FIDO FDO DEMO

SIMPLE DEMO Using Java Applications and AIO server

Clone the github repository: https://github.com/fido-device-onboard/pri-fidoiot/tree/master

- From the main folder:
  mvn clean install
- From folder <pri-fidoiot/component-samples/demo/scripts>:
  bash demo_ca.sh
  bash web_csr_req.sh
  bash user_csr_req.sh
  ./keys_gen.sh
  This command generates the folder “secrets” and “service.env” (inside the scripts folder)
- Copy **secrets** and **service.env** in .../component-samples/demo/<component>
  Where <component> = AIO, MANUFACTURER, RV, OWNER, RESELLER
  (Replace existing files)
- Copy only **secrets** folder in <component> = DB (do not copy service.env)

- START DATABASE
  From the folder .../component-samples/demo/db run command
  sudo docker compose up --build -d

- START AIO SERVER
  From the folder .../component-samples/aio (NOTE: it is NOT demo/aio) run command
  mvn clean install
  From the folder .../component-samples/demo/aio, export all environment variables contained in service.env. Example:
  export api_password=default
  export api_user="bla bla bla"
  (check using “echo $api_user”)
  Then, run the command
  java -jar aio.jar

- START DEVICE
  From the folder .../component-samples/device (NOTE: it is NOT demo/device) run command **mvn clean install**
  From the folder .../component-samples/demo/device
  java -jar device.jar

... success

SIMPLE DEMO Using Docker and Aio

- START AIO
  From the folder .../component-samples/aio (NOTE: it is NOT demo/aio) run the command
  mvn clean install

From the folder .../component-samples/demo/aio run the command
sudo docker compose up --build

- START DEVICE
  From the folder .../component-samples/device (NOTE: it is NOT demo/device) run the command
  mvn clean install
  From the folder .../component-samples/demo/device

sudo docker compose up --build

NOTE: Before launching the device app, to start the protocol from scratch (i.e. from the DI phase), delete the device credentials
-> go to folder component-samples/demo/device/app-data and manually delete **credentials.bin**

SIMPLE DEMO Using Individual components

NOTE: before running this demo, it is necessary to do "mvn clean installs":
from main folder pri-fidoiot: mvn clean install
from folder component-samples/aio: mvn clean install
from folder component-samples/device: mvn clean install

SWITCH Digest authentication
By default, all the server components use https and require a TLS certificate to authenticate the requests sent to the various api-endpoints. This behaviour can be disabled by enabling Digest Authentication:
Go to folder component-samples/demo/<component>/**WEB-INF** and change the security constraints inside **web.xml.** In particular, replace and insert this code:

```
<security-constraint>
   <web-resource-collection>
      <web-resource-name>apis</web-resource-name>
      <url-pattern>/api/v1/*</url-pattern>
   </web-resource-collection>
   <auth-constraint>
      <role-name>api</role-name>
   </auth-constraint>
   <user-data-constraint>
     <transport-guarantee>NONE</transport-guarantee>
   </user-data-constraint>
  </security-constraint>
  <login-config>
     <auth-method>DIGEST</auth-method>
  </login-config>
```

Then, back to component-samples/demo/<component> and modify the file **tomcat-users.xml**. In particular, change the line associated with user as follows:

```
<user username="default" password="default" roles="api"/>
```

This must be done for <component> = manufacturer, owner, reseller, rv

NOTE: **default:default** this are username and password to specify inside api requests

DI BETWEEN DEVICE AND MANUFACTURER

START MANUFACTURER
1) Switch Digest Authentication on the Manufacturer
2) Start Manufacturer using docker
   From the folder component-samples/demo/manufacturer, run the command
   sudo docker compose up --build

POST to the il Manufacturer to update the IP address of the RV Server (this is the IP address that will be inserted inside the OVHeader)

```
curl -D - --digest -u default:default --location --request POST
'http://host.docker.internal:8039/api/v1/rvinfo' --header 'Content-Type: text/plain' --data-raw
'[[[5,"host.docker.internal"],[3,8041],[12,2],[2,"127.0.0.1"],[4,8041]]]'
```

Response: 200 OK

START DEVICE
1) Go to folder component-samples/demo/device and change inside the file service.yml the line corresponding to di-url to make it point to port 8039 (this is the port on which the manufacturer is waiting for a connection)

```
di-url: http://host.docker.internal:8039
```

2) If present, delete credentials.bin from folder component-samples/demo/device/app-data
3) Launch the device from folder component-samples/demo/device using the command
   java -jar device.jar

START OWNER
1) Switch Digest Authentication on the Owner
2) Start Owner using docker from the folder component-samples/demo/owner
   sudo docker compose up --build

GET certificate of the Owner (response = owner certificate in PEM format)

```
curl -D - --digest -u default:default --location --request GET
'http://host.docker.internal:8042/api/v1/certificate?alias=SECP256R1' --header 'Content-Type:
text/plain'
```

It is returned the certificate of the Owner. Example:

```
-----BEGIN CERTIFICATE-----
MIIBRDCB7KADAgECAgh2Z8k55hJHnjAKBggqhkjOPQQDAjAXMRUwEwYDVQQDDAxG
ZG9FbnRpdHkgQ0EwIBcNMjQwMTMwMTMyMjE1WhgPMjA1MzA4MjUxMzIyMTVaMBcx
FTATBgNVBAMMDEZkb0VudGl0eSBDQTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0I
A
BD3544yjk3DV3xxRy1RxnVmyGSYLJ6w/7ymwQLEqMnbajGvIgcQlnNB/vHiFdCUK
PFiWj4MP6UGGFvlbH+Bc2UejIDAeMA8GA1UdEwEB/wQFMAMBAf8wCwYDVR0PBAQD
AgGGMAoGCCqGSM49BAMCA0cAMEQCIF/Ly6X9R70YJlteVq8TVCDfPYPjCpOrdx4D
PXMgEK7rAiB+A7QoErpo6bolXbTEE62OYryOvUnhrXk/pTzOArZQ+g==
-----END CERTIFICATE-----
```

GET serial number of the OV released by the Manufacturer. Response = serial number of the OVs generated by the Manufacturer, not extended yet to owner/reseller

```
curl -D - --digest -u default:default --location --request GET
'http://host.docker.internal:8039/api/v1/deviceinfo/100000' --header 'Content-Type: text/plain'
```

The list of OV IDs generated in the last n seconds is returned. Example of response body:

```
[{"serial_no":"232E9EFD","timestamp":"2024-01-31 10:36:37.188","uuid":"0066fc5d-3dff-4326-
ba98-7862986f8f9c","alias":"SECP256R1"},
{"serial_no":"3E4BE84D","timestamp":"2024-01-30 16:02:33.201","uuid":"82ea7dde-bca5-490a-
a4b0-70dc17767996","alias":"SECP256R1"},
{"serial_no":"C9E400DE","timestamp":"2024-01-30 13:22:14.306","uuid":"074eb766-4463-47cf-
b788-7aba81715ff6","alias":"SECP256R1"}]
```

POST to extend the OV to the Owner (Owner becomes the new owner of the device). POST message is sent to the manufacturer, that will sign the new entry containing the public key of the Owner

```
curl -D - --digest -u default:default --location --request POST
"http://host.docker.internal:8039/api/v1/mfg/vouchers/${serial_no}" --header 'Content-Type:
text/plain' --data-raw "owner certificate PEM"
```

Example of request:

```
curl -D - --digest -u default:default --location --request POST
"http://host.docker.internal:8039/api/v1/mfg/vouchers/232E9EFD" --header 'Content-Type:
text/plain' --data-raw "-----BEGIN CERTIFICATE-----
MIIBRDCB7KADAgECAgh2Z8k55hJHnjAKBggqhkjOPQQDAjAXMRUwEwYDVQQDDAxG
ZG9FbnRpdHkgQ0EwIBcNMjQwMTMwMTMyMjE1WhgPMjA1MzA4MjUxMzIyMTVaMBcx
FTATBgNVBAMMDEZkb0VudGl0eSBDQTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0I
A
BD3544yjk3DV3xxRy1RxnVmyGSYLJ6w/7ymwQLEqMnbajGvIgcQlnNB/vHiFdCUK
```

PFiWj4MP6UGGFvlbH+Bc2UejIDAeMA8GA1UdEwEB/wQFMAMBAf8wCwYDVR0PBAQD
AgGGMAoGCCqGSM49BAMCA0cAMEQCIF/Ly6X9R70YJlteVq8TVCDfPYPjCpOrdx4D
PXMgEK7rAiB+A7QoErpo6bolXbTEE62OYryOvUnhrXk/pTzOArZQ+g==
-----END CERTIFICATE-----"

The response contains the extended Ownership Voucher

-----BEGIN OWNERSHIP VOUCHER-----
hRhlWNWGGGVQAGb8XT3/Qya6mHhimG+PnIGFggVVdGhvc3QuZG9ja2VyLmludGVy
bmFsggNDGR9pggxBAoICRUR/AAABggRDGR9pakRlbW9EZXZpY2WDCgFYWzBZMBMG
ByqGSM49AgEGCCqGSM49AwEHA0IABIsqYQAS663sT0G8ncNTimfsXjjF3c0ms430
6F3hXyuUHrIis6J+GESn/qzWs0txGzFAq7mdblZbDlZ/MpcSNaqCL1ggv0UVKWu5
De9cLJ1ZAbxuzqloW95L35QtXTFyX7FyXm2CBVggDmuoPdQjAq9tCXyxKjDIm8Vl
FtNmuiN814Jxz/XcXuqCWQE1MIIBMTCB2aADAgECAggyOJxNk1dFYzAKBggqhkjO
PQQDAjAXMRUwEwYDVQQDDAxGZG9FbnRpdHkgQ0EwIBcNMjQwMTMxMTAzNjM2Wh
gP
MjA1MzA4MjYxMDM2MzZaMBUxEzARBgNVBAMMCkZkbyBEZXZpY2UwWTATBgcqhkj
O
PQIBBggqhkjOPQMBBwNCAASKq9e/IS/aVqY7r41Jph06YmVngNT6aAKXifvrBSTe
4bMWSCv37sFLoH7XhGJSfLXuDuH9c6/DqK7jyOYDsrF6ow8wDTALBgNVHQ8EBAMC
B4AwCgYIKoZIzj0EAwIDRwAwRAIgdNG+5L6FFBXTI2aOHFIJtu09PZ6mIL5Tz2nE
XWo7Dp8CIDENQVSQlnS72t+/Wn4FMe/fUAsPNKk2evNgrNZqF1siWQFJMIIBRTCB
7KADAgECAghlJrieqU9CDzAKBggqhkjOPQQDAjAXMRUwEwYDVQQDDAxGZG9FbnRp
dHkgQ0EwIBcNMjQwMTMwMTMyMjEzWhgPMjA1MzA4MjUxMzIyMTNaMBcxFTATBgNV
BAMMDEZkb0VudGl0eSBDQTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABIsqYQAS
663sT0G8ncNTimfsXjjF3c0ms4306F3hXyuUHrIis6J+GESn/qzWs0txGzFAq7md
blZbDlZ/MpcSNaqjIDAeMA8GA1UdEwEB/wQFMAMBAf8wCwYDVR0PBAQDAgGGMAoG
CCqGSM49BAMCA0gAMEUCIBugcYB80BZvdB1FHBrI0SkmljrxIrujTyKXSl2dertr
AiEA22BsymuhftR5h40R4OQjI59AIi4nFAttZyMp2di12CCB0oRDoQEmoFiqhIIv
WCDd2W+L9BjOepmK8Gw+HoRogV2+krAehPvrTKunUKvAV4IvWCBUJf6lxl38tGDg
yah4VYMRG5Fn6YysL58F0GaU1S7fO/aDCgFYWzBZMBMGByqGSM49AgEGCCqGSM49
AwEHA0IABD3544yjk3DV3xxRy1RxnVmyGSYLJ6w/7ymwQLEqMnbajGvIgcQlnNB/
vHiFdCUKPFiWj4MP6UGGFvlbH+Bc2UdYQPNHEf/IIrdo7NUYILqfX3xcTIeTVaJo
1XsSIbrGZA5vu333gabS4Lj4GWDY8cnOnITaPvMn1KwAp2Ue93nCnYI=
-----END OWNERSHIP VOUCHER-----

POST to deliver the extended OV to the Owner (sent to the Owner API)

```
curl  -D - --digest -u default:default --location --request POST
"http://host.docker.internal:8042/api/v1/owner/vouchers" --header 'Content-Type: text/plain' --data-
raw '${voucher}'
```

Example of request:

```
curl  -D - --digest -u default:default --location --request POST
"http://host.docker.internal:8042/api/v1/owner/vouchers" --header 'Content-Type: text/plain' --data-
raw '-----BEGIN OWNERSHIP VOUCHER-----
hRhlWNWGGGVQAGb8XT3/Qya6mHhimG+PnIGFggVVdGhvc3QuZG9ja2VyLmludGVy
bmFsggNDGR9pggxBAoICRUR/AAABggRDGR9pakRlbW9EZXZpY2WDCgFYWzBZMBMG
ByqGSM49AgEGCCqGSM49AwEHA0IABIsqYQAS663sT0G8ncNTimfsXjjF3c0ms430
6F3hXyuUHrIis6J+GESn/qzWs0txGzFAq7mdblZbDlZ/MpcSNaqCL1ggv0UVKWu5
```

De9cLJ1ZAbxuzqloW95L35QtXTFyX7FyXm2CBVggDmuoPdQjAq9tCXyxKjDIm8Vl
FtNmuiN814Jxz/XcXuqCWQE1MIIBMTCB2aADAgECAggyOJxNk1dFYzAKBggqhkjO
PQQDAjAXMRUwEwYDVQQDDAxGZG9FbnRpdHkgQ0EwIBcNMjQwMTMxMTAzNjM2Wh
gP
MjA1MzA4MjYxMDM2MzZaMBUxEzARBgNVBAMMCkZkbyBEZXZpY2UwWTATBgcqhkj
O
PQIBBggqhkjOPQMBBwNCAASKq9e/IS/aVqY7r41Jph06YmVngNT6aAKXifvrBSTe
4bMWSCv37sFLoH7XhGJSfLXuDuH9c6/DqK7jyOYDsrF6ow8wDTALBgNVHQ8EBAMC
B4AwCgYIKoZIzj0EAwIDRwAwRAIgdNG+5L6FFBXTI2aOHFIJtu09PZ6mIL5Tz2nE
XWo7Dp8CIDENQVSQlnS72t+/Wn4FMe/fUAsPNKk2evNgrNZqF1siWQFJMIIBRTCB
7KADAgECAghlJrieqU9CDzAKBggqhkjOPQQDAjAXMRUwEwYDVQQDDAxGZG9FbnRp
dHkgQ0EwIBcNMjQwMTMwMTMyMjEzWhgPMjA1MzA4MjUxMzIyMTNaMBcxFTATBgNV
BAMMDEZkb0VudGl0eSBDQTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABIsqYQAS
663sT0G8ncNTimfsXjjF3c0ms4306F3hXyuUHrIis6J+GESn/qzWs0txGzFAq7md
blZbDlZ/MpcSNaqjIDAeMA8GA1UdEwEB/wQFMAMBAf8wCwYDVR0PBAQDAgGGMAoG
CCqGSM49BAMCA0gAMEUCIBugcYB80BZvdB1FHBrI0SkmljrxIrujTyKXSl2dertr
AiEA22BsymuhftR5h40R4OQjI59AIi4nFAttZyMp2di12CCB0oRDoQEmoFiqhIIv
WCDd2W+L9BjOepmK8Gw+HoRogV2+krAehPvrTKunUKvAV4IvWCBUJf6lxl38tGDg
yah4VYMRG5Fn6YysL58F0GaU1S7fO/aDCgFYWzBZMBMGByqGSM49AgEGCCqGSM49
AwEHA0IABD3544yjk3DV3xxRy1RxnVmyGSYLJ6w/7ymwQLEqMnbajGvIgcQlnNB/
vHiFdCUKPFiWj4MP6UGGFvlbH+Bc2UdYQPNHEf/IIrdo7NUYILqfX3xcTIeTVaJo
1XsSIbrGZA5vu333gabS4Lj4GWDY8cnOnITaPvMn1KwAp2Ue93nCnYI=
-----END OWNERSHIP VOUCHER-----'

The response contains the UUID of the Voucher stored on the Owner app. The extended OV is stored within the db in the table ONBOARDING_VOUCHER

Example UUID returned in the response:
0066fc5d-3dff-4326-ba98-7862986f8f9c

POST to configure the OWNER IP on the Owner. This is the IP address on which the Owner will wait for a connection from the device. This IP is the one stored on the Rendezvous Server database inside the blob

```
curl -D - --digest -u default:default --location --request POST
'http://host.docker.internal:8042/api/v1/owner/redirect' --header 'Content-Type: text/plain' --data-
raw '[[null,"host.docker.internal",8043,5]]'
```

Response: 200 OK

GET to the Owner to retrieve the guid of the devices for which the Owner owns a valid OV

```
curl -D - --digest -u default:default --location --request GET
"http://host.docker.internal:8042/api/v1/owner/vouchers" --header 'Content-Type: text/plain'
```

A list of device GUIDs is returned as output. The last added in chronological order is the one saved in the first line. Example of response:

```
0066fc5d-3dff-4326-ba98-7862986f8f9c
074eb766-4463-47cf-b788-7aba81715ff6
82ea7dde-bca5-490a-a4b0-70dc17767996
```

START RENDEVOUZ SERVER
1) Switch Digest Authentication on RV
2) From the folder component-samples/demo/rv, run the command
    sudo docker compose up --build

POST to store the Owner certificate on the Server
NOTE: This is the countermeasure described in the app-note. As a parameter, the request wants the Owner certificate in PEM format.

```
curl -D - --digest -u default:default --location --request POST
"http://host.docker.internal:8040/api/v1/rv/allow" --header 'Content-Type: text/plain' --data-raw
"$owner_certificate"
```

Example of request:

```
curl -D - --digest -u default:default --location --request POST
"http://host.docker.internal:8040/api/v1/rv/allow" --header 'Content-Type: text/plain' --data-raw "----
-BEGIN CERTIFICATE-----
MIIBRDCB7KADAgECAgh2Z8k55hJHnjAKBggqhkjOPQQDAjAXMRUwEwYDVQQDDAxG
ZG9FbnRpdHkgQ0EwIBcNMjQwMTMwMTMyMjE1WhgPMjA1MzA4MjUxMzIyMTVaMBcx
FTATBgNVBAMMDEZkb0VudGl0eSBDQTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0I
A
BD3544yjk3DV3xxRy1RxnVmyGSYLJ6w/7ymwQLEqMnbajGvIgcQlnNB/vHiFdCUK
PFiWj4MP6UGGFvlbH+Bc2UejIDAeMA8GA1UdEwEB/wQFMAMBAf8wCwYDVR0PBAQD
AgGGMAoGCCqGSM49BAMCA0cAMEQCIF/Ly6X9R70YJlteVq8TVCDfPYPjCpOrdx4D
PXMgEK7rAiB+A7QoErpo6bolXbTEE62OYryOvUnhrXk/pTzOArZQ+g==
-----END CERTIFICATE-----"
```

Response 200 OK

GET to trigger the Owner to execute phase TO0 with the Server. Param = GUID of the device

```
curl -D - --digest -u default:default --location --request GET
"http://host.docker.internal:8042/api/v1/to0/${device_guid}" --header 'Content-Type: text/plain'
```

Example of request,

```
curl -D - --digest -u default:default --location --request GET
"http://host.docker.internal:8042/api/v1/to0/0066fc5d-3dff-4326-ba98-7862986f8f9c" --header
'Content-Type: text/plain'
```

If all goes well, this message appears on the console of the Owner
TO0 COMPLETED FOR GUID ...

POST to configure the Owner Service Info package

```
curl -D - --digest -u default:default --location --request POST
'http://host.docker.internal:8042/api/v1/owner/svi' --header 'Content-Type: text/plain' --data-raw '[
  {"filedesc" : "setup.sh", "resource" : "http://www.google.com"}
]'
```

Response 200 OK

START DEVICE TO EXECUTE PHASES TO1 AND TO2
From the folder component-samples/demo/device, run the command

```
java -jar device.jar
```

If all goes well, this message appears on the console of the Device

```
===== FDO TO2 SUCCESS =====
```

DEMO WITH RESELLER

SWITCH Digest authentication
By default, all the server components use https and require a TLS certificate to authenticate the requests sent to the various api-endpoints. This behaviour can be disabled by enabling Digest Authentication:
Go to folder component-samples/demo/<component>/**WEB-INF** and change the security constraints inside **web.xml.** In particular, replace and insert this code:

```
<security-constraint>
   <web-resource-collection>
      <web-resource-name>apis</web-resource-name>
      <url-pattern>/api/v1/*</url-pattern>
   </web-resource-collection>
   <auth-constraint>
      <role-name>api</role-name>
   </auth-constraint>
   <user-data-constraint>
     <transport-guarantee>NONE</transport-guarantee>
   </user-data-constraint>
 </security-constraint>

 <login-config>
    <auth-method>DIGEST</auth-method>
 </login-config>
```

Then, back to component-samples/demo/<component> and modify the file **tomcat-users.xml**. In particular, change the line associated with user as follows:

```
<user username="default" password="default" roles="api"/>
```

Va fatto per: manufacturer, owner, reseller, rv

This must be done for <component> = manufacturer, owner, reseller, rv

NOTE: **default:default** this are username and password to specify inside api requests

NOTE: before running this demo, it is necessary to do “mvn clean installs”:
from main folder pri-fidoiot: mvn clean install
from folder component-samples/aio: mvn clean install
from folder component-samples/device: mvn clean install

Then, execute the scripts to generate the keys.

The first part is the same as the previous demo: switch on both manufacturer and device and let them perform the DI phase

START MANUFACTURER
1) Switch Digest Authentication on Manufacturer
2) Start Manufacturer using docker
   From the folder component-samples/demo/manufacturer, run the command
   sudo docker compose up --build

POST to the Manufacturer to update the IP address of the RV Server (This is the IP stored within the OVHeader)

```
curl -D - --digest -u default:default --location --request POST
'http://host.docker.internal:8039/api/v1/rvinfo' --header 'Content-Type: text/plain' --data-raw
'[[[5,"host.docker.internal"],[3,8041],[12,2],[2,"127.0.0.1"],[4,8041]]]'
```

Response: 200 OK

START DEVICE
1) Go to folder component-samples/demo/device and change inside the file service.yml the line corresponding to di-url to make it point to port 8039 (this is the port on which the manufacturer is waiting for a connection)

```
di-url: http://host.docker.internal:8039
```

2) If present, delete credentials.bin from folder component-samples/demo/device/app-data
3) Launch the device from folder component-samples/demo/device using the command
   java -jar device.jar

START OWNER
1) Switch Digest Authentication on the Owner
2) Start Owner using docker from the folder component-samples/demo/owner
   sudo docker compose up --build

GET Owner certificate

```
curl -D - --digest -u default:default --location --request GET
'http://host.docker.internal:8042/api/v1/certificate?alias=SECP256R1' --header 'Content-Type:
text/plain'
```

START RESELLER
1) Switch Digest Authentication on the Reseller
2) Start Reseller using docker from the folder component-samples/demo/reseller

```
   sudo docker compose up --build
```

GET certificate of the Reseller

```
curl -D - --digest -u default:default --location --request GET
'http://host.docker.internal:8070/api/v1/certificate?alias=SECP256R1' --header 'Content-Type:
text/plain'
```

GET serial number of the OV and GUID of the device

```
curl -D - --digest -u default:default --location --request GET
'http://host.docker.internal:8039/api/v1/deviceinfo/100000' --header 'Content-Type: text/plain'
```

Example of response

```
[{"serial_no":"2D1A92AD","timestamp":"2024-02-01 16:10:13.117","uuid":"9cb04faa-b033-4361-
9fc9-ac54616d6966","alias":"SECP256R1"}]
```

POST TO MANUFACTURER to request the extension of the OV for the Reseller

```
curl -D - --digest -u default:default --location --request POST
"http://host.docker.internal:8039/api/v1/mfg/vouchers/${serial_number}" --header 'Content-Type:
text/plain' --data-raw "Reseller certificate in PEM"
```

The response contains the extended OV for the Reseller

POST to store the Reseller certificate inside the db table ONBOARDING_VOUCHER. NOTE: while extending the OV, the Reseller will automatically retrieve the value written inside this table

```
curl -D - --digest -u default:default --location --request POST
"http://host.docker.internal:8070/api/v1/owner/vouchers" --header 'Content-Type: text/plain' --data-
raw 'OV esteso per il reseller'
```

Response = guid of the device (for confirmation)

```
9cb04faa-b033-4361-9fc9-ac54616d6966
```

(FOR CONFIRMATION: GET to check that the OV has been entered correctly)

```
curl -D - --digest -u default:default --location --request GET
"http://host.docker.internal:8070/api/v1/owner/vouchers/${GUID}" --header 'Content-Type:
text/plain'
```

Should return the extended OV for the reseller

POST to extend the certificate for the Owner

```
curl -D - --digest -u default:default --location --request POST
"http://host.docker.internal:8070/api/v1/resell/${GUID}" --header 'Content-Type: text/plain' --data-raw "Owner certificate in PEM"
```

Response: OV extended RESELLER+OWNER

POST to insert extended OV Reseller+Owner inside db table ONBOARDING_VOUCHER

```
curl  -D - --digest -u default:default --location --request POST
"http://host.docker.internal:8042/api/v1/owner/vouchers" --header 'Content-Type: text/plain' --data-raw 'OV extended Reseller+Owner'
```

POST to configure the Owner IP on the Owner

```
curl -D - --digest -u default:default --location --request POST
'http://host.docker.internal:8042/api/v1/owner/redirect' --header 'Content-Type: text/plain' --data-raw '[[null,"host.docker.internal",8043,5]]'
```

Response 200 OK

START RENDEVOUZ SERVER
1) Switch Digest Authentication on the RV
2) From folder component-samples/demo/rv, run the command
   ```
   sudo docker compose up --build
   ```

POST to store the certificate of the Owner on the Server

```
curl -D - --digest -u default:default --location --request POST
"http://host.docker.internal:8040/api/v1/rv/allow" --header 'Content-Type: text/plain' --data-raw
"$owner_certificate"
```

Response 200 OK

GET to trigger the Owner to execute phase TO0 with the Server. Param = GUID of the device

```
curl -D - --digest -u default:default --location --request GET
"http://host.docker.internal:8042/api/v1/to0/${device_guid}" --header 'Content-Type: text/plain'
```

If all goes well, this message appears on the console of the Owner

TO0 COMPLETED FOR GUID ...

POST to configure the Owner Service Info package

```
curl -D - --digest -u default:default --location --request POST
'http://host.docker.internal:8042/api/v1/owner/svi' --header 'Content-Type: text/plain' --data-raw '[
  {"filedesc" : "setup.sh", "resource" : "http://www.google.com"}
]'
```

Response 200 OK

START DEVICE TO EXECUTE PHASES TO1 AND TO2
From the folder component-samples/demo/device, run the command

```
java -jar device.jar
```

If all goes well, this message appears on the console of the Device

```
===== FDO TO2 SUCCESS =====
```

ATTACKS TESTED:

- maufacturer extends OV for both Owner and Reseller (two different OVs)
- the extended OV for the reseller I then also extend to the Owner
- reseller certificate stored in the Server trust store
- owner certificate stored in the Server trust store

Two OVs (owner, reseller+owner). Use the OV only with an extension for the Owner. Success even if we have two different OVs

ATTACK 2 (attack in phase TO0)

Since the Owner is the only Java application we have to perform the TO0 protocol phase, we try to modify the certificate+private key saved on the Owner by acting directly on the database. In particular, we save the Reseller's certificate+private key on the Owner entry in the db

```
Three extended OV    MANUFACTURER -> RESELLER -> OWNER
                                  -> RESELLER -> RESELLER
                                  -> RESELLER
```

For the last two, store certificate+private key of the Reseller inside the entry of the Owner

To access to the database:

```
sudo docker exec -it db-fdo-db-1 mysql --user=root --password=<password> --database=emdb
```

show tables;	to show tables present in the db
describe <table_name>;	to have a description of the fields present in a table

Certificates of the various entities (Owner, Reseller, Manufacturer) are in the table certificate_data
To update the certificate on the Owner and assign it the OV of the Reseller:

update certificate_data set data = (select data from certificate_data where name = 'reseller.p12')
where name = 'owner.p12';

To stop all docker running containers
sudo docker stop $(sudo docker ps -a -q)

ATTACK 3: MITM TO1 phase
Leverage the code contained within the `verification-fdo/fdo-hack-code-server` repository and the specifications detailed in **Test 2** of the accompanying paper.

ATTACK 4: MITM TO2 phase
Leverage the code contained within the `verification-fdo/fdo-hack-code-server` repository and the specifications detailed in **Test 3** of the accompanying paper.