

1. Solution Architecture	2
1.1 ART0512 Architecture Overview	4
1.2 ART0651 System Context	6
2. Technical Documentation	18
2.1 IDE Setup	19
2.2 Component Startup Instructions	24
2.3 NGROK Setup	26
3. Description Reference Implementation	28
4. Security	29
4.1 Guidance	31
4.2 SecDevOps	32
4.3 Security Acceptance Criteria	34
4.4 Security Meetings	36
4.5 Vulnerability Management	37
4.6 Weekly status	41

Solution Architecture

SSI@Bundeskanzleramt / Digital Driver's License Solution Architecture

Version	Date	Author	Comment
0.1	13.07.2021		Initial version

This page provides an overview of the sub-pages describing the solution architecture of the vaccination pass solution.

Table of Contents

#	Title	Confluence Page	Short Description	Owner	Status
ART 0512	Architecture Overview	ART0512 Architecture Overview	An overview of the solution architecture and its key concepts.		DRAFT
ART 0651	System Context	ART0651 System Context	An overview of the systems the DDL solution integrates with.		DONE
ART 0515	Component Model		An overview and description of the components of the DDL solution architecture.		TODO
ART 0513	Architectural Decisions	Architectural Decisions	A list of key architectural decisions impacting the solution design.		TODO
ART 0507	Non Functional Requirements	Non-functional Requirements	A list of nonfunctional requirements upon which the architectural design is motivated.		TODO

Introduction

Scope and Purpose

The purpose of this document is to provide an overview of the key solution architecture viewpoints.

The contents of this document are enhanced with the progress of this project.

Intended Audience

This document provides the Bundeskanzleramt and IBM an overview of the defined solution scope and documents agreements on how the solution is realized. It applies to IBM and its partners.

Distinction

A comprehensive description of the underlying Hyperledger indy network is not the focus of this document.

Open Topics

- None

Glossary

Term	Definition
SSI	Self Sovereign Identity

- Working Document from Project -

ART0512 Architecture Overview

Version	Date	Author	Comment
0.1	13.07.2021		Initial version
0.2	13.12.2021		added Architecture Overview Diagram
1.0	17.12.2021		added Architectural Goals, pushed to version 1

Table of Contents

- 1 [Architectural Goals](#)
 - 1.1 [Reference implementation: Light-weight, API-driven](#)
 - 1.2 [Built for Interoperability](#)
 - 1.3 [Built for Privacy](#)
- 2 [Architecture Overview Diagram](#)

Architectural Goals

This chapter describes the general architectural goals of the solution.

Reference implementation: Light-weight, API-driven

The developed component serves as a reference implementation, to be used for understanding the concept of Germany's Digital Driver's license from a verifier point of view.

It is intended to be extended and adapted to the verifier's need, but must not be seen as production-ready. The code is published under Apache 2.0 license on [Github](#).

The reference implementation aims to simplify integration for verifying parties and to abstract the complexity of interacting with agent-to-agent communication, handle cryptographic proof of device bound credentials, and to provide business-driven API endpoints.

Built for Interoperability

The verification controller should support a multitude of use cases providing a mechanism to securely and reliably verify a digital driver's license credential of a holder: For example, car rental or corporate vehicle management. The following key aspects are taken into consideration:

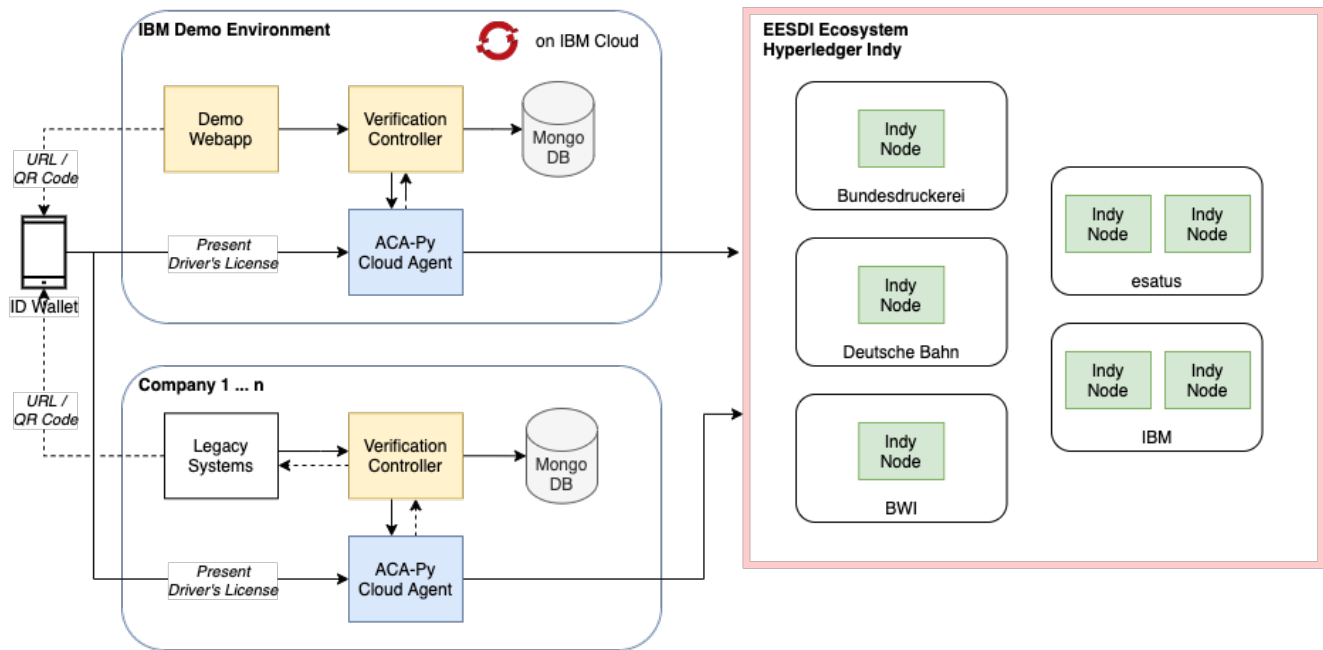
1. The component should be compatible with digital driver's license SSI credentials, issued by Bundesdruckerei and Kraftfahrtbundesamt (KBA).
2. Interfaces must be agnostic so that any system may consume the service.
3. Open standards and protocols must be used to support a variety of external systems.

Built for Privacy

The reference implementation must follow the *principle of least privilege*, using as minimal sensitive data as required to verify the holder's validity of their digital driver's license. The Hyperledger Indy / Aries blockchain solution is used to put holders in control of what data they share. Little to no personal data must be persisted within the verification controller.

Architecture Overview Diagram

The diagram below provides a high-level overview of the solution diagram.



Source: [Draw.io](https://draw.io)

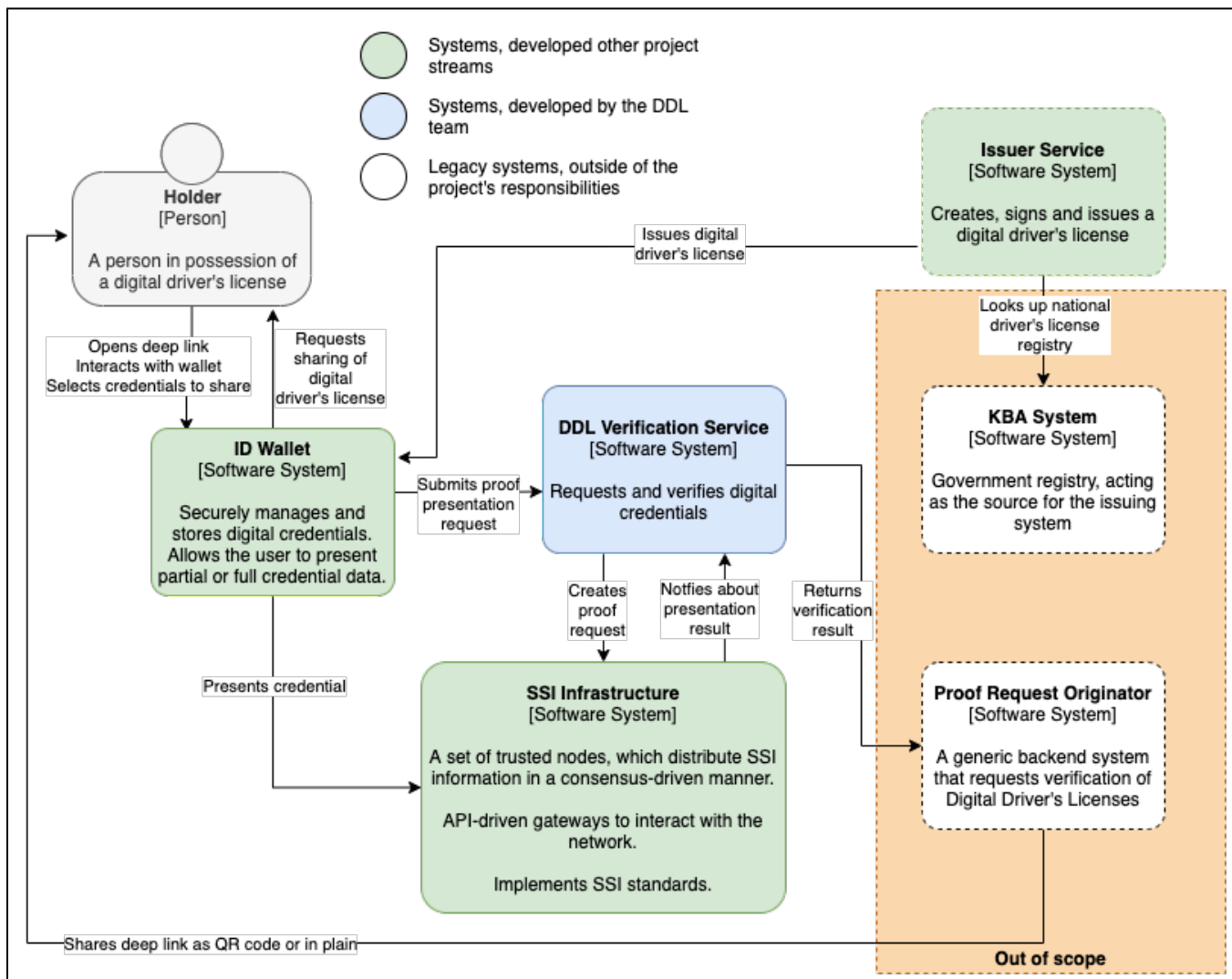
ART0651 System Context

Version	Date	Author	Comment	Status
0.1	14.07.2021		Initial version	DRAFT
0.2	06.08.2021		Review, Added Actors, Sequence Diagram, reworked the whole document	DRAFT
0.3	07.08.2021		Prepare for Review	REVIEWED
0.4	17.08.2021		Incorporate Feedback, updated context diagram, updated sequence diagram, a complete overhaul of all chapters	REVIEWED
0.5	19.08.2021		Incorporate Feedback, Prepare for final Review	IN REVIEW
1.0	25.08.2021		Resolved all pending comments, moved to 1.0 and marked as done	DONE

Table of Contents

- 1 [System Context Diagram](#)
 - 1.1 [Actors](#)
 - 1.2 [Software Systems](#)
- 2 [Key Integration Use Cases](#)
 - 2.1 [Proof Request Originator Integration](#)
 - 2.1.1 [Deep Link Generation \(INT01\)](#)
 - 2.2 ["Proof Presentation" request creation](#)
 - 2.2.1 [Create Proof Request \(INT02\)](#)
 - 2.3 ["Proof Presentation" verification \(INT03\)](#)
 - 2.4 [Callback Proof Request Originator \(INT04\)](#)
 - 2.5 [Revocation](#)
- 3 [Error Handling](#)
 - 3.1 [The Holder rejects to share their credentials](#)
 - 3.2 [The Holder does not respond to the approval request](#)
 - 3.3 [The Holder tries to provide a proof with incorrect first name, last name or birthdate](#)
 - 3.4 [The Holder does not meet the requirements on its Digital Driver License credential.](#)
 - 3.5 [The DDL Verification Service has an internal technical issue.](#)
- 4 [Complementing material](#)
 - 4.1 [Schemas](#)
 - 4.1.1 [Verification Request Schema](#)
 - 4.1.2 [Proof Request Schema](#)
 - 4.1.3 [Digital Driver's License Schema](#)
 - 4.1.4 [Proof Response Schema](#)

System Context Diagram



Source: [ddl-art0651-context.drawio](#)

Actors

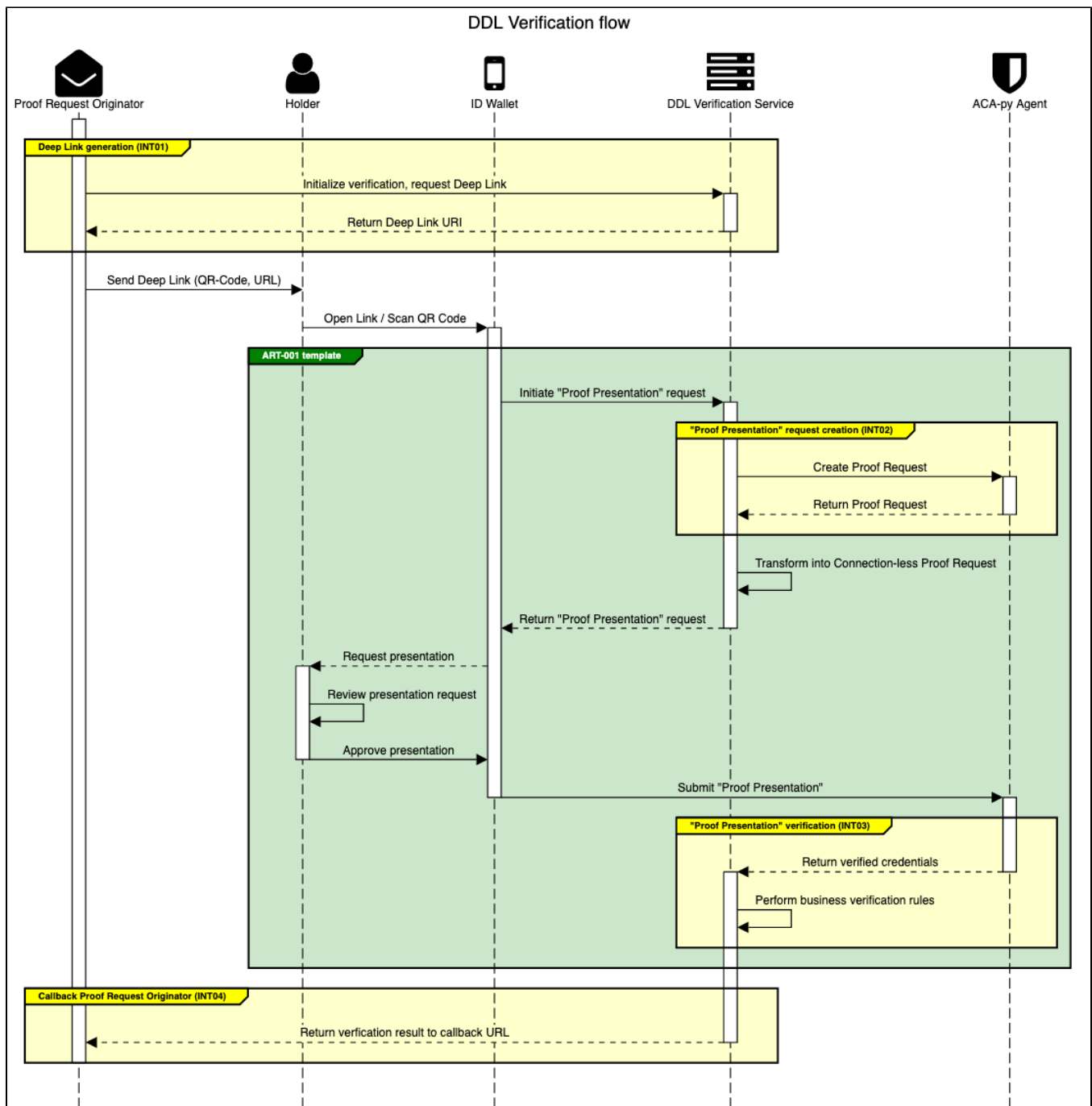
- **Holder:** A person who is in possession of SSI credentials, in this case, a Digital Driver's License.

Software Systems

- **DDL Verification Service:** The solution described by this document, allowing the verification of a *holder's* Digital Driver's License.
- **ID Wallet:** A mobile wallet to securely store and manage a *holder's* SSI credentials, and orchestrating credential exchange ceremonies (i.e. credential issuing, proof requests, proof presentation).
- **SSI Infrastructure:** The infrastructure and protocols, based on a Hyperledger Indy network and Cloud agents.
- **Issuer Service:** The service operated by *Bundesdruckerei BDr*, responsible for issuing digital versions of driver's licenses.
- **KBA System:** The national registry of driver's licenses, operated by *Kraftfahrt Bundesamt KBA*.
- **Proof Request Originator:** A generic backend system operated by the *Vehicle Fleet Operators*, that requests verifications of Digital Driver's Licenses, integrating into the *DDL Verification Controller*.

Key Integration Use Cases

Integration follows the flow described in [ART005 Verification Integration Pattern](#) with the customization described in the following subchapter.



Source

```
title DDL Verification flow

fontawesome f2b6 Proof Request Originator
fontawesome f007 Holder
fontawesome f10b ID Wallet
fontawesome f233 DDL Verification Service
fontawesome f132 ACA-py Agent

activate Proof Request Originator
group #yellow Deep Link generation (INT01) #black
Proof Request Originator->DDL Verification Service:Initialize verification, request Deep Link
activate DDL Verification Service
Proof Request Originator<--DDL Verification Service:Return Deep Link URI
deactivate DDL Verification Service
end
Proof Request Originator->Holder:Send Deep Link (QR-Code, URL)
Holder->ID Wallet:Open Link / Scan QR Code
activate ID Wallet
group #green ART-001 template #white
ID Wallet->DDL Verification Service:Initiate "Proof Presentation" request
activate DDL Verification Service
group #yellow "Proof Presentation" request creation (INT02) #black
DDL Verification Service->ACA-py Agent:Create Proof Request
activate ACA-py Agent
DDL Verification Service<--ACA-py Agent:Return Proof Request
deactivate ACA-py Agent
end
DDL Verification Service->DDL Verification Service:Transform into Connection-less Proof Request
ID Wallet<--DDL Verification Service:Return "Proof Presentation" request
deactivate DDL Verification Service
Holder<--ID Wallet: Request presentation
activate Holder
Holder->Holder:Review presentation request
Holder->ID Wallet: Approve presentation
deactivate Holder
ID Wallet->ACA-py Agent:Submit "Proof Presentation"
activate ACA-py Agent
deactivate ID Wallet
group #yellow "Proof Presentation" verification (INT03) #black
DDL Verification Service<--ACA-py Agent:Return verified credentials
deactivate ACA-py Agent
activate DDL Verification Service
DDL Verification Service->DDL Verification Service:Perform business verification rules
end
end
group #yellow Callback Proof Request Originator (INT04) #black
Proof Request Originator<--DDL Verification Service:Return verification result to callback URL
deactivate DDL Verification Service
end
deactivate Proof Request Originator
```

Proof Request Originator Integration

This section describes an interaction pattern to use the DDL Verification Service as Source to verify the validity of a *holder's* Digital Driver's License. This interaction is based on the architectural templates [ART001 Verify SSI Credentials](#), [ART003 Device Binding Verification](#), and [ART005 Verification Integration Pattern](#). These are the key principles for the design decisions:

- Verifying a Digital Driver's License must be possible without SSI-specific knowledge for the *holder*.
- Integration points must be generic to support a multitude of *Proof Request Originators*.

Deep Link Generation (INT01)

The *Proof Request Originator* requests a deep link from the *DDL Validation Controller* and distributes the link as a transformed QR-code and URL to the *holder* by email (or other channels).

This verification request includes the following metadata fields:

- Working Document from Project -

- `callbackURL` | *not null*: A webhook URL to which the *DDL Validation Controller* will return the result of the verification process.
- `validUntil` | *null*: An optional expiry date for the generated "Verification Request"

Additional fields are included in the schema and populated by the *DDL Validation Controller*.

Upon successful communication with the *ACA-py* agent and generation of the "Verification Request", all metadata gets persisted and referenced by a generated *verificationId*. This ID will be used in subsequent phases to refer to the "Verification Request" entry.

The payload returned to the *Proof Request Originator*:

Verification Request Response

```
{
  "uri": "string",
  "verificationId": "string"
}
```

"Proof Presentation" request creation

Upon activation of the *ID Wallet* with a given Deep Link, it performs a RESTful call to the *DDL Verification Service*. The API call provides the *verificationId* reference attribute, to look up the corresponding verification request generated in INT01.

Create Proof Request (INT02)

The *DDL Verification Service* creates a "Proof Request" to request the presentation of the required attributes with restrictions on the Digital Driver's License credential schema definition. Additionally, self-attested attributes (ART006) can be requested without schema restrictions. This, for example, allows for requesting additional information from the holder for the purpose of custom business rules validation, or, for activating device-bound credentials (refer to [ART003 Device Binding Verification](#)).

Upon successful communication with the *ACA-py* agent and generation of the "Proof Request", the returned `threadId` will be updated to the persisted metadata record in INT01.



Thread ID

The `threadId` acts as a correlation id, allowing for mapping subsequent "Proof Presentations" to the original "Proof Request".

The converted "Connection-less Proof Request" is being returned as a Base64-encoded DIDComm URI.

"Proof Presentation" verification (INT03)

Once the "Proof Presentation" request got processed by the *ID Wallet* the *DDL Verification Service* implements the Digital Driver's License-specific business verification logic.

1. **Query Metadata:** Query metadata record persisted in INT01 and INT02, using the provided `threadId`.
2. **Check ACA-py verification result**
3. **Check Device Binding Challenge:** As describe in [ART003 Device Binding Verification](#), the signature challenge is verified using the public key provided through the `masterID.hardwareDID` attribute.
4. **Check DDL business verification rules:** Perform custom business rules for verifying the presented Digital Driver's License attributes.
5. **Delete Proof record from ACA-py**

Callback *Proof Request Originator* (INT04)

Once the Digital Driver's License SSI credential has been verified, the "Verification Response" is returned to the callback URL (Webhook), originally provided in INT01.

Verification Result Response

```
{
  "code": "number",
  "verified": "boolean",
  "verificationId": "string",
  "data": {
    "keyvalueset"
  },
  "message": string
}
```

Revocation

In future releases of the *DDL Verification Controller*, the *KBA System* is going to be queried for revoked credentials.



WIP

Please note, that this functionality is not available in the current release.

Error Handling

The following error scenarios may occur during the process of verifying the Digital Driver License credential. This chapter describes how the Proof Request Originator will be informed about an error state.

- The response schema is based on *ddl.proof.response.schema.json*.
- User errors are within the 4xx error range.
- System errors are within the 5xx error range.
- A successful verification evaluation is indicated with a 200 status code. However, the *data.verified* body has to be evaluated.
- In case of an error 4xx or 5xx, the returned body will be empty.

The Holder rejects to share their credentials

In case the user rejects to share their credential, the process will stop. As of today, the ID Wallet app does not provide any feedback information to the Verification Service. The DDL Verification Service system implements a timeout, though, in case no response is received from the wallet within 120 seconds, it will presume the proof request failed and will POST the following body to the *callbackURL*.

```
{
  "code": 408,
  "message": "Request Timeout Error"
}
```

The Holder does not respond to the approval request

The user may remain inactive on the ID Wallet app and does not continue the process. The DDL Verification Service will timeout after 120 seconds and presume the proof request failed. The following body will be sent as POST to the *callbackURL*.

```
{
  "code": 408,
  "message": "Request Timeout Error"
}
```

The Holder tries to provide a proof with incorrect *first name*, *last name* or *birthdate*

The deep link includes the first name, last name, and birthdate in the metadata. This data is checked against the base id. If the provided attributes do not match with that data the proof can not be accepted. The DDL Verification Service will POST the following body to the *callbackURL*.

```
{
  "code": 404,
  "message": "Not found"
}
```

The Holder does not meet the requirements on its Digital Driver License credential.

e.g. The user (Holder) does not have a driver's license to operate a motorcycle. The DDL Verification Service will POST the following body to the callbackURL.

```
{
  "code": 200,
  "data": {
    "verified": false
  }
}
```

The DDL Verification Service has an internal technical issue.

In case there is any technical issue with the backend system, e.g. the database is not available, the DDL Verification Service system will POST the following body to the callbackURL.

```
{ "code": 500, "message": "Internal Server Error" }
```

Complementing material

Schemas



WIP

Please note, that these schemas are currently subject to the integration and use case discussions. They will be updated accordingly. However, they already include the key aspects to support inbound and outbound communication.

Verification Request Schema

DDL Verification Request Schema

```
{
  "$schema": "http://json-schema.org/draft-07/schema",
  "$id": "https://ssi.ddl.example.com/ddl.verification.request.schema.json",
  "type": "object",
  "title": "DDL meta data schema definition",
  "description": "DDL schema for exchanging meta data for a proof request processing.",
  "default": {},
  "examples": [
    {
      "callbackURL": "https://webhook.example.com/47110815",
      "validUntil": "2021-10-03",
      "id": "123456789",
      "presentationExchangeId": "123456789",
      "verificationId": "123456789",
      "verifier": "DDL Verifier",
      "data": {}
    }
  ],
  "required": [
    "callbackURL"
  ],
  "properties": {
    "callbackURL": {
      "$id": "#/properties/callbackURL",
      "type": "string",
      "title": "The callbackURL schema",
      "description": "An explanation about the purpose of this instance.",
      "examples": [
        "https://webhook.example.com/47110815"
      ]
    },
    "validUntil": {
      "$id": "#/properties/validUntil",
      "type": "string",
      "title": "Date constraint",
      "description": "A date constraint for the presented credential."
    }
  }
}
```

- Working Document from Project -

```

        "examples": [
            "2021-10-03"
        ]
    },
    "id": {
        "$id": "#/properties/id",
        "type": "string",
        "title": "Unique identifier",
        "description": "An unique identifier",
        "examples": [
            "47110815"
        ]
    },
    "presentationExchangeId": {
        "$id": "#/properties/presentationExchangeId",
        "type": "array",
        "title": "Unique presentation exchange identifier",
        "description": "An unique presentation exchange identifier",
        "examples": [
            "47110815"
        ]
    },
    "verificationId": {
        "$id": "#/properties/verificationId",
        "type": "array",
        "title": "Unique verification identifier",
        "description": "An unique verification identifier",
        "examples": [
            "47110815"
        ]
    },
    "verifier": {
        "$id": "#/properties/verifier",
        "type": "array",
        "title": "Identification of the verifier",
        "description": "Identification of the verifier",
        "examples": [
            "verify-corp"
        ]
    },
    "data": {
        "$id": "#/properties/data",
        "type": "object",
        "title": "The data schema",
        "description": "An explanation about the purpose of this instance.",
        "examples": [
            {}
        ],
        "required": [],
        "additionalProperties": true
    }
},
"additionalProperties": true
}

```

Proof Request Schema

DDL Request Schema

```

{
    "$schema": "http://json-schema.org/draft-07/schema",
    "$id": "https://ssi.ddl.example.com/ddl.proof.request.schema.json",
    "type": "object",
    "title": "DDL proof request schema",
    "description": "The ddl proof request schema describing the expected payload to initiate a proof request.",
    "default": {},
    "examples": [
        {

```

- Working Document from Project -

```

        "firstname": "Max",
        "lastname": "Mustermann",
        "birthdate": "1983-03-17",
        "classes": ["B"]
        "callbackUrl": "https://callback.example.com/api/v1/936d6626-1b85-497e-b64e-518195b4747f"
    }
},
"required": [
    "firstname",
    "lastname",
    "birthdate",
    "classes",
    "callbackUrl"
],
"properties": {
    "firstname": {
        "$id": "#/properties/firstname",
        "type": "string",
        "title": "The firstname schema",
        "description": "The firstname of the holder to start the digital driver license verification.",
        "default": "",
        "examples": [
            "Max"
        ]
    },
    "classes": {
        "$id": "#/properties/classes",
        "type": "array",
        "minItems": 1,
        "title": "The driver license classes",
        "description": "The requested driver license classes.",
        "items": {
            "type": "string",
            "enum": ["A", "A1", "A2", "B", "BE", "B96", "C1", "C1E", "C", "CE", "D1", "D1E", "D", "DE"]
        }
    },
    "lastname": {
        "$id": "#/properties/lastname",
        "type": "string",
        "title": "The lastname schema",
        "description": "The lastname of the holder to start the digital driver license verification.",
        "default": "",
        "examples": [
            "Mustermann"
        ]
    },
    "birthdate": {
        "$id": "#/properties/birthdate",
        "type": "string",
        "title": "The birthdate schema",
        "description": "The date of birth of the credential holder in the ISO 8601 format.",
        "default": "",
        "examples": [
            "1983-03-17"
        ]
    },
    "callbackUrl": {
        "$id": "#/properties/callbackUrl",
        "type": "string",
        "title": "The callbackUrl schema",
        "description": "The callback to be called once the verification has been evaluated. A post request
will be submitted to that endpoint containing the proof response. see also ddl.proof.response.schema.json.",
        "default": "",
        "examples": [
            "https://callback.example.com/api/v1/936d6626-1b85-497e-b64e-518195b4747f"
        ]
    }
},
"additionalProperties": true
}

```

Digital Driver's License Schema

Digital Driver's License Schema v0.2

```
{
  "txn": {
    "data": {
      "data": {
        "attr_names": [
          "licenseCategoryB1_Restrictions",
          "generalRestrictions",
          "licenseCategoryDE_DateOfIssuance",
          "licenseCategoryA2_Restrictions",
          "licenseCategoryD_DateOfExpiry",
          "licenseCategoryA1_Restrictions",
          "licenseCategoryT_DateOfIssuance",
          "licenseCategoryD_DateOfIssuance",
          "dateOfIssuance",
          "licenseCategoryL_Restrictions",
          "licenseCategoryCE_DateOfExpiry",
          "licenseCategoryC1_DateOfIssuance",
          "licenseCategoryD_Restrictions",
          "licenseCategoryB_Restrictions",
          "issuingEntity",
          "licenseCategoryA1_DateOfIssuance",
          "licenseCategoryD1E_DateOfExpiry",
          "licenseCategoryCE_DateOfIssuance",
          "firstName",
          "licenseCategoryC1E_Restrictions",
          "licenseCategoryM_Restrictions",
          "licenseCategoryC1E_DateOfIssuance",
          "licenseCategoryC_DateOfIssuance",
          "licenseCategoryDE_DateOfExpiry",
          "licenseCategoryD1_DateOfExpiry",
          "licenseCategoryB_DateOfIssuance",
          "licenseCategoryT_Restrictions",
          "licenseCategoryD1E_Restrictions",
          "licenseCategoryM_DateOfIssuance",
          "licenseCategoryD1_DateOfIssuance",
          "licenseCategoryBE_Restrictions",
          "licenseCategoryAM_DateOfIssuance",
          "licenseCategoryC_DateOfExpiry",
          "licenseCategoryC1_Restrictions",
          "licenseCategoryCE_Restrictions",
          "licenseCategoryDE_Restrictions",
          "licenseCategoryA_DateOfIssuance",
          "licenseCategoryL_DateOfIssuance",
          "licenseCategoryC1_DateOfExpiry",
          "licenseCategoryBE_DateOfIssuance",
          "placeOfBirth",
          "licenseCategoryC_Restrictions",
          "licenseCategoryAM_Restrictions",
          "licenseCategoryD1E_DateOfIssuance",
          "academicTitle",
          "licenseCategoryC1E_DateOfExpiry",
          "id",
          "licenseCategoryB1_DateOfIssuance",
          "licenseCategoryD1_Restrictions",
          "familyName",
          "licenseCategoryA2_DateOfIssuance",
          "hardwareDID",
          "licenseCategoryA_Restrictions",
          "dateOfBirth"
        ],
        "name": "Digitaler Fuehrerschein",
        "version": "0.2"
      }
    }
  },
  "metadata": {
```

- Working Document from Project -

```

    "digest": "a8220a414d02ae5e59a342641dc2efc6147192fe57b60bf01aa79d96b8020f57",
    "from": "9hsRe5jdzAbbyLAsT6sPc",
    "payloadDigest": "b8be86d7c03eb4a8b8d99a4b6f474598d9a3edd9505169d1498c4b522bb6b9bd",
    "reqId": 1628857671741074700
  },
  "protocolVersion": 2,
  "type": "101",
  "typeName": "SCHEMA"
},
"txnMetadata": {
  "seqNo": 393,
  "txnId": "9hsRe5jdzAbbyLAsT6sPc:2:Digitaler Fuehrerschein:0.2",
  "txnTime": "2021-08-13T12:27:51.000Z"
}
}
}

```

Source: <https://eesdi-test.esatus.com/tx/EESDI-Test-Ledger/domain/393>

Proof Response Schema

DDL Proof Response Schema

```

{
  "$schema": "http://json-schema.org/draft-07/schema",
  "$id": "http://example.com/ddl.proof.response.schema.json",
  "type": "object",
  "title": "DDL proof response schema",
  "description": "The ddl proof response schema describing the response payload after verification.",
  "default": {},
  "examples": [
    {
      "code": 200,
      "data": {
        "verified": true
      },
      "message": ""
    }
  ],
  "required": [
    "code"
  ],
  "properties": {
    "code": {
      "$id": "#/properties/code",
      "type": "integer",
      "title": "The code schema",
      "description": "Represents the status code for the verification process. These codes are aligned on the http status codes, hence 200 OK, 400 Bad Request, 500 Internal Server Error",
      "default": 200,
      "examples": [
        200, 400, 500
      ],
      "minimum": 100,
      "maximum": 511
    },
    "data": {
      "$id": "#/properties/data",
      "type": "object",
      "title": "The data schema",
      "description": "The data payload, only provided if the code is 200",
      "default": {},
      "examples": [
        {
          "verified": true
        }
      ],
      "required": [
        "verified"
      ]
    }
  ]
}

```

- Working Document from Project -

```

    ],
    "properties": {
      "verified": {
        "$id": "#/properties/data/properties/verified",
        "type": "boolean",
        "title": "The verified schema",
        "description": "Represents the verification process result. true means the holder has
successfully proofed, that he is meeting the digital driver license requirements.",
        "default": false,
        "examples": [
          true
        ]
      }
    },
    "additionalProperties": true
  },
  "message": {
    "$id": "#/properties/message",
    "type": "string",
    "title": "The message schema",
    "description": "Contains the error message in case of an response code 400 or 500.",
    "default": "",
    "examples": [
      ""
    ]
  }
},
"additionalProperties": true
}

```


Technical Documentation

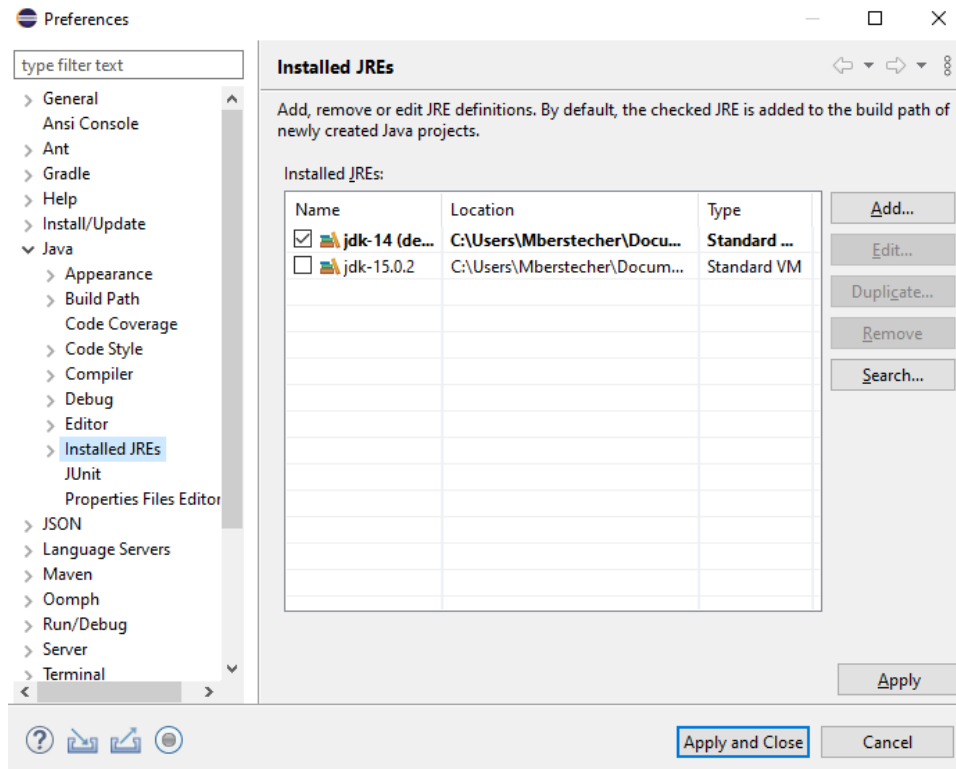
- [IDE Setup](#)
- [Component Startup Instructions](#)
- [NGROK Setup](#)

IDE Setup

All information on how to setup the IDE will be explained here.

Prerequisites:

- Docker ([Windows](#))
- Eclipse ([2021-03 R](#))
- Java SE Development Kit ([14.0.2](#))
 - If needed or you use multiple JDKs use the following steps:
 1. Window Preferences
 2. Java Installed JREs
 3. Add or select jdk-14



- Download [Code Style](#)

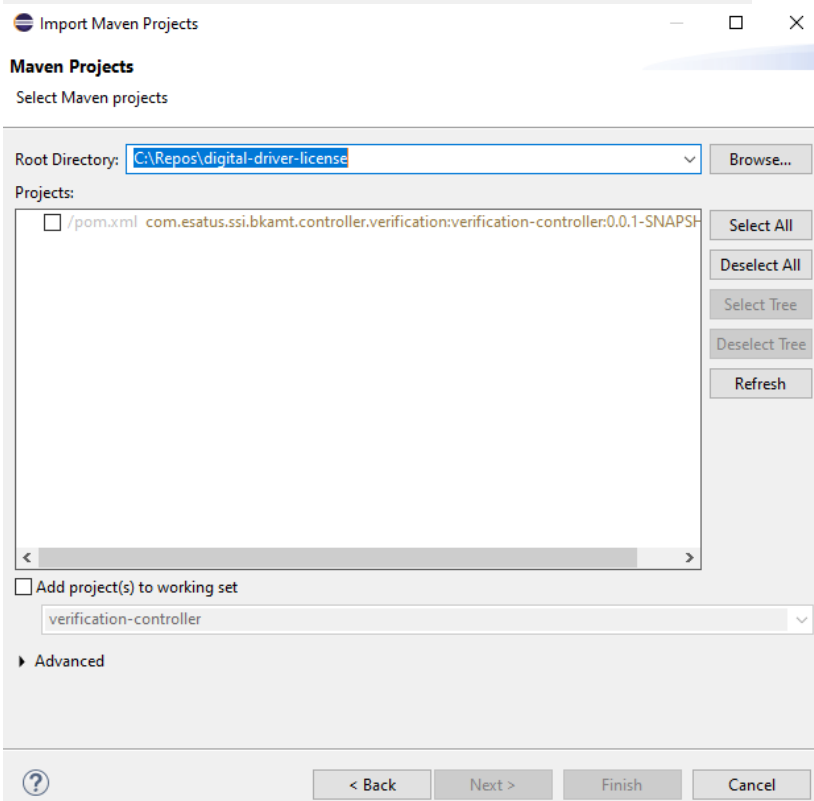
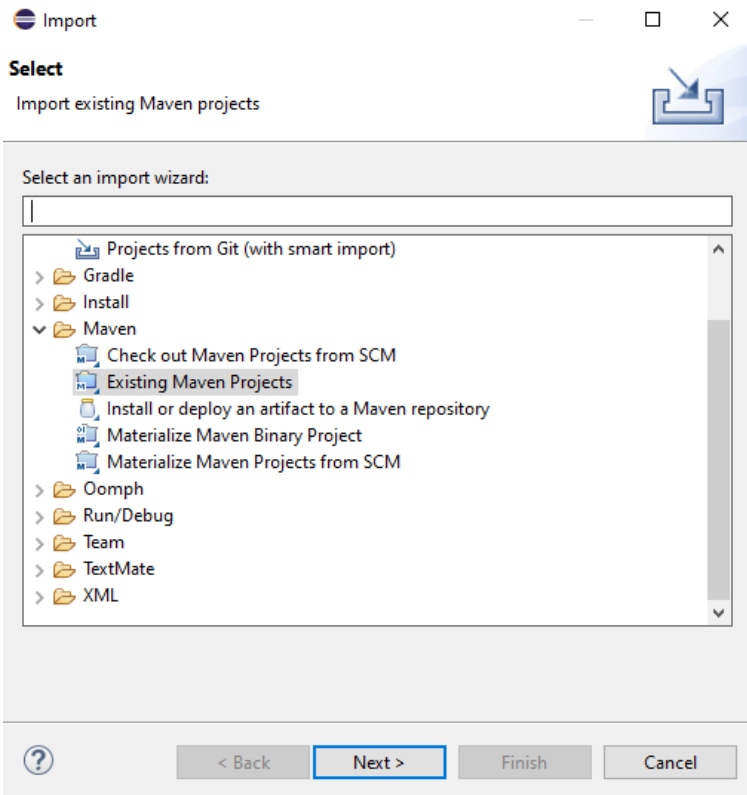
Import the project:

Code of ther verification-controller in [GitLab](#)

1. Download and save the project in the folder of your choice

Import the project in Eclipse:

1. File Import...
2. Select *Exisisting Maven Projects* under the folder Maven
3. Choose the righ directory of the Project, select the *pom.xml* file and click **Finish**



Run configuration:

- ✓ Be sure to start the docker of the database first

Database:

- Working Document from Project -

1. Go to the root directory of the project in any shell
2. Start the database with the following command

```
docker-compose -f src/main/docker/controller-mongo-mongoadmin.yml up -d
```

```
docker-compose -f src/main/docker/controller-mongo-mongoadmin.yml down -v
```

Verification-Controller:

There are two options to start the verification-controller

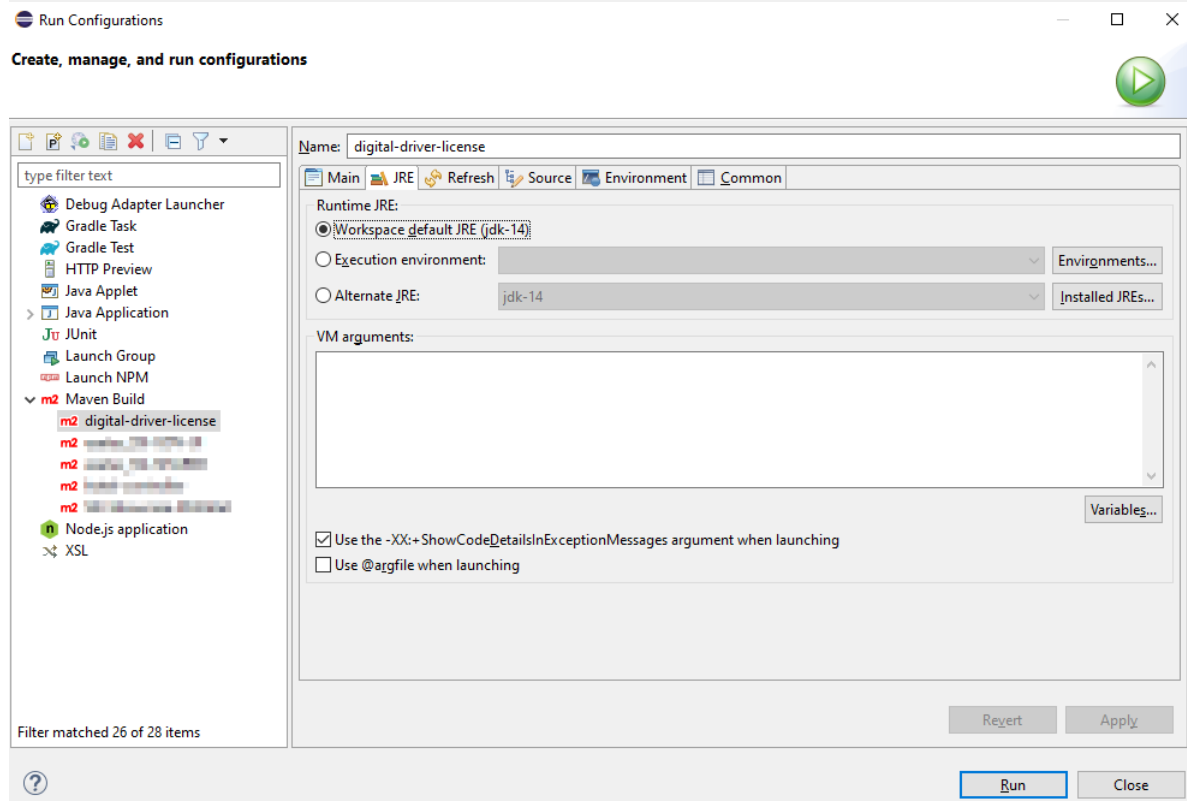
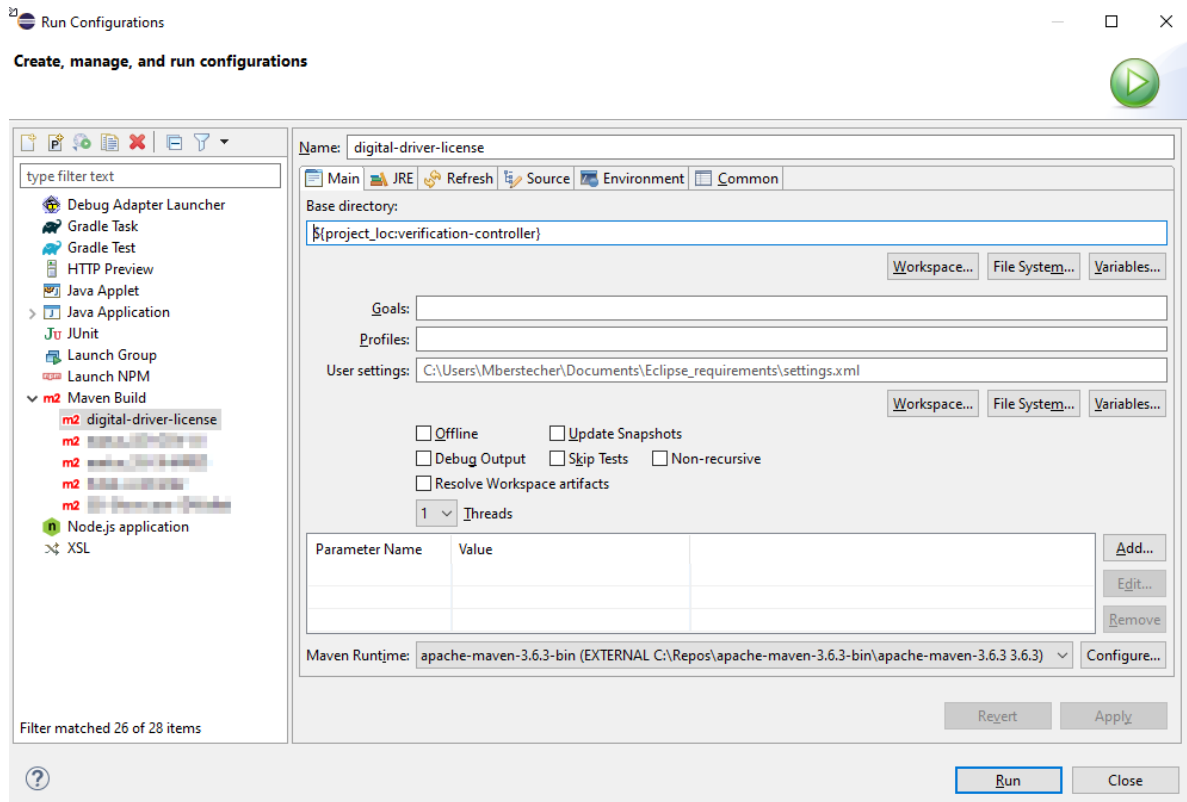
First Option:


1. Go to the root directory of the project in any shell
2. Start the verification-controller with the following command

```
./mvnw
```

Second Option:

1. Run Run As Maven Build
2. Check the run configuration and click **Run**



 Further details on how to use the verification-controller are in the README

- Working Document from Project -

Component Startup Instructions

Prerequisites

Environment variables

Navigate to `src/main/docker/` and rename the `.env-default` file to `.env`. This file is used by docker to set all required environment variables passed into the docker containers.

There are different variables to set:

1. VERIFY AGENT (ACA-PY)

- `VERIFY_AGENT_GENESIS_URL`: URL of the genesis file the agent uses
- `VERIFY_AGENT_WALLET_KEY`: Key to generate the wallet with and unlocks the wallet
- `VERIFY_AGENT_API_KEY`: Secures all requests to ACA-Py (Send view X-API-KEY Header)
- `VERIFY_AGENT_WEBHOOK_APIKEY`: Send from ACA-Py in X-API-Key the application endpoints

2. Network

- `IP_ADDRESS`: Your current IP-Address

3. MONGO DB

- `MONGODB_USERNAME`: Mongodb username (defaults to admin123)
- `MONGODB_PASSWORD`: Mongodb password (defaults to pass123)

Add `verification-agent-url`, `verification-agent-key`, `credential-definition-id` in `application-dev.yml` file.

Development

To start your application in the dev profile, open the terminal, navigate to the `verification-controller` folder and run the following commands:

```
docker-compose -f src/main/docker/agent-mongodb.yml up -d
./mvnw
```

The first step will deploy a MongoDB instance. The second step will deploy the application.

Swagger-Ui

Swagger UI will be available at the following URL

```
http://localhost:8090/swagger-ui/index.html
```

Note: The API key can be configured in `src/main/resources/config/application-dev.yml` (application properties) file which can be used to interact with the API.

MongoDB

There is an database init script `mongo-init.js` located in `src/main/docker/mongodb/` which connects to the mongodb on port 27017. The scripts creates an admin user with username: admin123 and password: pass123.

After the connection was established successfull it creates a new user:

```
username: user123
password: 123pass
```

You can use this user to connect to the database with your favourite MongoDB access tool. Here we use AdminMongo. You will find more information about how to use it below.

AdminMongo

AdminMongo is running in Port 8092

```
- localhost:8092
- Connection-Name: Verification Controller
- Working Document from Project -
```

```
- Connection-String: mongodb://user123:123pass@ddl-mongodb:27017/VerificationController?authSource=VerificationController
```

Initial Verifier

The database is initialized with a demo verifier with the following values:

```
name: demo
password: $2y$10$AW0Zit2JNBcTI0UDpPmc4OM72nm86AyvoOfV7GJOP4iropj9IuyVS
```

The password in plain text is secure. When you try to interact with the api endpoints use the plain value in the X-AUTH-HEADER.

To create a new verifier you have to create a password hash with bcrypt with a strength factor of 12 Password length may not be greater than 72 bytes because that the maxim password length bcrypt supports only

You can use any online bcrypt password generator, e.g. <https://www.appdevtools.com/bcrypt-generator>

Endpoints

As mentioned above all endpoints are documented via swagger on <http://localhost:8090/swagger-ui/index.html>

Currently, there are three different security mechanisms for these endpoints:

- /api/proof: No Authentication
- /topic/present_proof: The X-AUTH-HEADER is checked against the value configured in the application-dev.yml under ssibk: verification:controller:apikey
- /*: All requests to any other routes have to include a value in the X-AUTH-HEADER which matches the api-key of a verifier stored in the database (remember they are hashed in the database so don't use the hash in the header, use the plan value instead)

Testing with your mobile device and the ID Wallet App

To test the whole application with your mobile phone you need to make sure the following prerequisites are met:

- You have the ID Wallet app installed on your mobile device
 - iOS: <https://apps.apple.com/at/app/id-wallet/id1564933989>
 - Android: <https://play.google.com/store/apps/details?id=com.digitalenabling.idw&hl=de&gl=US>
- All containers are running without any errors
- You have a tool like ngrok installed on your system (<https://ngrok.com/>). You can use any other tool which provides the same functionality but this how-to uses ngrok. See the docs of you favourite tools on how to use it

Configure your IP Address

Make sure the correct IP Address of your PC in /src/main/docker/.env

```
# NETWORK
IP_ADDRESS=192.168.152.43
```

Start the application and execute ngrok

Open three terminals and navigate to the folder ngrok is located in each instance of your shell (e.g. Ngrok is in C:\dev\ngrok navigate with `cd C:\dev\ngrok`)

- `.\ngrok.exe http 8090`
- `.\ngrok.exe http 10000`
- `.\ngrok.exe http 10080`

Open demo page

When the application runs with dev profile there is demo website where you can generate a QR-Code to scan with your Wallet-ID App. To see this page call

Ngrok now creates three public endpoints which are tunneled to your local endpoints

NGROK Setup

Download: [ngrok](#)

Ports: 10000, 10080, 8090

Command

```
ngrok.exe http <port>
```

To create a secure endpoint for each local one execute

Command

```
ngrok.exe http 8089
ngrok.exe http 10000
ngrok.exe http 10080
```

For each command you will get a secure ngrok endpoint that looks like this:

Copy the two endpoints for port **10000** and **8089**.

Paste these two values in your applications-dev.yml (you do not need to enter the value for port 10000, so just ignore it):

YML Configuration

```
ssibk:
  verification:
    controller:
      apikey: 123
      endpoint: https://xxx.ngrok.io # PORT: 8089 -> Enter the correct ngrok url for PORT 8089

  agent:
    apikey: secure
    apiurl: localhost:10080
    recipientkey: xxx
    endpoint: https://xxx.ngrok.io # the DDL agent endpoint | PORT: 10000 -> Enter the correct ngrok url
for PORT 10000
    endpointName: DDL Verification

  ddl:
    credential_definition_ids: xxx
    requested_attributes: xxx
```

- Working Document from Project -

```
expiryCheck:  
  attribute: Ausstellungsdatum  
  format: yyyyMMdd  
  validity: 0
```

Description Reference Implementation

What is the reference implementation ?

The reference implementation is a template for a software system that enables the verification of digital driver's licenses within the SSI ecosystem.

Partner companies acting as verifiers within the SSI ecosystem have the opportunity to use this as a technical reference for their own implementations. The core functionality includes verification of digital driver's licenses issued within the ecosystem. Other functionalities includes device binding, API / webhook integration with originator systems, configurable queries and verification of attributes to be presented.

Since the reference implementation is a template and not a production-ready software, it needs to be adapted to the respective system landscape of the individual verifier. Thus, it only serves as an orientation guide with recommendations for an implementation and will be updated on a regular base.

Where can I find more information ?

There is a public developer handbook on Github that is available to everyone. The developer handbook contains onboarding documents, relevant documentation and general recommendations for a successful and secure implementation. For this purpose, please ensure that you are complied with the company-specific security requirements and general best practices.

Link to developer handbook: <https://github.com/My-DIGI-ID/ddl-verification-controller/blob/develop/README.md>

The Confluence page here is only accessible to a limited group of users, namely partner companies that are already in the implementation phase of the verification service.

Please keep in mind that we, as the publisher of this reference implementation, do not assume any liability and warranty.

Security

Organization

Security team

Name		Project role
<div><div></div><div></div></div>		

Communication

- preferably within Slack-Channel #ssi-ddl or direct contact in Slack

Scope

Prio 1:

- interne Sicherheit mit [REDACTED]. (DevOps, Entwicklung)
- Sicherheitskonzeption? mit bdr abstimmen (bdr für issuing service selbst, wir dann für verifizierungs service) Sebastian Luther ansprechen

Prio 2:

- Vulnerability-Mgmt (Konzept erstellen)

Prio 3:

- übergreifende Sicherheitsperspektive mit allen Stakeholder, offene Themen identifizieren

Out of scope:

- Pentesting, Sebastian Luther fragt SecureSystems an

Objectives and tasks

zu überarbeiten

Priorities categories: 0 = must, highest prio; 1 = should, middle prio; 2 = could, lowest prio

- ☐ align with Impfpass-team regarding priorities of BSI regarding DevSecOps

Topic	Task	Priority	Status
	- Working Document from Project -		

SecDevOps	DevOps assessment à https://owasp samm.org /model/	0	<input checked="" type="checkbox"/> assessment meetings with frontend/backend devs <input checked="" type="checkbox"/> assessment meeting with Hartwig for DevOps
	Setting up a scanner for secrets in the repo (including commits)	0	
	Setting up static code analysis (recommendations, setup can be done by the team)	0	
	Setting up third party vulnerability scanning tools (libraries/container)	0	
	Setting up patch process	1	
	Define security release conditions (gates, checkpoints, guardrails, milestones...)	1	
	Confluence/Jira + Git MFA	2	
	Review onboarding process + roles/rights	2	Existing documentation (seems not to be complete for all project members): https://ibm.ent.box.com/folder/135615606465?s=9jfrktx0s0gm1g0yfbjo187ro8m6frd "On-and off-boarding is conducted by the Project Manager."
	Security awareness training for project members	0	General DS&P training done: https://ibm.ent.box.com/folder/135623585104 <input type="checkbox"/> more specific training for architects and developer <input type="checkbox"/> introduction of our team and activities in "all-hands" maybe in the daily
	Security testing (support) - tbd	tbd	
Security Concept (BSI 200-x)	Create artifacts for security concept (overview, threats/risks, guidelines) first usage of app 08.08. ideally done until end of july	0	<ul style="list-style-type: none"> ■ Boris in discussion with Heike regarding RKI+BSI involvement <input type="checkbox"/> meeting scheduled with architects (██████████)
	Gather reference documents (from CovPass, evatus, Bundesdruckerei, etc.)	0	Security plan (hotel-uc): https://ibm.ent.box.com/file/799731831625?s=10odthdbjhccqkri33osp7j20til76o Datenflüsse (hotel-uc): https://ibm.ent.box.com/file/782163414326?s=135s60doe8s8dsb36tple0vhe3bzm62
Secure architecture	Threat modeling with IriusRisk	0	<input type="checkbox"/> ██████████ ensure access and dedicated project in IR <input type="checkbox"/> create threat model with architects ██████████
	Abuse case meetings for HotelCheckin, Impfpass and KBA with arch-/dev-team	0	<input type="checkbox"/> set up meeting ██████████ <input type="checkbox"/> verify current solution
	Listing all tls endpoints and checking the used cipher suites	1	
	Identifying critical code parts (like for jwt signing) and planning code audits for them	1	
Secure operations	Review existing constructs / concepts	2	
	Identifying metrics/measures to track the state of security a. using issues found by devops tools /scanner b. establish risk management	1	

Guidance

Recommended TLS versions:

- The use of **TLS 1.2** and **TLS 1.3** is **recommended**.
- **TLS 1.0** and **TLS 1.1** are **not recommended**.
- **SSLv2** and **SSLv3** are **not recommended**.

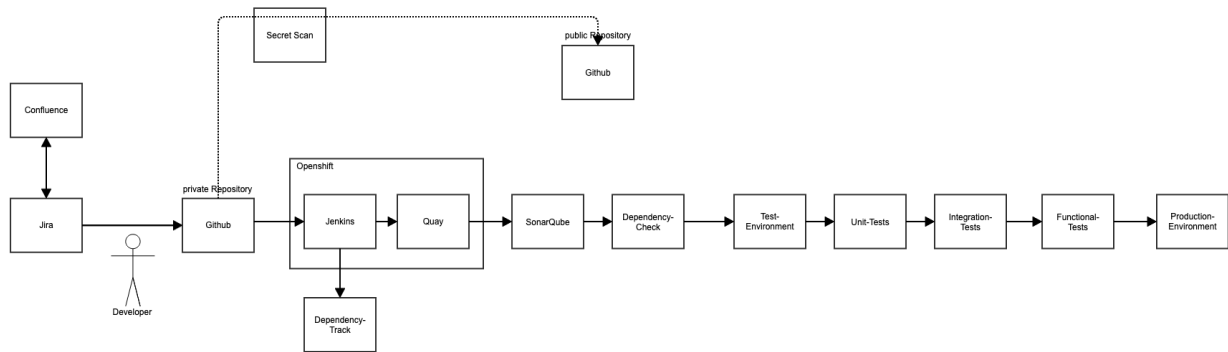
Cipher-Suites

In TLS 1.2, cipher suites are usually specified with the naming convention TLS_AKE_WITH_Enc_Hash, with AKE denoting an (authenticated) key agreement process, Enc denoting an encryption process with operating mode and hash denoting a hash function. The hash function is used by an HMAC (Keyed-Hash Message Authentication Code), which is used for the PRF (PseudoRandom Function) for key derivation. If Enc is not an authenticated encryption process (Authenticated Encryption with Associated Data, AEAD for short), the HMAC is also used to ensure the integrity of the record protocol. It is generally recommended to only use cipher suites that meet the requirements of the algorithms and key lengths of [TR-02102-1].

The full lists of the recommended cipher suites can be found in: **Technical Guideline regarding the use of TLS:** <https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102-2.pdf>

SecDevOps

1) SAMM Assessment



2) Security requirements (security stories)

- Use static code analysis to prevent the use of insecure code patterns (e.g. correctly implement SonarQube for the CI/CD)
- Block commit/publish of secrets and tokens (e.g. using Truffle-Hog)
- Scan all libraries and used third party software for vulnerabilities (e.g. using Dependency-Track)
- Scan used images and containers for security issues (e.g. clair)

3) Specific assessments

a) Code review

Code review performed for Hotel-Checkin use-case.

SQL Injection:

- All queries to the database are currently done via prepared statements using the default SPRING methodology using *org.springframework.data.mongodb.repository.MongoRepository*
- However, most DTO objects implement an "*object.toString()*" method which creates Strings using user input without sanitization. This might lead to a problem if the *AuditEventConverter* saves data using the "*convertDataToStrings()*" method, because the comment says that "This method will allow to save additional data.". This is for the future to examine.
- For the future: If dynamic statements are being made please check if the data is sanitized before used as par of the statement.

Other Findings:

- Default *BCryptPasswordEncoder()* is used, which doesn't get initialised using a random vector and which also does not use a salt. (e.g. <https://github.com/My-DIGI-ID/hotel-controller/blob/open-source-git-hub/src/main/java/com/ibm/ssi/controller/hotel/service/impl/UserServiceImpl.java> line 88)
- Passwords in code (e.g. *mongoinit.js*)

- Working Document from Project -

b) Secret Scan

Prior publication of the DDL verification component code, a scan for passwords, tokens and other secrets has been performed. The results are saved here:



`pw_regex_match.txt`



`pw_high_entropy.txt`

Security Acceptance Criteria

Acceptance criteria for different stages of the secure development lifecycle:

Application Architecture and Design

- Data flow and process flow diagrams are current and assessed for risk by Information Security
- System and application components are hardened according to Information Security requirements and checked for vulnerabilities through manual or automated security testing. All critical and high vulnerabilities will be remediated prior to release.
- Components are segregated based on compliance requirements and Information Security Data Handling Requirements.

Authentication, Authorization, Accounting

- All pages and resources require authentication except those specifically intended to be public.
- Forms containing credentials are not filled in by the application. Pre-populating a form could mean that credentials are stored in plaintext or a reversible format.
- All authentication controls are enforced on the server side.
- All authentication controls fail securely to ensure attackers cannot log in.
- All authentication decisions can be logged, without storing sensitive session identifiers or passwords. This should include requests with relevant metadata needed for security investigations.
- Information enumeration is not possible via login, password reset, or forgot account functionality.
- Verify there are no default passwords in use for the application framework or any components used by the application (such as "admin /password").
- Anti-automation is in place to prevent breached credential testing, brute forcing, and account lockout attacks.
- All authentication credentials for accessing services external to the application are encrypted and stored in a protected location.
- Account lockout is divided into soft and hard lock status, and these are not mutually exclusive. If an account is temporarily soft locked out due to a brute force attack, this should not reset the hard lock status.
- If shared knowledge based questions (also known as "secret questions") are required, the questions must not violate privacy laws and are sufficiently strong to protect accounts from malicious recovery.
- The system can be configured to disallow the use of a configurable number of previous passwords.
- All authentication challenges, whether successful or failed, should respond in the same average response time.
- Secrets, API keys, and passwords must not be included in the source code, or online source code repositories.
- Administrative interfaces are not accessible to untrusted parties.

Access Control

- Directory browsing is disabled unless deliberately desired. Additionally, applications should not allow discovery or disclosure of file or directory metadata, such as Thumbs.db, .DS_Store, .git or .svn folders.
- Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.
- A centralized mechanism (including libraries that call external authorization services) is used for protecting access to each type of protected resource.
- All access control decisions can be logged and all failed decisions are logged.
- The application has additional authorization (such as step up or adaptive authentication) for lower value systems, and / or segregation of duties for high value applications to enforce anti-fraud controls as per the risk of application and past fraud.
- The application correctly enforces context-sensitive authorization to not allow unauthorized manipulation by means of parameter tampering.

Input Validation Handling

- The runtime environment is not susceptible to buffer overflows, or that security controls prevent buffer overflows.
 - Input validation routines are enforced on the server side and input validation failures result in request rejection and are logged.
 - A single input validation control is used by the application for each type of data that is accepted.
 - All SQL queries, HQL, OSQL, NOSQL and stored procedures, calling of stored procedures are protected using prepared statements or query parameterization, and thus not susceptible to SQL injection
 - The application is not susceptible to LDAP Injection, or that security controls prevent LDAP Injection.
 - The application is not susceptible to OS Command Injection, or that security controls prevent OS Command Injection.
 - The application is not susceptible to Remote File Inclusion (RFI) or Local File Inclusion (LFI) when content is used that is a path to a file.
 - The application is not susceptible to common XML attacks, such as XPath query tampering, XML External Entity attacks, and XML injection attacks.
 - Ensure that all string variables placed into HTML or other web client code is either properly contextually encoded manually, or utilize templates that automatically encode contextually to ensure the application is not susceptible to reflected, stored and DOM Cross-Site Scripting (XSS) attacks.
 - If the application framework allows automatic mass parameter assignment (also called automatic variable binding) from the inbound request to a model, verify that security sensitive fields such as "accountBalance", "role" or "password" are protected from malicious automatic binding.
 - Verify that the application has defenses against HTTP parameter pollution attacks, particularly if the application framework makes no distinction about the source of request parameters (GET, POST, cookies, headers, environment, etc.)
 - All input data is validated, not only HTML form fields but all sources of input such as REST calls, query parameters, HTTP headers, cookies, batch files, RSS feeds, etc; using positive validation (whitelisting), then lesser forms of validation such as greylisting (eliminating known bad strings), or rejecting bad inputs (blacklisting).
 - Structured data is strongly typed and validated against a defined schema including allowed characters, length and pattern (e.g. credit card numbers or telephone, or validating that two related fields are reasonable, such as validating suburbs and zip or post codes match).
 - Unstructured data is sanitized to enforce generic safety measures such as allowed characters and length, and characters potentially harmful in given context should be escaped (e.g. natural names with Unicode or apostrophes, such as O'Hara)
- Working Document from Project -

- Untrusted HTML from WYSIWYG editors or similar are properly sanitized with an HTML sanitizer and handle it appropriately according to the input validation task and encoding task.
- For auto-escaping template technology, if UI escaping is disabled, ensure that HTML sanitization is enabled instead.
- Data transferred from one DOM context to another, uses safe JavaScript methods, such as using `.innerText` and `.val`.
- When parsing JSON in browsers, that `JSON.parse` is used to parse JSON on the client. Do not use `eval()` to parse JSON on the client.
- Authenticated data is cleared from client storage, such as the browser DOM, after the session is terminated.

Encryption At Rest

- All cryptographic modules fail securely, and errors are handled in a way that does not enable oracle padding.
- Cryptographic algorithms used by the application have been validated against FIPS 140-2 or an equivalent standard.
- Sensitive passwords or key material maintained in memory is overwritten with zeros as soon as it no longer required, to mitigate memory dumping attacks.
- Verify that all keys and passwords are replaceable, and are generated or replaced at installation time.
- Random numbers are created with proper entropy even when the application is under heavy load, or that the application degrades gracefully in such circumstances.

Error Handling and Logging

- Security logs are protected from unauthorized access and modification.
- All non-printable symbols and field separators are properly encoded in log entries, to prevent log injection.
- Log fields from trusted and untrusted sources are distinguishable in log entries and audit log or similar allows for non-repudiation of key transactions.
- The logs are stored on a different partition than the application is running with proper log rotation.
- Time sources should be synchronized to ensure logs have the correct time.

Data Protection and Verification

- List classification of data processed by the application according to Software Company's Information Classification Standard. This should be documented in data flow diagrams. There must be explicit enforcement in accordance with the Software company Data Handling Matrix requirements.
- Verify all sensitive data is sent to the server in the HTTP message body or headers (i.e., URL parameters are never used to send sensitive data).
- The application sets appropriate anti-caching headers as per the risk tier of the application and data it handles.
- On the server, all cached or temporary copies of sensitive data stored are protected from unauthorized access or purged/invalidated after the authorized user accesses the sensitive data.
- Verify the application minimizes the number of parameters in a request, such as hidden fields, Ajax variables, cookies and header values.
- Data stored in client side storage (such as HTML5 local storage, session storage, IndexedDB, regular cookies or Flash cookies) does not contain sensitive data or PII.
- Sensitive information maintained in memory is overwritten with zeros as soon as it no longer required, to mitigate memory dumping attacks.

Communications Security

- Verify that a path can be built from a trusted CA to each Transport Layer Security (TLS) server certificate, and that each server certificate is valid. Also ensure that certificate paths are built and verified for all client certificates using configured trust anchors and revocation information.
- Approved version of TLS and set of cipher suites is used for connections (including both external and backend connections) that pass data classified as RESTRICTED or CONFIDENTIAL according to the Software company Information Classification Standard on public networks, and that it does not fall back to insecure or unencrypted protocols. Ensure the strongest alternative is the preferred algorithm.
- Ensure that a single standard TLS implementation is used by the application and components, that it is configured in accordance with Information Security's Encryption Standard.

HTTP Security Configuration Verification

- Application accepts only a defined set of required HTTP request methods, such as GET and POST are accepted, and unused methods (e.g. TRACE, PUT, and DELETE) are explicitly blocked.
- Every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8, ISO 8859-1).
- HTTP headers added by a trusted proxy or SSO devices, such as a bearer token, are authenticated by the application.
- Suitable X-FRAME-OPTIONS headers are in use for sites where content should not be viewed in a 3rd-party X-Frame.
- HTTP headers or any part of the HTTP response do not expose detailed version information of system components.
- Verify that all API responses contain X-Content-Type-Options: nosniff and Content-Disposition: attachment; filename="api.json" (or other appropriate filename for the content type).
- Content security policy (CSPv2) is in place that helps mitigate common DOM, XSS, JSON, and JavaScript injection vulnerabilities.
- X-XSS-Protection: 1; mode=block header is in place to enable browser reflected XSS filters.

Files and Resources Verification

- Avoid Flash, Active-X, Silverlight, NACL, client-side Java or other client side technologies not supported natively via W3C browser standards.
- Enforce Same-Origin Policy and/or implement Content Security Policy to protect against XSS, Clickjacking and other code injection attacks.

Security Meetings

Kundenworkshops

Datum	Titel	Unterlagen/Protokoll

Interne Meetings

Datum	Titel	Unterlagen/Protokoll
tbd	Sichere Nutzung Projekttools	in Vorbereitung durch [REDACTED]

Vulnerability Management

A **vulnerability** is defined in the ISO 27002 standard as “A weakness of an asset or group of assets that can be exploited by one or more threats” (International Organization for Standardization, 2005).

Vulnerability management is the process in which vulnerabilities in IT are identified and the risks of these vulnerabilities are evaluated. This evaluation leads to correcting the vulnerabilities and removing the risk or a formal risk acceptance by the management of an organization (e.g. in case the impact of an attack would be low or the cost of correction does not outweigh possible damages to the organization).

Vulnerability management is not **vulnerability scanning**. Despite the fact both are related, there is an important difference between the two. Vulnerability scanning consists of using a computer program to identify vulnerabilities in networks, computer infrastructure or applications. Vulnerability management is the process surrounding vulnerability scanning, also taking into account other aspects such as risk acceptance, remediation etc.

Vulnerability Scanners

For Operations

TBD:

1. Define if we want to manage the scanner or outsource it
2. Outsource or buy, install, configure and use the scanner.

Possible scanners:

- Qualys
- Nessus
- OpenVAS (kostenlos und vom BSI genannt)

Wer trifft make or buy Entscheidung?

Wer kann den Scanner kaufen?

Wer übernimmt Verantwortung für den Scanner?

For Development

Activity	Scanner	Owned by	Dev-Stage	Why?
Static Code Analysis	SonarQube		post-commit	Prevents the use of insecure functions/code
Static Code Analysis	SonarLint?		pre-commit	"
Secrets Leak-Check	git-secrets /Trufflehog		pre-publish	Prevents the leakage of passwords, tokens, secrets in the public repository
Software Composition Analysis	Dependency-Track		post-commit	Shows if vulnerability in library.
Container/Image Security Check	Quay		pre-deployment	Shows vulnerabilities in containers/images

For more information have a look at the DevSecOps section

Roles and responsibilities

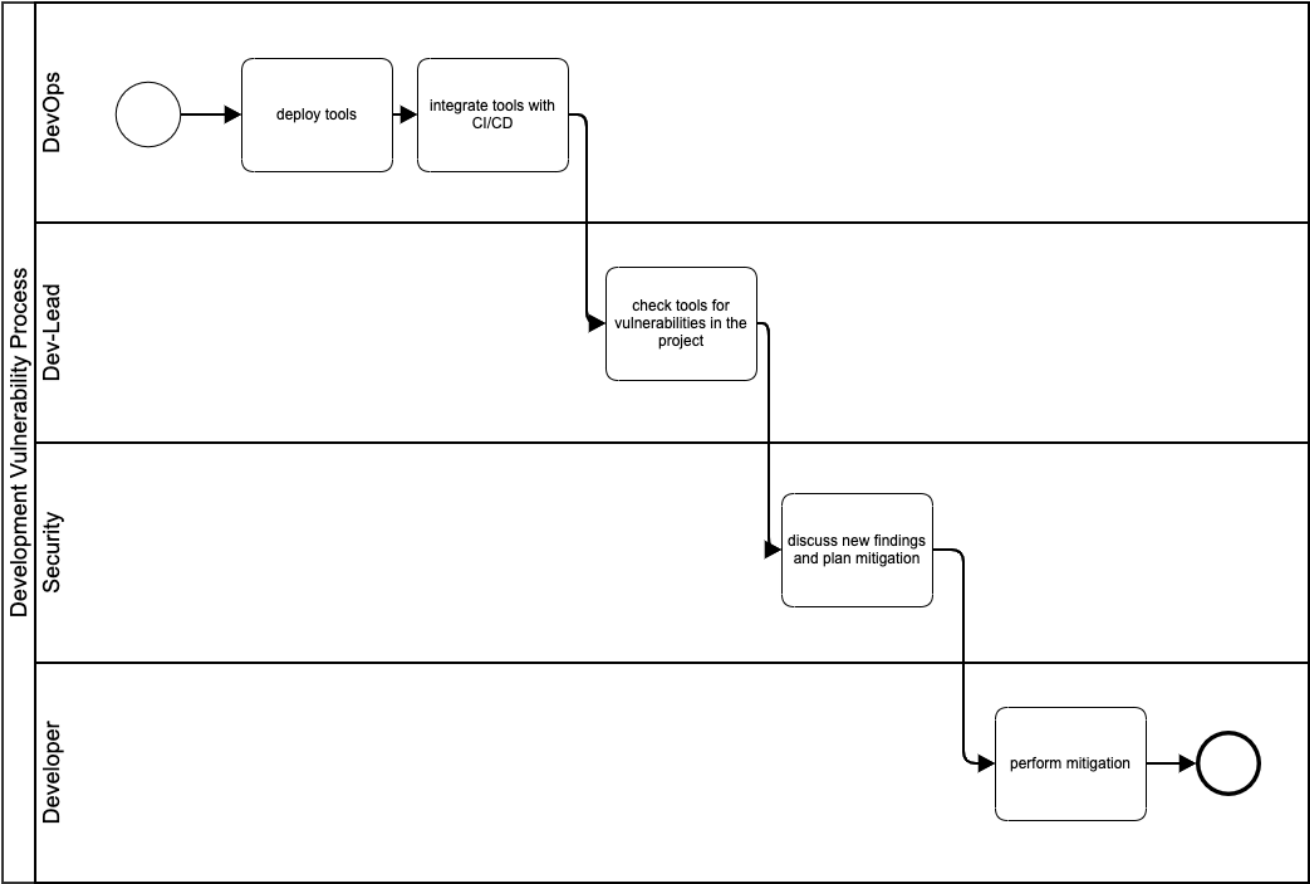
The following roles need to be identified within the organisation:

Role	Description	Contact
Security Officer	The security officer is the owner of the vulnerability management process. This person designs the process and ensures it is implemented as designed.	(IBM Security & Frauenhofer)
Vulnerability Engineer	The vulnerability engineer role is responsible for configuring the vulnerability scanner and scheduling the various vulnerability scans.	TBD
Asset Owner	The asset owner is responsible for the IT asset that is scanned by the vulnerability management process. This role should decide whether identified vulnerabilities are mitigated or their associated risks are accepted.	See inventory
IT System Engineer	The IT system engineer role is typically responsible for implementing remediating actions defined as a result of detected vulnerabilities.	See inventory

Vulnerability Management Process

- Working Document from Project -

Development



Operations

Preparation

Vulnerability Scan

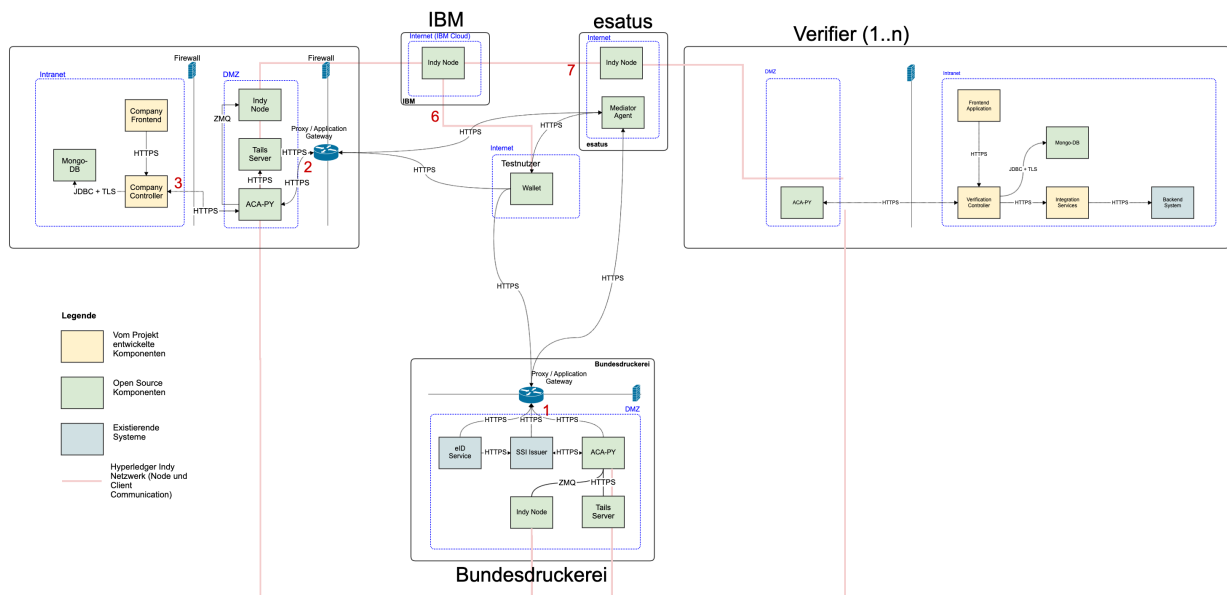
Asset owners need to be informed before the scan. Since vulnerability scanning typically involves sending a large number of packets to systems, they might sometimes trigger unusual effects such as – for example - disrupting network equipment. However, since vulnerability scanning is mainly limited to scanning and not exploiting, risks are minimal.

Define remediation actions

Implement remediating actions

Rescan

Inventory



IBM

Application	Location	Developed by	Owner	Engineer
Company Controller	Intranet			
Company Frontend	Intranet			
MongoDB	Intranet			
Indy Node	DMZ			
Tails Server	DMZ			
ACA-PY	DMZ			
Firewall	Intranet DMZ			
Firewall	DMZ Application Gateway			
Proxy Application Gateway	Internet			
Indy Node	Internet			

Mobile App User

Application	Location	Developed by	Owner
Wallet	Smartphone		
Smartphone OS	Smartphone		

KBA

Application	Location	Developed by	Owner	Engineer
eID Service	DMZ	existing (KBA?)		
Indy Node	DMZ	open source (Hyperledger)		
Tails Server	DMZ	open source (Hyperledger)		
ACA-PY	DMZ	open source (Hyperledger)		
SSI Issuer	DMZ	existing (KBA?)		

Company (BMW/VW/...)

Application	Location	Developed by	Owner	Engineer

- Working Document from Project -

Company Controller	DMZ			
Company Frontend	DMZ			
ACA-PY	DMZ			
MongoDB	Intranet			
Company Integration Service	Intranet			
Existing Backend System	Intranet			