DEUTSCHE
BUNDESBANK
EUROSYSTEM

Trigger Solution

Onboarding Guide - Technical Annex

Last updated: ~~6 June~~3 September 2024

Versions

Version	Chapter	Change information	Date	Name and departmental ID number
1.0	All	Created	10/11/23	Deutsche Bundesbank
1.1	All	Publication	13/12/23	Deutsche Bundesbank
1.2	6.1, 6.3, 6.4, 6.6	Further clarifications on the digital signature, the certificate issuance and signing process, new chapter on certificate and user revocation	04/04/24	Deutsche Bundesbank
1.3	4.2.4, 6.2, 6.3.1	Information about the „User-Agent"; further clarification on OID info in the CSR; Additional an example of an openssl command to generate a CSR	06/06/24	Deutsche Bundesbank
1.4	4.1, 5.2.6	Hint for API users added, Updated to correct channel name	03/09/2024	Deutsche Bundesbank

Table of contents

Seite

1 Overview

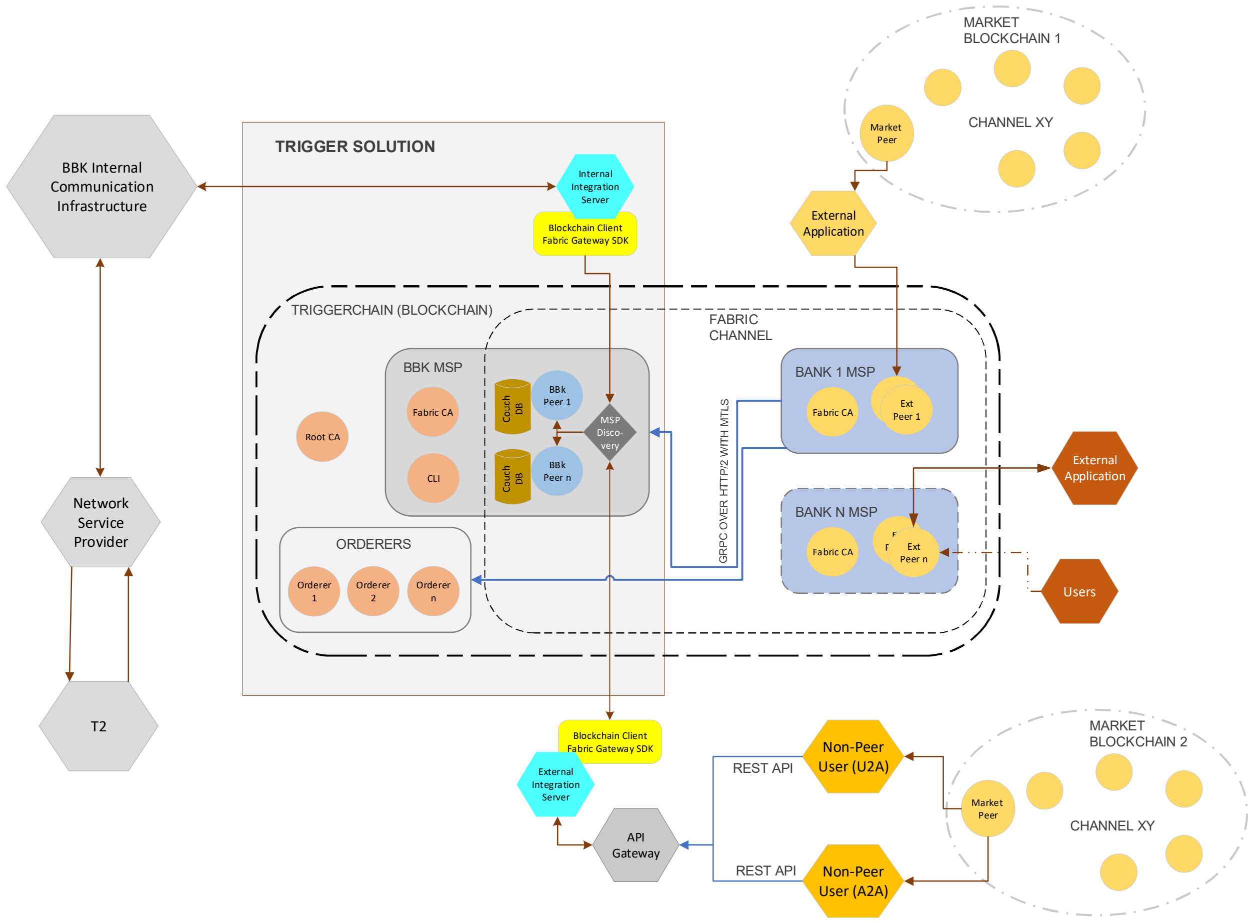
The Trigger Solution is a blockchain-based solution that allows participants to settle Eligible DVP transactions and Eligible Payments in central bank money. The system is based on Hyperledger Fabric 2.5 and is able to take input from numerous Eligible Market DLT Platforms. When an Eligible DVP transaction or an Eligible Payment is confirmed on an Eligible Market DLT Platform, the participant in the Trigger Solution initiates a payment instruction which leads to a payment in the RTGS component of T2.

1.1 Hyperledger Fabric

Hyperledger Fabric 2.5 is being used in the Trigger Solution due to its advanced features that align with the requirement of keeping payment instructions private between the concerned parties. The utilization of **Hyperledger Fabric's private data collection** capabilities offers a robust framework for achieving this privacy while maintaining the benefits of a distributed ledger system.

2 Architecture

2.1 Architecture Diagram



2.2 Components

- **Hyperledger Fabric Orderers:** The Hyperledger Fabric orderers are responsible for ordering transactions on the Trigger Solution blockchain network. They ensure that transactions are processed in a secure and deterministic manner.

- **Hyperledger Fabric Peers:** The Hyperledger Fabric peers are responsible for maintaining the Trigger Solution blockchain ledger and processing transactions. They also store the cryptographic keys that are used to sign transactions.

- **Root Certificate Authority (CA):** The Root CA is hosted by Bundesbank and is responsible for issuing certificates to the Sub CA of participants of the Trigger Solution.

- **Sub Certificate Authority (Sub CA):** This Sub CA will be used to issue certificates for its own peers to authenticate the nodes on the network and to encrypt communications between the nodes (i.e. TLS communication certificates). It will also be used to issue

certificates to its users (i.e. authorisation certificates). These certificates are used to authorize them to perform certain actions on the network. This Sub CA must be owned and hosted by every peer participant in the network.
In case of API users, the Deutsche Bundesbank owned Sub CA will be used to issue users certificates.

- **Trigger Solution Server behind API Gateway:** The Trigger Solution Server is responsible for managing the Trigger Solution network. It handles requests from users and applications, and it coordinates the activities of the other components of the network.

- **Eligible Market DLT Platforms:** The Eligible Market DLT Platforms are the blockchains that are used to track the ownership of assets. They are interacting with the Trigger Solution network via the Interoperability Mechanism to allow for the seamless settlement Eligible DVP transactions and Eligible Payments.

- **External Applications:** External applications can interact with the Trigger Solution network through the API Gateway. This allows external applications to submit requests to the Trigger Solution Server and to receive notifications about events that occur on the network.

3 Integration

There are two primary options available to banks:

API USER

The API approach allows for connection to the Trigger Solution using exposed REST APIs offered by Bundesbank. Through these APIs, participants can build and configure their applications to communicate with the Trigger Solution. This approach offers a simplified integration process, making it accessible for institutions with less blockchain experience.

PEER NODE

The peer node approach involves hosting own Hyperledger Fabric nodes. This entails setting up and configuring dedicated nodes within the Trigger Solution network. Participants gain full control over their nodes, enabling direct interaction with the underlying blockchain network. While this approach demands a certain level of blockchain expertise, it offers the advantages of customization and enhanced security. In addition, the deployment of own smart contracts in the Trigger Solution operated by Deutsche Bundesbank is possible to cater for complex transactions requiring for example the coordination of input from several Eligible Market DLT platforms. However, the Deutsche Bundesbank node will not execute these smart contracts.

Both approaches cater to diverse technical capacities, offering flexibility to choose the option that best aligns with a participant’s resources and requirements.

4 API USER

4.1 Technical Requirements

In order to use the API the registration for Bundesbank ExtraNet for receiving the necessary authentication information (User ID and password) is necessary (please see Trigger Solution Onboarding Guide chapter 2.3). Via API an A2A- and/or U2A-Access to the Trigger Solution is offered.

Proper security measures should be implemented to ensure the confidentiality and integrity of data exchanged with the APIs.

For testing (in advance of Experiments and Trials) as well as for Experiments the Trigger Solution test environment will be used, while Trials will be conducted in the Trigger Solution Production environment.

The URLs for U2A access are as follows:

Environment	URL
Tests (for Experiments and Trials)	https://experiment-triggersolution.bundesbank.de
Experiments	https://experiment-triggersolution.bundesbank.de
Trials	https://triggersolution.bundesbank.de

The API endpoints for A2A access are as follows:

Environment	URL
Tests (for Experiments and Trials)	https://api-a.bundesbank.de/experiment
Experiments	https://api-a.bundesbank.de/experiment
Trials	https://api.bundesbank.de/triggersolution

The API endpoints are strictly https, hence it can only be reached on port 443.

Component	Minimum System Requirements
Application Server	CPU: Dual-core or higher
	RAM: 4GB or more
	Storage: 20GB or more (SSD recommended)
Programming	Programming language runtime environment (e.g., Python, Java, Node.js)
	API client libraries or frameworks
Network	Stable internet connection with low latency (100 Mbps or higher recommended)
Security Measures	Secure coding practices
	Encryption mechanisms (HTTPS, data encryption)

	Monitoring and logging tools for application performance and troubleshooting
API Specifications	Understanding of OpenAPI 3.0 as a consumer

Note: API users~~,~~ should always get provided access to U2A and A2A together. Shall IT departments provide access to U2A URL, it must be ensured that A2A URL access is also provided.

4.2 Onboarding Steps

4.2.1 Step 1: API Documentation Review

Begin by thoroughly reviewing the API documentation provided by Deutsche Bundesbank as Solution Provider in the restricted service area “Trigger Solution” on the Deutsche Bundesbank website (see chapter 2.1 Trigger Solution Onboarding Guide) . This is a JSON file and is written as per OpenAPI 3.0 specifications, hence it can be imported easily to any REST API tools like Postman etc.

Understand the available endpoints, request and response formats, authentication mechanisms (API keys, Basic Auth, etc.), rate limits, and any specific guidelines for API usage. This step is crucial for gaining a comprehensive understanding of how to interact with the Trigger Solution through the exposed APIs.

4.2.2 Step 2: Firewall Setup / Opening

The API gateway of Deutsche Bundesbank is restricted to allowed users only. Hence it is required to get in touch with Deutsche Bundesbank onboarding team and supply them with your IP or IP range which must be whitelisted before you can start using the endpoints. Please send the details to triggersolution@bundesbank.de after having submitted the initial Trigger Solution Registration Form.

4.2.3 Step 3: Application Development (only relevant if an A2A-connection shall be established)

Develop the application that will interact with the exposed APIs. Choose a suitable programming language (Python, Java, Node.js, etc.) based on your team's expertise and application requirements. Utilize programming frameworks and libraries to simplify API integration and streamline the development process. Set up the necessary environment variables, configuration files and follow other best practices for application development.

4.2.4 Step 4: API Integration (only relevant if an A2A-connection shall be established)

Integrate the API endpoints into your application's codebase. Implement functions or methods to send HTTP requests (GET, POST, PUT, DELETE) to the Deutsche Bundesbank APIs, passing the required parameters and headers as specified in the documentation. Utilize API client libraries or packages to handle authentication, and error handling. Ensure that the application can successfully communicate with the APIs and retrieve the expected responses.

May some of your applications or your HTTP Libraries use the HTTP Header key “User-Agent”, we strongly recommend that you use your own custom value for this header. Some default HTTP Libraries values for User-Agent header may be classified as Untruthful and API requests might be therefore blocked by security mechanism.
Optionally, you can request Triggersolution team to whitelist your User-Agent configuration if necessary.

4.2.5 Step 5: Digital signature process

After the previous steps have been completed, the participants need to get a digital certificate (authorisation certificate) from the Sub CA of Deutsche Bundesbank in order to be able to approve payment instructions. This will be a public/private keypair, which can be generated as per steps in chapter 6.

5 PEER NODE

5.1 Technical Requirements

The peer node approach involves hosting own Hyperledger Fabric nodes. This entails setting up and configuring dedicated nodes within the Trigger Solution network. Participants gain full control over their nodes, enabling direct interaction with the underlying blockchain network.

Deploy a minimum of three servers (physical or virtual) for a basic setup: one for the endorsing peer, one for couchdb and the last one for the Hyperledger Fabric (or any other) CA (Certificate Authority).

Component	Minimum System Requirements
Peer	CPU: Dual-core or higher
	RAM: 2GB or more
	Storage: 50GB or more (SSD recommended)
CouchDB	CPU: Dual-core or higher
	RAM: 4GB or more
	Storage: 20GB or more (SSD recommended)
Fabric CA	CPU: Single-core or higher
	RAM: 1GB or more
	Storage: 5GB or more (SSD recommended)
Network	Stable internet connection with low latency (100 Mbps or higher recommended)
	Network security configurations (firewalls, security groups)
Docker	Docker Engine installed
	Docker Compose for orchestration if no K8S (optional)
	Kubernetes for scalability (optional & recommended)

5.2 Onboarding Steps

This section is an overview of steps to be performed for the onboarding of a participant to the network.

5.2.1 Step 1: System preparation

This step involves setting up the necessary infrastructure to host Hyperledger Fabric nodes.

- Selection of suitable hardware, such as servers or cloud instances, with adequate computational power, memory, and storage.
- Install the required operating system (e.g., Ubuntu, CentOS) and Docker to create containers for deploying nodes.
- Configure network settings, including static IP addresses and domain names, to ensure reliable communication within the network.

5.2.2 Step 2: Node Deployment and Configuration

- Deploy and configure the Hyperledger Fabric nodes according to your network's specifications.
- Utilize tools like Docker Compose / Kubernetes to define and manage the containers that make up the network.
- Set up the committing peer node, a Hyperledger Fabric CA node. A CLI can also be setup optionally to interact with peer node.
- Configure the cryptographic materials, including TLS certificates and cryptographic keys, to ensure secure communication between nodes. Refer Step 4 below.
- Establish connections between nodes to create a functional Hyperledger Fabric network. Verify that the nodes are up and running, and that they can communicate effectively within the network.

5.2.3 Step 3: Connectivity Establishment

Participants must also ensure to open the connectivity and share their intended IP address(es) and PORT(s) so that is it reachable by other peers on the network. This is crucial for private data dissipation through gossip protocol. Please send the details to triggersolution@bundesbank.de after having submitted the initial Trigger Solution Registration Form.

All peer participants must establish the connectivity to the following after they provided IP address(es) and PORT(s) are unlocked in the firewall by Deutsche Bundesbank:

For testing (in advance of Experiments and Trials) as well as for Experiments the Trigger Solution test environment will be used, while Trials will be conducted in the Trigger Solution Production environment.

TEST ENV:

peer1.experiment.triggersolution.bundesbank.de on port 443
peer2.experiment.triggersolution.bundesbank.de on port 443
orderer1.experiment.triggersolution.bundesbank.de on port 443
orderer2.experiment.triggersolution.bundesbank.de on port 443
orderer3.experiment.triggersolution.bundesbank.de on port 443

PROD ENV:

peer1.triggersolution.bundesbank.de on port 443
peer1.triggersolution.bundesbank.de on port 443
orderer1.triggersolution.bundesbank.de on port 443

orderer2.triggersolution.bundesbank.de on port 443
orderer3.triggersolution.bundesbank.de on port 443

5.2.4 Step 3: Setup a Sub CA

Participants need to start with first deploying a Sub Certificate Authority. This Sub CA will be used to issue certificates for its own peers (i.e. TLS communication certificates) and users (i.e. authorisation certificates). This Sub CA must be signed by the Root CA of Bundesbank Trigger Solution. The following steps are needed to achieve this:

- Start by creating a private key and CSR (certificate signing request) for the sub CA. Read the official documentation for more information and commands: https://hyperledger-fabric.readthedocs.io/en/release-2.5/deployment_guide_overview.html#step-three-set-up-your-cas

- Share the CSR with Deutsche Bundesbank Administrators via mail to triggersolution@bundesbank.de and get the signed certificate (public key) as a response.

- Start the CA node with the public and private keypairs you have from above step. More info here: https://hyperledger-fabric-ca.readthedocs.io/en/latest/deployguide/cadeploy.html#optional-deploy-an-intermediate-ca

- Next, MSP certificate and identities need to be created. More info here: https://hyperledger-fabric.readthedocs.io/en/release-2.5/deployment_guide_overview.html#step-four-use-the-ca-to-create-identities-and-msps

5.2.5 Step 4: MSP Definition in Channel Configuration

Once you have gathered all the certificates and MSPs you need, you're almost ready to create a node. Before any node can be deployed, its configuration file must be customized. For the peer, this file is called core.yaml. You have three main options for tuning your configuration.

- Edit the YAML file bundled with the binaries.

- Use environment variable overrides when deploying. (recommended)

- Specify flags on CLI commands.

More details on above process can be found here in the official documentation: https://hyperledger-fabric.readthedocs.io/en/release-2.5/deployment_guide_overview.html - creating-a-peertarted, we need to generate MSP definition JSON or YAML file which would contain all the public information about the peer, such as public keys, MSPID's and endpoints etc. To generate this file, refer to: https://hyperledger-fabric.readthedocs.io/en/release-2.5/channel_update_tutorial.html#generate-the-org3-crypto-material

This MSP definition file must be shared with the Deutsche Bundesbank administrators (trigger-solution@bundesbank.de), who will then take over the subsequent step. The Deutsche Bundesbank administrators will incorporate the provided cryptographic materials and certificates into the channel configuration using the configtxlator tool from the Hyperledger Fabric suite. This inclusion ensures that each node's identity and access privileges are accurately defined within the channel, thus enhancing the overall security and integrity of the network. Hence, while participants initiate the cryptographic setup, it is the Deutsche Bundesbank administrators who seamlessly integrate these materials into the channel configuration.

This step requires coordination between the node administrator of the participants and the Trigger Solution team of Deutsche Bundesbank.

5.2.6 Step 5: Network Interaction

After Deutsche Bundesbank's administrators confirm the channel update was successful, and the participant's MSPID was added to the channel, participants need to interact with the Trigger Solution network, for which you can proceed to try joining the "~~triggersolution~~triggerchannel" channel. Refer to: https://hyperledger-fabric.readthedocs.io/en/release-2.5/channel_update_tutorial.html#generate-the-org3-crypto-material

5.2.7 Step 6: Chaincode provision

The chaincode package will be provided by Bundesbank Trigger Solution team via a restricted service area on the Deutsche Bundesbank website (see Trigger Solution Onboarding Guide chapter 2.1). It can be enhanced by the participant to the extent that it does not change the read-write sets of any function. Test the chaincode logic by invoking transactions and checking the endorsement results. This stage requires careful coding, validation, and thorough testing to ensure the chaincode function as intended.

For chaincode installation, refer to: https://hyperledger-fabric.readthedocs.io/en/release-2.5/cc_service.html#running-chaincode-as-an-external-service

5.2.8 Step 7: Security and Endorsement Policies

Security is paramount in blockchain networks. Configure the security settings for nodes by enabling mutual Transport Layer Security (mTLS) encryption. Ensure that only authorized entities can access and communicate with the nodes. Define endorsement policies that specify the minimum number of nodes required to approve a transaction. This adds an additional layer of security and prevents fraudulent or unauthorized transactions. Continuously monitor and update security settings to address potential vulnerabilities and maintain a secure network environment.

5.2.9 Step 8: Digital signature process

After the previous steps have been completed, the participants need to get a digital certificate (authorisation certificate) from the Sub CA in order to be able to approve payment instructions. This will be a public/private keypair, which can be generated as per steps in chapter 6.

6 Digital Signature Process

6.1 Overview

Out of various functions which the Trigger Solution has, there is an Approve function which basically translates to its literal meaning, the approval of the payment transaction.

This is a formal go-ahead by the payer bank and must be supplied with a valid digital signature regardless of the chosen connectivity option of the participant.

The payer bank can also empower a third party to perform this step, which then is referred to as Third Party Authorisation. The third party has to be the creator of the payment instruction, a minimal condition to access the payment instruction. The third party does not get access to the payer bank keys, and therefore can only use its own certificates and keys to generate the signatures.

6.2 Four-eyes principle

The Trigger Solution has implemented a functionality to approve (i.e. digitally sign) payment instructions in four-eyes mode. Each user's certificate must contain information about the privilege level of that user to sign the payment instructions (either two-eyes or four-eyes mode).

During the signing process, the chaincode will check the privilege of a user via a specific attribute string **attrs** under the certificate OID (Object Identifier) 1.2.3.4.5.6.7.8.1.

- If the value of string called "privilege" inside attrs json array in OID 1.2.3.4.5.6.7.8.1 for that user is set to 4E (being the approval in 4-eyes mode), the signature will be recorded and the payment instruction will now need to be signed by another user to confirm the approval.
- If the value of string called "privilege" inside attrs json array in OID 1.2.3.4.5.6.7.8.1 for that user is set to 2E (being the approval in 2-eyes mode), the signature will be recorded and the payment instruction will be approved, without requiring further approvals.

In case of API users, the value of this OID will be set automatically by Trigger Solution backend based on the ExtraNet registration (see Trigger Solution Onboarding Guide, chapter 2.3). Therefore, setting the OID 1.2.4.5.6.7.8.1 is not mandatory for API users. More details can be found below in the Issuance section.

While PEER users need to ensure the values for these OID are set when they issue user certificates from their own Sub CA.

6.3 Issuance

This sections describes two approaches to the certificate issuance step.
API USERS have the option to get issued a signed certificate upon a successful Certificate Signing Request (CSR) via general use case approach as described below in chapter 6.3.1, or via the GUI (described in U2A Use Case below in chapter 6.3.2) in order to get assistance to:

- Generate a private key
- Create the CSR request (from the generated or an own private key)
- Receive the Signed Certificate (from the generated CSR or a own CSR)

6.3.1 General Use Case

As a part of onboarding, the users will create a CSR (certificate signing request) with their real name or identity in CN of the certificate. The private key must be strictly kept secured and it shall remain with only them. This must be done by each user individually.

The CSR should be strictly of type ECDSA with elliptical curve as prime256v1 (also known as P-256)

The API USERS can then call the certificate signature API and get their CSR signed by the Deutsche Bundesbank Certificate authority.

All users will be required to supply the signature string and their public key in PEM format in the payload of the API call for approval.

The PEER USERS can get their CSR signed by the Sub CA which they have hosted as a part of node installation.

Illustration of a certificate creation process:

Step 1: Create a CSR using a certificate management tool (e.g. openssl) with at least the following mandatory fields: CN, O, OU, OID 1.2.3.4.5.6.7.8.1 (optional for API users), C, ST & L

Only for peer node users: The custom OID 1.2.3.4.5.6.7.8.1 must include a json key called attrs, under which there should be two sub keys:

- mspid: value set to MSPID/BIC code.
- privilege: value set to 4E for users using the four-eyes mode, OR value set to 2E for users using the two-eyes mode.

For more information on how to do this via fabric CA, read: Attribute-Based-Access-Control. Example below:

--id.attrs 'privilege=4E,mspid=TESTMSPID123'

Command line example for CSR generation:

```
openssl req -new -sha256 -key private_key.pem -out certificate_signing_request.csr
-subj "/C=DE/ST=Hessen/L=Frankfurt/O=TriggerchainMember/OU=NA/CN=exnusername"
```

Note: In some Unix or Windows systems, the subject parameter in the command line might require to escape the first character in order to work properly. You can achieve this by adding a dummy first key/value. For example: `openssl req -new -sha256 -key private_key.pem -out certificate_signing_request.csr -subj "//x=1/C=DE/ST=Hessen/L=Frankfurt/O=TriggerchainMember/OU=NA/CN=exnusername`
Note: The above example does not include OID in the subject, and therefore is corresponding only to an API user.

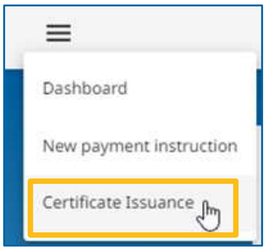
Step 2: For API USERS: Call the signCSR API endpoint with the CSR file as payload. More details on the API usage can be found in the API documentation (json file) provided in the restricted area “Trigger Solution” on the Deutsche Bundesbank website (see Trigger Solution Onboarding Guide chapter 2.1)

For PEER USERS: Use your own Sub CA to get your CSR signed.

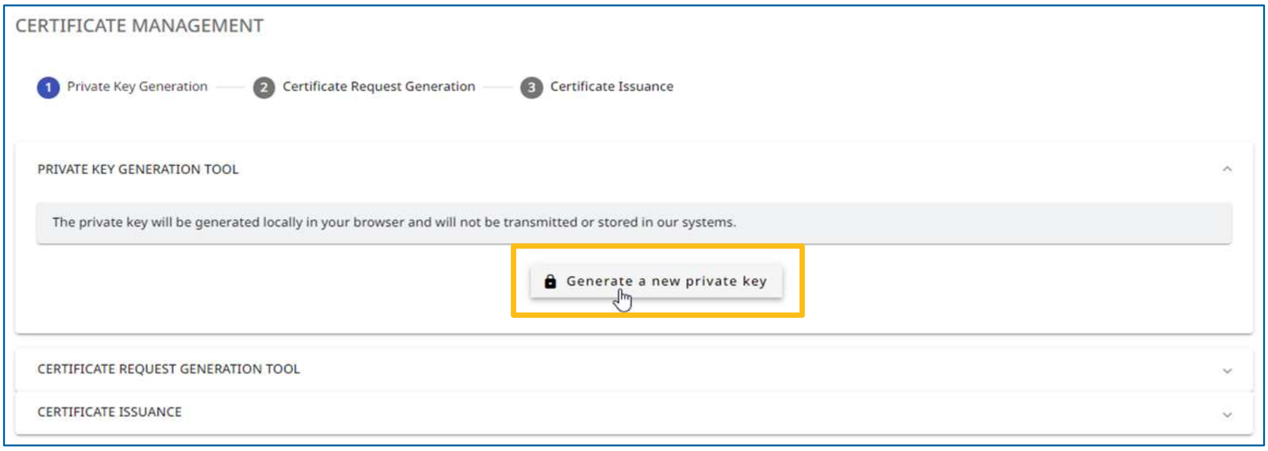
6.3.2 U2A Use Case (API USERS only)

Access and log-in to the frontend application.

1. **Open the Screen “Certificate Issuance”**



2. **Press the button “Generate a new private key**


CERTIFICATE MANAGEMENT

1 Private Key Generation — 2 Certificate Request Generation — 3 Certificate Issuance

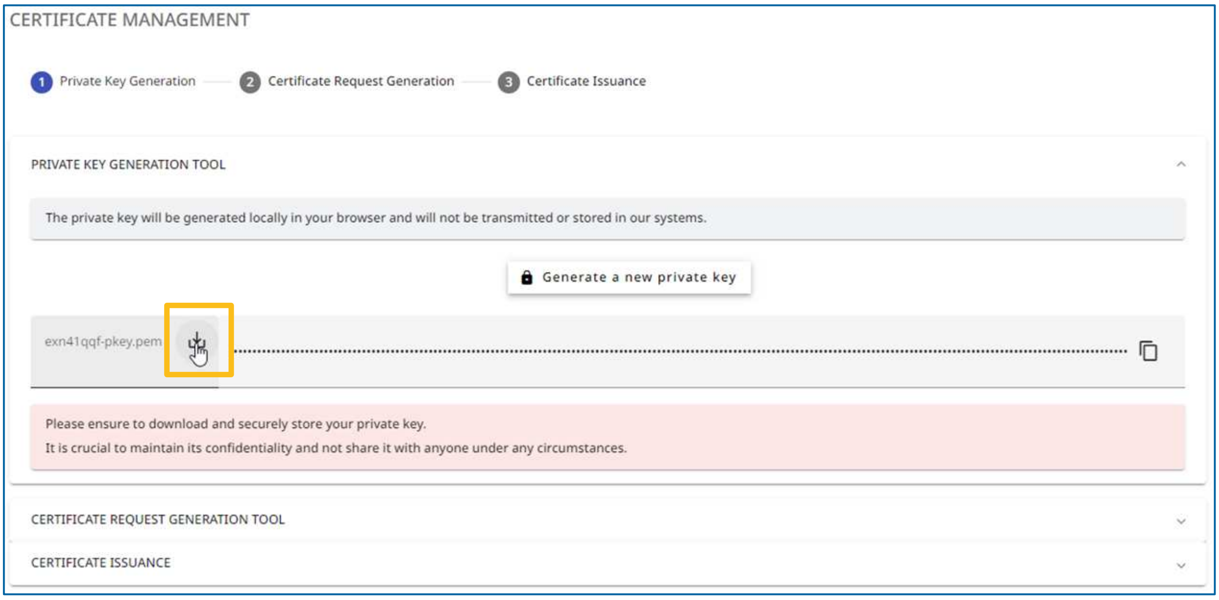
PRIVATE KEY GENERATION TOOL

The private key will be generated locally in your browser and will not be transmitted or stored in our systems.

CERTIFICATE REQUEST GENERATION TOOL

CERTIFICATE ISSUANCE

3. Download and save the generated private key



CERTIFICATE MANAGEMENT

1 Private Key Generation — 2 Certificate Request Generation — 3 Certificate Issuance

PRIVATE KEY GENERATION TOOL

The private key will be generated locally in your browser and will not be transmitted or stored in our systems.

Generate a new private key

exn41qqf-pkey.pem

Please ensure to download and securely store your private key.
It is crucial to maintain its confidentiality and not share it with anyone under any circumstances.

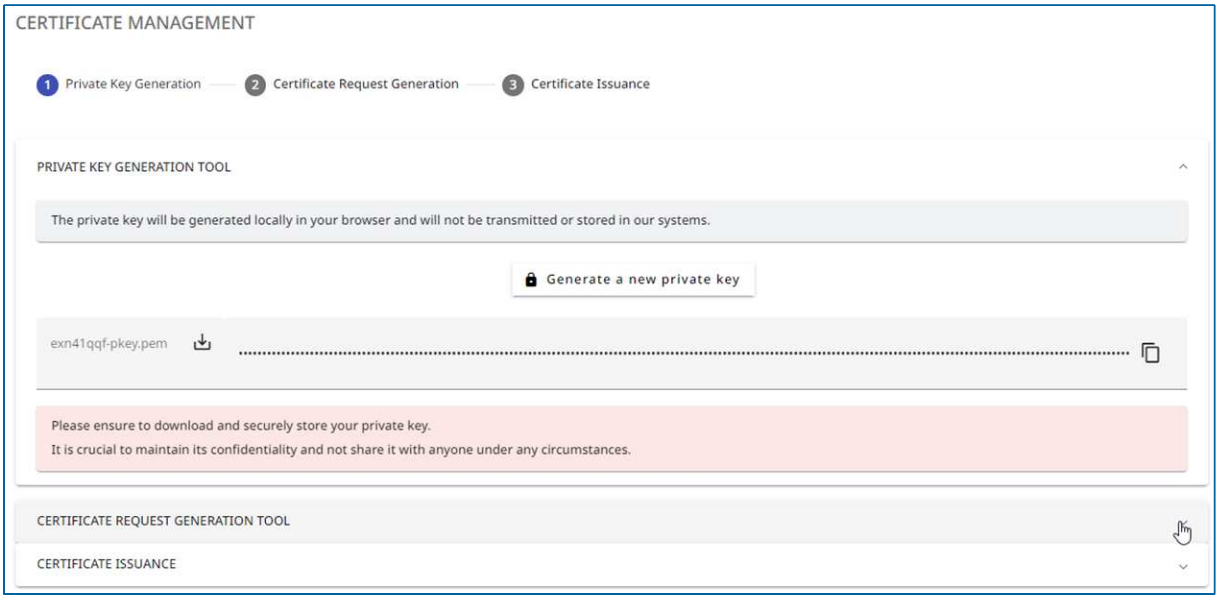
CERTIFICATE REQUEST GENERATION TOOL

CERTIFICATE ISSUANCE

This step is really important!

The downloaded and saved private key is needed for the authorisation process of any payment instruction.

4. Expand the section “Certificate Request Generation Tool”



CERTIFICATE MANAGEMENT

1 Private Key Generation — 2 Certificate Request Generation — 3 Certificate Issuance

PRIVATE KEY GENERATION TOOL

The private key will be generated locally in your browser and will not be transmitted or stored in our systems.

Generate a new private key

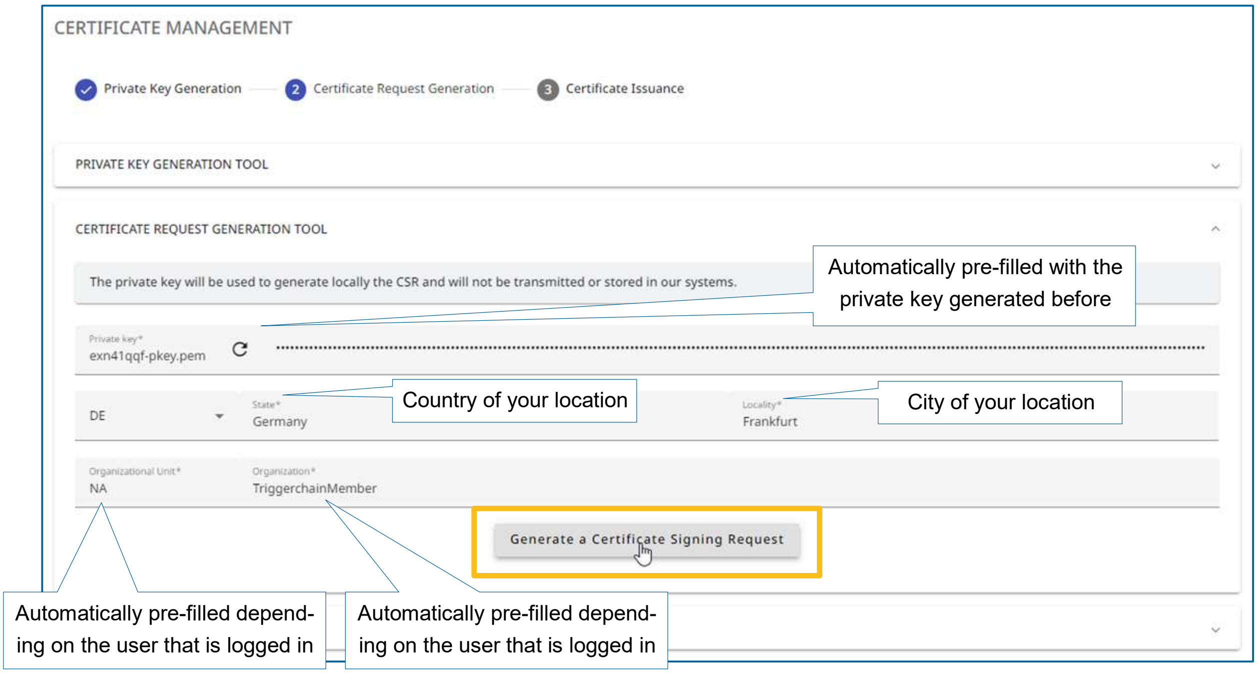
exn41qqf-pkey.pem

Please ensure to download and securely store your private key.
It is crucial to maintain its confidentiality and not share it with anyone under any circumstances.

CERTIFICATE REQUEST GENERATION TOOL

CERTIFICATE ISSUANCE

5. **Enter the mandatory fields and push the button “Generate a Certificate Signing Request”**


CERTIFICATE MANAGEMENT
Private Key Generation
2 Certificate Request Generation
3 Certificate Issuance
PRIVATE KEY GENERATION TOOL
CERTIFICATE REQUEST GENERATION TOOL
Generate a Certificate Signing Request

The private key will be used to generate locally the CSR and will not be transmitted or stored in our systems.

Private key*: exn41qqf-pkey.pem

DE

State*: Germany

Locality*: Frankfurt

Organizational Unit*: NA

Organization*: TriggerchainMember

Automatically pre-filled with the private key generated before

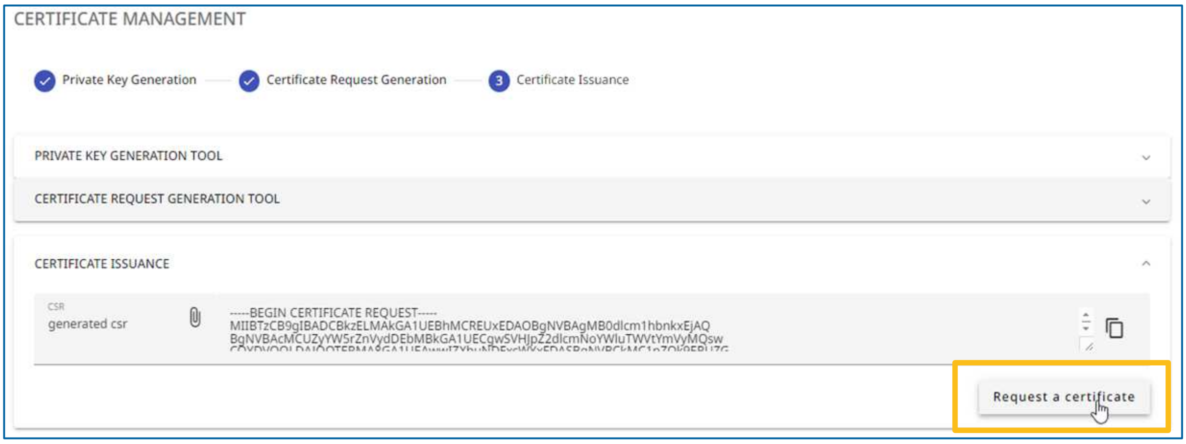
Country of your location

City of your location

Automatically pre-filled depending on the user that is logged in

Automatically pre-filled depending on the user that is logged in

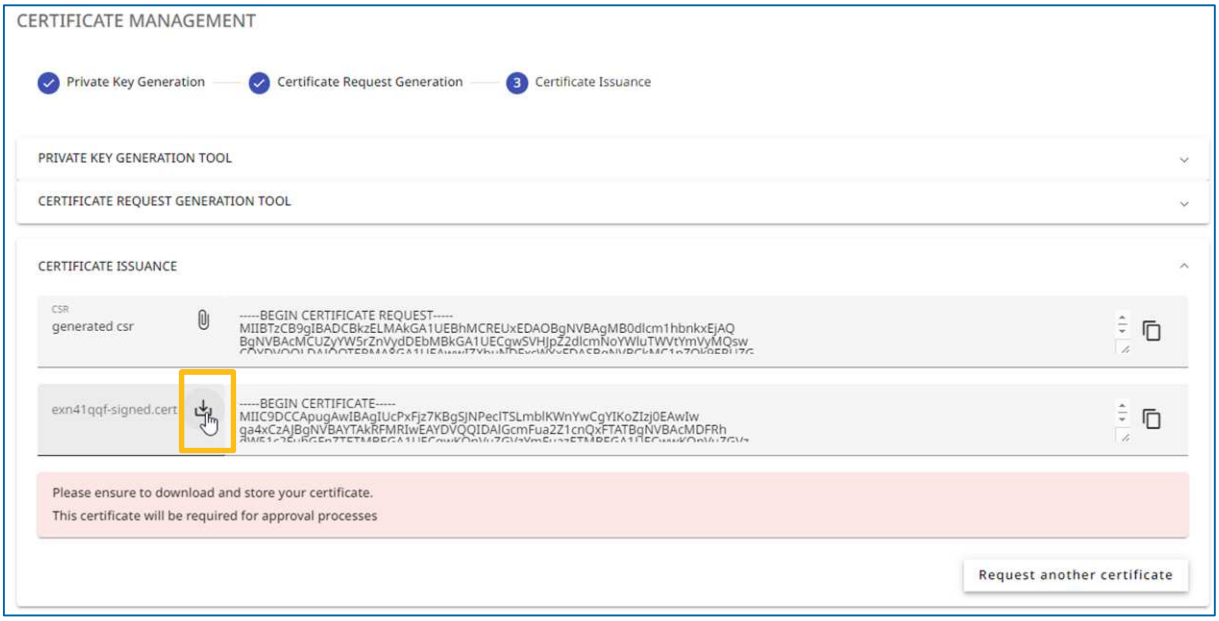
6. **Push the button “Request a certificate”**
After pushing the button to generate a Certificate Signing Request some fields will be filled automatically. The button “Request a certificate” can immediately be pressed.



CSR: generated csr

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBTzCB9gIBADCBkzELMAkGA1UEBhMCREUxEDAOBgNVBAgMB0dlcm1hbnkxEjAQ
BgNVBAcMCUZyYW5rZnVydDEbMBkGA1UECgwSVHJpZ2dlcmNoYWluTWVtYmVyMQsw
```

7. **Download and store the signed certificate**



CERTIFICATE MANAGEMENT

Private Key Generation
Certificate Request Generation
3 Certificate Issuance
PRIVATE KEY GENERATION TOOL
CERTIFICATE REQUEST GENERATION TOOL
CERTIFICATE ISSUANCE

CSR
generated csr

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBTzCB9gIBADCBkzELMAkGA1UEBhMCREUxEDAOBgNVBAgMB0dlcm1hbnkxEjAQ
BgNVBAcMCUZyYW5rZnVydDEbMBkGA1UECgwSVHJpZ2dlcmNoYWluTWVtYmVyMQsw
```

exn41qqf-signed.cert

```
-----BEGIN CERTIFICATE-----
MIIC9DCCApugAwIBAgIUcPxFjz7KBgSJNPecITSLmblKWnYwCgYIKoZIzj0EAwIw
ga4xCzAJBgNVBAYTAkRFMRIwEAYDVQQIDAlGcmFua2Z1cnQxFTATBgNVBAcMDFRh
```

Please ensure to download and store your certificate.
This certificate will be required for approval processes

This step is really important!

The downloaded signed certificate is needed for the authorisation process of any payment instruction.

6.4 Signing

This sections describes two approaches to sign a payment instruction during the payment instruction approval process.
API USERS have the option to generate the signature via general use case approach as described below in chapter 6.4.1, or via the GUI (described in U2A Use Case below in chapter 6.4.2) in order to get assistance to:

- Generate a signature based on Payment Instruction Details (from an own private key)
- Approve the Payment Instruction (with the generated or a self-generated signature)

6.4.1 General Use Case

Users sign a SHA256 sum of a JSON string of ID + AMOUNT + PAYERBANK + RECEIVERBANK + HTLCHASH + HTLCTIMEOUT, generating a unique transaction signature.

Note: The HTLC Hash and HTLC Timeout would remain blank if the payment instruction is non-HTLC.

Signature and public key are included in the payload for operations like payment instruction approval.

Illustration of a signing process:

Step 1: Suppose the payment instructions JSON is:

```
{
    "id": "6daf1139-35a1-4360-bbf7-32aadaef2c92",
    "payerBank": "ZBANKMSP1",
    "receiverBank": "ZBANKMSP2",
    "orderingCustomer": "",
    "beneficiaryCustomer": "",
    "creator": "ZBANKMSP3",
    "correlationId": "AC9I-23hjm23k3pn4b94or3s ",
    "htlcHash": "810ff2fb242a5dee4220f2cb0e6a519891fb67f2f828a6cab4ef8894633b1f50",
    "htlcSecret": "",
    "state": "PREPARE",
    "currency": "EUR",
    "amount": 2345,
    "amountString": "23.45",
    "htlcTimeOut": "2024-02-06T12:45:00.000Z",
    "htlcTimeOutLocale": "2024-02-06 13:45:00",
    "htlcTimeOutTimestamp": "1707223500000",
    "modifiedOn": "2024-02-06T10:32:05.746Z",
    "modificationDateLocale": "2024-02-06 11:32:05",
    "modificationDateTimestamp": "1707215525746",
    "createdAt": "2024-02-06T10:30:15.201Z",
    "creationDateLocale": "2024-02-06 11:30:15",
    "creationDateTimestamp": "1707215415201",
    "sign": [],
    "errorCode": "",
    "errorDescription": ""
}
```

Step 2: Calculate the string to sign. It will be ID + AMOUNT + PAYERBANK + RECEIVERBANK + HTLCHASH + HTLCTIMEOUT.
That becomes as follows in this case:

```
6daf1139-35a1-4360-bbf7-32aadaef2c9223.45ZBANKMSP1ZBANKMSP2810ff2fb242a5dee4220f2cb0e6a519891fb67f2f828a6cab4ef8894633b1f501707223500000
```

Note: The amount should be the string value containing the decimal separator
Note: The htlc timeout should be epoch value of the timeout date (13 characters)

Note: In case of non-HTLC payment this would look like:

```
6daf1139-35a1-4360-bbf7-32aadaef2c9223.45ZBANKMSP1ZBANKMSP2
```

Step 3: Calculate sha256sum of above string.

Command line example:

```
echo -n "6daf1139-35a1-4360-bbf7-32aadaef2c9223.45ZBANKMSP1ZBANK-
MSP2810ff2fb242a5dee4220f2cb0e6a519891fb67f2f828a6cab4ef8894633b1f501707223500000"
| openssl dgst -sha256
```

Outcome in stdin:

```
074890de8318aec02772d0de5ee0e0ea060c3f66a395c70d4a1a2663a3dc49ce
```

Step 4: Sign the above string using private key.

The output of signing is what we will refer to as “signature string” in base64 format.

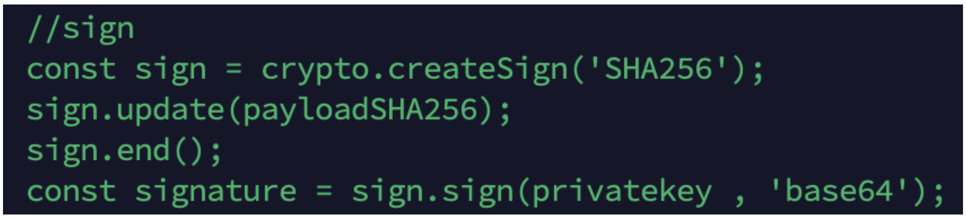
Few signature generation examples from code:

- Using OpenSSL

```
echo -n "074890de8318aec02772d0de5ee0e0ea060c3f66a395c70d4a1a2663a3dc49ce" | openssl
dgst -sha256 -sign private_key.pem | openssl base64 -out signature_string.txt
```

Where private_key.pem is user’s private key and signature string is stored in a file called signature_string.txt

- Using NODEJS:



```
//sign
const sign = crypto.createSign('SHA256');
sign.update(payloadSHA256);
sign.end();
const signature = sign.sign(privatekey , 'base64');
```

Alternative Step 2, 3 and 4:

```
echo -n $(echo -n "6daf1139-35a1-4360-bbf7-32aadaef2c9223.45ZBANKMSP1ZBANK-
MSP2810ff2fb242a5dee4220f2cb0e6a519891fb67f2f828a6cab4ef8894633b1f501707223500000" |
openssl dgst -sha256 | cut -d' ' -f2) | openssl dgst -sha256 -sign private_key.pem |
openssl base64 -A -out signature_string.txt
```

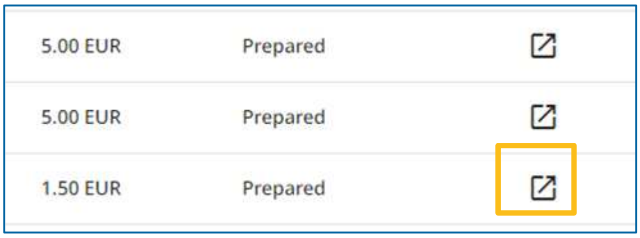
Note: The echo wrapping functions ensure the formatting, the –n option removing the potential trailing newline characters
Note: The cut function is used to remove the “(stdin)= ” prefix to the sha256 converted string
Note: The –A option for the openssl base64 function removes the training newline characters of the generated signature

Step 5: Call the approve API with post request and pass the signature string and public key (user certificate public PEM) in the payload under “sign” section. More details can be found in the shared in the API documentation (json file).

6.4.2 U2A Use Case

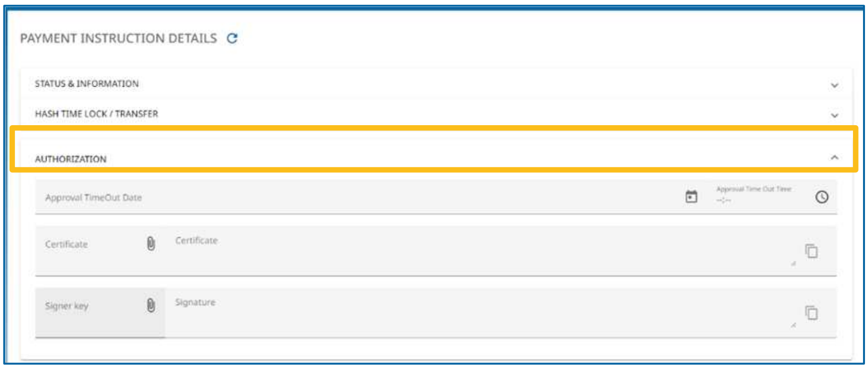
Access and log-in to the frontend application.

1. **Open a payment instruction**

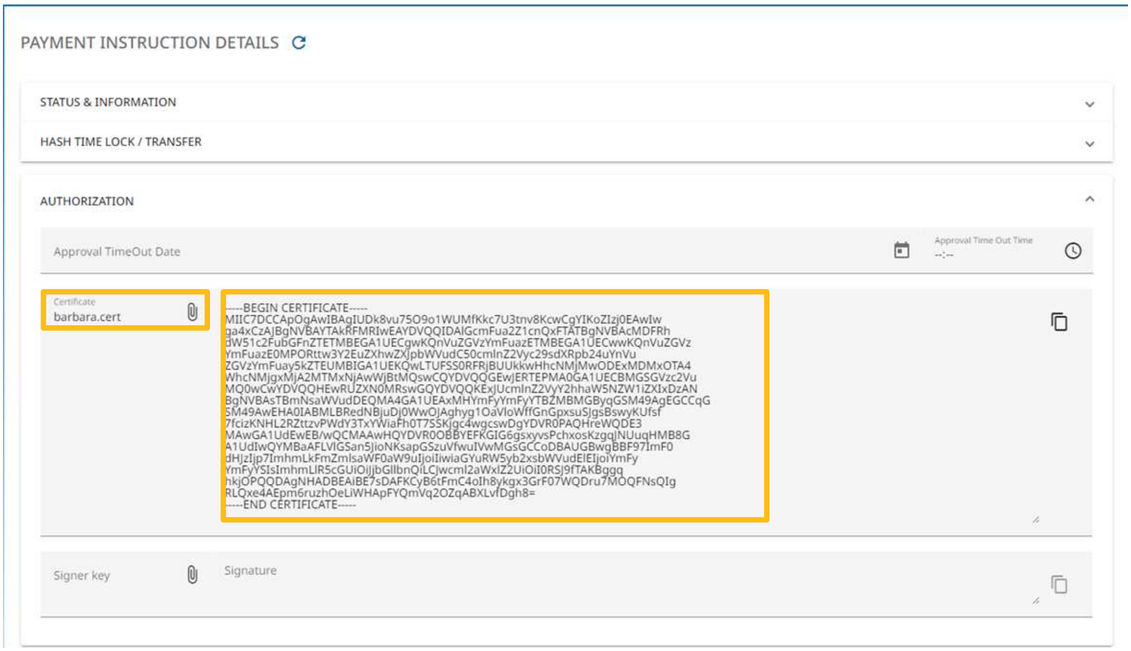


5.00 EUR	Prepared
5.00 EUR	Prepared
1.50 EUR	Prepared

2. **Go to the Authorization section**



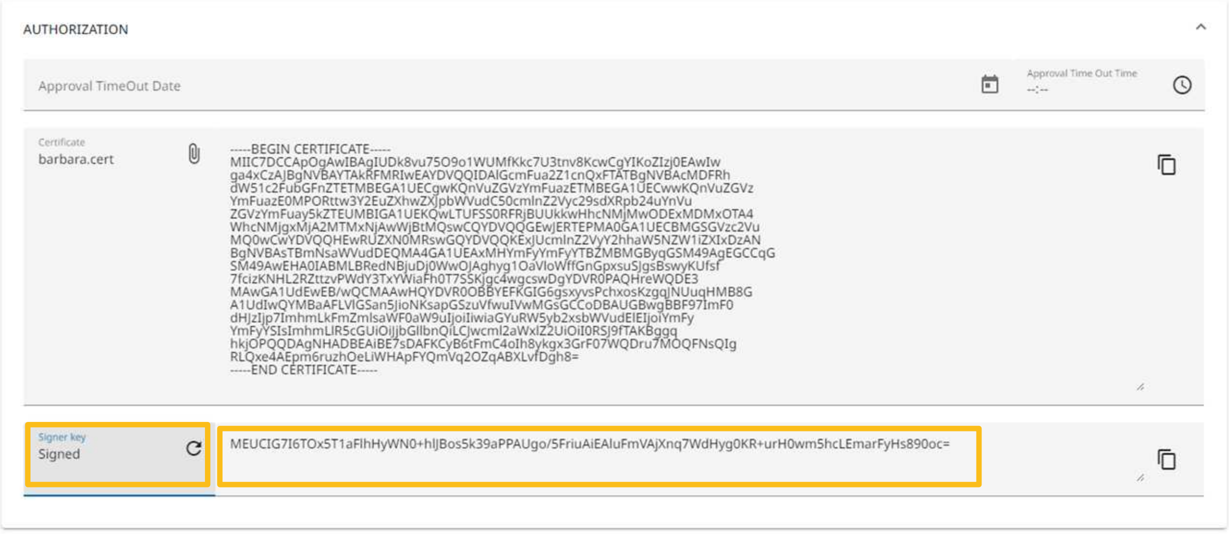
3. **Load or Paste the Certificate in the Certificate Field**



Certificate
barbara.cert

```
-----BEGIN CERTIFICATE-----
MIIC7DCCApOgAwIBAgIUDk8vu75O9o1WUMfKkc7U3tnv8KcwCgYIKoZIzj0EAwIw
ga4xCzAJBgNVBAYTAkRFMRIwEAYDVQQIDAlGcmFua2Z1cnQxFTATBgNVBAcMDFRh
dW51c2FubGFnZTETMBEGA1UECgwKQnVuZGVzYmFuazETMBEGA1UECwwKQnVuZGVz
YmFuazE0MPORttw3Y2EuZXhwZXJpbWVudC50cmlnZ2Vyc29sdXRpb24uYnVu
ZGVzYmFuay5kZTEUMBIGA1UEKQwLTUFSS0RFRjBUUkkwHhcNMjMwODExMDMxOTA4
WhcNMjgxMjA2MTMxNjAwWjBtMQswCQYDVQQGEwJERTEPMA0GA1UECBMGSGVzc2Vu
MQ0wCwYDVQQHEwRUZXN0MRswGQYDVQQKExJUcmlnZ2VyY2hhaW5NZW1iZXIxDzAN
BgNVBAsTBmNsaWVudDEQMA4GA1UEAxMHYmFyYmFyYTBZMBMGByqGSM49AgEGCCqG
SM49AwEHA0IABMLBRedNBjuDj0WwOJAghyg1OaVloWffGnGpxsuSJgsBswyKUfsf
7fcizKNHL2RZttzvPWdY3TxYWiaFh0T7SSKjgc4wgcswDgYDVR0PAQHreWQDE3
MAwGA1UdEwEB/wQCMAAwHQYDVR0OBBYEFKGIG6gsxyvsPchxosKzgqJNUuqHMB8G
A1UdIwQYMBaAFLVlGSan5JioNKsapGSzuVfwuIVwMGsGCCoDBAUGBwgBBF97ImF0
dHJzIjp7ImhmLkFmZmlsaWF0aW9uIjoiIiwiaGYuRW5yb2xsbWVudElEIjoiYmFy
YmFyYSIsImhmLlR5cGUiOiJjbGllbnQiLCJwcml2aWxlZ2UiOiI0RSJ9fTAKBggq
hkjOPQQDAgNHADBEAiBE7sDAFKCyB6tFmC4oIh8ykgx3GrF07WQDru7MOQFNsQIg
RLQxe4AEpm6ruzhOeLiWHApFYQmVq2OZqABXLvfDgh8=
-----END CERTIFICATE-----
```

4. **Paste a self-generated signature or Generate a new signature with the private key**



AUTHORIZATION

Approval TimeOut Date

Approval Time Out Time: --:--

Certificate: barbara.cert

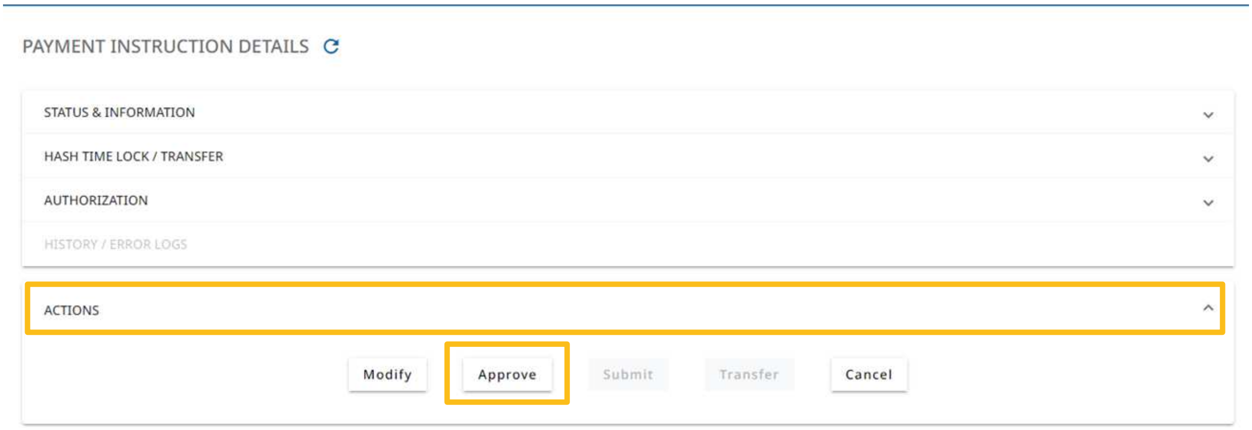
```
-----BEGIN CERTIFICATE-----
MIIC7DCCApOgAwIBAgIUDk8vu75O9o1WUMfKkc7U3tnv8KcwCgYIKoZIzj0EAwIw
ga4xCzAJBgNVBAYTAkRFMRIwEAYDVQQIDAlGcmFua2Z1cnQxFTATBgNVBAcMDFRh
dW51c2FubGFnZTETMBEGA1UECgwKQnVuZGVzYmFuazETMBEGA1UECwwKQnVuZGVz
YmFuazE0MPORttw3Y2EuZXhwZXJpbWVudC50cmlnZ2Vyc29sdXRpb24uYnVu
ZGVzYmFuay5kZTEUMBIGA1UEKQwLTUFSS0RFRjBUUkkwHhcNMjMwODExMDMxOTA4
WhcNMjgxMjA2MTMxNjAwWjBtMQswCQYDVQQGEwJERTEPMA0GA1UECBMGSGVzc2Vu
MQ0wCwYDVQQHEwRUZXN0MRswGQYDVQQKExJUcmlnZ2VyY2hhaW5NZW1iZXIxDzAN
BgNVBAsTBmNsaWVudDEQMA4GA1UEAxMHYmFyYmFyYTBZMBMGByqGSM49AgEGCCqG
SM49AwEHA0IABMLBRedNBjuDj0WwOJAghyg1OaVloWffGnGpxsuSJgsBswyKUfsf
7fcizKNHL2RZttzvPWdY3TxYWiaFh0T7SSKjgc4wgcswDgYDVR0PAQHreWQDE3
MAwGA1UdEwEB/wQCMAAwHQYDVR0OBBYEFKGIG6gsxyvsPchxosKzgqJNUuqHMB8G
A1UdIwQYMBaAFLVlGSan5JioNKsapGSzuVfwuIVwMGsGCCoDBAUGBwgBBF97ImF0
dHJzIjp7ImhmLkFmZmlsaWF0aW9uIjoiIiwiaGYuRW5yb2xsbWVudElEIjoiYmFy
YmFyYSIsImhmLlR5cGUiOiJjbGllbnQiLCJwcml2aWxlZ2UiOiI0RSJ9fTAKBggq
hkjOPQQDAgNHADBEAiBE7sDAFKCyB6tFmC4oIh8ykgx3GrF07WQDru7MOQFNsQIg
RLQxe4AEpm6ruzhOeLiWHApFYQmVq2OZqABXLvfDgh8=
-----END CERTIFICATE-----
```

Signer key: Signed

MEUCIG7I6TOx5T1aFlhHyWN0+hlJBos5k39aPPAUgo/5FriuAiEAluFmVAjXnq7WdHyg0KR+urH0wm5hcLEmarFyHs890oc=

Note: The private key is used to and only to generate locally the signature from the client browser, and is not transmitted or stored to backend systems.

5. **Approve the Payment instruction from the Actions Section**



6.5 Verification

Signature verification occurs at the Trigger Solution level.

Custom OID, Issuer, and signature string undergo validation.

6.6 Revocation of a certificate and user

If a Trigger Solution participant connected via API (U2A or A2A) wants to revoke a certificate and the respective ExtraNet user, please contact triggersolution@bundesbank.de.

Please provide the following information:

First and last name (if applicable) of the person / application owning the certificate / user

User ID of the ExtraNet user.

The Trigger Solution Team will then perform the revocation in the Trigger Solution and ExtraNet.

Deutsche Bundesbank
Wilhelm-Epstein-Straße 14
60431 Frankfurt am Main
Deutschland