SAMSUNG ARTIK™ Modules

Samsung Training Lab

TABLE OF CONTENTS

Pre-requisites 1
Lab1: Onboard an ARTIK 053s 2
  1.1 Prepare ARTIK 053s Board for Onboarding 2
  1.2 Samsung ARTIK Onboarding Application 2
Lab 2: Security APIs 7
  2.1 Key Management APIs 7
  2.2 Secure Storage APIs 9
  2.3 Encryption/Decryption APIs 9

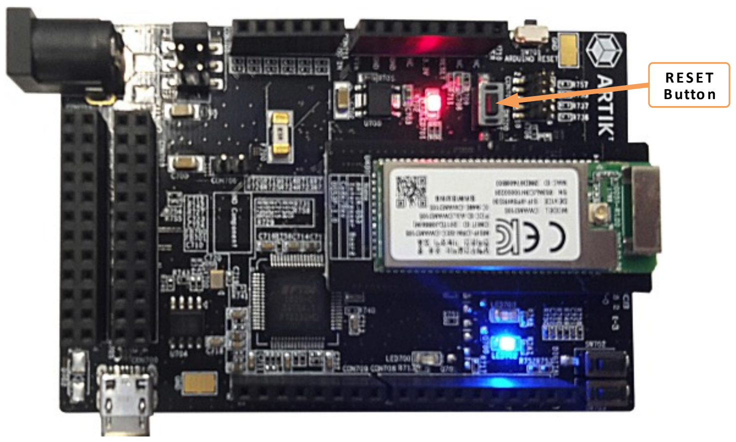
Pre-requisites

1. Create an ARTIK Cloud account.

Lab1: Onboard an ARTIK 053s

1.1 Prepare ARTIK 053s Board for Onboarding

1. Power up your ARTIK053 board and push the Reset button.

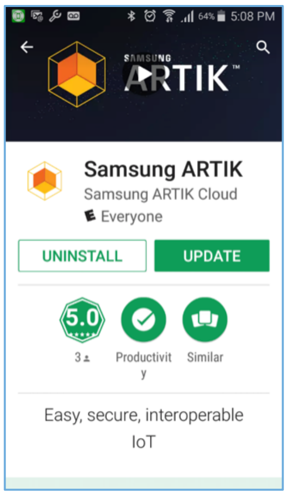


2. Run *onboard* from ARTIK 053s TASH shell.

```
TASH>>onboard
TASH>>Onboarding service version 1.6.0
Starting supplicant in foreground...
Starting AP ARTIK_286d97409408
Web server started
ARTIK Onboarding Service started
```

1.2 Samsung ARTIK Onboarding Application

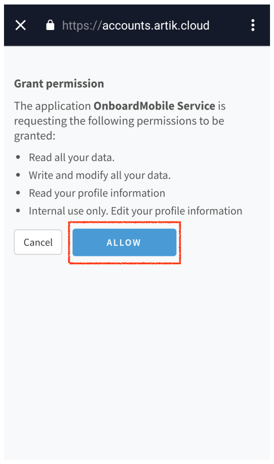
1. Download Samsung ARTIK App to your mobile device:



- For Android phones, the app can be downloaded from Google Play.

- For iPhone, the app can be found at the iTunes App Store (search for Samsung ARTIK). The app logo looks like:

2. Launch ARTIK app on your mobile phone, sign in with your own account and grant permission to the onboarding app.



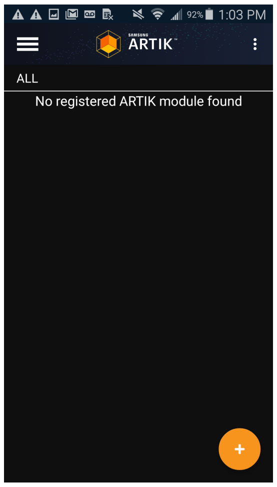
https://accounts.artik.cloud

Grant permission

The application **OnboardMobile Service** is requesting the following permissions to be granted:

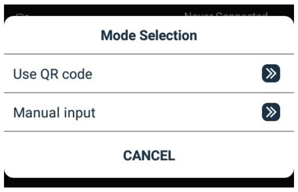
- Read all your data.
- Write and modify all your data.
- Read your profile information
- Internal use only. Edit your profile information

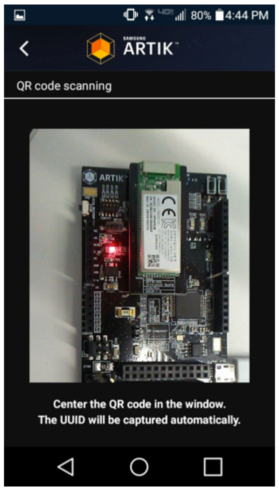
3. Click the "+" button to add a device.



No registered ARTIK module found

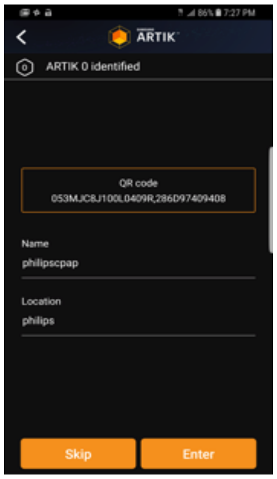
3. Press the arrows next to the Use QR Code.



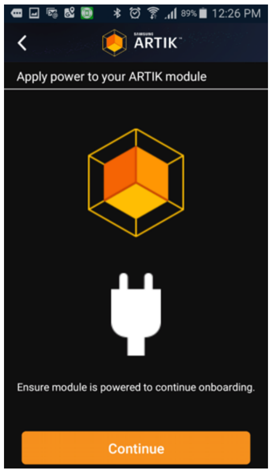

ARTIK
QR code scanning
Center the QR code in the window.
The UUID will be captured automatically.

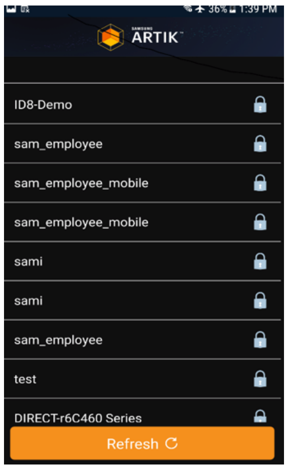
4. Scan the QR code.

5. Enter a device name in the Name field and press Enter.



6. Press Continue


ARTIK
Apply power to your ARTIK module
Ensure module is powered to continue onboarding.
Continue


ARTIK

ID8-Demo

sam_employee

sam_employee_mobile

sam_employee_mobile

sami

sami

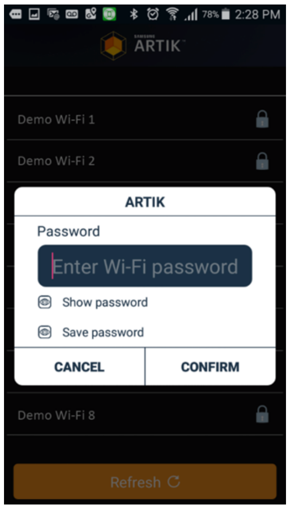
sam_employee

test

DIRECT-r6C460 Series

Refresh

7. Select the SSID to be used by the ARTIK 053s.


ARTIK

Demo Wi-Fi 1

Demo Wi-Fi 2

ARTIK

Password

Enter Wi-Fi password

Show password

Save password

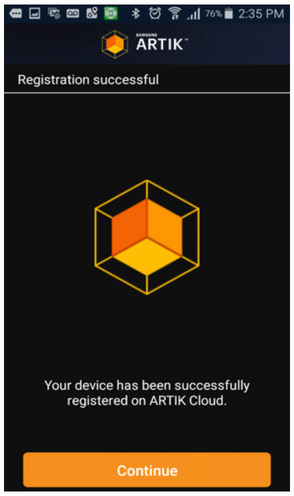
CANCEL | CONFIRM

Demo Wi-Fi 8

Refresh

8. Enter the password and press Confirm.

9. After registration is successful, click Continue.

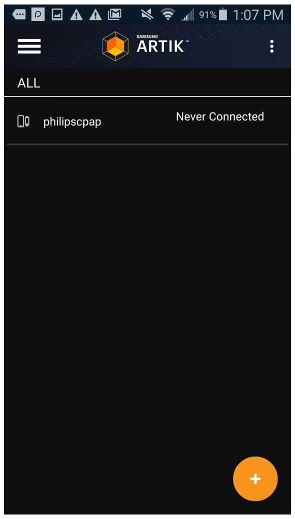

ARTIK

Registration successful

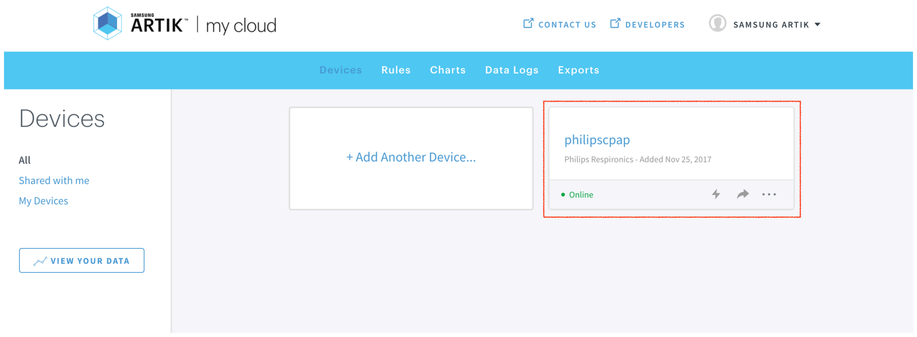
Your device has been successfully registered on ARTIK Cloud.

Continue

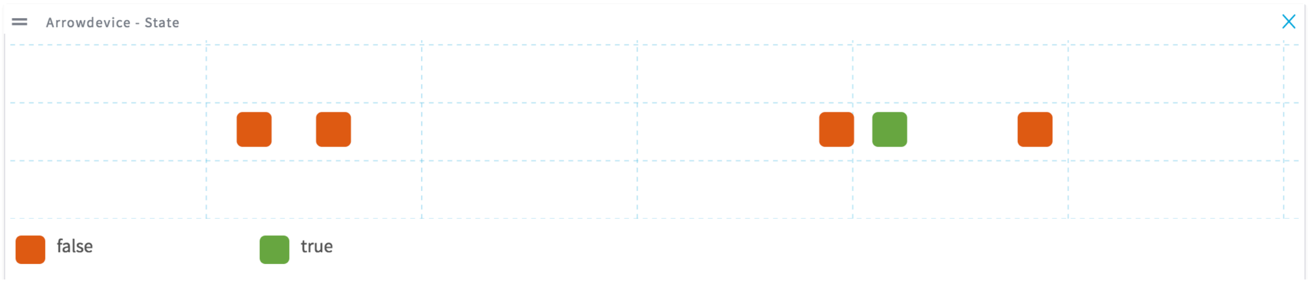
10. New device shows up.



The ARTIK 053s has been successfully on-boarded to the Cloud. From your ARTIK Cloud portal, you can see the newly onboarded ARTIK 053s device.



There is a blue LED on the ARTIK053s board. Click on “VIEW YOUR DATA” button, and you should be able to see LED state streamed to the Cloud.



From your mobile onboarding app, go to ACTIONS tab, and you can set on or off the onboard LED.

Lab 2: Security APIs

The Samsung ARTIK solution includes a Security Library that consists of APIs to communicate with the Samsung Secure Environment from the non-secure OS. The Security Library offers specific functions related to key management, authentication, secure storage and post-provisioning etc.

2.1 Key Management APIs

We can use Security Key Management APIs to retrieve factory injected certificates and keys, or generate your own certificate/key-pair.

Here is a code snippet to retrieve factory default certificate.

```
unsigned char buf[4096];
unsigned int buf_len = 4096;

printf("  . SEE Write Secure Storage ...\n");
if (see_get_certificate(buf, &buf_len, FACTORY_ARTIK_CERT, 0) != SEE_OK)
{
    printf("Fail\n  ! see_get_certificate\n");
    return 1;
}
printf("Write %s to Secure Storage data slot 0\n", input);

print_crt(1, buf, buf_len);

return 0;
```

We have a sample application security_api pre-installed. If you launch the sample app as below, you can see the ARTIK default certs.

```
TASH >> security_api factorycert

…
ARTIK CERT ...
  - cert. version     : 3
  - serial number     : 59:42:F1:8F:AF:47:09:5A:0E:90:99:97:EB:32:E0:4B
  - issuer name       : C=KR, O=Samsung Semiconductor ARTIK, OU=ARTIK
Root CA, CN=ARTIK Root CA
  - subject name      : C=KR, O=Samsung Semiconductor ARTIK, OU=ARTIK
Root CA, CN=ARTIK Root CA
  - issued  on        : 2017-06-15 20:43:59
  - expires on        : 2067-06-15 20:43:59
  - signed using      : ECDSA with SHA256
  - EC key size       : 256 bits
  - basic constraints : CA=true
  - key usage         : Key Cert Sign, CRL Sign


  - cert. version     : 3
  - serial number     : 59:42:F5:7C:AC:C1:06:31:11:5F:E4:B0:C8:CC:12:2D
  - issuer name       : C=KR, O=Samsung Semiconductor ARTIK, OU=ARTIK
Root CA, CN=ARTIK Root CA
  - subject name      : C=KR, O=Samsung Semiconductor ARTIK, OU=ARTIK
High Security Device CA, CN=ARTIK High Security Device CA
  - issued  on        : 2017-06-15 21:00:44
  - expires on        : 2067-06-15 21:00:44
  - signed using      : ECDSA with SHA256
  - EC key size       : 256 bits
  - basic constraints : CA=true, max_pathlen=0
  - key usage         : Key Cert Sign, CRL Sign


  - cert. version     : 3
  - serial number     : 01:01:17:07:24:00:00:06:F1
  - issuer name       : C=KR, O=Samsung Semiconductor ARTIK, OU=ARTIK
High Security Device CA, CN=ARTIK High Security Device CA
  - subject name      : C=KR, O=Samsung Semiconductor ARTIK, OU=ARTIK
High Security Device, CN=SIP-0P5WRS30 (01011707-2400-0006-f10f-
88024a777642)
  - issued  on        : 2017-07-24 00:26:23
  - expires on        : 2047-07-24 00:26:23
  - signed using      : ECDSA with SHA256
  - EC key size       : 256 bits
  - key usage         : Digital Signature, Non Repudiation
  - ext key usage     : TLS Web Client Authentication, TLS Web Server
Authentication
```

2.2 Secure Storage APIs

Secure Storage APIs help us read/write data to data slots in secure storage. Here is a code snippet that writes a string to secure storage data slot 0, then reads it back.

```
unsigned char input[20]= "ArrowTraining";
unsigned int len = sizeof(input);
unsigned char output[20];

printf("  . SEE Write Secure Storage ...\n");
if (see_write_secure_storage(input, len, 0) != SEE_OK) {
    printf("Fail\n  ! see_write_secure_storage\n");
    return 1;
}
printf("Write %s to Secure Storage data slot 0\n", input);

printf("  . SEE Read Secure Storage ...");
if (see_read_secure_storage(output, &len, 0) != SEE_OK) {
    printf("Fail\n  ! see_read_secure_storage\n");
    return 1;
}
printf("Read Secure Storage data slot 0: %s\n", output);

return 0;
```

Run *security_api* with *securestorage*, and you should see the results below.

```
TASH >> security_api securestorage
. SEE Write Secure Storage ...
Write ArrowTraining to secure storage data slot 0
 . SEE Read Secure Storage ...
Read secure storage data slot 0:ArrowTraining
```

2.3 Encryption/Decryption APIs

```
unsigned char input[128]= "Arrow Training in Denver, Jan, 2018";
unsigned char output[128];
unsigned char check[128];

printf("  . SEE RSA encryption ...\n");
see_setup_key(1192, SECURE_STORAGE_TYPE_KEY_RSA,0);
if (see_rsa_encryption(0, MBEDTLS_RSA_PKCS_V15, output, &outlen, input,
inlen) != SEE_OK) {
    printf("Fail\n  ! see_rsa_encryption\n");
    return 1;
}
printf("see_rsa_encryption success\n");

printf("  . SEE RSA decryption ...");
if (see_rsa_decryption(0, MBEDTLS_RSA_PKCS_V15, check, &checklen,
output, outlen) != SEE_OK) {
    printf("Fail\n  ! see_rsa_decryption\n");
    return 1;
}
printf("rsa decryption: %s\n", output);
```

Run *security_api* with *rsaencdec*, and you can see how we use RSA to encrypt a string then decrypt it.

```
TASH>> security_api rsaencdec
TASH>>----------------------------------------
TESTCASE : .SEE RSA encryption …
----------------------------------------
[0] see_rsa_encryption success
. SEE RSA decryption …
rsa decryption: Arrow Training in Denver, Jan, 2018
```