin;admin;9;36;2;REST-API Testserver JO
        PRIMERGY RX4770 M4;REST-Demo2;Added by
script;10.172.124.247;admin;admin;9;38;2;REST-API Testserver JO
        PRIMERGY RX2530 M1;REST-Demo3;Added by
script;10.172.124.147;admin;admin;9;40;1;REST-API Testserver JO

        $ ism_add_server ism_nodes.csv
        2023-01-05 17:18:20 -- Registering node 10.172.124.223 ... OK
        2023-01-05 17:18:29 -- BG-Retrieving information from NodeID 10290 ...
PID=2635

--------------------------------------------------------------------------------
        2023-01-05 17:18:30 -- Registering node 10.172.124.247 ... OK
        2023-01-05 17:18:38 -- BG-Retrieving information from NodeID 10291 ...
PID=2671

--------------------------------------------------------------------------------
        2023-01-05 17:18:39 -- Registering node 10.172.124.147 ... OK
        2023-01-05 17:18:48 -- BG-Retrieving information from NodeID 10292 ...
PID=2707

--------------------------------------------------------------------------------
```

After this the new servers should appear within 3D view:



`ism_run_refreshnodes [<filter>]`

Retrieves current node infos. Without argument all nodes are refreshed. If you want to refresh only specific nodes just enter a [filter].

This might be useful to update node infos after changes (e.g. FW) as ISM does this only once a day.

```
        $ ism_run_refreshnodes "type=server&rackid=1"
        2023-01-05 10:29:40 -- Reading node list ...
        2023-01-05 10:29:44 -- BG refreshing NodeId 10177 (10.172.124.101)
[PID=2264]
        2023-01-05 10:29:44 -- BG refreshing NodeId 10180 (10.172.124.113)
[PID=2266]
        2023-01-05 10:29:44 -- BG refreshing NodeId 10181 (10.172.124.125)
[PID=2269]
        2023-01-05 10:29:45 -- BG refreshing NodeId 10118 (10.172.124.233)
[PID=2274]
        2023-01-05 10:29:45 -- BG refreshing NodeId 10157 (10.172.124.87)
[PID=2280]
        2023-01-05 10:29:45 -- BG refreshing NodeId 10191 (10.172.124.225)
[PID=2286]
        2023-01-05 10:29:46 -- BG refreshing NodeId 10192 (10.172.124.145)
[PID=2293]
        2023-01-05 10:29:47 -- BG refreshing NodeId 10230 (10.172.124.203)
[PID=2301]
        2023-01-05 10:29:47 -- BG refreshing NodeId 10143 (10.172.124.221)
[PID=2306]
        2023-01-05 10:29:48 -- BG refreshing NodeId 10117 (10.172.124.231)
[PID=2313]
```

`ism_run_gfupdate [-s]`

This command refreshs (synchronizes) the ISM VA internal repository with Fujitsu's internet repository (aka GlobalFlash). Without argument all available firmware/driver components are downloaded. When you use option `-s` then it runs in *smart* mode which means only software components are downloaded for servers and their components that are registered in ISM VA.

> You should have enough disk space within your ISM VA to prevent it from running out of space. The whole GlobalFlash repository needs more than 20 GByte!

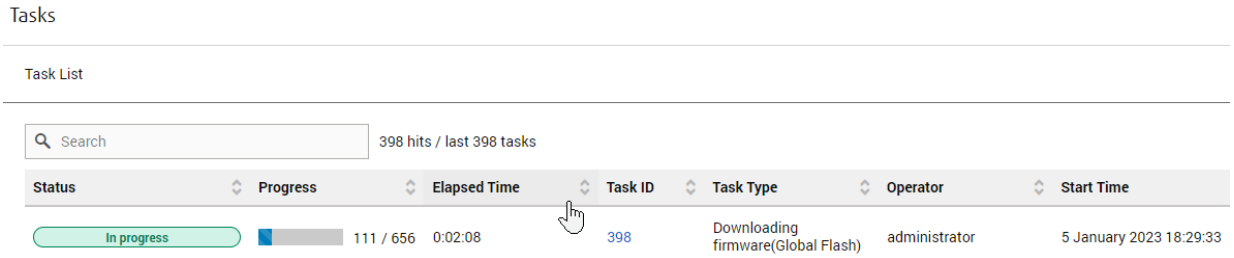
As it is a good idea to synchronize your ISM VA repository on regurlar schedule you should add a line to your crontab to archive this like:

```
0 23 * * * ism_run_gfupdate -s
```

Then this job is done each day at 11pm.

```
        $ ism_run_gfupdate -s
        2023-01-05 18:27:35 -- Retrieving meta data - Please wait ~2 minutes
...  done
        2023-01-05 18:28:56 -- Saving meta data.
        2023-01-05 18:29:00 -- Smart filtering in progress. This takes some
time ...
        2023-01-05 18:29:29 -- Starting download of firmware/drivers ...
        {
          "SchemaType": "https://10.172.125.85:25566/ism/schema/v2/System
/SystemSettingsFirmwareFtsFirmwareDownload-POST-Out.0.0.1.json",
          "MessageInfo": [],
          "IsmBody": {
            "TaskId": "398",
            "CancelUri": "https://10.172.125.85:25566/ism/api/v2/system
/settings/firmware/ftsfirmware/download/cancel"
          }
        }
        2023-01-05 18:29:35 -- Cleaning up.
```

Then you can see a task within the GUI that is downloading the required software components to ISM VA. Of course this task can take a long time depending on how many components have to be downloaded.



ism_set_thresholds [<warn> [<critical> [<filter>]]]

This command defines some power threscholds for nodes. If power consumption is about *warning* or *critical* threshold then an event is raised. Systems with warning or critical values can also be seen in 3D view when you select "Power Consumption".

Without arguments defaults values will be used. You can see them in the example below:

```
        $ ism_set_thresholds
```

```
        2023-01-05 17:58:23 -- Log in to ISM if necessary ...
        2023-01-05 17:58:27 --    Session_Id=5f6b3a3fb9587f464dd62943d1acdadb
        2023-01-05 17:58:27 -- Using filter "type=server&nodetag=powercheck"
        2023-01-05 17:58:27 -- Setting upper power thresholds (300W/400W) to:
rx100s8-124-84-irmc rx2530m6-4-77
        2023-01-05 17:58:35 -- Logging out
```

ism_show_racks

Shows *RackId* and *RackName* for all racks as table.

```
        $ ism_show_racks
        RackId  RackName
        ======= ==============================
        1       Test Rack #2 42HE
        7       HQ Storage Rack #2
        8       Test DC FFM #1 Storage Rack
        9       HQ Server Rack #1
        10      Test DC FFM #3 Infrastruktur Rack
        14      R1
```

ism_show_isos

Shows the ISO files that have been uploaded to ISM VA.

```
        $ ism_show_isos
         ID     Filename
        ===     ============================
        9       VMware-ESXi-7.0.3.update03-19193900-Fujitsu-v530-1.iso
        2       VMware-ESXi-6.7.0-14320388-Fujitsu-v480-1.iso
        3       en_windows_server_2019_updated_april_2020_x64_dvd_12d6dc63.iso
        10      SVIM14.21.11.07.iso
        11      SLE-15-SP4-Full-x86_64-GM-Media1.iso
        5       rhel-8.0-x86_64-dvd.iso
        6       SVIM13.20.10.06.iso
        7       VMware_ESXi_7.0.0_15843807_Fujitsu_v500_1.iso
        8       VMware-ESXi-7.0.1.update01-16850804-Fujitsu-v510-1.iso
```

ism_del_iso [<id>]

Deletes uploaded ISO files. If no argument is given then it runs in interactive mode (can be cancelled by SIGINT signal, Ctrl-C).

```
        $ ism_del_iso
```

```
 ID Filename
=== ============================
  9 VMware-ESXi-7.0.3.update03-19193900-Fujitsu-v530-1.iso
  2 VMware-ESXi-6.7.0-14320388-Fujitsu-v480-1.iso
  3 en_windows_server_2019_updated_april_2020_x64_dvd_12d6dc63.iso
 10 SVIM14.21.11.07.iso
 11 SLE-15-SP4-Full-x86_64-GM-Media1.iso
  5 rhel-8.0-x86_64-dvd.iso
  6 SVIM13.20.10.06.iso
  7 VMware_ESXi_7.0.0_15843807_Fujitsu_v500_1.iso
  8 VMware-ESXi-7.0.1.update01-16850804-Fujitsu-v510-1.iso
Please enter ID that should be deleted: 4711
invalid ID - Try again
Please enter ID that should be deleted:
... (truncated)
```

Security concerns

Even if it is possible to enter user names and passwords via commandline parameters to all commands: This should be used only in non critical environments (e.g. for testing). Otherwise this data could be read by any other user (e.g. by `ps -ef`).

The preferred and secure way to provide those critical data to the scripts is by defining those data in either `.ism_env` file, in your `~/.profile` (or `~/.bash_profile`) file or by exporting vars in your shell (e.g. `export ISM_USER=administrator:mysecretpassword`).

And of course, those files should be readable only by their owner (e.g. `chmod go-rwx .ism_env ~/.profile`)!

Bibliography

- [ISM_REST_API] Fujitsu: REST API Reference Manual, October 2022

Further links to documents, API specifications, tools and more can you find here.

You can convert/render this document to HTML with command line tool `asciidoc` or can open it in your favorite browser after installing the `Asciidoctor.js` addon.