SEMS Lite

examples and tooling based on SE051x

PUBLIC

Customer Application Support

OCTOBER 2020

SECURE CONNECTIONS
FOR A SMARTER WORLD

AGENDA

- PSP Content
  - Documents, Boards, Samples
- SEMS Lite flow
  - Customer onboarding
  - Key rotation
  - Update script generation
  - Update script usage
- Applet development
  - JCOP Applet data backup according to GP Amd H
- Hands-on
  - Put an applet using SEMS Lite on SE051P
  - Delete it again

SUPPORT RESOURCES SE051 P

- Homepage with all documents and SW for the generic type – P is only mentioned, no further info
  - https://nxp.com/se051
- CAS SE051 Trainings Folder (readonly)
  - Collection of Trainingsslides + extra SW for SE05x
- https://nxp1.sharepoint.com/:f:/s/CAS2SecureProducts/EmjzPjx2_QZNg0flUQQ3PgwBzecID4viXE6dQyFScDkFrQ?e=DwHlUe
- Docstore
  - IoT Security\Applet Development
- Ticketing/Communities
  - **NXP internal Support Community for authentication (NXP only – post questions here)**
    - https://community.nxp.com/community/fsl-employee/secure-authentication-internal
  - Public NXP Support Community for authentication (accessibly by customers – look for answers and give answers here)
    - https://community.nxp.com/community/identification-security/secure-authentication
  - Collabnet for support tickets (only selected users, used for exchange with R&D)
    - https://collabnet.nxp.com/sf/tracker/do/listArtifacts/projects.jcop_support/tracker.iot

SEMS Lite for Applet Development

SECURE CONNECTIONS
FOR A SMARTER WORLD

SEMS LITE

- **SEMS**
  Secure **E**lement **M**anagement **S**ervice as defined in Global Platform Amendment I
  Defines how card content can be managed based on distributable scripts
  Mainly used in mobile applications
- **SEMS Lite:**
  a shrinked down implementation of Global Platform SEMS
- **LS CGT:**
  This document is an extension to Global Platform Amd-I specification. SEMS allows to generate offline scripts that can be broadcasted to perform card content management operations over a group of secure elements. This document is an extension to the Global Platform Amendment-I specification where the overall architecture of the solution is explained. This document lists the NXP specific extensions to the solution and also describes on how the LS CGT tool can be used to generate keypairs, sign certificates, encode scripts and decrypt keys.

DOCUMENTATION

- Global Platform Amendment I - Executable Load File Upgrade
  Describes how loaded files (applets) in JavaCard systems can be updated and include data save/restore functionality
- Global Platform Amendment H - Secure Element Management Service
  Define card content management by delegation mechanism. SE diversified specific key data is no longer required to launch an administration session. Instead signed update commands are used.
- LS CGT UM (included in ls-cgt in docstore)
  This document is an extension to Global Platform Amd-I specification. SEMS allows to generate offline scripts that can be broadcasted to perform card content management operations over a group of secure elements. This document is an extension to the Global Platform Amendment-I specification where the overall architecture of the solution is explained. This document lists the NXP specific extensions to the solution and also describes on how the LS CGT tool can be used to generate keypairs, sign certificates, encode scripts and decrypt keys.

DIFFERENCES SEMS TO SEMS LITE

		Mobile SEMS	IoT SEMS-Lite	Explanation
SEMS as standartized in GP Amd. I	On chip ownership management.	✓	✗	Separation of applet managing parties from each other. The chip remebers the which certificate can manage which applet.
	Certificate field based priviledge management	✓	✗	Priviledges are mostly hardcoded in SEMS-Lite: Creation of Security Domains, keyInjection/keyCreation, Loading and deletion of ELFs, Instantiation and deletion of applets, Applet personalization with non-diversified data, Key rotation of the SEMS on-card entity in case of change of ownership or for security reasons
	Implemntation Detail: Keeping an additional SEMS registry on chip	✓	✗	In mobile SEMS, for the on chip ownership and priviledge management an additional registry, to the JCOP registry, is kept in SEMS. In SEMS-Lite this has been removed.
	Security Domain Creation/Deletion/KeyInjection/KeyCreation	✓	✗	
	GP Amd. H Upgrade preserving User Data	✓	✓	
Mobile Addons	Self Update	✓	✗	The SEMS Updater Applet can update itself in the field.
	Factory Reset	✓	✗	Bringing the system including data back into the original factory state.
	Legacy Loaderservice support	✓	✗	Before the standardization of SEMS in GP Amd. I, NXP already offered mobile Loaderservice. Some customers still use the legacy loaderservice functionality.
	↳Legacy Loaderservice AID	✓	✗	To be fully backwards compatible the legacy Application Identifier needs to be selectable.
	↳Legacy Loaderservice Commands	✓	✗	Allowing customers to execute older Loaderservice scripts on the mobile SEMS offering.
	↳Legacy Applet Migration preserving User Data	✓	✗	Original Loaderservice mechanism to preserve user data on chip (before Amd. H)
	Updater Applet is multi selectable	?	✗	
	Update Package genertion/signing via NXP TrustProvisioning Eng.	✓	✗	Script generation, if not done by the customer, and signing involves NXP TrustProvisioning Engineering and is done for a limited set of high volume key account customers.
IoT Addons	Off Chip ownership management	✗	✓	Allowing the mDAP signing of NXP only when the RID is associated with that Customer. Thus isolating parties from replacing each others applets.
	GP Amd H. GET STATUS (ELFs, Versions, Instances) in PLAIN	✗	✓	In SEMS-Lite: To allow offline updates and offline resolution of certain scenarios it is important to get the system state in plain (normally it would be an encrypted SEMS command)
	Update Package genertion/signing via Scalable Webservice	✗	✓ (future)	Script generation, if not done by the customer, and signing is done via a scalable webservice (EdgeLock2Go) and is done for a large set of low to medium volume customers.
Middleware Agent	Offline deployment to eSE	✓	✓	The complete update script is kept on the Host while applying onto the eSE.
	Streaming deployment to eSE	✗	✓	The host can only buffer a part of the update script during the update and is streaming it towards the eSE.

QUESTIONS

- How is SEMS Lite access-protected?
  - It is not specially protected – everybody who has an applicable script and access to an SE interface can run it
- What needs to be considered when doing a SEMS Lite update of an applet in the applet development?
  - Implement acc. GP Amd H
- Who (which applet) can retrieve the backup data?
  - Only whitelisted AIDs in the certificate. Multiple AIDs can be whitelisted
- How does the versioning work?
  - ELF AID Each ELF is versioned with a version according to JCVM specification - section 4.5 CAP and Package Versions (https://docs.oracle.com/javacard/3.1/related-docs/JCVMS/JCVMS.pdf). Which is <major>.<minor> both major and minor are in the range 0 to 127 each.
  - GP Amd. H allows updates only to higher (or equal) versions.

USED KEYS IN SEMS LITE CONTEXT

- SK = Secret Key
- PK =Public Key
- AP = Application Provider

Type	Usage	Name	Location
Secret Signature Key	Signs scripts to be uploaded to the SE51	SK.AP.AUT	Customer
Public Signature Key	Used to verify upload scripts by SEMS Lite	PK.AP.AUT	Certificate as issued by NXP
Secret Certificate Signature Key	Signs SEMS Lite Certificates	SK.CA-SEMS.AUT	NXP
Public Certificate Signature Key	Public key in SE051 to verify upload certificates	PK.CA-SEMS.AUT	SEMS Lite Applet
Public Encryption Key	Public encryption key for SEMS Lite commands	PK.CA-SEMS.ENC	NXP + Customer
Secret Encryption Key	Secret Decryption key inside SE051 for SEMS Lite commands decryption	SK.CA-SEMS.ENC	SEMS Lite Applet

SEMS LITE FLOW

- Customer gets access to documentation "Applet Development" in docstore (customer needs to be explicitly accepted for SE051P development upfront)
- Customer creates keypair which will be used to sign their data (`SK.AP.AUT`) (type: brainpool256_r1)
- Customer shares public key (uncompressed, standard when openSSL is used) of this keypair + wanted ELF.AID (JCOP-Tools: Package-AID) with NXP
  - NXP checks RID (first 5bytes of ELF.AID, check to not conflict with IoT applet and not other customers)
    - AID needs to be min 6 bytes (5 bytes RID + min 1 bytes)
- NXP provides
  - certificate which contains ELF.AID (Executable load file ID) + Public encryption key PK.CA-SEMS.ENC
  - Key rotation script for SE051P SEMS keys to rotate PK.CA-SEMS.AUT to a customer-individual key
- Customer edits SEMS-Lite script with ELF.AID/cert/key and runs LS-CGT to create the upload script
- Customer converts the ls-cgt script further and apply it to the secure element

SEMS LITE PROCESS WITH INVOLVED KEYS

NXP Production SE051P

- NXP SK.AP.AUT known by NXP
- PK.CA-SEMS.AUT inside SE051
- SK.CA-SEMS.ENC inside SE051
- PK.CA-SEMS.ENC known to NXP

Key rotation customer request

- New SK.AP.AUT created by customer
- PK.AP.AUT sent to NXP

Key rotation - NXP action

- Creation of new SK.CA.SEMS.AUT
- Create script which rotates PK.CA-SEMS.AUT to PK.CA-SEMS.AUT
- Sign script for key rotation with NXP SK.AP.AUT
- Send to customer

Certificate creation - NXP action

- Create Certificate which allows customer PK.AP.AUT to upload/install/update /delete applets
- Certificate signed by SK.CA-SEMS.AUT
- Send certificate + PK.CA-SEMS.ENC to customer

Apply key rotation - customer action

- Convert .json with sems-lite-generator.py
- Apply key rotation script to ICs

Applet Upload script – customer action

- Use ls-cgt to generate encrypted update script using SK.AP.AUT and PK.CA-SEMS.ENC
- Convert with tool MulticastPackage.py
- Apply applet update script to ICs

Sample Key Rotation

SECURE CONNECTIONS
FOR A SMARTER WORLD

PRACTICAL STEPS
KEY CREATION EXAMPLE

- Create keyset with curve brainpoolP256r1, has to be done in a secure way by customer
- Examples for demo purpose, either:
  - Using openSSL:

```
openssl ecparam -name brainpoolP256r1 -out brainpoolP256r1.pem
openssl ecparam -name brainpoolP256r1 -genkey -noout -out SEMS-keypair.pem
openssl ec -in SEMS-keypair.pem -pubout -out SEMS-pub.pem
```

    - Send SEMS-pub.pem + ELF.AID to NXP
  - Using ls-cgt:
    - Execute to generate a txt containing the keypair:

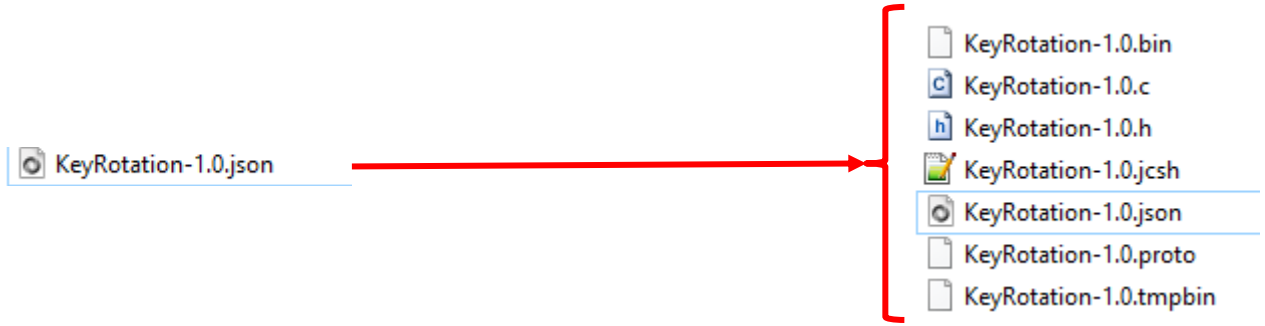
```
java -jar ls-cgt.jar -genkey -paramType brainpoolP256r1 -keyout SK.AP.AUT.txt
-------------------------------------------------------------------------------
Welcome to NXP Tools LS-CGT - v1.3.0 (Fri, 3rd July 2020)
(c) 2019 NXP India
-------------------------------------------------------------------------------

[2020710-17:6:3|KeyPair] New key pair AP.AUT written to SK.AP.AUT.txt
```

    - Send line with PK.AP.AUT to NXP, like:

```
PK.AP.AUT =
04xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxx
```

CONVERT KEY ROTATION SCRIPT

- Customer gets
  - KeyRotation-1.0.json – script to rotate from NXP to customer-individual keys (PK.CA-SEMS.AUT), signed and encrypted by NXP
  - SEMS-Lite_Cert_SP_<customername>.txt
  - SEMS_CA_ENC.txt – NXP provided public key for SEMS Lite script encryption
- Convert .json to usable files, use script in simw-top\semslite\tools\sems-lite-generator:
  `python semslite_json_converter.py [path to directory with .json file]`



EXECUTE KEY ROTATION SCRIPT

- Use sems-lite-agent:
  - Include .h/.c in MCU firmware projects
  - Execute .bin with sems-lite-cli
    - Compile MW with SE051 settings (applet version = 6.X) to get sems-lite-cli compiled
- Or execute .jcsh with JCShell (checks of sems-lite-agent bypassed)

SEMS LITE STATE
BEFORE KEY ROTATION, READ WITH SEMS_LITE_CLI_APP.EXE

```
uid: 040050017fe56cdabb1bc5043102d1870000
SSS Status: SUCCESS
appletVer: 01031211
SSS Status: SUCCESS
PbkeyId:
04184f096bb3eb67760d193d43a3e6017bd701499ef0fd5d0cbb12ddf58f6a0fff6524438deeee1c05de7b65cf749eeb907c769e2e36f23662101758352ac16c85
SSS Status: SUCCESS
Tear Status: 0
SSS Status: SUCCESS
Upgrade Status: 0
SSS Status: SUCCESS
EnckeyId: 00000000000000000000000000000000063709317042150
SSS Status: SUCCESS
CA-SEMS Identifier: 00000000000000000063709317142150
SSS Status: SUCCESS
CA-SEMS Key Identifier: 0063709317142150
App   :INFO :availableCODMemory 3840
App   :INFO :availableCORMemory 3722
App   :INFO :availablePersistentMemory 61548
App   :INFO :availableIDX 2
App   :INFO :Free PHeap Central Gap 61516
App   :INFO :Free freeTransient 3840
SSS Status: SUCCESS
```

SEMSLITE-CLI SAMPLE SCRIPT RUN

```
sems_lite_cli_app.exe --loadpkg "u:\se051\SE051P\SEMS_Lite\trial_2020-08-10\FromChrisRotation\build_ns\Production_A4A6_Run3_keyrotation_ROT_Production_A4A6_Run3_Test_CAS2Key3AE1\KeyRotation-1.0.bin"
App   :INFO :Using PortName='COM34' (ENV: EX_SSS_BOOT_SSS_PORT=COM34)
Opening COM Port '\\.\COM34'
sss   :INFO :atr (Len=35)
      01 A0 00 00    03 96 04 03    E8 00 FE 02    0B 03 E8 00
      01 00 00 00    00 64 13 88    0A 00 65 53    45 30 35 31
      00 00 00
App   :INFO :SEMS Lite Check Status Word: Success.
SEMS Lite Status: SUCCESS
```

SEMS LITE STATE
AFTER KEY ROTATION, READ WITH SEMS_LITE_CLI_APP.EXE

```
uid: 040050017fe56cdabb1bc5043102d1870000
SSS Status: SUCCESS
appletVer: 01031211
SSS Status: SUCCESS
PbkeyId:
049f1e2b97574bdd6f56fb6f13944f704f467ad011b716bf2cf7e27f8f28b1bd9751068e3838ee94e7d203e94ce7c4c7d38bf5b84a532b30e0b00c6d0962997c98
SSS Status: SUCCESS
Tear Status: 0
SSS Status: SUCCESS
Upgrade Status: 0
SSS Status: SUCCESS
EnckeyId: 00000000000000000000000000000000063709317042150
SSS Status: SUCCESS
CA-SEMS Identifier: 00000000000000000063709317742150
SSS Status: SUCCESS
CA-SEMS Key Identifier: 0063709317742150
App   :INFO :availableCODMemory 3840
App   :INFO :availableCORMemory 3722
App   :INFO :availablePersistentMemory 61548
App   :INFO :availableIDX 2
App   :INFO :Free PHeap Central Gap 61516
App   :INFO :Free freeTransient 3840
SSS Status: SUCCESS
```

COMPARE BEFORE/AFTER

- Data read with sems_lite_cli_app.exe:

```
PbkeyId before:
04184f096bb3eb67760d193d43a3e6017bd701499ef0fd5d0cbb12ddf58f6a0fff
6524438deeee1c05de7b65cf749eeb907c769e2e36f23662101758352ac16c85
PbkeyId after:
049f1e2b97574bdd6f56fb6f13944f704f467ad011b716bf2cf7e27f8f28b1bd97
51068e3838ee94e7d203e94ce7c4c7d38bf5b84a532b30e0b00c6d0962997c98
EnckeyId: 0000000000000000000000000000000000063709317042150
CA-SEMS Key Identifier: 0063709317142150
CA-SEMS Key Identifier: 0063709317742150
```

Update script creation

SECURE CONNECTIONS
FOR A SMARTER WORLD

PROCESS OF UPDATE SCRIPT GENERATION

- Tools and steps needed by customer
  - ls-cgt: from docstore
  - Other scripts and code: All in Plug&Trust MW

ls-cgt.jar

- **Inputs:**
  - Command file: Plain.txt
  - Key File encryption: SEMS_CA_ENC.txt
  - Key and certificate for signing: SK.SP.AUT.txt
  - Cap file
- **Output:**
  - encrypted.txt

→

MulticastPackageCli.py

- **Inputs:**
  - Ls-cgt script Encrypted.txt
  - Multicast-package config file: config.json
- **Out:**
  - Multicast package: package.json

→

semslite_json_converter.py

- **Inputs:**
  - Multicast-package package.json
- Out – update package:
  - .bin
  - .c/.h
  - .jcsh

→

Upload, sems-lite-cli

- **Input:**
  - Update script

LS-CGT.jar

SECURE CONNECTIONS
FOR A SMARTER WORLD

PLAIN.TXT

- Plain.txt contains the commands which should be executed by SEMS Lite
- NXP provides templates for this cases with SE051 in the AN:
  - Applet upload
  - Applet delete
  - Applet update
- More complex scenarios possible (multiple applets, AIDs, …) → ls-cgt user manual

PLAIN.TXT
EXAMPLE UPLOAD APPLET

```
manage_channel -lcid 01 -open
select -lcid 01 -aid A0000001510000 00
authenticate -lcid 01 -kvn 30 -seclevel 11 -sdaid A0000001510000 00
load -lcid 01 -cap <CAP_full_file_path> -sdaid A0000001510000 00
install -makeselectable -lcid 01 -elfaid <ELF_AID_of_applet> -elmaid
<ELM_AID_of_applet> -aid <New_applet_AID> -privileges 00 -params C900
select -lcid 01 -aid A0000001510000 00
manage_channel -lcid 01 –close
#The last line of this file has to be left empty
```

PLAIN.TXT
EXAMPLE DELETE APPLET

```
manage_channel -lcid 01 -open
select -lcid 01 -aid A0000001510000 00
authenticate -lcid 01 -kvn 30 -seclevel 11 -sdaid A0000001510000 00
delete -r -lcid 01 -aid <AID_of_applet_to_delete>
manage_channel -lcid 01 -close
#The last line of this file has to be left empty
```

PLAIN.TXT
EXAMPLE UPDATE APPLET

```
manage_channel -lcid 01 -open
select -lcid 01 -aid A000000151000000
authenticate -lcid 01 -kvn 30 -seclevel 11 -sdaid A000000151000000
begin_manage_elf_upgrade -lcid 01 -cap <CAP_full_file_path> -elfaid
<ELF_AID> -action 03 -elfaidlist <ELF_AID>
apdu -lcid 01 -apdu 80EA0100#(A1#(4F#(<ELF_AID>)800104))
end_manage_elf_upgrade
load -lcid 01 -cap <CAP_full_file_path> -sdaid A000000151000000
select -lcid 01 -aid A000000151000000
manage_channel -lcid 01 –close
#The last line of this file has to be left empty
```

LS-CGT SCRIPT FOR LOAD AND INSTANTIATE APPLET USING SEMS LITE

```
# The purpose of this script is to Upload package and create instances of applications
using SEMS Lite
#
# List the AID to be used in the script here.
/set-var APP.ELF.AID 000102030405060708       #Identifier of ELF
/set-var APP.ELM.AID 000102030405060708       #Identifier of ELM (Package AID)
/set-var APP.INSTANCE.AID 000102030405060708  #Identifier of Application (Instance AID)
/set-var ISD.AID A0000001510000 00

# Select ISD and authenticate
select -lcid 01 -aid ${ISD.AID}
authenticate -lcid 01 -kvn 30 -seclevel 11 -sdaid ${ISD.AID}

#OPTIONAL: To apply binding of the script execution to an individual device
#binding_se -sn ${UUID.NUMBER}

# Load cap file(s). Applies to both upgrade and just load
load -lcid 01 -cap path\nthELFv1.cap -sdaid ${ISD.AID}

# Create applet instance
install -makeselectable -lcid 01 -elfaid ${TEST.ELF.AID} -elmaid
${TEST.ELM.AID} -aid ${TEST.APP1.AID} -privileges 00 -params
C900
```

LS-CGT SCRIPT FOR DELETE APPLET USING SEMS LITE

```
# The purpose of this script is to DELETE an application or package and its
applications using SEMS Lite
#
# List the AID to be used in the script here.
/set-var APP.ELF.AID Identifier of ELF
/set-var APP.ELM.AID Identifier of ELM
/set-var APP.INSTANCE.AID Identifier of Application
/set-var ISD.AID A0000001510000 00

# Select ISD and authenticate
select -lcid 01 -aid ${ISD.AID}
authenticate -lcid 01 -kvn 30 -seclevel 11 -sdaid ${ISD.AID}

#OPTIONAL: To apply binding of the script execution to an individual device
#binding_se -sn ${UUID.NUMBER}

# Either: Delete only the test application instance
# delete -lcid 01 -aid ${APP.INSTANCE.AID}
# or: Delete the test package along with its applications
delete -r -lcid 01 -aid ${APP.ELF.AID}
```

PREPARATION OF KEYS AND CERTS FOR SEMS LITE

- These further files needs to be available:
- SK.AP.AUT.txt

  CERT.AP.AUT = **<Your_CERT.SP.AUT>**

  SK.AP.AUT = **<Your_SK.SP.AUT>**

  PK.AP.AUT = **<Your_PK.SP.AUT>**
- SEMS_CA_ENC.txt (as received from NXP):

  SK.RT.ENC = **<NXP_SK.CA-SEMS.ENC>**

  PK.RT.ENC = **<NXP_PK.CA-SEMS.ENC>**
- Cap file (in case of upload/update)
- Then run ls-cgt – it will create the encrypted commands in encrypted.txt

Script conversion

SECURE CONNECTIONS
FOR A SMARTER WORLD

CONVERSION OF ENCRYPTED.TXT TO A MULTICAST PACKAGE

- Two tools needed:
  - MulticastPackageCli.py
    Combines additional constraints as defined in a config.json
    - Allows to specify minimum memory limits which needs to be free
    - Enabled the SEMS Lite agent to do additional checks before the script continues to run against the SEMS Lite applet
      →Here logical/functional checks are specified

    Output: Multicast package - .json file
  - semslite_json_converter.py
    converts the .json into file formats usable for the update examples
    .bin → sems-lite-cli
    .c/.h → embedded examples
    .jcsh → jcshell execution (does bypass sems-lite-agent checks!)

SEMS Lite CLI

SECURE CONNECTIONS
FOR A SMARTER WORLD

SEMS LITE CLI

- A command line client to interact with the SEMS Lite applet
- Allows to interactively check the state and execute update packages
- Compile able on Windows and Linux
- Delivered within the MW:
  - Source code
  - Ready compiled for Windows in simw-top\binaries

SEMS LITE CLI INTEGRATED HELP

```
c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe
App   :INFO :Give one of Following Options for SEMS Lite Test
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--loadpkg] path-to-SEMS-Lite-applet-package-binary-file
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--loadpkgwt] path-to-SEMS-Lite-applet-package-binary-file tear_time
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--getuid]
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--getappcontents] optional-app-aid
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--getpkgcontents] optional-pkg-aid
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--getPbkeyId]
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--semslitegetversion]
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--getsignature] File-Name-to-store-signature
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--checkTear]
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--checkUpgradeProgress]
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--getENCIdentifier]
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--testapplet] applet-aid, apdu-command
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--getCAIdentifier]
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--getCAKeyIdentifier]
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--getpkgversion]
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--getFreePHeap]
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--getECParameter]
App   :INFO :c:\se050_mw\simw-top\binaries\ex\VCOM-se051_sems_lite_cli_app.exe [--getFIPSInfo]
```

SEMS LITE CL
GETUID

```
root@raspberrypi:/home/pi/03.00.00_rc/simw-
top_build/raspbian_native_se050_t1oi2c/bin# ./sems_lite_cli_app --getUID
SSS Status: SUCCESS
uid: 04005001597f1882148cd4043003c1870000
```

SEMS LITE CL
CHECKTEAR

```
root@raspberrypi:/home/pi/03.00.00_rc/simw-
top_build/raspbian_native_se050_t1oi2c/bin# ./sems_lite_cli_app --
checkTear
SSS Status: SUCCESS
Tear Status: 0
```

SEMS LITE CL GETENCIDENTIFIER

```
root@raspberrypi:/home/pi/03.00.00_rc/simw-top_build/raspbian_native_se050_t1oi2c/bin# ./sems_lite_cli_app --getENCIdentifier
SSS Status: SUCCESS
EnckeyId: 00000000000000000000000000000000000063709317042148
```

SEMS LITE CL CHECKCAIDENTIFIER

```
root@raspberrypi:/home/pi/03.00.00_rc/simw-top_build/raspbian_native_se050_t1oi2c/bin# ./sems_lite_cli_app --checkCAIdentifier
SSS Status: SUCCESS
CA-SEMS Identifier: 00000000000000000063709317142148
```

SEMS LITE CL GETFREEPHEAP

```
root@raspberrypi:/home/pi/03.00.00_rc/simw-
top_build/raspbian_native_se050_t1oi2c/bin# ./sems_lite_cli_app --
getFreePHeap
App   :INFO :availableCODMemory 2272
App   :INFO :availableCORMemory 598
App   :INFO :availablePersistentMemory 25332
App   :INFO :availableIDX 1
App   :INFO :Free PHeap Central Gap 25316
App   :INFO :Free freeTransient 2272
SSS Status: SUCCESS
App   :INFO :freePHeapCentralGap Insufficient: 25316 < 26544
```

SE051P – Applet Development

SECURE CONNECTIONS
FOR A SMARTER WORLD

PUBLIC

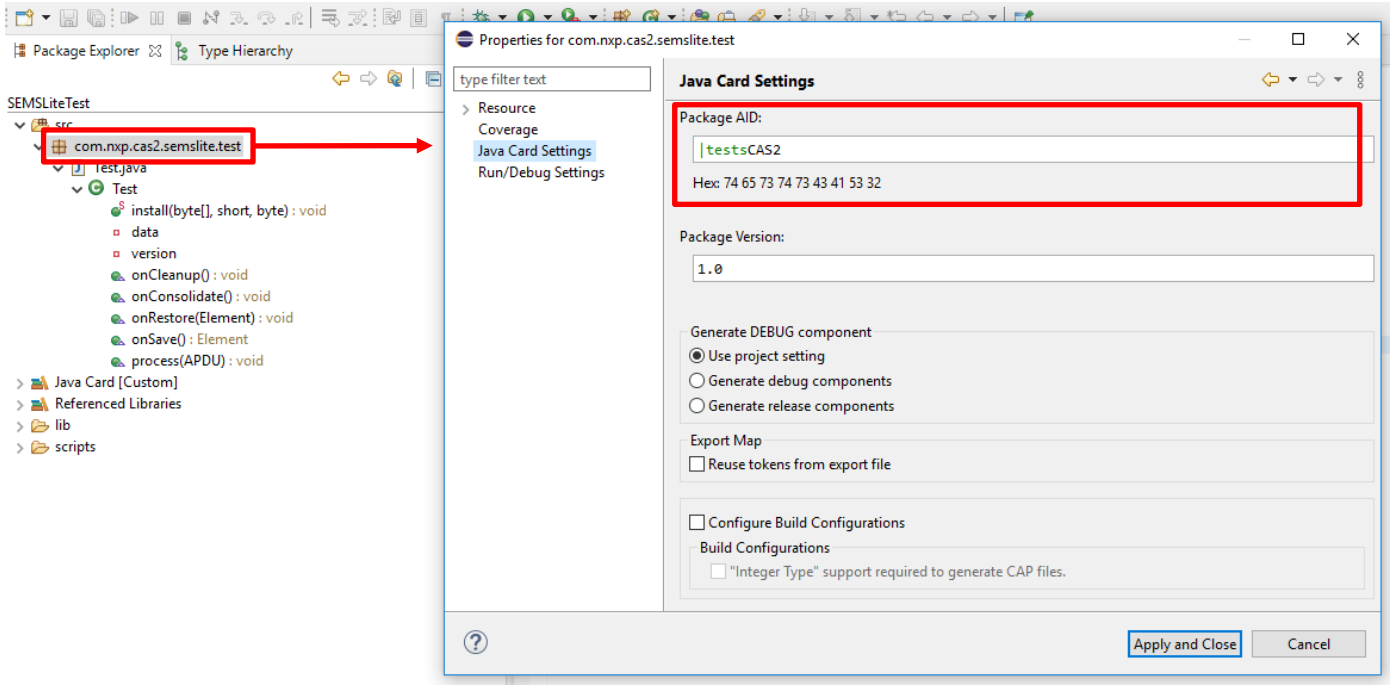
DOCUMENTATION FOR APPLET DEVELOPMENT

- All software and documents for Applet development are available after approval in docstore – folder IoT Security/Applet Development

Use	Docstore number	Document Title	Content
JCOP Tools	623410	JCOP 4 Tools (IOT) 5.34 (1.0)	Eclipse plugin for JCOP development + SE051 API + default simulator
JCOP Tools Simulator	630101	JCOP 4 Simulator Release Package (IOT) RC3 For The JCOP 4 Tools 5.34 RC1 (0.1)	SE051 simulator matching released SE051 OEFs
JCOP UGM	581812	JCOP 4.7 SE051 User Guidance And Administrator Manual (1.2)	User Guideance Manual for operation of JCOP, JCOP API Extensions description
LS CGT Tool	633610	LS-CGT/SEMS-CGT Version 1.3 (1.0)	Tool to generate applet update scripts
AN JCOP Tools + SEMS Lite Quickstart	641010	AN12909 How to develop JCOP applets on EdgeLock SE051 using JCOP Tools	Getting started with Applet Development and SEMS Lite usage on SE051P

PACKAGE AID FOR SEMS LITE IN SEMS LITE CERTIFICATE

- For SEMS-Lite certificate the package AID is relevant




```
Card Manager AID   :  A000000151000000
Card Manager state :  OP_READY

    Application:  SELECTABLE (--------) (--------) (----) "testsCAS2sems"
    Load File  :      LOADED (--------) A0000001515350
     Module    :                       A000000151535041
    Load File  :      LOADED (--------) "testsCAS2"
     Module    :                       "testsCAS2sems"
```

SEMS LITE APPLET CODE FOR SAVE/RESTORE

- Imports

```
import org.globalplatform.upgrade.Element;
import org.globalplatform.upgrade.OnUpgradeListener;
import org.globalplatform.upgrade.UpgradeManager;
```

. .

- Install Method

```
public static void install(byte[] bArray, short bOffset, byte bLength) {
    // GP-compliant JavaCard applet registration
    if (UpgradeManager.isUpgrading()) {
        new com.nxp.cas2.semslite.test.Test().register(bArray, (short) (bOffset + 1), bArray[bOffset]);
        return;
    }
    new com.nxp.cas2.semslite.test.Test().register(bArray, (short) (bOffset + 1), bArray[bOffset]);
}
```

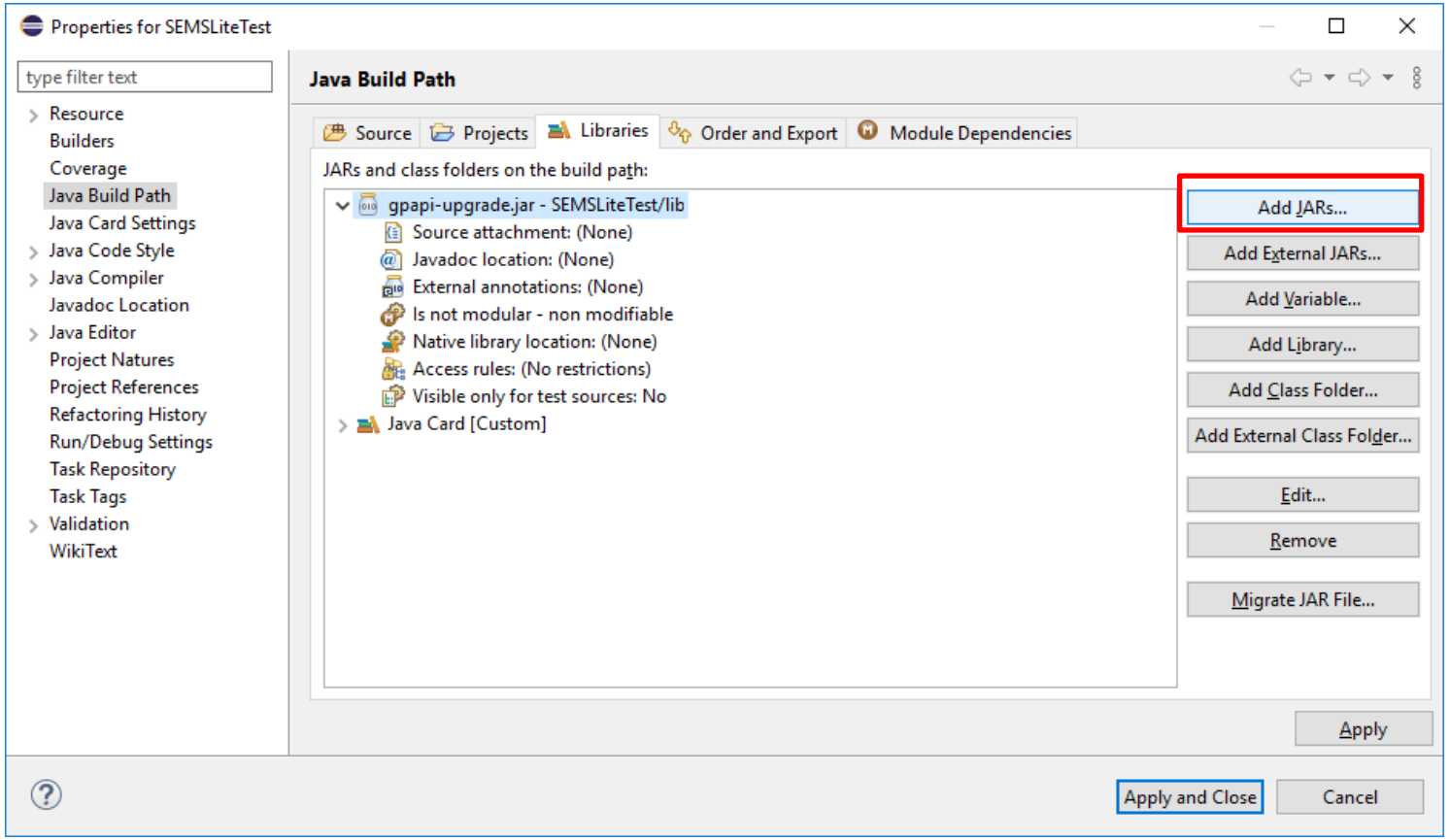
…

- Save/Restore

```
public Element onSave() {
    Element e = UpgradeManager.createElement(Element.TYPE_SIMPLE, (short)8, (short)1);
    e.write(data);
    return e;
}
public void onRestore(Element e) {
    e.initRead();
    data = (byte[])e.readObject();
}…
```

INSTALL GP UPDATE LIB

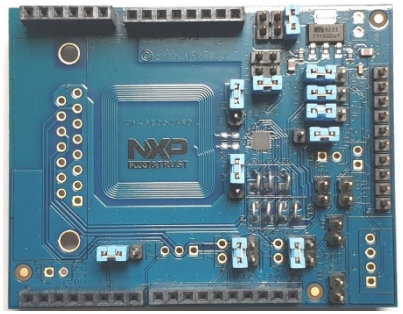
- API from https://globalplatform.org/specs-library/globalplatform-card-api-org-globalplatform-upgrade-v1/
- put gpapi-upgrade.jar in project folder
- Use "Add JARs…"



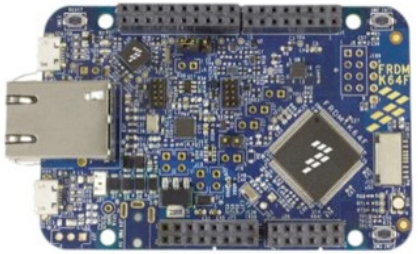
Handson

SECURE CONNECTIONS
FOR A SMARTER WORLD

PREREQUISITES



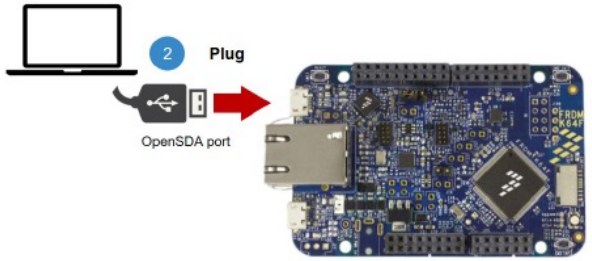
- SE051P board (blue)

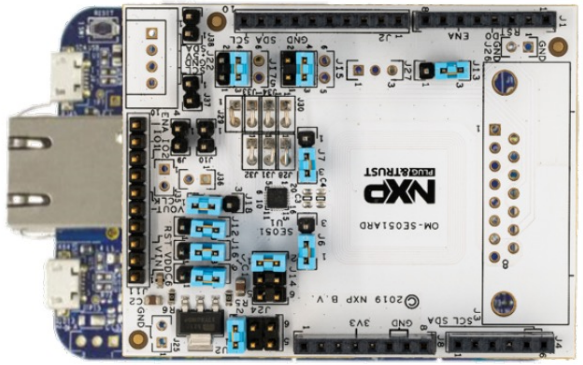


- K64F
  - Board
  - Micro-USB cable
- Ensure to have following SW ready:
  - Python 3.x 32bit
- All SW packages are shared as well in "[Training]\SW Windows"

HANDS ON GENERAL BOARD PREPARATION

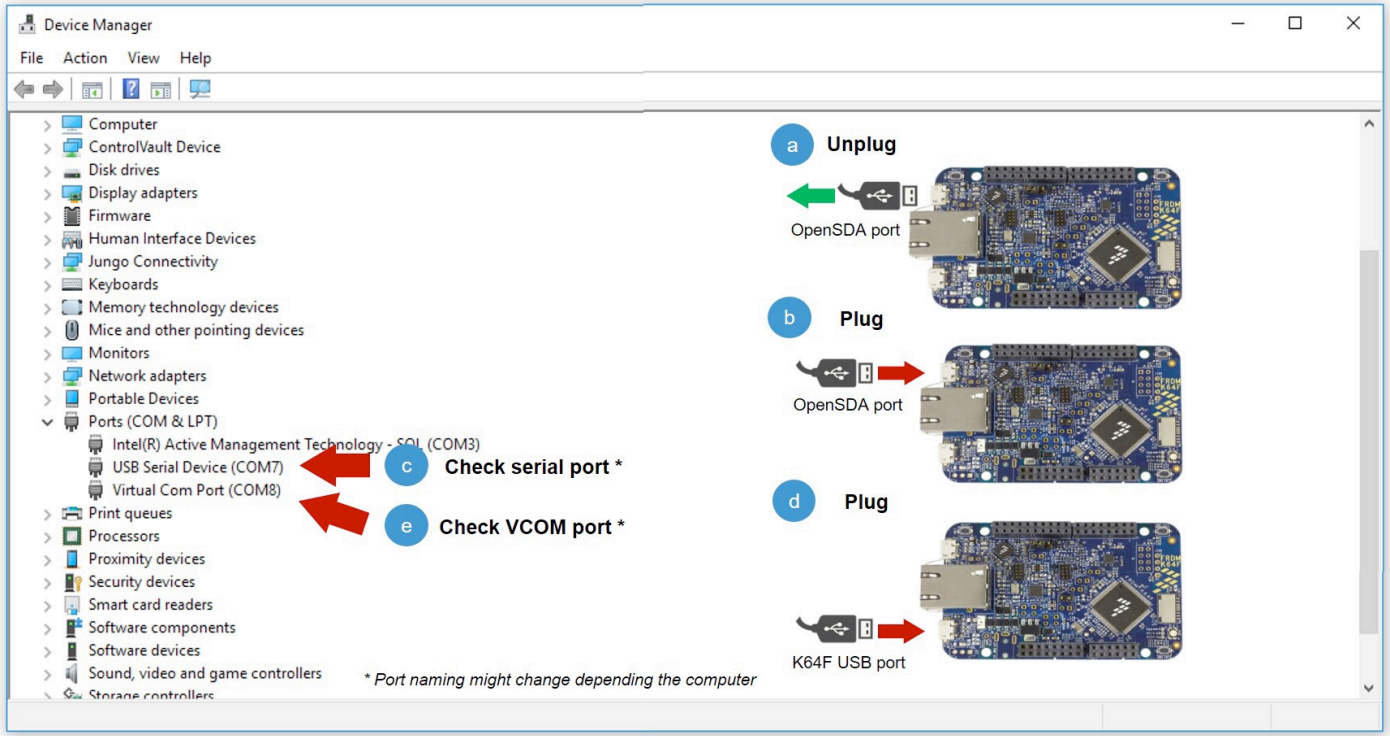
- Prepare board according to section 3 "Boards setup" of AN12398-Quick_start_guide_se050_vs_projects (2.0)
  - Potentially update DAPLink firmware on K64F
  - Check jumpers (should be already in default position)
  - Connect K64F and SE051ARD board
- Put VCOM binary on K64F according to section 4.4 "Flash FRDM-K64F with VCOM software" of AN12398-Quick_start_guide_se050_vs_projects (2.0) (USB debug port needs to be connected)





HANDS ON GENERAL BOARD PREPARATION

- Connect second USB port and find the new COM port displayed
- If no USB port is recognized, install driver https://os.mbed.com/handbook/Windows-serial-configuration
- You can connect both USB port at the same time



Identify the Product

SECURE CONNECTIONS
FOR A SMARTER WORLD

IDENTIFY THE PRODUCT
GET INFO

- Run Get info, specify the COM port of the second VCOM as argument, e.g:

```
C:\se050_mw\simw-top\binaries\ex>VCOM-se05x_GetInfo.exe COM34
```

- Check the applet version and OEF ID – should match this data:
  - OEF 0xA4A6
  - No IoT Applet

GET INFO SE051P

```
C:\se050_mw\03.00.05\simw-top\binaries\ex>VCOM-se05x_GetInfo.exe COM34
App   :INFO :PlugAndTrust_v03.00.05_20201014
App   :INFO :Running VCOM-se05x_GetInfo.exe
App   :INFO :Using PortName='COM34' (CLI)
Opening COM Port '\\.\COM34'
sss   :INFO :atr (Len=35)
      01 A0 00 00    03 96 04 03    E8 00 FE 02    0B 03 E8 00
      01 00 00 00    00 64 13 88    0A 00 65 53    45 30 35 31
      00 00 00
App   :WARN :#####################################################
App   :INFO :uid (Len=18)
      04 00 50 01    90 60 5A DE    64 30 42 04    5C 6A 32 C9
      6F 80
App   :INFO :Running VCOM-se05x_GetInfo.exe
App   :INFO :Using PortName='COM34' (CLI)
Opening COM Port '\\.\COM34'
smCom :ERROR:Can not select Applet=SE050:C'
smCom :ERROR:Failed (SW_FILE_NOT_FOUND) selecting Applet.  (Len=16)
      A0 00 00 03    96 54 53 00    00 00 01 03    00 00 00 00
sss   :ERROR:SM_RjctConnect Failed. Status 6A82
App   :ERROR:sss_session_open failed
App   :ERROR:ex_sss_session_open Failed
App   :INFO :No IoT applet found
App   :WARN :#####################################################
App   :INFO :Tag value - proprietary data 0xFE = 0xFE
App   :INFO :Length of following data 0x45 = 0x45
App   :INFO :Tag card identification data (Len=2)
      DF 28
App   :INFO :Length of card identification data = 0x42
App   :INFO :Tag configuration ID (Must be 0x01) = 0x01
App   :INFO :Configuration ID (Len=12)
      00 05 A4 A6    14 5E 7C CC    16 9A 7B C5
App   :INFO :OEF ID (Len=2)
      A4 A6
App   :INFO :Tag patch ID (Must be 0x02) = 0x02
App   :INFO :Patch ID (Len=8)
      00 00 00 00    00 00 00 01
App   :INFO :Tag platform build ID1 (Must be 0x03) = 0x03
App   :INFO :Platform build ID (Len=24)
      4A 33 52 33    35 31 30 32    39 42 34 31    31 31 30 30
      4C 09 54 E7    3E 77 3C 6E
App   :INFO :JCOP Platform ID = J3R351029B411100
App   :INFO :Tag FIPS mode (Must be 0x05) = 0x05
App   :INFO :FIPS mode var = 0x00
App   :INFO :Tag pre-perso state (Must be 0x07) = 0x07
App   :INFO :Bit mask of pre-perso state var = 0x00
App   :INFO :Tag ROM ID (Must be 0x08) = 0x08
App   :INFO :ROM ID (Len=8)
      2E 5A D8 84    09 C9 BA DB
App   :INFO :Status Word (SW) (Len=2)
      90 00
App   :INFO :se05x_GetInfoPlainApplet Example Success !!!...
App   :WARN :#####################################################
App   :INFO :cplc_data.IC_fabricator (Len=2)
      47 90
App   :INFO :cplc_data.IC_type1 (Len=2)
      D3 21
App   :INFO :cplc_data.Operating_system_identifier (Len=2)
      47 00
App   :INFO :cplc_data.Operating_system_release_date (Len=2)
      00 00
App   :INFO :cplc_data.Operating_system_release_level (Len=2)
      00 00
App   :INFO :cplc_data.IC_fabrication_date (Len=2)
      02 11
App   :INFO :cplc_data.IC_Serial_number (Len=4)
      27 11 08 99
App   :INFO :cplc_data.IC_Batch_identifier (Len=2)
      59 15
App   :INFO :cplc_data.IC_module_fabricator (Len=2)
      00 00
App   :INFO :cplc_data.IC_module_packaging_date (Len=2)
      00 00
App   :INFO :cplc_data.ICC_manufacturer (Len=2)
      00 00
App   :INFO :cplc_data.IC_embedding_date (Len=2)
      00 00
App   :INFO :cplc_data.IC_OS_initializer (Len=2)
      0D 5C
App   :INFO :cplc_data.IC_OS_initialization_date (Len=2)
      6A 37
App   :INFO :cplc_data.IC_OS_initialization_equipment (Len=4)
      31 31 30 38
App   :INFO :cplc_data.IC_personalizer (Len=2)
      00 00
App   :INFO :cplc_data.IC_personalization_date (Len=2)
      00 00
App   :INFO :cplc_data.IC_personalization_equipment_ID (Len=4)
      00 00 00 00
App   :INFO :cplc_data.SW (Len=2)
      90 00
App   :INFO :ex_sss Finished
```

HANDS ON EXAMPLES

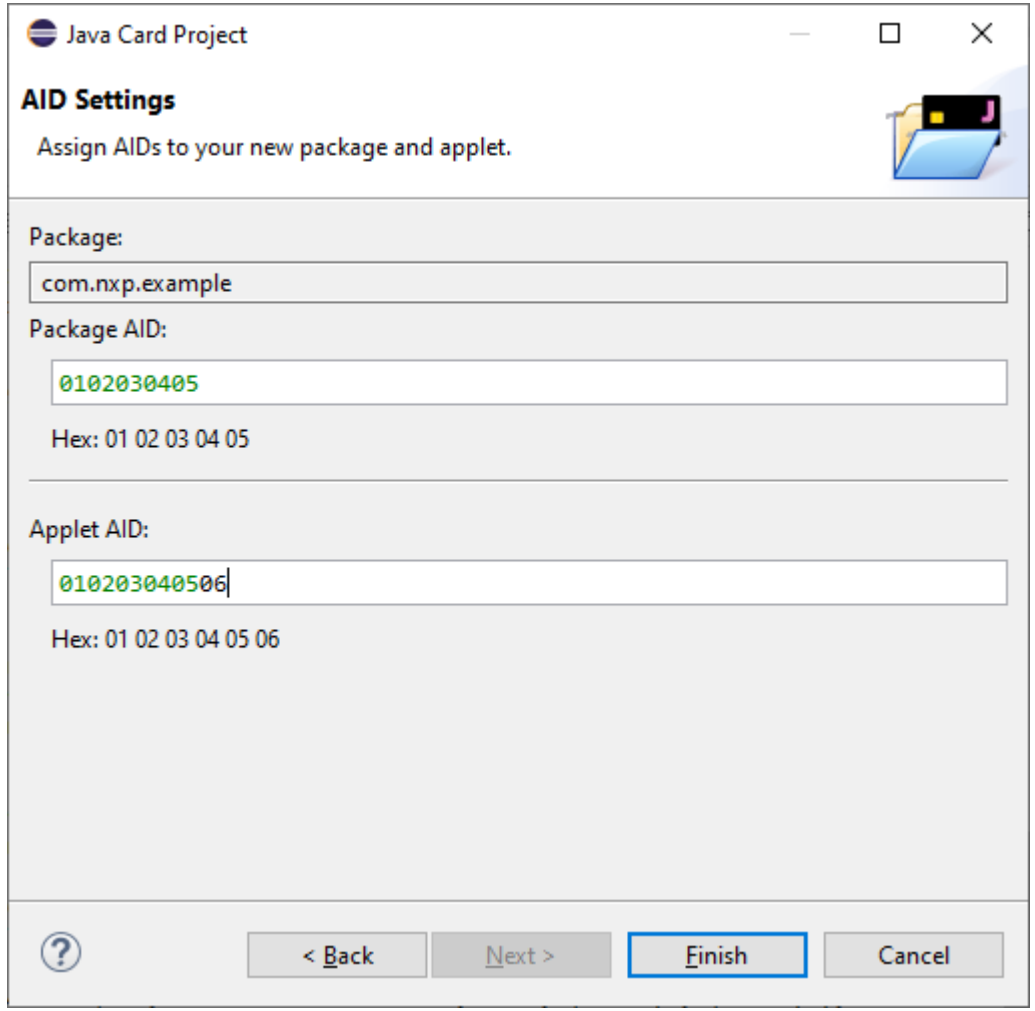
- Please fetch for this training the new MW 03.00.05 from the Trainings folder [training]\20_MW
- Sequence
  - Fetch and Install Tools
  - Fetch SEMS Lite Keyrotation and Certificate
  - Generate cap file
  - Install ls-cgt
  - Convert and apply key-rotation script to sample
  - Create ls-cgt config and run ls-cgt
  - Create config for multicast package (.json)
  - Convert multicast package in runnable script
  - Run the script

HANDSON – DIFFERENCE TO CUSTOMER APPLICATION NOTE KEY USAGE

- Normal flow is that every customer creates his own keys and NXP provides individual key rotation script and certificate for this customer on his AID
- For the training pre-generated keys are used so that everybody has the same setup
- Generated key, key-rotation script and certificate for the package AID 0102030405 are here:
  [Training]\70_DemoCode\SEMS Lite SE051P NPXCASFAE_1
  - 00 customer generated key
    contains customer-keypair SK.SP.AUT.txt
  - 01 request data sent to NXP
    customer specified AID, PK.SP.AUT.txt (public key!) to NXP
  - 02 delivery from NXP
    NXP sends back key rotation script, certificate and SEMS_CA_ENC.txt (script encryption key, public)

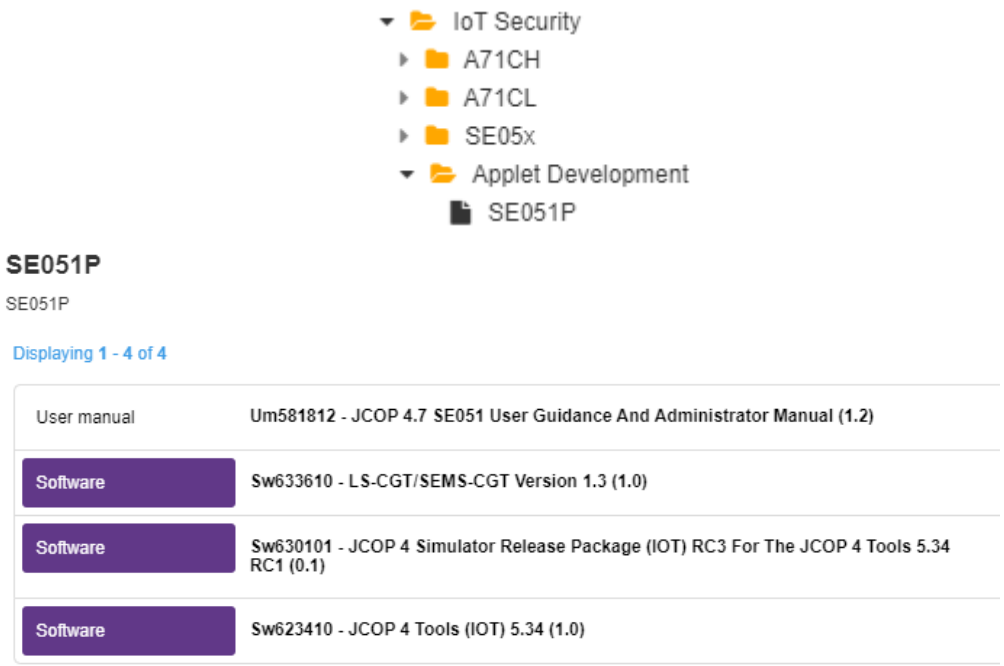
PACKAGE AID FOR THE NXPCASFAE_1 CERTIFICATE

- Don’t forget to specify the exact allowed package AID
- AppletAID is also important to be known later



HANDSON
TOOLS INSTALLATION

- Tools installation procedure is described in [Training]\10_Documentation\Applet Development\AN12909-How_to_develop_JCOP_applets_using_JCOP_Tools (Draft).pdf
  It is a Draft – contains some comments with corrections
- Fetch SW from:
  - JCOP Tools and Simulator with ls-cgt from docstore in IoT Security\Applet Development
  - Eclipse and GlobalPlatform update API from Trainingsfolder: [Training]\50_SW-Windows\Applet Development
- Install also extra simulator and GP update API as described in the AN12909



RUN/DEBUG ECHO APPLET IN SIMULATOR

- Run the example (section 4.2.3)
  - Configure Correct Simulator
- Run a debug session (section 4.2.4)
- Specify Java compiler compliance level if cap conversion fails (4.3.2)

- You can skip the trace analyzer section 4.3.3 (feel free to test)

DEPLOYMENT OF JAVA CARD APPLETS IN EDGELOCK SE051 USING SEMS LITE

- Install ls-cgt (section 5.1.2)
  - Run ls-cgt once to accept the license agreement
  - Try to run `java -jar ls-cgt.jar -help` to check that ls-cgt runs
- Compilation of sems-lite-cli can be skipped (section 5.1.3)
- Obtain credentials can be skipped (section 5.2 / 5.2.1) – fetch them from [Training]\70_DemoCode\SEMS Lite SE051P NPXCASFAE_1\02 delivery from NXP
- Convert the key rotation .json with sems-lite-generator (section 5.2.2) to .bin
  `python semslite_json_converter.py <multicast_package_path>`
  (potentially need to install python modules first: `pip install –r requirements.txt`)
- execute it on the sample with
  `sems_lite_cli_app.exe --loadpkg <path to KeyRotation-NXPCASFAE_1-1.0.bin>`
- Before running this command, you can check with commands on the next slide the active keys in the sample

BEFORE KEY ROTATION

```
C:\se050_mw\simw-top\binaries\ex>VCOM-se051_sems_lite_cli_app.exe --getCAidentifier
CA-SEMS Identifier: 00000000000000000063709317142150

C:\se050_mw\simw-top\binaries\ex>VCOM-se051_sems_lite_cli_app.exe --getCAKeyidentifier
CA-SEMS Key Identifier: 0063709317142150

C:\se050_mw\simw-top\binaries\ex>VCOM-se051_sems_lite_cli_app.exe --getENCIdentifier
EnckeyId: 00000000000000000000000000000000063709317042150

C:\se050_mw\simw-top\binaries\ex>VCOM-se051_sems_lite_cli_app.exe --getPbKeyID
PbkeyId:
042c08037208c3ce8c263a6619d25f810d730b28759eee19a567d2852045f06a089769a266633c02ea212862c34ba0
59d89c1cce23becbf69bfa1a4e8ca967ae42

C:\se050_mw\simw-top\binaries\ex>VCOM-se051_sems_lite_cli_app.exe --getUid
uid: 0400500190605ade643042045c6a32c96f80
```

AFTER KEY ROTATION

```
C:\se050_mw\simw-top\binaries\ex>VCOM-se051_sems_lite_cli_app.exe --getCAidentifier
CA-SEMS Identifier: 00000000000000000063709317742150

C:\se050_mw\simw-top\binaries\ex>VCOM-se051_sems_lite_cli_app.exe --getCAKeyidentifier
CA-SEMS Key Identifier: 0063709317742150

C:\se050_mw\simw-top\binaries\ex>VCOM-se051_sems_lite_cli_app.exe --getENCIdentifier
EnckeyId: 00000000000000000000000000000000063709317042150

C:\se050_mw\simw-top\binaries\ex>VCOM-se051_sems_lite_cli_app.exe --getPbKeyID
PbkeyId:
0454b8dc8a2706f806df635a066a34c810a157e99cbdb0c51b4bfb54849ffb45d78e09a983ec3ecaf01fd66a7cfde5
0399897cdd9e320e629db7bb72d6c30a62d0

C:\se050_mw\simw-top\binaries\ex>VCOM-se051_sems_lite_cli_app.exe --getUid
uid: 0400500190605ade643042045c6a32c96f80
```

ROTATED KEY PK.CA-SEMS.AUT

→ the key PK.CA-SEMS.AUT got changed

- Now every script needs to have a certificate which can be validated against this PK.CA-SEMS.AUT
- The certificate which was supplied by NXP is signed with SK.CA-SEMS.AUT

→ This certificate can be used now to upload new applets

PREPARE APPLET UPLOAD SCRIPT (PLAIN SCRIPT)

- Modify the applet upload script template from AN (section 5.1.3)
  (and also in training folder in 70_DemoCode\SEMS Lite SE051P NPXCASFAE_1\04 applet upload script\ )

```
manage_channel -lcid 01 -open
select -lcid 01 -aid A000000151000000
authenticate -lcid 01 -kvn 30 -seclevel 11 -sdaid A000000151000000
load -lcid 01 -cap <CAP_full_file_path> -sdaid A000000151000000
install -makeselectable -lcid 01 -elfaid <ELF_AID_of_applet> -elmaid <ELM_AID_of_applet> -aid <New_applet_AID> -privileges 00 -params C900
select -lcid 01 -aid A000000151000000
manage_channel -lcid 01 -close
#The last line of this file has to be left empty
```

- Parameters:
  - <CAP_full_file_path>
    Full path to .cap file (copy it preferably to <LS_CGT_installation_directory>\LSCGT\usage\env\capfiles)
  - <ELF_AID_of_applet>
    Executable load module AID (Package AID, 0x0102030405 in this example)
  - <ELM_AID_of_applet>
    Applet AID (010203040506 in this example)
  - <New_applet_AID>
    Applet Instance AID (010203040506 in this example)
- Store this file in <LS_CGT_installation_directory>\LSCGT\usage\env\cgtData\plainScripts as plain.txt

CREATE A SEMS LITE ENCRYPTED SCRIPT

- Copy file SEMS_CA_ENC.txt (as got from NXP – "02 delivery from NXP") to <LS_CGT_installation_directory>\LSCGT\usage\env\cgtData\keys
- Concatenate certificate data (SEMS-Lite_Cert_SP_NXPCASFAE_1_1.crt) and your customer key data (SK.SP.AUT.txt) into one file named (SK.SP.AUT.txt) which then looks like:
  CERT.AP.AUT = **<Your_CERT.SP.AUT>**
  SK.AP.AUT = **<Your_SK.SP.AUT>**
  PK.AP.AUT = **<Your_PK.SP.AUT>**
- And place this file in <LS_CGT_installation_directory>\LSCGT\usage\env\cgtData\cert
- Copy the .cap file in <LS_CGT_installation_directory>\LSCGT\usage\env\capfiles
- **Note: The naming of the files SEMS_CA_ENC.txt and SK.SP.AUT.txt cannot be chosen freely as ls-cgt expects exactly these names**

CREATE A SEMS LITE ENCRYPTED SCRIPT
RUN LS-CGT

- Go to <LS_CGT_installation_directory>\LSCGT\usage\tools
- And execute:

```
java -jar ls-cgt.jar -k "..\env\cgtData\keys\SEMS_CA_ENC.txt" -c "..\env\cgtData\cert\SK.SP.AUT.txt" -i
"..\env\cgtData\plainScripts\plain.txt" -o "..\env\cgtData\encryptedScripts\encrypted.txt"
```

```
-----------------------------------------------------------------------------
Welcome to NXP Tools LS-CGT - v1.3.0 (Fri, 3rd July 2020)
(c) 2019 NXP India
-----------------------------------------------------------------------------

[2020927-20:1:32|Cert_SkSign_PkEnc] Loading file
C:\se050_mw\SE051P\LSCGT\usage\tools\..\env\cgtData\cert\SK.SP.AUT.txt
[2020927-20:1:32|Cert_SkSign_PkEnc] Certificate and Private Key found
[2020927-20:1:32|certFiles] Parsed CertFile: ..\env\cgtData\cert\SK.SP.AUT.txt
[2020927-20:1:32|script_Ls_plainAll] Parsed LS script: ..\env\cgtData\plainScripts\plain.txt
[2020927-20:1:33|cert_sksign_pkenc] Script encrypted
[2020927-20:1:33|cert_sksign_pkenc] Encrypted script written to:
..\env\cgtData\encryptedScripts\encrypted.txt
```

- You have now an encrypted script

ENCRYPTED.TXT

- 7f2181e49310000000000000000000000000000000142100000000000000000000063709317742150 5f2010000000000004e585043415346414
55f319501824508006370931774215053154808000100000002000 14c01004d060701020304055f374041b34842f7af9b0bae3935772fb66aa7
e07ce3a496ebb079e20f41c449a01d26a4f5cf7c24da4c4898943ad33a874ca7f4d9eb0c4497022b13ff1926943d27977f49438641047933f9638
5252db1da928914695e11a523ff4e57a5c620505759dfc9d7fa417686e0857e4fb5f3eb196a55c6e5306ce4f231464197dc50fe3f0892be9f64e8f
a6081ba4181b704695b9ed2fb689ba8e38d32cbcc2501db8902005f25489ca916874f0d4983220f9fd2bb168215e1d6903c99322fd72de57771
052de28dd4a8292cb53a9c3d1b8e36b99673940d23ba18e5afbf28f0bfe6d8a27ecd9c99c49615ace774b7899623b200d9c70c3da777263a870
94b8ec686304402201ba930140d13418cf6b93179265fae8bced0e99c9c0fc3f7e0fac34afc38a86802208f998f855de07b25ed40222b2ac2656a
a3414bc63d983622ba4fed6f06d0e3d0404580a000004035ed3a6cc601b3e056db467b7160ac528a53b6c2a1041f0550fe499b2b2fd2e15b0a5
616886518366a193bf97dcbd43179c6db314b1890af56d6ecf0d308edb9404580a00000401543fc2b55c3db2579c244ddc1d8df3a3b1045e238
ad4e88679eab239c6749e5d2e7b266cae580bb2c2da7d84ecb9ae4b18117f616c107cf4de6d66dd60f3336404580a0000040aed1d6d956382bf
f87b0e4160714ab4268cdb5c86683e3a6d5326175c5fa75cc85643b033e0dcf72a07163d833d8f6e61dc4297b1cbb763cb5b0b925aa789a6640
5580a0000050ea10740a31b76bf97830e6cf0e73b71da64bc800395f3d355d0ff299ae4509cf9488eed16cc179e3ad7b8282ad0930022021f9de
cd55ac0bc60e5915a9bfd8344011ae6f6079a827e375c9246b315d224081f580a00000f083b9565f0859eb8d75e051155f0d61e42b968790729
9a2bf0a8dc262119b3f6b620e9745828274fff5c4ff3278dbd411d65cbadac7b45e4441cf9c75528a67d323ff80ce3294a3072e71a589242bf5cc9
89b1cd92eea435ca3397500f85985f77badf6b4b12542e6b50ab2a68856a70f0c01bac8405b3c42ee0062b8a5e17515bc2356680737a89acd3e
449ec075c11eabe8c5fa81a5faffdf55b3fda6f3f92d55b5236e4ff99e217e840b57cd6c8ada1196bc9a9d68690adf606ff8bd4075033223629effa6
6a113456ee438f2b96da061ad4da04adb192f872e31dca49539b43b9ea759ec9b6ac7b4103432699341640819580a0000090c57dd13722467
9143649db4781649dbbfba21be52788260b3d571876e8f48262dc7b77cfafb8c852d63a62480d89bd23467ee542e2b1471b11c392ede84ea2e
a57f903d609424829db902b3fd48d36a03306e408fc07f04b050a3584cddd00a12db665ec619ac985fc1e6a5219933816833ac6e9c0ba892328f
23e7dca81645ae5386a73166c23f619156b2d43f3c42d405580a00000506a47db97e9472c465024d27898843a5af2006aa1e195e699d033263
7b4cd0716c6b0e3e76790b56bfc3640d8dbaa31f192e4b6faa9d371b31099fb70aeb5e0bc04df5a07543adcc30dbb864322b68843404580a000
00405d80ccdc4436a52e61a49170853104042e583ee8da1eae54c89a644132e4770b703c4446dd652deb562d81b25ddde2dbfb3a7b0b7509a
de5559548cb8226a029404580a080004061abc4d019cb81a359209d0ae745920e5d8012f524c2e82b615d626edba5c394df3f619804a92fdb4
057192dc43dd748ea778adc52bc498ce80524c014b81119

CONVERT ENCRYPTED.TXT TO MULTICAST PACKAGE TO .JSON

- Like described in section 5.4.2:
- open <MW_installation_path>\simw-top\semslite\tools\sems-lite-generator\Config\ExampleConfig.json in a text editor and save it afterwards as config.json
- Modify the configuration fields, important fields:
  - TargetEntityID
    You can obtain this value using the SEMS Lite CLI tool with the –getCAIdentifier
    Note: *make sure to send the command after you have run the key rotation script*
  - requiredFreeBytesNonVolatileMemory
    Add the required non-volatile memory for the applet in bytes. This is the total amount of memory required, including the temporary additional memory required when updating the applet – 0 means no check needed
  - requiredFreeBytesTransientMemory
    Add the required RAM for the applet in bytes. This is the total amount of memory required, including the temporary additional memory required when updating the applet – 0 means no check needed
  - MulticastPackageName
    Give a name of your choice to the multicast package;
  - MulticastPackageDescription
    Provide a description for the multicast package that will be generated;
  - SubComponentMetaData
    provide a list of metadata information about the applets. In this case, the name of the applet, the AID of the applet (ELF AID) and the version of the applet;
    Note: *the SubComponentMetadata parameter must be empty when you want to delete an applet from the SE.*
  - MulticastPackageVersion
    Provide a version for the multicast package.
- ***Note:*** the JSON configuration file may contain many more configuration and metadata parameters. All parameters are defined in MulticastPackage.jsonschema file in <MW_installation_path>\simw-top\semslite\tools\semslite-generator\schema. The JSON schema is used by the tool to validate the configuration before creating the multicast package.

Edited config.json:

```
{
    "Copyright": "Copyright 2020 NXP",
    "TargetCommercialName": "SE051P2HQ1/Z011A",
    "Target12nc": "123456789012",
    "TargetEntityID": "00000000000000000063709317742150",
    "requiredFreeBytesNonVolatileMemory": 0,
    "requiredFreeBytesTransientMemory": 0,
    "MulticastPackageName": "Install_EchoApplet",
    "MulticastPackageDescription": "Install EchoApplet",
    "SubComponentMetaData": [
        {
            "name": "Example Echo Applet",
            "aid": "010203040506",
            "version": "0.1"
        }
    ],
    "MulticastPackageVersion": "1.0"
}
```

GENERATE MULTICAST JSON PACKAGE

- Convert encrypted.txt using config.json to a multicast package:

```
cd <MW_installation_path>\simw-top\semslite\tools\sems-lite-generator
python MulticastPackageCli.py --script_file <encrypted_sems_script> --
config_file <multicast_config_file_path> --out
<output_folder>\<output_package_name>.json
```

e.g. (generate output folder and run conversion)

```
mkdir c:\se050_mw\SE051P\installApplet\out
cd C:\se050_mw\simw-top\semslite\tools\sems-lite-generator
python MulticastPackageCli.py --script_file
c:\se050_mw\SE051P\LSCGT\usage\env\cgtData\encryptedScripts\encrypted.txt
--config_file "c:\se050_mw\SE051P\installApplet\config.json" --out
c:\se050_mw\SE051P\installApplet\out\installEchoApplet.json
```

CONVERT MULTICAST JSON PACKAGE FOR SEMS LITE CLI

- Convert again using semslite_json_converter.py:
  python semslite_json_converter.py <multicast_package_path>
- E.g.

```
C:\se050_mw\simw-top\semslite\tools\sems-lite-generator>python semslite_json_converter.py "c:\se050_mw\SE051P\installApplet\out"
c:\se050_mw\SE051P\installApplet\out\installEchoApplet
OEF ID : A4A6
```

- Update files created:

installEchoApplet.bin
installEchoApplet.c
installEchoApplet.h
installEchoApplet.jcsh
installEchoApplet.json
installEchoApplet.proto
installEchoApplet.tmpbin

APPLY THE UPDATE FILE

- Run SEMS Lite CLI:
  - VCOM-se051_sems_lite_cli_app.exe –loadpkg <sems_lite_script_path>\<sems_lite_script>.bin
  - Dont' forget to specify the COM-port to be used before:
    set EX_SSS_BOOT_SSS_PORT=COM34

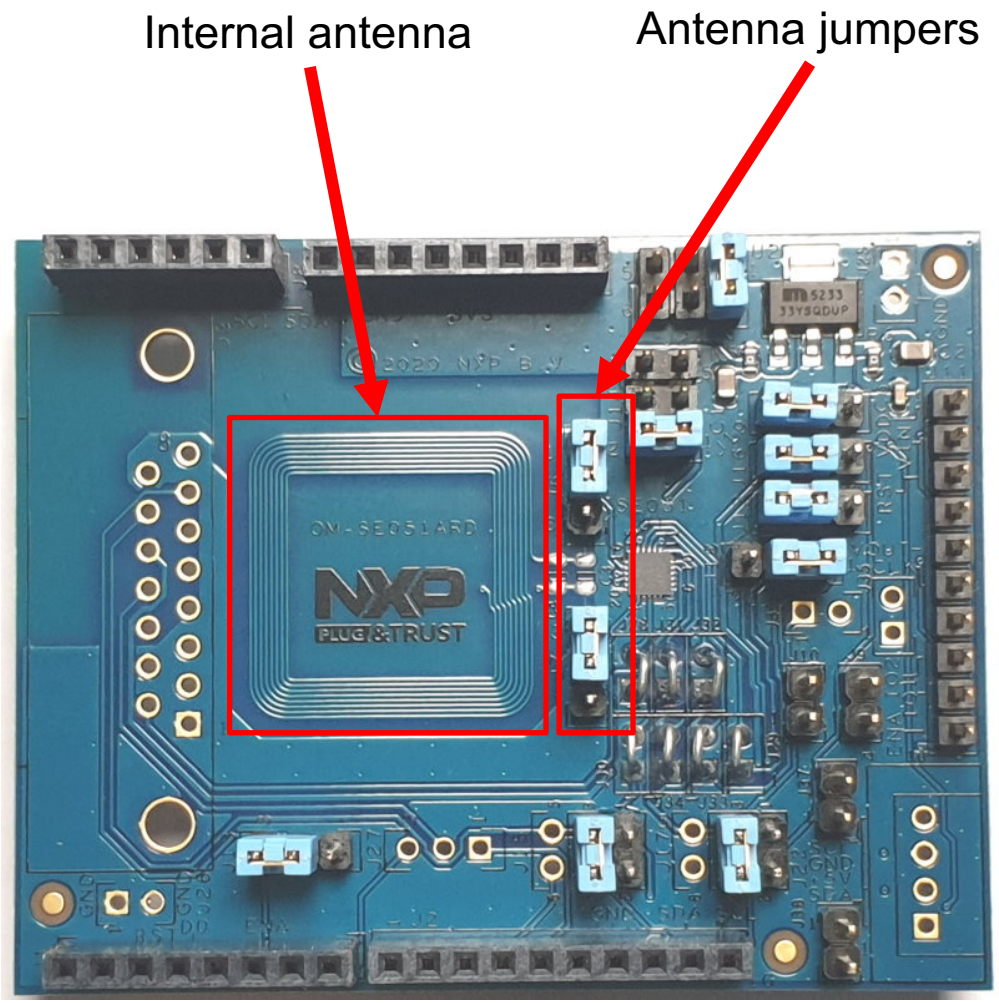
```
C:\se050_mw\simw-top\binaries\ex>VCOM-se051_sems_lite_cli_app.exe --loadpkg
c:\se050_mw\SE051P\installApplet\out\installEchoApplet.bin
App    :INFO :Using PortName='COM34' (ENV: EX_SSS_BOOT_SSS_PORT=COM34)
Opening COM Port '\\.\COM34'
sss    :INFO :atr (Len=35)
       01 A0 00 00    03 96 04 03    E8 00 FE 02    0B 03 E8 00
       01 00 00 00    00 64 13 88    0A 00 65 53    45 30 35 31
       00 00 00
App    :INFO :Skip min previous version verification.
App    :INFO :SEMS Lite Check Status Word: Success.
SEMS Lite Status: SUCCESS
```

CONNECT TO SE051P USING JCSHELL TO SEND AN APDU

- Connect with jcshell, either:
  - let the K64F board emulate a CCID reader for you (next) slide
  - use a contactless reader and use the contactless interface (see below)

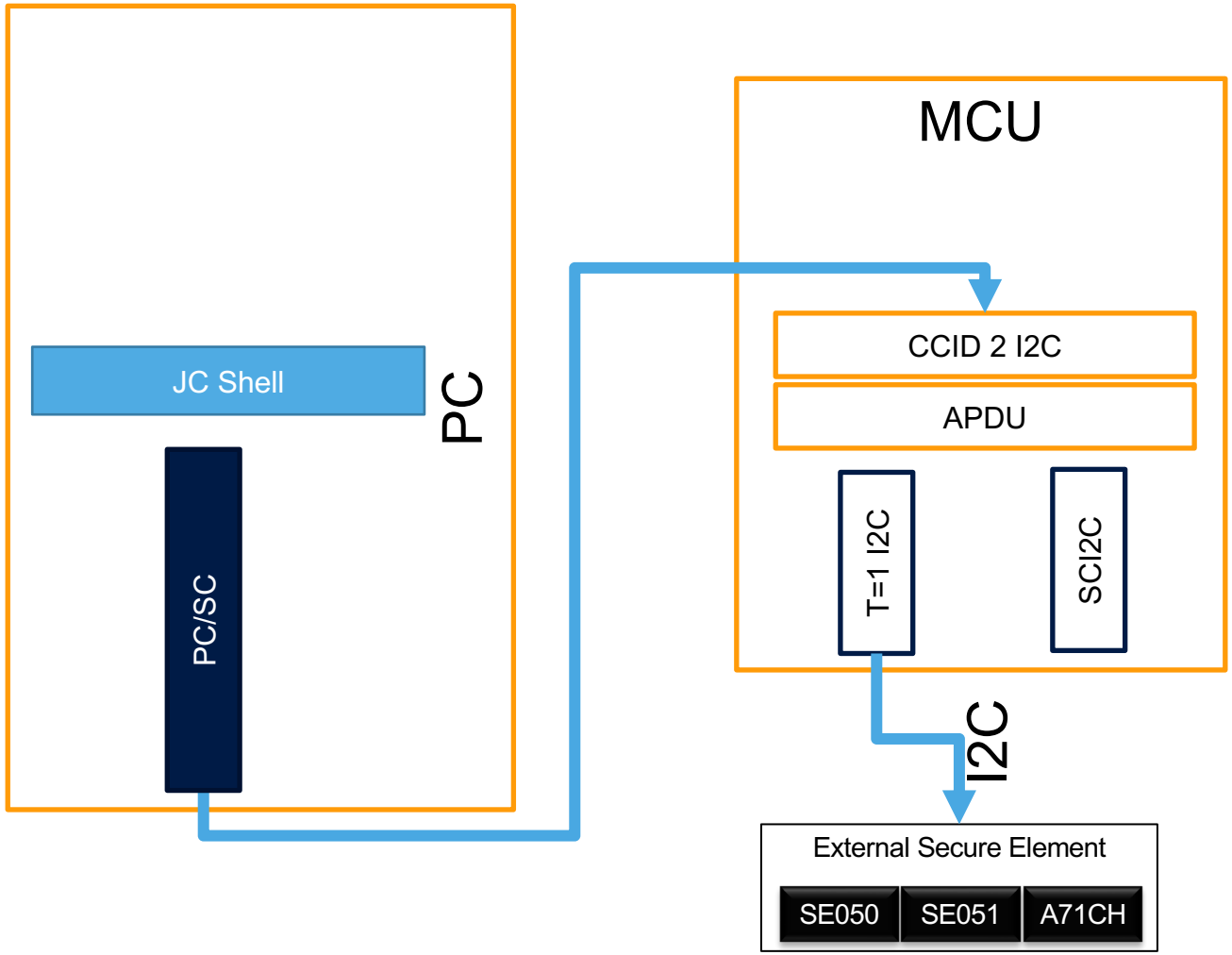
- Note: Contactless only works when the contact side is not powered!

Jumper	Description	Configuration
J6, J7	Antenna	1-2: via DB15 connector 2-3: Internal antenna



CCID (PC/SC) INTERFACE (E.G. KINETIS K64F) FOR JCSHELL

- MCU uses CCID to I²C bridge firmware
- Example implementation in MW included for K64F
  - Source
  - Binary
- Copy FW from simw-top\binaries\a7x_ccid-T1oI2C-frdmk64f-SE050x.bin on K64F board →CCID reader on host USB port active
- Connect with jcshell to this reader using PCSC: /term



USE JCSHELL TO TEST THE INSTALLED APPLET

- Connect to reader:
  /term
  /atr
- Select the applet and send the echo request

```
cm>  /select 0102030405
 => 00 A4 04 00 05 01 02 03 04 05 00                    ...........
 (872700 nsec [SYS], 4 JavaCard byte codes [DEV])
 <= 90 00                                               ..
Status: No Error
cm>  /send 80AA00000401020304
NAD=FF
 => 80 AA 00 00 04 01 02 03 04                          .........
 (722600 nsec [SYS], 22 JavaCard byte codes [DEV])
 <= 80 AA 00 00 04 01 02 03 04 90 00                    ...........
Status: No Error
```

DELETE THE APPLET

- Create an applet delete script
  - Based on templates in the slides or the AN
- Filled templates and output files are as well in [Training]\ 70_DemoCode\SEMS Lite SE051P NPXCASFAE_1\10 appletDelete
- You can directly run the generated .jcsh files in JCShell
  Be aware: No checks done as with sems-lite-agent

SECURE CONNECTIONS
FOR A SMARTER WORLD