AN12665

EdgeLock SE05x for Blockchain ID

Rev. 1.1 — 22 August 2022 Application note

Document information

Information	Content
Keywords	EdgeLock SE05x, Blockchain, Hyperledger® Sawtooth
Abstract	This application note describes how to leverage EdgeLock SE05x for Blockchain applications.

Revision history

Revision history

Revision number	Date	Description
1.1	2022-08-22	Remove "ECDAA" in Section 2
1.0	2021-04-22	First document release

1 Introduction

The blockchain technology is increasingly being used in the industry as a way to keep track of logs and transactions produced by connected IoT devices. These devices may belong to internal and external entities and organizations that do not necessarily share a trust relationship (e.g. in a supply chain). Blockchains ensure, through cryptographic mechanisms, the integrity and immutability of the chain of transactions at any time, so that each party can trust the information that is published.

To make sure that only authorized devices are publishing transactions in the blockchain, transactions can be signed by IoT devices and verified by the blockchain upon reception. Storing securely the private keys that are used to sign transaction requests in a Secure Element (SE) such as EdgeLock SE05x is therefore of utmost importance to guarantee the authenticity of transactions.

EdgeLock SE05x is a ready-to-use SE solution that provides a secure, CC EAL 6+ certified tamper-resistant hardware to accommodate all the security needs of an IoT device. EdgeLock SE05x provides a root of trust at the IC level and gives an IoT system a state-of-the-art, edge-to-cloud security capability right out of the box. EdgeLock SE05x secure memory allows the user to protect mission critical cryptographic keys, such as the private keys used to sign Blockchain transactions, and use those keys to perform cryptographic operations in EdgeLock SE05x secure hardware.

This document introduces the Blockchain technology and demonstrates how EdgeLock SE05x can be leveraged to support the authentication of blockchain transactions. A demo example that showcases the EdgeLock SE05x functionalities in the context of Blockchain is also presented.

2 Blockchain for the industry 4.0

The blockchain is a distributed, shared ledger formed by a chain of ordered data blocks that contain one or more records or transactions. The main distinguishing feature of the blockchain is the impossibility for any given party to retroactively change the data stored inside a block. This is achieved by linking blocks using cryptography: every block contains the transaction data, the hash of the block and the hash of the previous block. The hash of a block is used to verify the integrity of the block and it is calculated before the new block is added to the chain. The hash of the previous block is also included in the hash computation, so blocks are cryptographically linked to form a chain as shown in Figure 1.

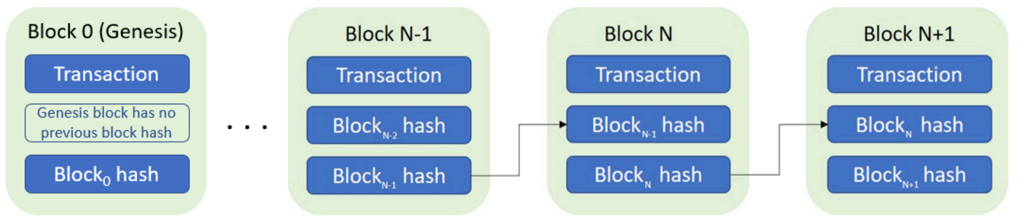


Figure 1. Sequence of blocks linked to each other forming a chain.

By recurring the chain backward from the last block to the first generated block (genesis block) it is possible to verify the integrity of the whole chain. Tampering with the content of a block in the chain would change the hash of the block and with it the hash of all subsequent blocks. This property makes blockchains resistant to attacks aimed at tampering with the integrity of the stored information, even if such information is public and available to untrusted parties.

The reliability of blockchains is also ensured by the distributed nature of the ledger. Instead of using a central entity to manage the chain, blockchains take advantage of a peer-to-peer network where every node has its own copy of the ledger. When new transactions are added to the blockchain, they are sent to special nodes called validators. Validators verify transactions and broadcast the result to all other nodes. A consensus algorithm takes care of determining if transactions are valid or not. If the majority of validators behave legitimately, the integrity of the blockchain can be ensured.

Depending on whether the identity of the participants in the network is known a priori or not, blockchains can be divided in two categories: permissionless blockchains and permissioned blockchains.

Permissionless blockchains are public and openly accessible by anyone having an internet connection. Each user may maintain a copy of the ledger and may act as validator for transactions. Any member in a permissionless blockchain can access and create transactions at any time. The access model provided by permissionless blockchains might not be adequate for all use cases, especially if data has to be kept private and made accessible only to a few authorized parties.

Permissioned blockchains solve this problem by providing a closed ecosystem where all the participants are known at any point in time. This allows an entity or organization to enforce participation policies and to control access to transaction details, thus providing enhanced privacy, improved auditability and increased efficiency. Moreover, permissioned blockchains require less memory and time resources to run. It is for these reasons that permissioned blockchains are becoming more and more popular among industry-level enterprises and businesses for which security, identity, and role definition are important or even mandatory requirements.

Permissioned blockchains are ideal to support a wide array of business-related use cases, including:

- **Commercial financing:** businesses need visibility when purchasing goods and services so they can avoid and/or resolve disputes. Using blockchain, they can have complete visibility of the order-to-delivery pipeline and in this way reduce the time required to resolve disputes.
- **Supply Chain management**: businesses suffer a lack of transparency as products move along the supply chain. By assigning it a unique identity, a product can be tracked as it moves along the supply chain, from the supplier facilities to the customer facilities. A transaction can be registered in a shared blockchain whenever the product changes hands. Through the blockchain, authenticity of purchased goods and services can be verified by customers.
- **Internet of Things**: due to its inherent scale and distributed nature, IoT greatly benefits from the blockchain technology. The blockchain can be used to keep a permanent record of measurement logs collected in the field by IoT devices and to track down a device that generated a particular log.

EdgeLock SE05x can be used to support the integration of the abovementioned use cases in the customer's own blockchain solution. EdgeLock SE05x can be leveraged to provide a unique identity to the device and to securely store private keys that are necessary to sign new blockchain transactions. The secrecy of the private key of the device is required to guarantee the authenticity of the records in the blockchain. In fact, the ownership of the information stored in the blockchain about a device is linked to the unique identity of the device and to the signature the device performs on the transaction request using its private key (the public key is known to the blockchain). When a new transaction is produced by the device, EdgeLock SE05x can be leveraged to create a signed transaction request before adding it to the blockchain. EdgeLock SE05x natively supports all common cryptographic signature algorithms (RSA, ECDSA, EdDSA) and hash algorithms (SHA-1 to SHA-512) for this purpose. Private keys that are used to sign transactions are securely stored in EdgeLock SE05x tamper-resistant environment and never leave the boundaries of the secure element. For additional security, EdgeLock SE05x comes with a set of device-unique pre-injected credentials, identifiers and certificates that can be used to secure blockchain transactions and for secure authentication of a device to a permissioned blockchain network.

3 EdgeLock SE05x for Blockchain project example

EdgeLock SE05x provides a high level of security to the participants of a Blockchain network, acting as secure storage for device keys and credentials, including the device unique identifier, the keypair used to sign new Blockchain transaction requests and, optionally, the TLS credentials to connect to the blockchain.

Figure 2 shows how EdgeLock SE05x can be used to support Blockchain applications for IoT devices.

Note: *this document does not illustrate how to leverage EdgeLock SE05x to establish a TLS connection with the Blockchain. Please refer to EdgeLock SE05x for secure connection to OEM cloud application note for more information about EdgeLock SE05x TLS support.*

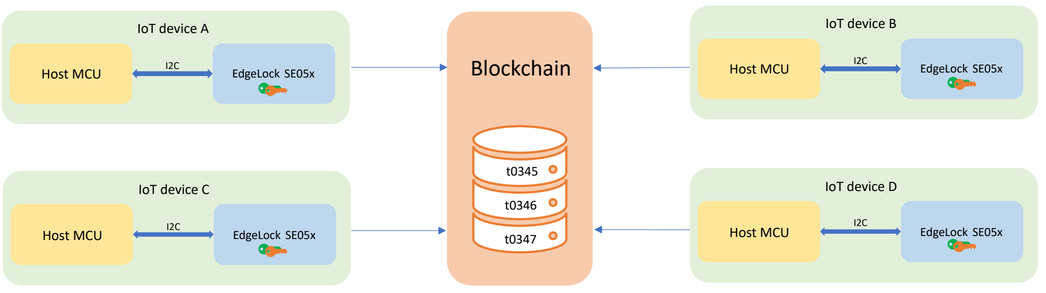


Figure 2. Adding EdgeLock SE05x to IoT devices participating to the Blockchain.

The Blockchain project example showcases how to leverage EdgeLock SE05x to register transactions in a Blockchain deployed using Hyperledger® Sawtooth. This section describes how to run the Blockchain demo example included in the EdgeLock SE05x Plug & Trust middleware using the OM-SE05xARD and FRDM-K64F boards.

Figure 3 describes the initial setup of the Blockchain project example.

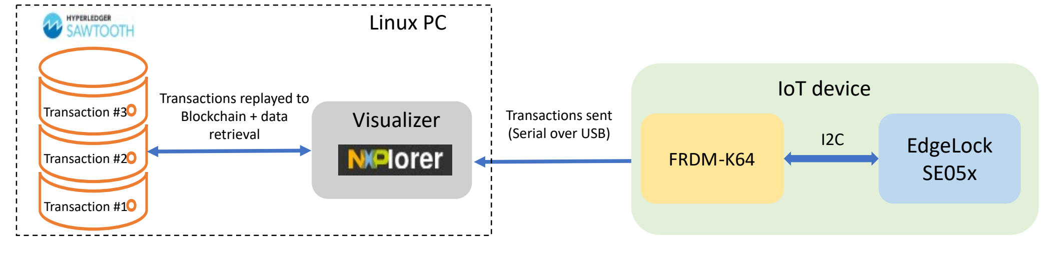


Figure 3. Initial setup of Blockchain project example

The IoT device is realized using an FRDM-K64F development board and an OM-SE05xARD board. The OM-SE05xARD is an ideal development kit to evaluate EdgeLock SE05x features and build a proof of concept or a prototype of an IoT-enabled blockchain solution before going into production. The OM-SE05xARD comes with headers and connectors that allow the user to access the EdgeLock SE05x interfaces, including the I2C slave lines to connect to a host MCU board. The Blockchain used in the example is deployed using Hyperledger® Sawtooth, a professional solution used to develop Blockchain applications. The NXPlorer tool provides a visual interface to easily analyze what goes on inside the Hyperledger® Sawtooth blockchain and relays the transactions that are sent from the FRDM-K64F. Hyperledger® Sawtooth and NXPlorer runs in a Linux PC with Ubuntu 16.04 operating system.

To simplify the setup of the project example and avoid complex network configurations, the connection between the FRDM-K64F board and the Linux PC is done using a serial connection over USB. In this configuration, transactions are sent over the serial interface and the NXPlorer tool acts as a proxy to relay such transactions to the Hyperledger® Sawtooth blockchain. In a real scenario, transactions would be sent to the Blockchain using standard internet protocols.

Even if Hyperledger® Sawtooth is a permissioned blockchain, this demonstration will not discuss permissioning capabilities. The permissioning is handled by the blockchain configuration and not by the device. We will assume that the device is at some point of time authorized by the blockchain through the blockchain management functions.

Note*: the Blockchain project example has only been fully tested on Linux (Ubuntu 16.04). However, the example might also work in other Linux distributions and Windows.*

Note: *the procedure described in this section and the Blockchain demo example are provided only for evaluation purposes. The procedure described in the following sections must be adapted and adjusted accordingly for a commercial deployment.*

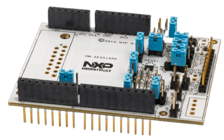
3.1 Hardware required

This guide provides detailed instructions on how to run the Blockchain project example using the hardware shown in the list below. However, this demonstration also works with LPC55S69 board (with and without EdgeLock SE05x) and iMXRT1050 board (with EdgeLock SE05x). Other platforms might be supported with some adaptation to the code.

Table 1. EdgeLock SE05x development boards.

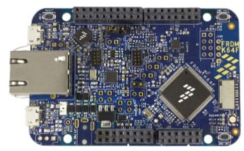
Part number	12NC	Description	Picture
OM-SE050ARD	935383282598	SE050 Arduino® compatible development kit	

Table 1. EdgeLock SE05x development boards. *...continued*

Part number	12NC	Description	Picture
OM-SE051ARD	935399187598	SE051 Arduino® compatible development kit	

Note: The pictures in this guide will show OM-SE050ARD, but OM-SE051ARD can be used as well with the same configuration.

Table 2. FRDM-K64F details

Part number	12NC	Content	Picture
FRDM-K64F	935326293598	Freedom development platform for Kinetis K64, K63 and K24 MCUs	

3.2 Import and run the demo project example

This section describes how to assemble the boards and flash them with the firmware required to run the Blockchain project example using a Windows PC. You can refer to FRDM-K64F quickstart guide for detailed instructions of how to download and compile the EdgeLock SE05x Plug&Trust middleware, install and configure the required software tools (MCUXpresso and TeraTerm), import the FRDM-K64F MCUXpresso project and setup the FRDM-K64F board.

Follow these steps to compile and flash the blockchain demo example firmware to the FRDM-K64F board:

1. As described in *Import project examples from CMake-based build system* section of the FRDM-K64F quickstart guide, we will be using the EdgeLock SE05x Plug&Trust middleware for this setup. Follow the instructions of the quick start guide until you reach the section *Import PlugAndTrustMW project example in MCUXpresso workspace*, then do as follows:
   a. Download the ZIP file containing the source code of the `se05x_sawtooth` project and extract its content. The ZIP file is provided in the NXP website as a related file download. You can find the download link just below the download link of this application note.
   b. Navigate to `<MW_path>/simw-top/demos` folder;
   c. Copy the entire folder `se05x_sawtooth` extracted from the ZIP file in `<MW_path>/simw-top/demos`;
   d. Open with a text editor the *CMakeList.txt* in `<MW_path>/simw-top/demos` and add a line at the end of the file with the following content: `ADD_SUBDIRECTORY(se05x_sawtooth)`. Make sure to save the file after the change is applied.

2. Open MCUXpresso and import the FRDM-K64F project that is included in the EdgeLock SE05x Plug & Trust middleware as described in [FRDM-K64F quickstart guide]() (section *Import PlugAndTrustMW project example in MCUXpresso workspace* and section *Import cmake_projects_frdm64f project example in MCUXpressoworkspace*);
3. Select and build the `se05x_sawtooth` project example as shown in [Figure 4]():
   (a) In the Project Explorer window, go to Debug folder and open the Makefile file;
   (b) The BUILD_TARGET contains the name of the project to be executed. Write `se05x_sawtooth` in the BUILD_TARGET variable;
   (c) Click on the small arrow next to the hammer icon in the top menu bar of MCUXpresso;
   (d) Select Debug (Debug build). Wait a few seconds until the build operation completes.

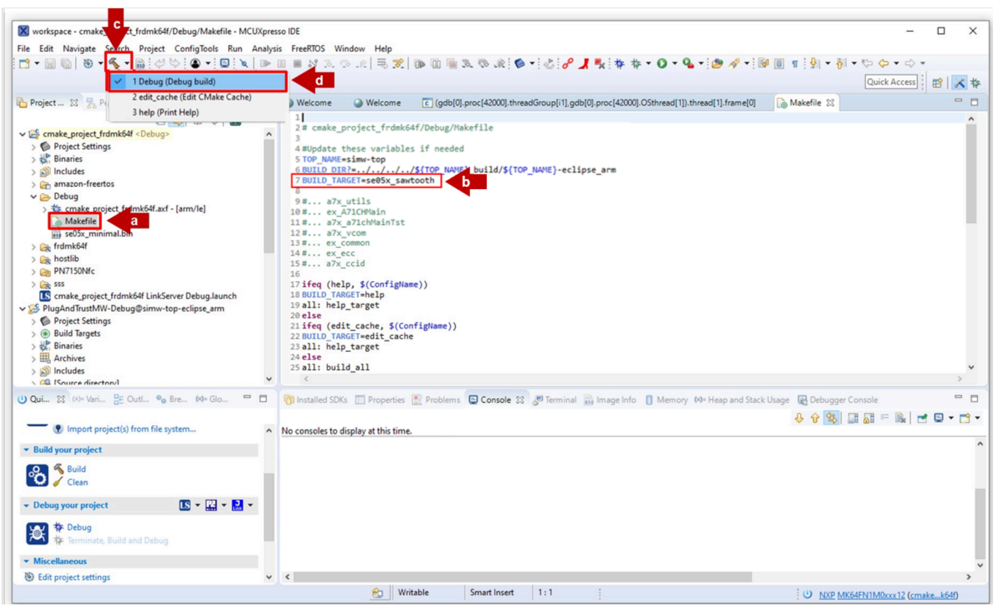


Figure 4. Change the value of BUILD_TARGET to build the correct example.

4. Connect the OM-SE05xARD board on top of the FRDM-K64F as shown in [Figure 5](). The OM-SE05xARD and FRDM-K64F boards can be directly connected using the Arduino headers present in both boards.

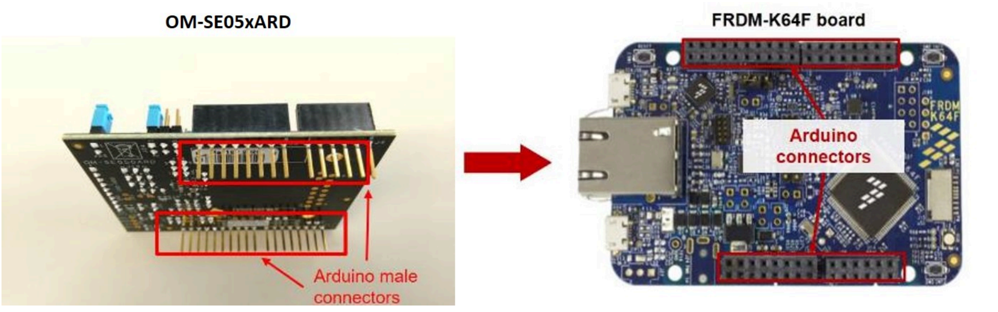


Figure 5. Connect the boards.

Double check that the two boards are connected as shown in [Figure 6]():

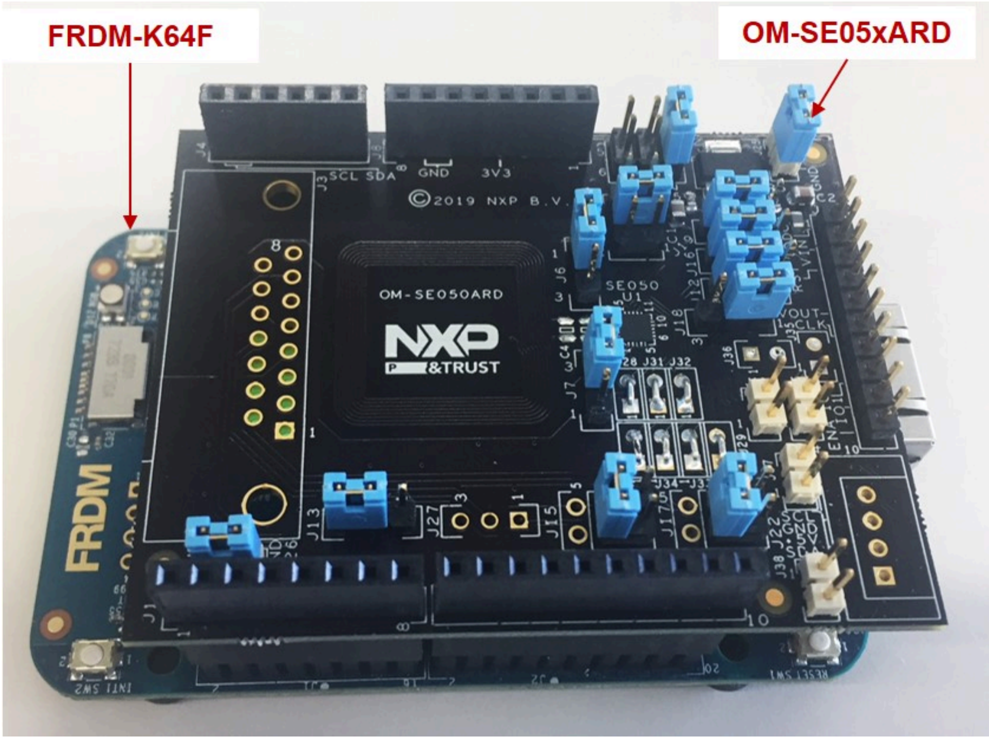


Figure 6. Boards connected.

5. Connect the FRDM-K64F board to the computer through the OpenSDA port as shown in Figure 7.

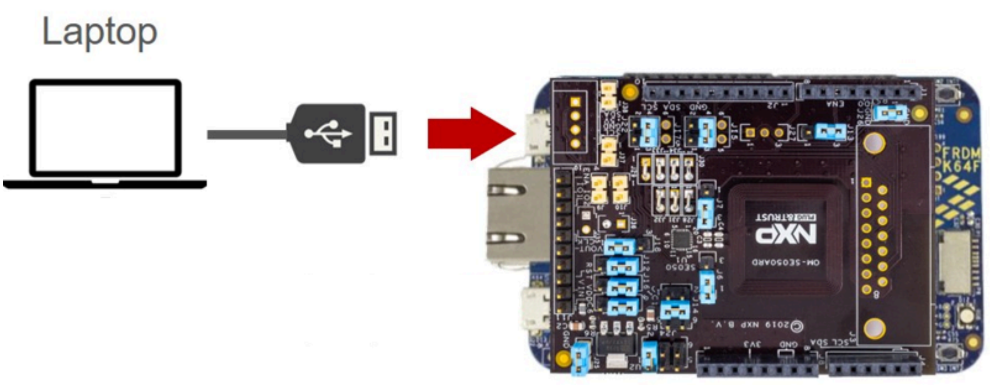


Figure 7. Connect the board to the Windows computer.

6. Download the demo application to the board as shown in Figure 8:
(1) Click on the Debug button in MCUXpresso
(2) In the popup window that appears, click on the OK button to download the firmware to the board. When the downloaded application starts, EdgeLock SE05x is provisioned with a new key pair that will be used to sign new blockchain transactions.

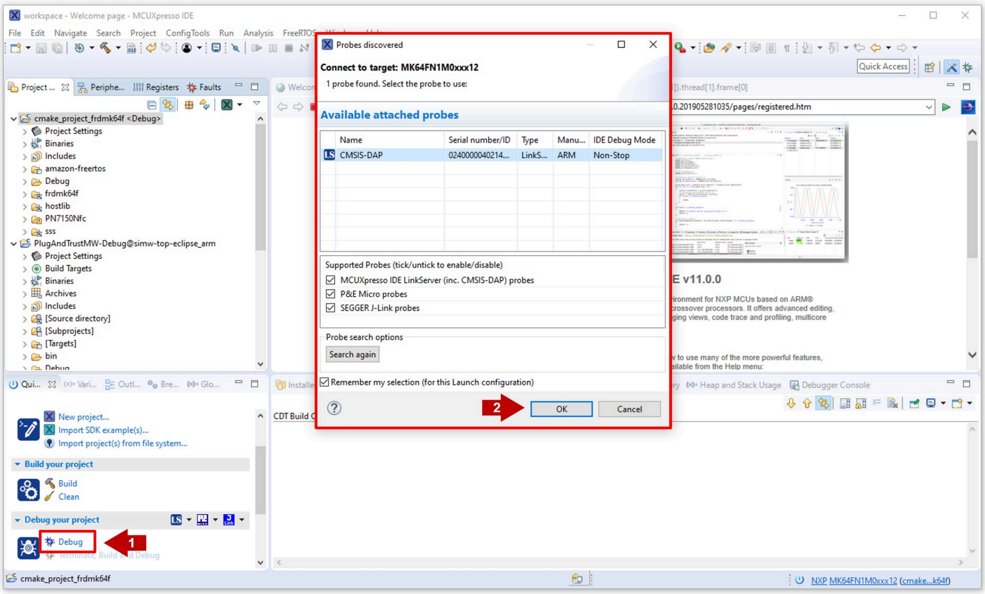


Figure 8. Flash firmware to the board

7. Open TeraTerm and connect to the board. If you click the reset button on the board to restart the application, you should be able to see a log with the public key that was provisioned in EdgeLock SE05x as shown in Figure 9:

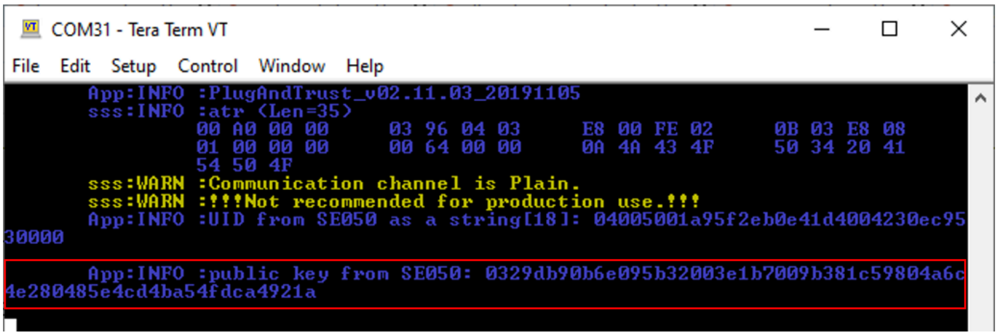


Figure 9. Public key from EdgeLock SE05x.

Figure 10 shows the state of the EdgeLock SE05x after running the project firmware in the FRDM-K64F board. When the application starts for the first time, EdgeLock SE05x is provisioned with a public key (green) and a private key (orange). Once the demo application is executed successfully for the first time, the EdgeLock SE05x will contain the key pair that will be used for all further invocations of the demo. When the switch SW3 (see Figure 22) is pressed on the FRDM-K64F, a new transaction is generated. The transaction is signed using the private key stored in EdgeLock SE05x and then sent over the serial port to the blockchain. The transaction also holds the public key that is later used by the blockchain to verify the signature.

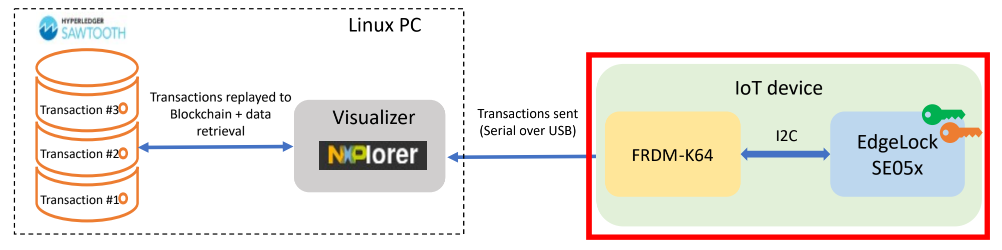


Figure 10. EdgeLock SE05x has been provisioned with public and private keys

3.3 Preparing the blockchain

Once the FRDM-K64F has been flashed with the demo project, it is necessary to deploy the Blockchain in the Linux PC before starting to register transactions. Three tools will be used to run the blockchain and visualize the transactions sent from the FRDM-K64F board:

- **Hyperledger® Sawtooth**: platform to deploy and run blockchain solutions that implement transaction-based shared ledgers. The blockchain records all the transactions chronologically as received by the connected devices.
- **Integerkey Transaction Family**: this transaction family is part of the Hyperledger® Sawtooth blockchain and allows users to set, increment and decrement the value of entries stored in a state dictionary. In this demo, the value of a key in the dictionary will be incremented every time a valid transaction is received in the blockchain.
- **NXPlorer**: graphic interface to visualize the status of the devices at any point in time, i.e. the number of times the SW3 switch on the FRDM-K64F board has been pressed per device. Additionally, and just for the purpose of this demo, NXPlorer acts as a proxy between the blockchain and the device to relay transactions sent over the serial port to the Hyperledger® Sawtooth blockchain.

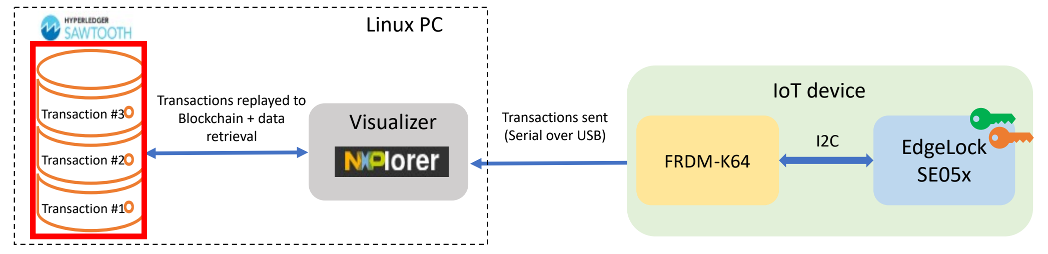


Figure 11. Preparing the blockchain for the demo.

3.3.1 Deploy the Hyperledger® Sawtooth Blockchain

Follow these steps to deploy the Hyperledger® Sawtooth blockchain in a Linux PC running Ubuntu 16.04:

1. Go to Section 4 to find the instructions of how to install the Docker Compose and Docker Engine components required to run Hyperledger® Sawtooth. Section 4 also explains how to download the Hyperledger® Sawtooth Compose file (*sawtooth-default.yaml*).
2. Move to the folder where you stored the *sawtooth-default.yaml* file and open a Terminal window, as shown in Figure 12.

Figure 12. Open a terminal in the folder where you downloaded the configuration file.

3. Make sure to terminate any previous running instance of the Hyperledger® Sawtooth Blockchain by sending the command:
   `sudo docker-compose -f sawtooth-default.yaml down`
4. Run the following command to start the Hyperledger® Sawtooth Blockchain in a new docker container:
   `sudo docker-compose -f sawtooth-default.yaml up`
5. If the Blockchain has been successfully deployed, you will see the execution finish with the message shown in Figure 13.

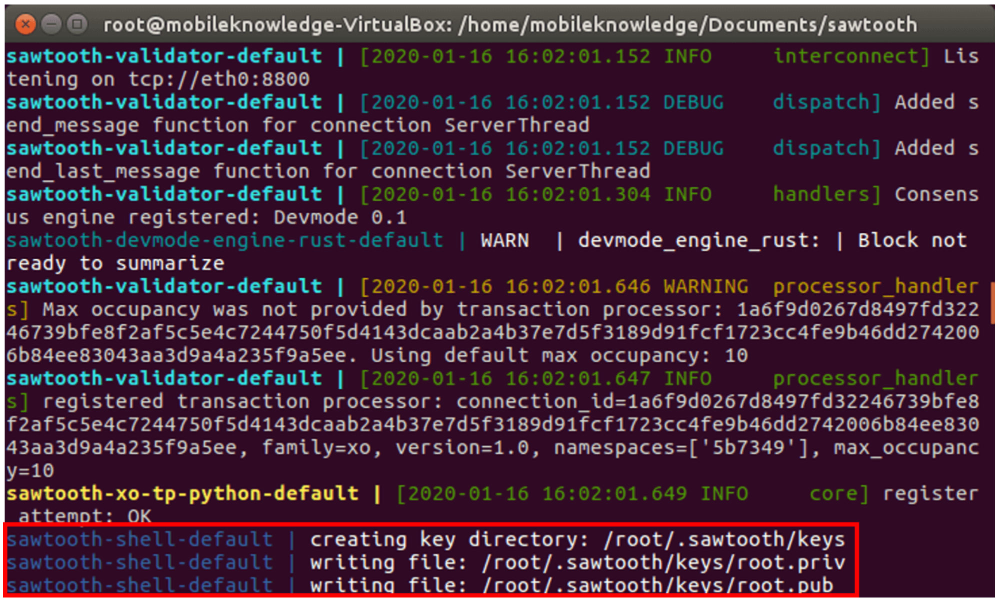


Figure 13. Hyperledger® Sawtooth successful deployment logs

6. If you don't see the same output as the one shown in Figure 13, run the following command:

`sudo docker-compose -f sawtooth-default.yaml up --force`
The command will force the deployment of the Hyperledger® Sawtooth blockchain and erase any previous instances and files that may still exist.

7. Once the Hyperledger® Sawtooth blockchain is running, it is important to leave the terminal window open. If you close the terminal window, the Hyperledger® Sawtooth process will stop and you won't be able to continue with the demo.

3.3.2 IntegerKey Transaction Family

The IntegerKey Transaction Family, also known as Intkey Transaction Family, allows users to set, increment and decrement the value of entries stored in a state dictionary in Hyperledger® Sawtooth. We will use the IntegerKey Transaction Family to increment the value of an entry each time a transaction from the FRDM-K64F board is registered in the blockchain. Open a new Terminal window and follow these steps to set a new entry in the dictionary:

1. First, you need to retrieve the `CONTAINER ID` of the sawtooth shell client. Type the following command to show all the containers running in the system:
   `sudo docker ps`
2. Write down the `CONTAINER ID` of the container whose `NAME` parameter is `"sawtooth-shell-default"` as shown in Figure 14.

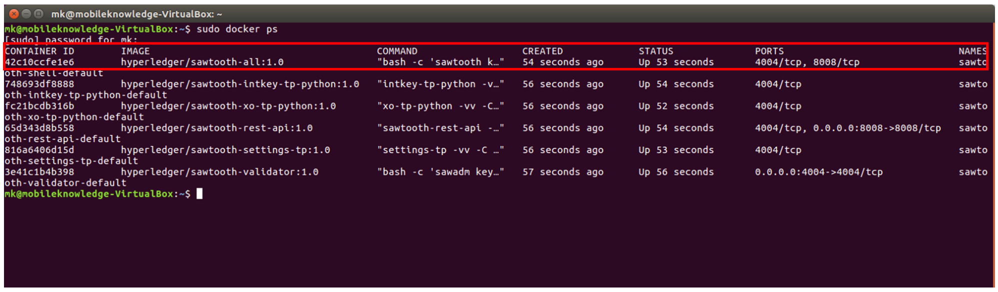


Figure 14. Retrieve the CONTAINER ID of the container called sawtooth-shell-default.

3. Run the following command to open a shell client in the `sawtooth-shell-default` container as shown in Figure 15:
   `sudo docker exec -it <CONTAIENR ID> bash`

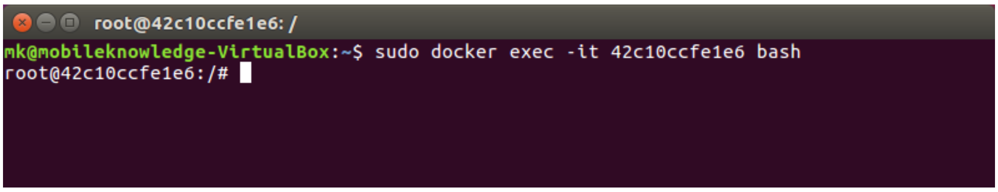


Figure 15. Login in the docker shell client.

4. Set an entry with the name `k64f_se050` and initial value `0` by typing the following command as shown in Figure 16:
   `intkey set k64f_se050 0 --url http://rest-api:8008`
   Note: *In the context of the demonstration, the permissioning capabilities of the blockchain are not used. This means that anybody has the right to increment the counter. The blockchain maintains the history of who has changed the value.*

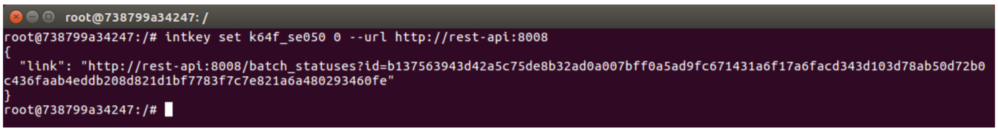


Figure 16. Set the key using the Intkey transaction family.

5. After setting the key, you will be able to see a log of the operation in the Sawtooth terminal as shown in Figure 17:

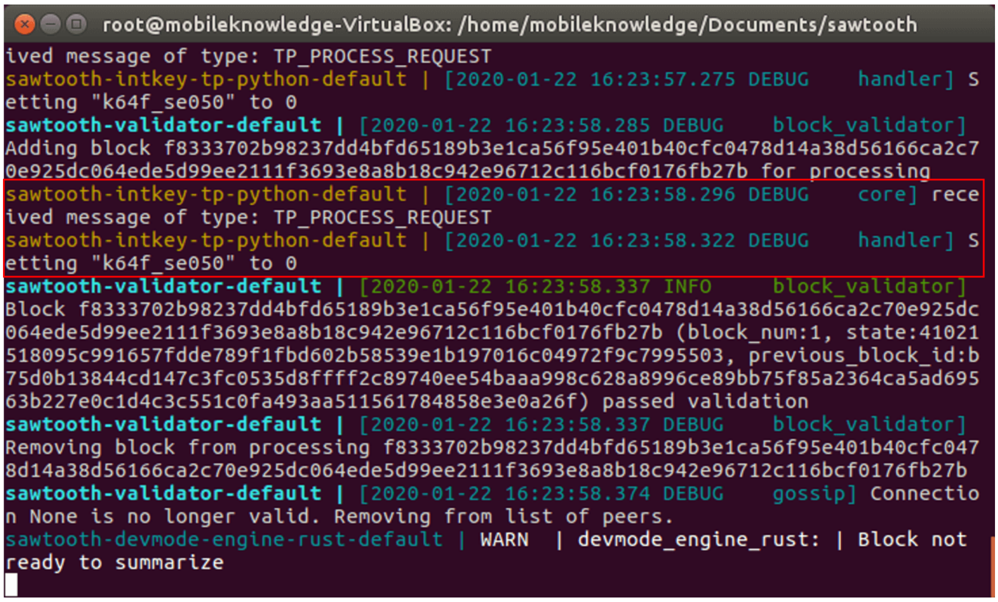


Figure 17. The request to set a key by the name of k64f_se050 is received in the Sawtooth terminal.

3.3.3 Run NXPlorer

Once you have successfully registered the key in the blockchain as described in Section 3.3.2, you can run NXPlorer and visualize the information of the key and the details about the transactions sent from the FRDM-K64F board.

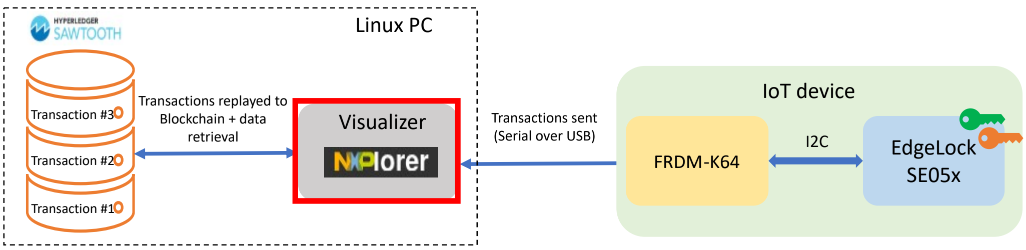


Figure 18. Run NXPlorer to visualize the data in the blockchain.

Follow these steps to run NXPlorer:

1. Go to Section 4.2 to find the instructions on how to download and install NXPlorer.

2. Go to the folder where you downloaded and installed NXPlorer, open a new terminal window and run the command:
`sudo ./NXPlorer multi -c example_config.json`
The `example_config.json` file is included in the NXPlorer package. Its purpose is to provide the necessary configuration to NXPlorer, including the configuration required to proxy transactions received from the serial port to the Sawtooth blockchain. If NXPlorer is properly started, you will see in the terminal the logs that inform you about the interfaces found and where NXPlorer is accessible, as shown in Figure 19.
Note: *if you receive an error, make sure that the board is connected to the Linux PC with a USB cable and that the serial port name in example_config.json is correct. In Ubuntu 16.04 the serial port should be /dev/ttyACMx, where x is a progressive number that might vary depending on your system. You can check the available serial ports by navigating to the /dev folder. Restart NXPlorer to apply the new configuration.*

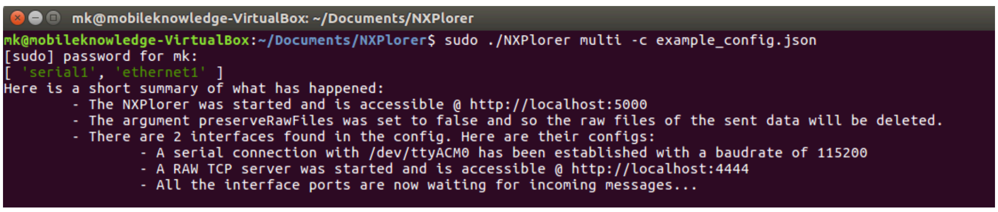


Figure 19. Run NXPlorer.

3. In a browser, open the URL where the NXPlorer is accessible (http://localhost:5000). You should be able to see a box that shows the key value that was set in Section 3.3.2 as shown in Figure 20.
Note: *Make sure not to close the terminal window where NXPlorer is running, otherwise you won't be able to receive transactions from FRDM-K64F*

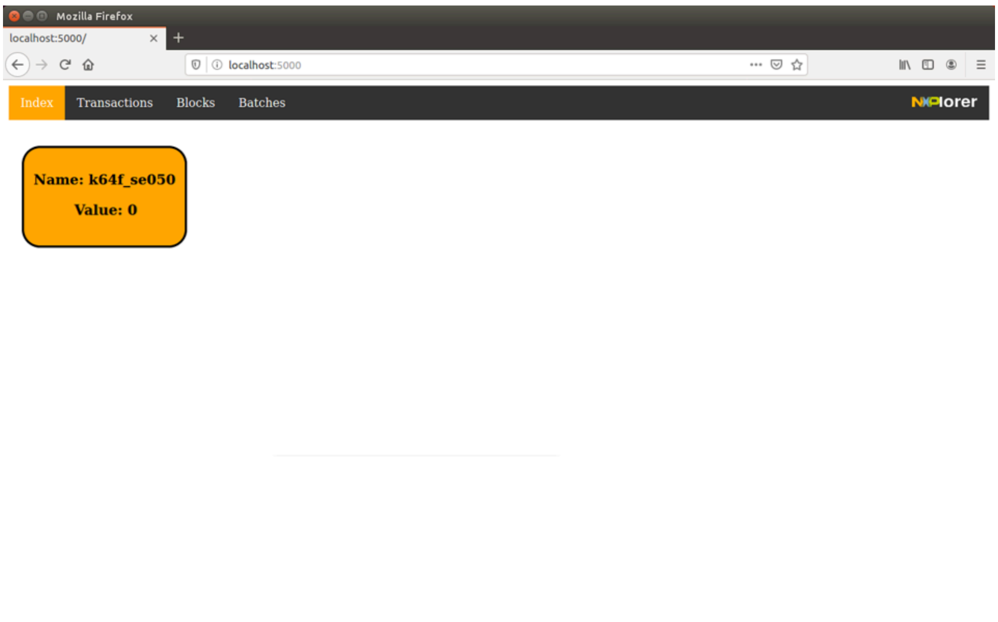


Figure 20. NXPlorer interface with the key k64f_se050 that has just been set.

3.4 Registering transactions from FRDM-K64F

Once the boards have been prepared and the Hyperledger® Sawtooth and NXPlorer tools have been deployed and are running, transactions can be sent from the FRDM-K64F using the SW3 button. Each time the button is pressed, a transaction is generated in the board and sent to the Hyperledger® Sawtooth blockchain. This transaction is signed with the private key provisioned in EdgeLock SE05x and holds the public key to verify the signature.

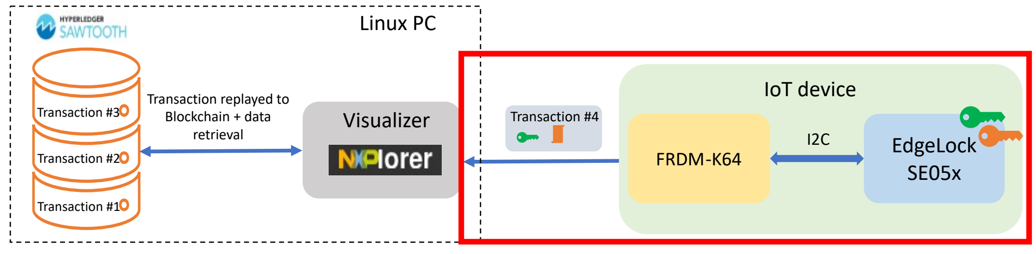


Figure 21. Transactions are sent from FRDM-K64F to the blockchain.

Once the transaction is received by the blockchain, the key defined in Section 3.3.2 will be incremented by one. Follow these steps:

1. Make sure the FRDM-K64F board is connected to the Linux PC using the USB cable and wait some seconds for the demo application to start running;
2. Click the SW3 of the FRDM-K64F to generate and send a new transaction;

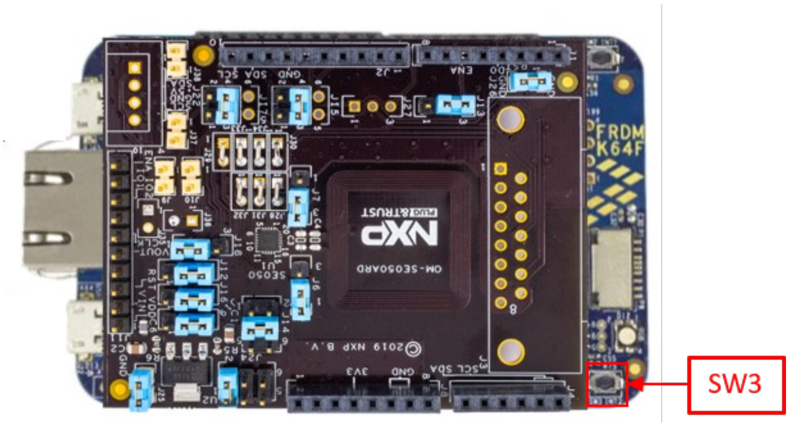


Figure 22. SW3 button in the FRDM-K64F.

3. In the Sawtooth terminal, you should be able to see that a new transaction has been received in the blockchain as shown in Figure 23

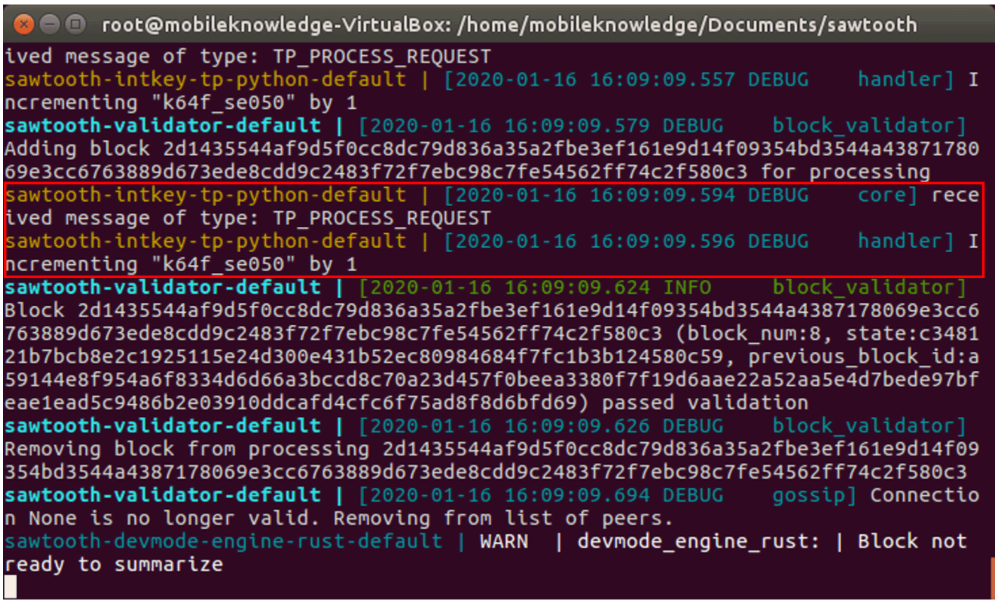


Figure 23. The transaction is received by the blockchain.

4. In addition, you can see that the value shown for the `k64f_se050` key in NXPlorer has been incremented by one as shown in Figure 24

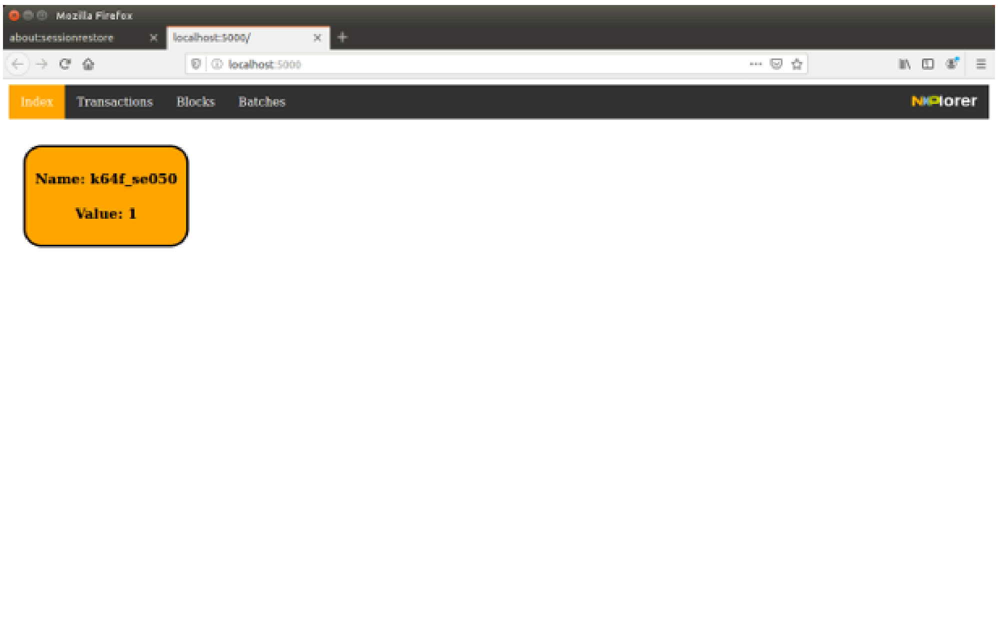


Figure 24. The key value has increased by one.

5. You can check the transaction information by (1) going to the Batches tab and then (2) clicking on the "Check transaction info" link as shown in Figure 25:

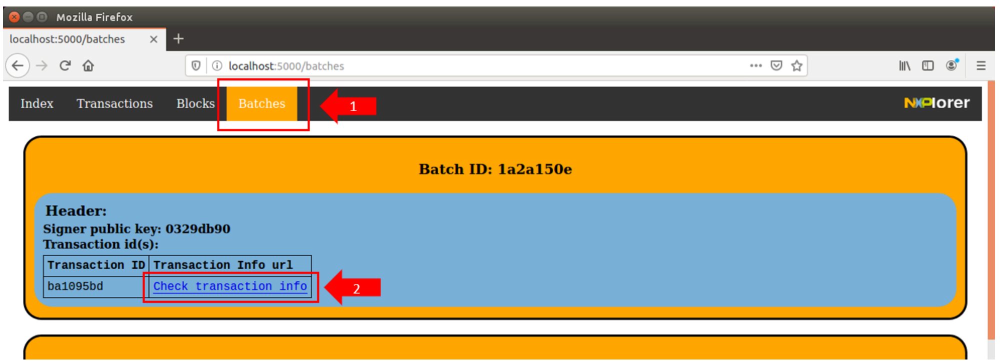


Figure 25. Check the information of the transaction.

6. You will see the transaction information in JSON format, including the public key of the signer as shown in Figure 26. The public key should coincide with the one that was previously logged in Figure 9.

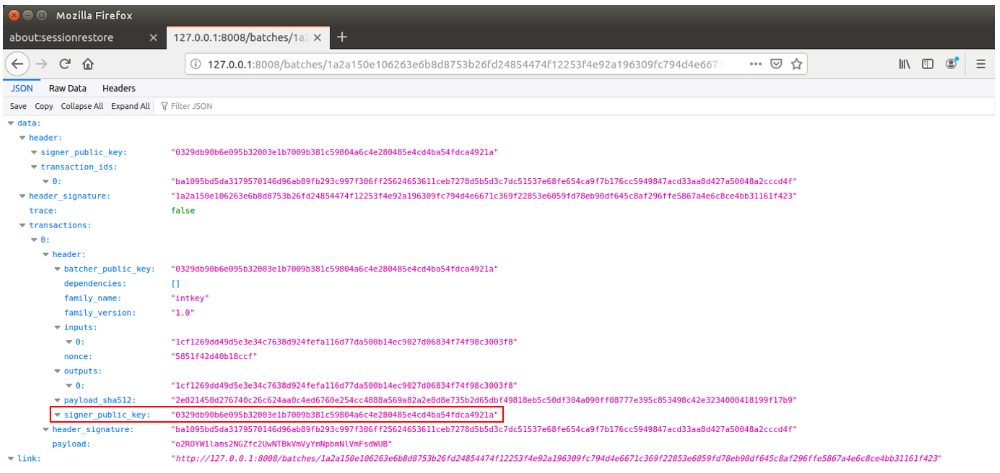


Figure 26. Transaction details.

4 Appendix: Installation of software components

This appendix provides instructions to download and install all the necessary software components to run the blockchain demo.

4.1 Hyperledger® Sawtooth

Hyperledger® Sawtooth is an enterprise solution for building, deploying and running blockchains. It provides an extremely modular and flexible platform for implementing transaction-based updates to shared ledgers between untrusted parties coordinated by consensus algorithms.

Start by downloading the Docker Compose file for the Sawtooth environment. This file specifies the process and configurations to build and run a simple Sawtooth environment. Follow these steps:

1. Go to https://sawtooth.hyperledger.org/docs/core/releases/1.1/app_developers_guide/docker.html and download the `sawtooth-default.yaml` file as shown in [Figure 27](#).

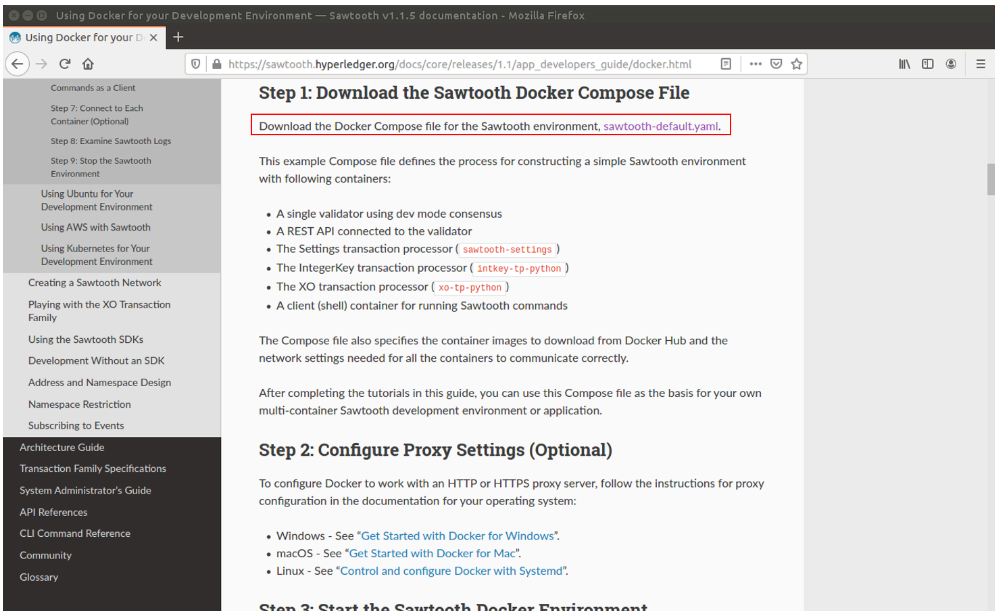


Figure 27. Download Docker Compose file.

2. Create a folder named Sawtooth under Documents and move the `sawtooth-default.yaml` file there.

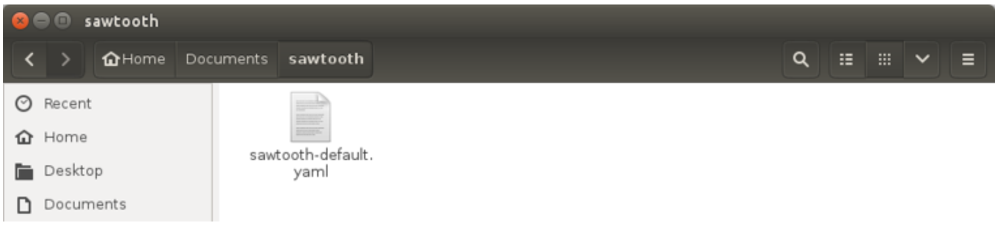


Figure 28. Save the yaml file in a new folder for future use.

After downloading the file, you need to download and install Docker Community Edition and Docker Compose in your Ubuntu 16.04 machine.

4.1.1 Docker Engine

Docker Engine is an open source technology for building and containerizing your applications. It will act as a client-server application with Sawtooth and integrate its basic functionalities with the IntegerKey transaction processors and Sawtooth commands.

Go to https://docs.docker.com/engine/install/ubuntu/ and follow the steps under "*Installation Methods*" to install the Docker Engine in your machine.

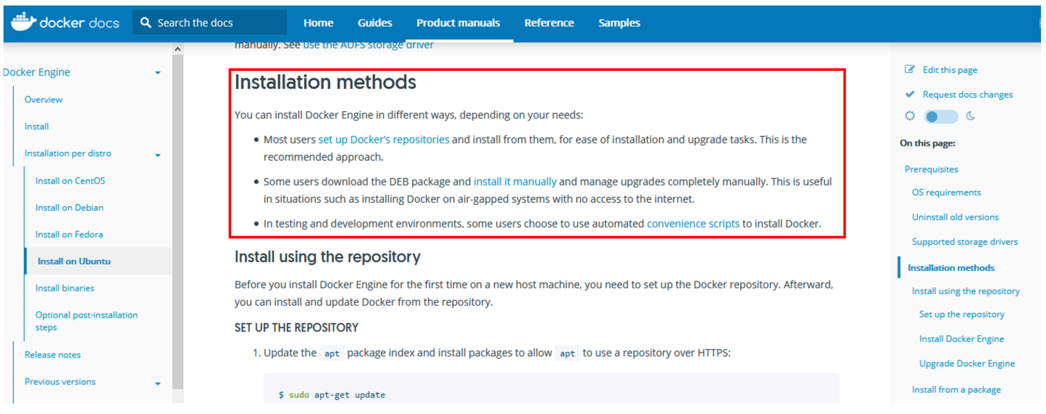


Figure 29. Docker page with detailed instruction for the installation.

4.1.2 Docker Compose

Docker Compose is a tool for defining and running multi-container Docker applications. You will use the Compose file you downloaded in Section 4.1 to configure and run the Sawtooth application.

Go to https://docs.docker.com/compose/install/ and choose Linux in the "*Install Compose*" section to install Docker Compose.

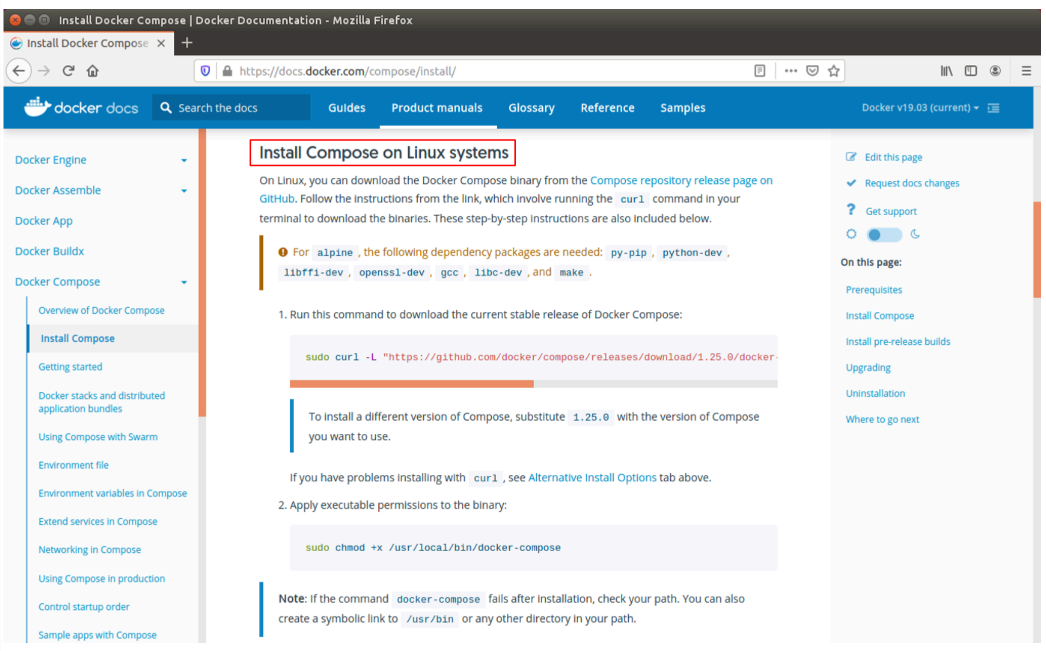


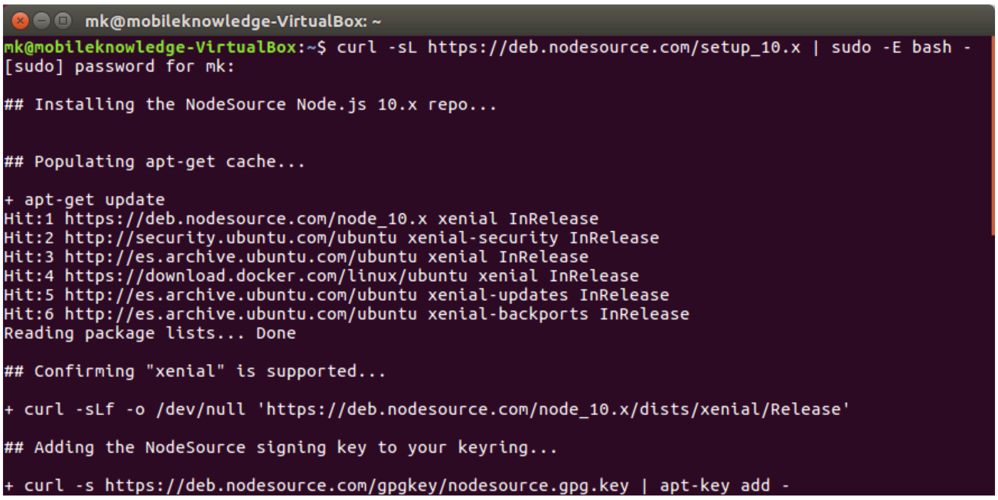
Figure 30. Docker page with detailed instruction for the installation.

4.2 NXPlorer

NXPlorer is a software tool used to show ledgers and updates for blockchains like Hyperledger® Fabric and Hyperledger® Sawtooth. It visualizes the data provided by APIs of different blockchains, turning JSON-encoded objects into human-readable objects.

This tool runs on Node.js and Node package manager (Npm). Follow these steps to download and install NXPlorer:

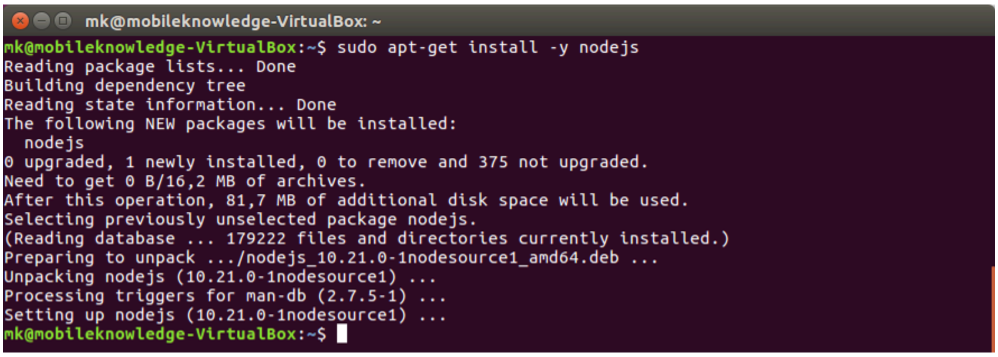
1. NXPlorer git repositoryClone the in a folder of your choice.
2. You can go to https://github.com/nodesource/distributions/blob/master/README.md for detailed instructions on how to install Node.js v10.x. Open a terminal window and type the following command to download the Node.js repository as shown in Figure 31:
   `curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash -`



```
mk@mobileknowledge-VirtualBox: ~
mk@mobileknowledge-VirtualBox:~$ curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash -
[sudo] password for mk:

## Installing the NodeSource Node.js 10.x repo...


## Populating apt-get cache...

+ apt-get update
Hit:1 https://deb.nodesource.com/node_10.x xenial InRelease
Hit:2 http://security.ubuntu.com/ubuntu xenial-security InRelease
Hit:3 http://es.archive.ubuntu.com/ubuntu xenial InRelease
Hit:4 https://download.docker.com/linux/ubuntu xenial InRelease
Hit:5 http://es.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:6 http://es.archive.ubuntu.com/ubuntu xenial-backports InRelease
Reading package lists... Done

## Confirming "xenial" is supported...

+ curl -sLf -o /dev/null 'https://deb.nodesource.com/node_10.x/dists/xenial/Release'

## Adding the NodeSource signing key to your keyring...

+ curl -s https://deb.nodesource.com/gpgkey/nodesource.gpg.key | apt-key add -
```

Figure 31. Dowload Node.js

3. Once the repository has been downloaded, run the following command to proceed with the Node.js installation as shown in Figure 32:
   `sudo apt-get install -y nodejs`



```
mk@mobileknowledge-VirtualBox: ~
mk@mobileknowledge-VirtualBox:~$ sudo apt-get install -y nodejs
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  nodejs
0 upgraded, 1 newly installed, 0 to remove and 375 not upgraded.
Need to get 0 B/16,2 MB of archives.
After this operation, 81,7 MB of additional disk space will be used.
Selecting previously unselected package nodejs.
(Reading database ... 179222 files and directories currently installed.)
Preparing to unpack .../nodejs_10.21.0-1nodesource1_amd64.deb ...
Unpacking nodejs (10.21.0-1nodesource1) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up nodejs (10.21.0-1nodesource1) ...
mk@mobileknowledge-VirtualBox:~$
```

Figure 32. Install Node.js

4. Move to the NXPlorer folder you created in step 1 and install the tool using Npm package manager. Run the following command in the terminal window as shown in Figure 33:
   `sudo npm install`

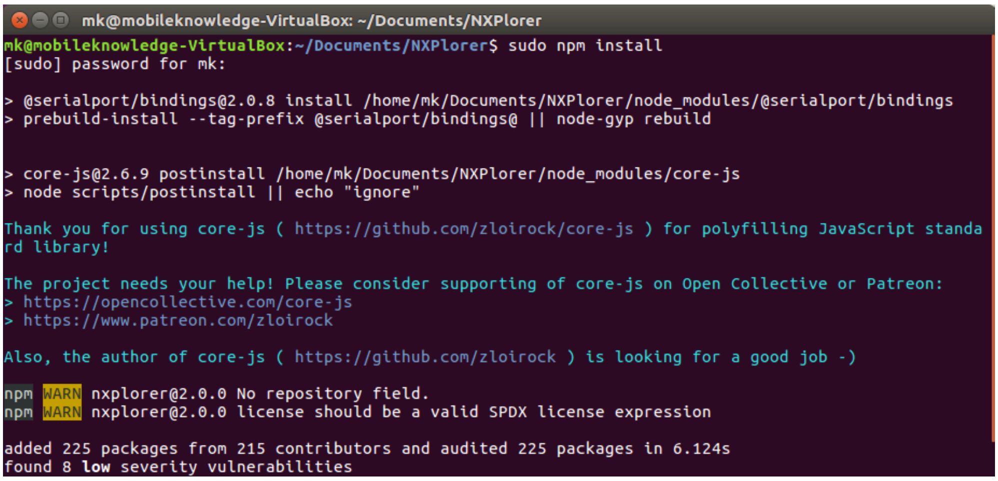


Figure 33. Install NXPlorer tool

5 Legal information

5.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

5.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

5.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Tables

Figures

Contents

Date of release: 22 August 2022