Pushing Performance

People | Power | Partnership

HARTING MICA Python Demo Container User Guide

Contents

1 Quickstart:

Connect your MICA to a webbrowser of your choice. Using Windows, you can access your MICA via

```
https://devicename
```

If your host is running an mdns application, you can access your MICA also via

```
https://devicename.local
```

You have to accept a Security Certificate. After Container installation, you have to configure network settings via Firmware -> Settings -> Network in order to access the Container via SSH.

2 SSH:

Under Windows, you can use putty to SSH into container by

```
containername-devicename
```

If your host is running an mdns application, you can access your container also via

```
containername-devicename.local
```

If you are using Linux and intend to connect to container by name via IPv6 make sure to resolve the link-local address first and append the network identifier. So eg. if using avahi as mdns client, run:

```
avahi-resolve -n -6 containername-devicename.local
```

in order to get the container IPv6 link-local address. Then connect via SSH:

```
ssh root@linklocal%zone-id
```

You need to append the zone-id / network interface, eg. eth0. If you are using ipv4, then using

```
ssh root@containername-devicename.local
```

is sufficient.

By Default login as user

```
root
```

with password

```
root
```

It is recommended to change your password after first login via

```
passwd
```

3 Running Services:

- SSH-Server running on port 22 (dropbear)
- mDNS Client (avahi-daemon)
- https://devicename/containername/API shows this README file
- https://devicename/containername/ABOUT shows container Meta information in JSON

Note: Running a HTTP webserver under port 80 will only be made available by MICA Basesystem via https://devicename(.local)/containername/. The webserver is redirected to port 443, hence encrypted, regardless of hosting a webserver with or without encryption within the container. Similar port redirection applies for websocket services running under port 8080.

4 Web Icons:

Web Icons **App.png** and **Header.png** can be found under **/META/web_icons**. In case new Web Container Icons are desired one can simply replace these files.

5 IPv6

Each MICA container is configured with two fixed IPv6 addresses. The Link Local address constitutes like

```
fe80::a:ed<mica-entity>:<mica-base/container>
```

with prefix

```
64
```

The first 11 Bytes are fixed for the HARTING domain, whereas the following 3 Bytes define the MICA entity. You can get this information from the MICA label on the back of your device. The second half of the MAC address represent these 3 Bytes. So if your MAC address was

```
00:0a:ed:49:92:ff
```

your IPv6 Link-local address of your mica would be

```
fe80::a:ed49:92ff:0
```

The last 2 Bytes is showing you the MICA base system / container identifier, whereas **0x0000**stands for MICA base system. Containers are iterated from number **0x8001**. So the first installed container will get **0x8001**, the second **0x8002** and so on.

The unique local address (ULA) can be derived quite similarly:

```
fd96:8d76:d432:0:a:ed<mica-entity>:<mica-base/container>
```

with prefix

```
64
```

Note: Make sure to assign a ULA with same prefix for you network interface in order to reach your mica/container via ULA.

6 Description

The Python Demo Container is a Busybox based container intended to develop python code via Web editor and interactive Web Shell. It runs under MICA Firmware versions >= 1.0.0.

Besides Python 3.4 (implemented via pyrun) this container provides:

- python3 modules jsonrpc, flipflop and websocket-client
- shellinaboxd in order to provide a webshell

7 Python Web-Shell

The Python webshell can be accessed via the container Web-GUI while logged in to MICA. The shell service is provided by shellinaboxd and runs under port 4200.

8 Python Editor JSON RPC HTTP Service

The functionality of the web editor is based on a fast common gateway interface (fast CGI) using lighttpd. The CGI python scripts implement JSON RPC calls that can be accessed (see also MICA Programming Guide for further information) via:

```
https://<mica-device>/<python-container>/editor/
```

The URL, server port etc. can be modified by changing

```
/etc/lighttpd/lighttpd.conf
```

The JSON RPC calls can be found under

```
/var/www/rpc_editor.py
```

The JSON RPC python script also uses the Single sign on service of the MICA base system by validating an auth token before running the requested call. (see also MICA Programming Guide for further information)

Note: for convenience purposes, id and json-rpc properties are left out. JSON RPC parameters are passed as object in this documentation, they can also be passed as array by appending all values in the order of object keys. Each JSON-RPC returns an error-message in case of an error.

8.1 Parameter Declaration

```
string auth token
string project
string script
string script_content
string script output

repeated project projects
repeated script scripts
```

8.2 Initialize Editor (Minimum Role: User)

Get all projects as an array.

Request

```
{
    "method" : "init",
    "params" : { "auth token" : auth token }
}
```

Response

```
{
    "result" : projects
}
```

8.3 Initialize Project (Minimum Role: User)

Activates a project. Get an array of all scripts of a project.

Request

```
{
    "method" : "init project",
    "params" : {
        "project" : project,
        "auth token" : auth token
    }
}
```

Response

```
{
    "result" : scripts
}
```

8.4 Get script content (Minimum Role: User)

Get file content of a script.

Request

```
{
    "method" : "get content",
    "params" : {
        "script" : script,
        "auth_token" : auth_token
    }
}
```

Response

```
{
    "result" : script_content
}
```

8.5 Add Project (Minimum Role: User)

Add a new project by creating a directory under **/home/python**.

Request

```
{
    "method" : "add project",
    "params" : {
```

```
        "project" : project,
        "auth token" : auth token
    }
}
```

Response

```
{
    "result" : "Success" | "project exists!"
}
```

8.6 Remove Project (Minimum Role: User)

Remove an existing project.

Request

```
{
    "method" : "remove project",
    "params" : {
        "project" : project,
        "auth_token" : auth_token
    }
}
```

Response

```
{
    "result" : "Success"
}
```

8.7 Add Script (Minimum Role: User)

Add a new script to the current project.

Request

```
{
    "method" : "add project",
    "params" : {
        "script" : script,
        "auth_token" : auth_token
    }
}
```

Response

```
{
    "result" : "Success" | "file exists!"
}
```

8.8 Remove Script (Minimum Role: User)

Remove a script from the current project.

Request

```
{
    "method" : "remove script",
    "params" : {
        "script" : script,
        "auth_token" : auth_token
    }
}
```

Response

```
{
    "result" : "Success"
}
```

8.9 Save Script (Minimum Role: User)

Save script content.

Request

```
{
    "method" : "save script",
    "params" : {
        "script" : script,
        "content" : content,
        "auth_token" : auth_token
    }
}
```

Response

```
{
    "result" : "Success"
}
```

8.10 Start Script (Minimum Role: User)

Start python script.

Request

```
{
    "method" : "start script",
    "params" : {
        "script" : script,
        "auth_token" : auth_token
    }
```

```
}
```

Response

```
{
    "result" : script_output
}
```