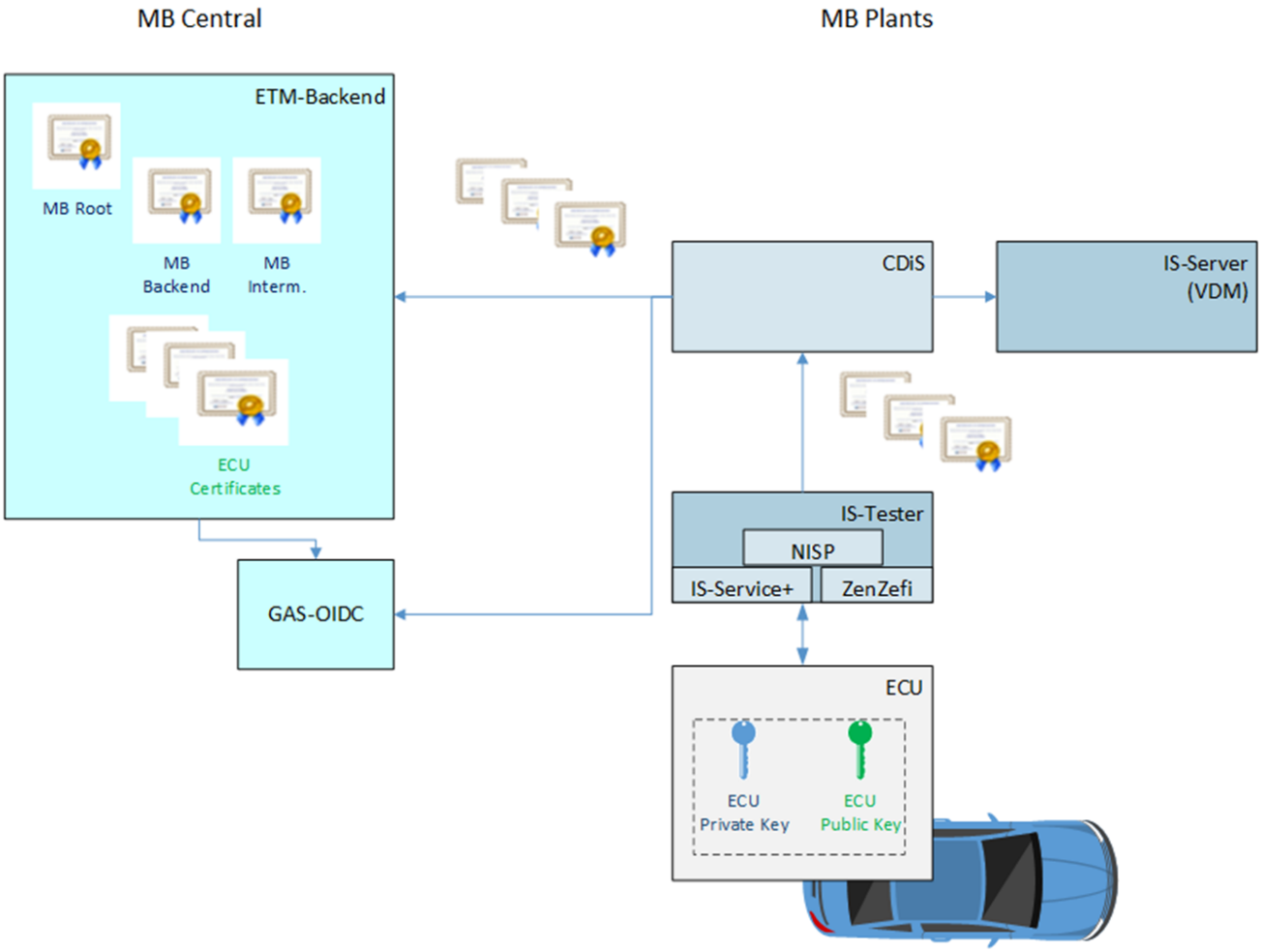
CDiS - Operation Manual

Purpose

CDiS is a microservice responsible to retrieve ECU Trust Model (ETM) certificates from the central (Automotive) PKI and forward them to tester devices.

Its main purpose is to have a decentralised copy of the ETM certificates in each plant to be able to provide them via its REST-API. A car needs up to 100 ETM certificates within a few seconds to keep the production flow. Periodical update cycles ensure that certificates are available at least hours before they are needed by production. In most cases they will be available way earlier depending on the plant and its supply chain.

CDiS will periodically delete old certificates to keep its database at a certain level to ensure fast respond times.



ETM-Backend (PKI):

- Provides ETM, root, backend- and intermediate certificates.
- Allows request of ECU-certificates based on CSR in case of emergency (certificate not available)
- Validates CSRs

CDiS:

- Requests new ETM certificates from ETM-Backend every x hours/minutes and stores them.
- Delivers ETM certificates or the certificate chain (intermediate, backend and root) upon request.
- In case of emergency, verifies if a CSR was sent by a coupled IS-Tester (validates if an additional signature was created with the ZenZefi coupling key) and submits it to ETM-backend to issue a certificate on-the-fly.

IS-Server (VDM):

- Provides ZenZefi coupling key to CDiS in case of emergency process

IS-Service+ (CDiM):

- Provides middleware to allow access from NISP to CDiS-API.
- Requests ETM certificates or certificate chains.
- Initiates emergency process for missing ETM certificates.

ZenZefi:

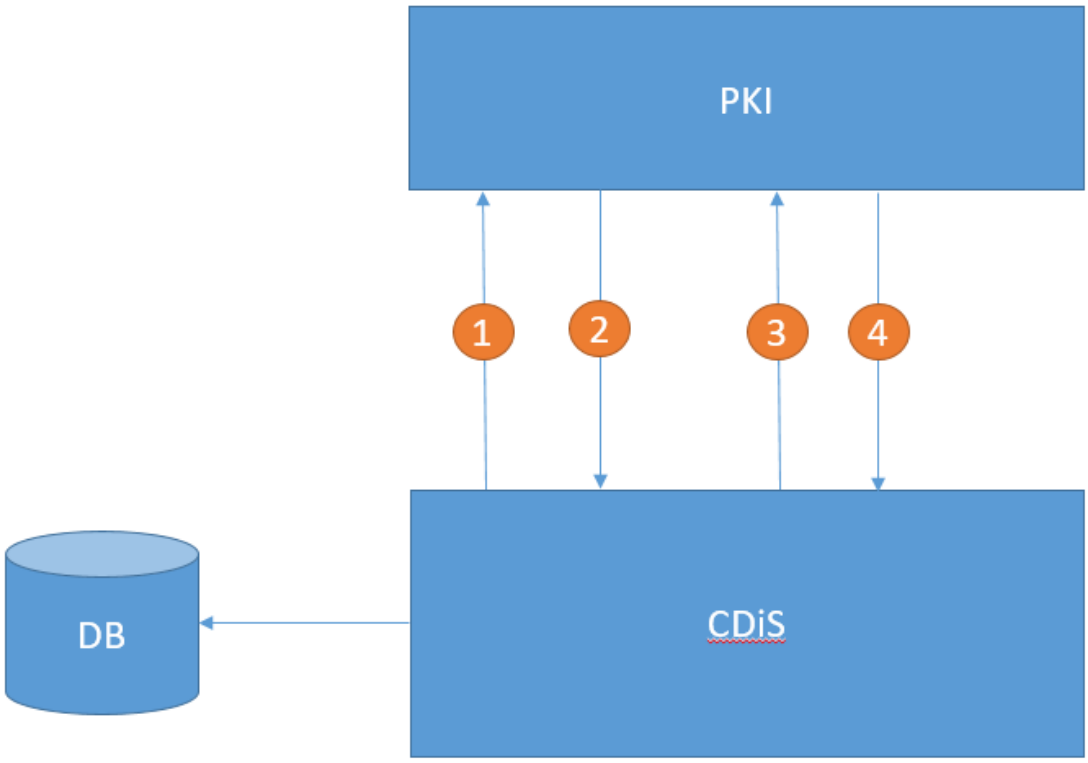
- In case of emergency process creates a signature over the CSR with its private coupling key.

NISP:

- Reads CSRs as input parameter for a certificate request and writes retrieved ETM-certificates back to the ECU.
- Calls the CDiS endpoint via IS-Service+ to request certificates.
- Provides Certificate Chain Partnumber as meta data to CDiS.

Processes

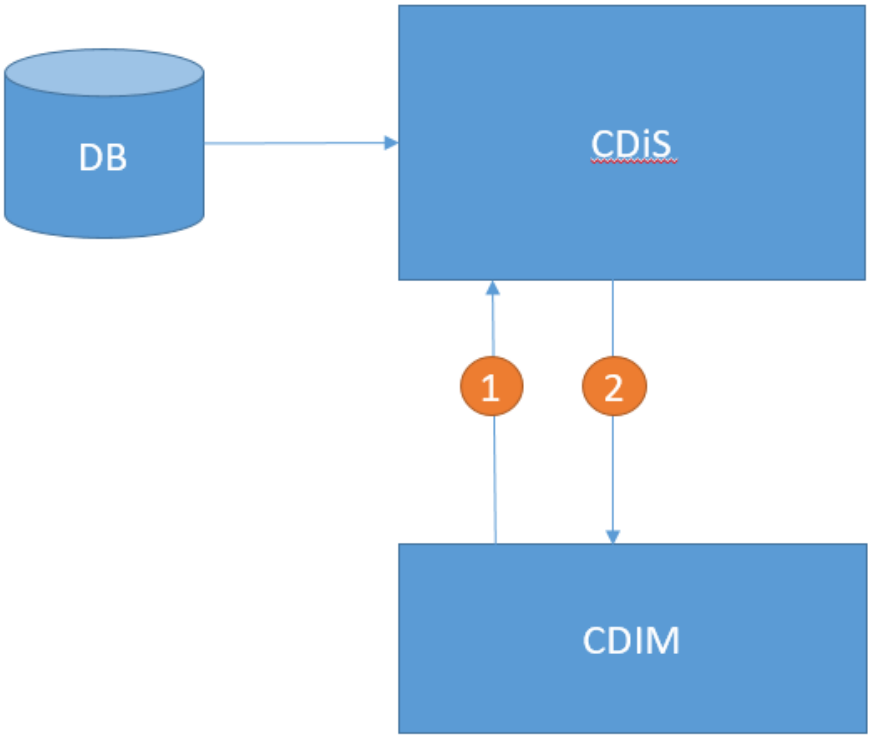
Retrieving certificates



1. CDiS requests all certificates between <start> and <end>
2. PKI sends a paginated list of public keys
3. CDiS requests the ETM certificate for one public key
4. PKI sends the ETM certificate

Steps 3 and 4 are repeated until all certificates are retrieved. Missing/error public key identifiers are stored in a separate db table for later retry. The update cycle can be parameterized (see chapter Cronjobs).

Providing certificates

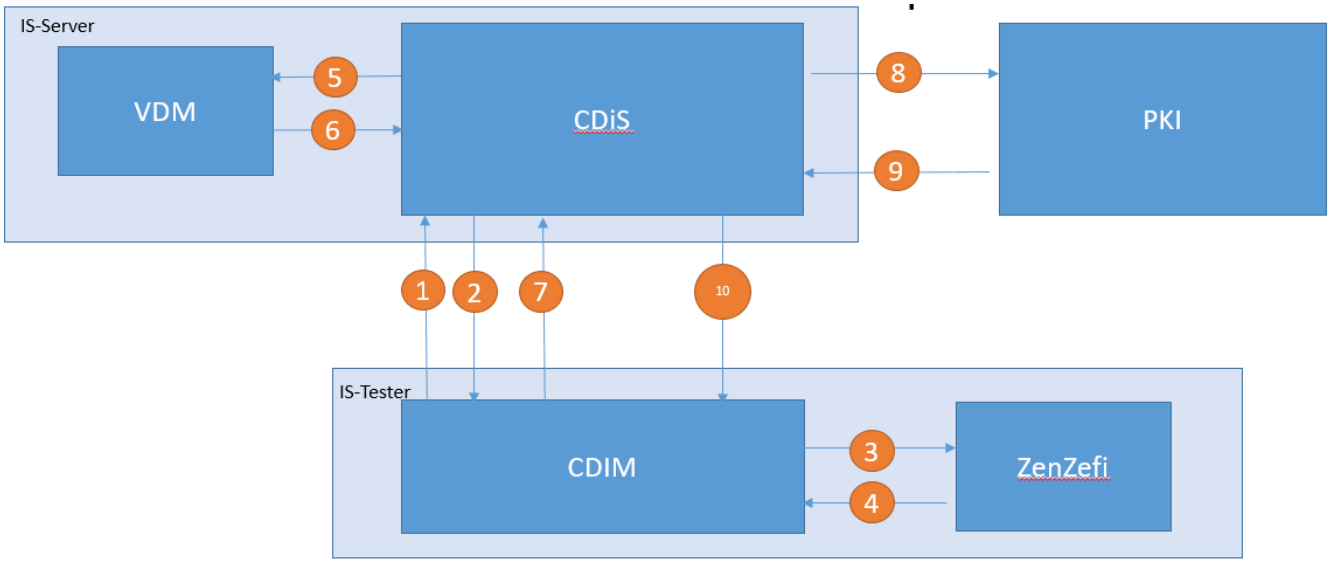


1. Request for an ETM certificate (single request using the public key as ident) (EP manual: https://<baseurl>/etmcertificates?csrPublicKey=ec172b93ad5e563bf4932c70e1245034c35467ef2efd4d64ebf819683467e2bf&partNrCertChain=1234567890)
2. Reply with ETM certificate.

These requests can be done in parallel.

The same process applies to Certificate Chains (different endpoint).

Emergency process



1. Request for certificate Certificate not in DB, not in PKI!
2. Reply negative: Authorization request to IS-Tester to send signed CSR and HostID to authorize device.

3. Request to create signature over CSR and HostID with ZenZefi private key
4. Reply with signature
5. Request public key of ZenZefi to verify signature (not coupled ZZ = ERROR)
6. Reply with public key of ZenZefi
7. Request: Signed CSR + HostID
8. Request to PKI to create the certificate on-the-fly (positive case)
9. Reply with ETM certificate
10. Reply with ETM certificate

The emergency process can be done in parallel for multiple certificates.

Additional information: If the certificate is already available in PKI it doesn't need to be issued on-the-fly. This step was excluded from the process description above to focus on the "real" emergency process where there is no ETM certificate on PKI side.

Deployment

Firewall Clearances

Outgoing

CDiS communicates with the central AM-PKI located at https://am-pki-backend.app.corpintra.net

CDiS authorizes itself using the GAS-OIDC workflow using the URL https://sso.mercedes-benz.com

Both URLs need to be accessible on port 443 (HTTPS).

The PostgreSQL database operates on port 64001. This port needs to be accessible from the host system.

Incoming

CDiS endpoints are accessible on port 443. The URL https://<baseurl>/vds/api/cdis needs to be opend from production to IS-Server.

Database

CDiS works with a dedicated PostgreSQL database. Its credentials have to be submitted to the CDiS container via the application.properties (ENV file later).

The PostgreSQL database is provided and operated by Infosys.

Availability: RC1 (Continuity critical)

Database optimization: OLTP

Performance: Real time

RTO: 12 hours

RPO: max 24 hours

Size (max): 250 GB

During the first run of CDiS the tables/schemes are created and the database is ready for usage.

Docker

CDiS is provided as a Docker image via the Mercdes-Benz standard docker registry "Harbor" located at: https://registry.app.corpintra.net/harbor/cdis.

PULL command:

docker pull registry.app.corpintra.net/cdis/cdis-jvm:latest

(For other versions replace :latest with the desired version)

A persistent folder where CDiS shall store its log files needs to be mapped like this:

source: <path to logs external> (e.g. /srv/vsm/cdis/log)
target: /var/log/cdis

Otherwise CDiS will create the log files inside the container. Rights have to be given to the container to write in this folder.

A folder containing the custom **application.properties** file needs to be mapped like this:

source: <path to properties> (e.g. /srv/vsm/cdis/)
target: /deployments/quarkus-app/config

Editing the application.properties is explained in the following chapter (Properties / Configuration).

An empty **cdis.env** file has to be placed in the same folder.

A docker-compose file as used in the IS-Server environment is available from https://git.i.mercedes-benz.com/MO-VSM/CDiS/blob/develop/cdis-resources/docker/docker-compose.cdis-cdis.yml_v2

This compose file already includes the mapping of the folders, so they just have to be created and rights for the container to write (logs) needed.

Properties / Configuration

CDiS is configured from outside the Docker container via the application.properties file. The latest application.properties template can be found in CDiS GIT:

CDiS/application.properties at develop · MO-VSM/CDiS (mercedes-benz.com)

The values marked with <> need to be adapted!

For encrypting the passwords please contact CDiS support. (common-steganography tool needed. mode -m CDIS_ENCRYPT)

Last change: 29.03.2023

```
#Place this file anywhere on the disk under an empty directory then mount it to the docker machine like this:
#docker run ...  -v /absolutepath/cdis-config:/deployments/quarkus-app/config ...
#IS-Server has a docker-compose file prepared
```

```
# ports
quarkus.http.port = 8080

# OIDC Prod
quarkus.oidc-client.auth-server-url = https://sso.mercedes-benz.com/as/authorization.oauth2
quarkus.oidc-client.client-id = <oidc client-id>
quarkus.oidc-client.credentials.secret = <oidc client-secret>

dbUrl=jdbc:postgresql://<baseurl>:64001/<db-name>

dbFlywayUserName=<db admin user>
dbFlywayUserPass=<encrypted password>

dbRegularUserName=<db app user>
dbRegularUserPass=<encrypted password>

#flyway properties
quarkus.flyway.second.migrate-at-start = true
quarkus.flyway.second.baseline-on-migrate = true
quarkus.flyway.second.baseline-version=0
#quarkus.flyway.second.schemas=cdis

#use for testing only
quarkus.flyway.second.clean-at-start = false


#swagger
quarkus.smallrye-openapi.path=/cdis/swagger
quarkus.swagger-ui.path=/cdis/swagger/ui
quarkus.swagger-ui.always-include=true
quarkus.smallrye-openapi.security-scheme=basic

# PKI URLs
%dev.pki.base.url = https://localhost:8999
pki.base.url = https://am-pki-backend-int.app.corpintra.net
pki.ecu.identifiers.url = ${pki.base.url}/api/etm/ca/v1/cert/identifiers
pki.ecu.certificates.url = ${pki.base.url}/api/etm/ca/v1/cert
pki.ecu.certificates.info.url = ${pki.base.url}/api/etm/ca/v1/cert/info
pki.ecu.certificates.cas.url = ${pki.base.url}/api/etm/ca/v1/cas

# JWT
jwt.key=<private key>


#default scheduler expression for automatic importing of certificates
cron.expr.default=0 0 0/1 ? * * *


#config for automatic importing of public keys
pki.certs.update.identifier.items = 1000
pki.certs.update.identifier.parallel = 5
pki.certs.update.identifier.retries = 3
pki.certs.update.identifier.retries.delay = 1

#config automatic importing of certificates
pki.certs.update.cert.parallel = 5
pki.certs.update.cert.retries = 3
pki.certs.update.cert.retries.delay = 5
pki.certs.update.error.throttle = 50
pki.certs.update.error.maxretry = 20

#default scheduler expression for automatic deletion of old certificates
delete.certificates.cron.expr=0 0 0 ? * * *
delete.certificates.older.then.days=<180>
```

```
# logging
#General log
quarkus.log.level=INFO
#Package log
quarkus.log.category."org.hibernate".level=INFO
quarkus.log.category."com.daimler.cdis".level=INFO
quarkus.log.category."org.apache.http".level=INFO
quarkus.log.category."org.postgresql.jdbc".level=OFF
quarkus.log.min-level=DEBUG
#File log
quarkus.log.file.enable=true
quarkus.log.file.level=INFO
quarkus.log.file.format=[%d{MM-dd HH:mm:ss.SSS}] [%-5p] [traceId=%X{traceId}, parentId=%X{parentId}, spanId=%X
{spanId}, sampled=%X{sampled}] [%t] [%C] [%M}] %i %s%e%n
quarkus.log.file.path=/var/log/cdis/cdis.info.log
%dev.quarkus.log.file.path=C:/tmp/cdis.info.log
#Rotation
quarkus.log.file.rotation.file-suffix=-yyyy-MM-dd.zip
quarkus.log.file.rotation.max-file-size=30M
quarkus.log.file.rotation.max-backup-index=30

# API log handler rotation
quarkus.log.handler.file."API_LOGGING_FILE".rotation.max-file-size=30M
quarkus.log.handler.file."API_LOGGING_FILE".rotation.max-backup-index=365
quarkus.log.handler.file."API_LOGGING_FILE".rotation.file-suffix=-yyyy-MM-dd.zip

#Console Log
quarkus.log.console.enable=true
quarkus.log.console.format=[%d{MM-dd HH:mm:ss.SSS}] [%-5p] [traceId=%X{traceId}, parentId=%X{parentId}, spanId=%X
{spanId}, sampled=%X{sampled}] [%t] [%C] [%M}] %i %s%e%n
#HTTP Log
quarkus.http.access-log.enabled=false


#VDM
%dev.com.daimler.cdis.certificates.integration.VDMRestClient/mp-rest/url=https://localhost:8999/vdm/tester-
authentications
com.daimler.cdis.certificates.integration.VDMRestClient/mp-rest/url=https://<baseurl>/vds/api/vdm/vdm/tester-
authentications
com.daimler.cdis.certificates.integration.VDMRestClient/mp-rest/hostnameVerifier=io.quarkus.restclient.
NoopHostnameVerifier
#VDM basic auth credentials used for VDMRestClient
vmd.rest.client.user=<basic auth user>
vmd.rest.client.password=<basic auth pasword>

quarkus.tls.trust-all=true

#Basic authentication/authorization for testerAuthentications endpoint
quarkus.http.auth.basic=true
quarkus.security.users.embedded.enabled=true
quarkus.security.users.embedded.plain-text=true

#swagger user passwords
quarkus.security.users.embedded.users.user=<user pw>
quarkus.security.users.embedded.users.admin=<admin pw>
```

First run: During the first run the database tables and scheme are set up. The initial update will start according to the cron.expr.default value which is full hour as default. The first update run will then fetch all certificates from one day prior until the current time.

Administration

SwaggerUI

CDiS is only accessible via its REST-API. Tools like curl, Postman or any other REST client can be used, but the most convenient way is the SwaggerUI.

It is originally located at: <baseurl>:<port>/cdis/swagger/ui/ (Path can be differnt if modified via a reverse-proxy like traefik or nginx).

All endpoints are secured using BasicAuth. There are two users: "user" and "admin" with different roles:

- user = Access to all endpoints that need to be accessed from the testet device excluding the ones for maintenance (currently: /etmcertificates, /etmcertificatechains and /authorizetester).
- admin = All endpoints

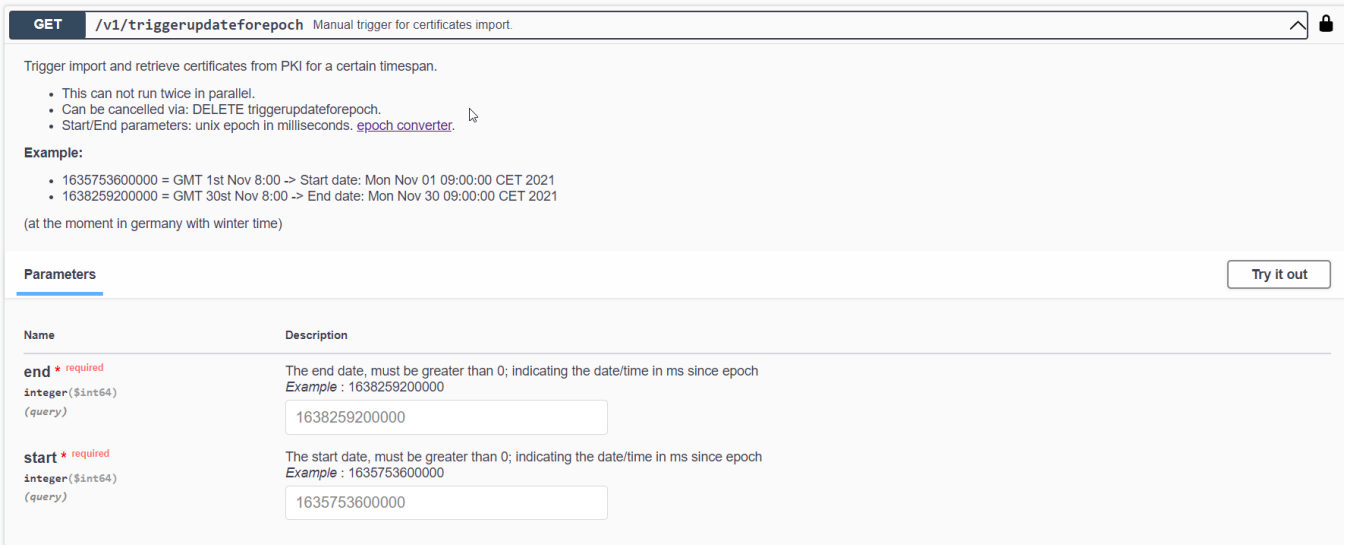
Although both users will work it is advised to use the "admin" user to access the SwaggerUI.

Initial load

After a new CDiS installation is set up and running an initial load of certificates needs to be triggered. This can be done using the endpoint /triggerupdateforepoch.

Just enter the start and end times and start the import by clicking on "Try it out". All certificates that were issued between the start and end times will be fetched.

There is a link to an online timestamp converter to assist creating the right timestamps.



Cronjobs

CDiS has 2 cronjobs running periodically:

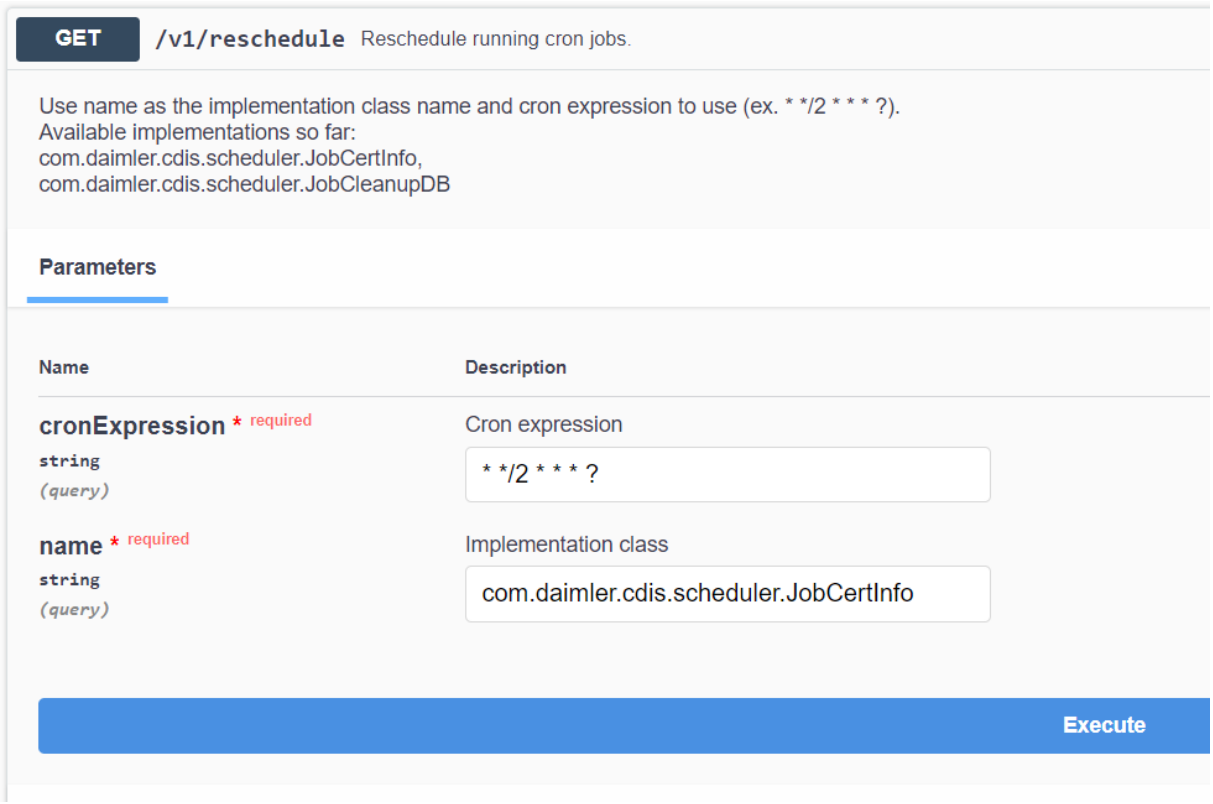
Update job (com.daimler.cdis.scheduler.JobCertInfo)

This job will retrieve all public keys from PKI for which a certificate was created since the last update run.

Cleanup job (com.daimler.cdis.scheduler.JobCleanupDB)

This job will delete all certificates older than the given number of days (standard value: 180, see application.properties parameter "delete.certificates.older.then.days")

Both can be parameterized using the REST-APIs endpoint /v1/reschedule. Best way is to use the SwaggerUI:



To calculate the desired cronExpression, this online tool can be used: https://www.freeformatter.com/cron-expression-generator-quartz.html

REST-API

CDiS REST-API offers the following endpoints. More detailed descriptions and usage see the SwaggerUI documentation.

Attention: The REST-API is protected by BasicAuth to prevent unauthorized access. Chapter "Properties Configuration" explains how to set the Users and Passwords.

There are two roles: User and Admin. The User role is used by technical clients while the Admin role is used by operators to use the SwagerUI.

Endpoint	Type	Role	Description
/v1/authorizetester	POST	User, Admin	Emergency process to verify a CSR submitted by a tester and forward it to PKI to issue a certificate
/v1/etmcertificatechains	GET	User, Admin	Certificate chain according to the partNumber submitted
/v1/etmcertificates	GET	User, Admin	Certificate according to the public key submitted
/v1/reschedule	GET	Admin	Reschedule the cronjobs (Delete, Update)
/v1/resetIdentifiersThreshold	GET	Admin	Reset all the identifiers that reached the maximum of retries in order to be retried on the next job execution
/v1/sysinfo	GET	Admin	Retrieve system settings/data
/v1/triggerupdateforepoch	GET	Admin	Manually trigger the update of certificates and retrieve certificates from PKI for a certain timespan
/v1/triggerupdateforepoch	DELETE	Admin	Delete the manually triggerd update of certificates (e.g. if it runs too long due to a wrong timespan chosen)

v1/verifyCsrAndSignature	POST	User, Admin	Verifies the existence of the submitted CSR and the validity of the signature (via submitted public key)

PKI API

Endpoint	Type	Description
/cert	GET	Get the certificate for a public key Example: https://am-pki-backend-int.app.corpintra.net/api/etm/ca/v1/cert?pubKey=38001c04f82447db8388277cd14ffb432f19edd413e690e7894bad7422a61244
/info	GET	Get all public keys created within a timespan Example: https://am-pki-int.es.corpintra.net/api/etm/ca/v1/cert/info?start=0&end=2635421260000&limit=100&page=0
/cert	POST	Emergency process. Submit CSR to get a certificate (JWT) Example Body: eyJhbGciOiJFUzI1NiJ9. eyJjc3IiOiJMUzB0TFMxQ1JVZEpUaUJEUlZKVVNVWkpRMEZVUlNCU1JWRlZSVk5VTFMwdExTMEtUVWxJYkUxSlIxbEJaMFZCVFVOT mVFbFVRV1pDWjA1V1FrRk5UVWRFUVhkTlJFMTNUVVJCZDAxRVFYZE5SRUYzVFVSQmQwMUVRWGROUkVGM1RVUkJjUXBOUVZWSFFYbDBiR05CVFdo QlNIbHBOelJCS3pOMlVVVkpSV3MxY1RsMU0zQnFibmxqUW10VGIzZHJVRTFNUjNwc2FWbHphVXNyUVc5RlNYZFJRVmxLQ2t0dldrbG9kbU5PUVZGclQw MVVUWGROVkVGMlFtY3dja0puUlVWQldscHJRWGRoUkdaUlZuTkNRalI0U0VGM1JsSkZNV1pTYTNkTlFsVlNUbGd3V2xNS1JFRldSVlJXT1ZOVVFYZEdVa1 V4WmxWc1NYZENVVmxFU3pKV2QwRXdSVUY2V2lzMEwxbHBkRTA0Y1RCTmNYTjJSakVyVkVOdFVHTm9XbmRFTTJ0MVJncFNRMnczUm14Mk56aFJkM mxsWVZCSlZFNUZaUzlFZEdnM1IzQllVMlJ2ZEVOalZ6SjZkRkZyUlZKVmQxQkJabVU0Wm5GRVJGRTlQUW90TFMwdExVVk9SQ0JEUlZKVVNVWkpRMEZV UlNCU1JWRlZSVk5VTFMwdExTMD0iLCJoYXJkd2FyZVBhcnROdW1iZXIiOiIxMjM0NTY3ODkwIn0. eM3bsFmgMcF_LY5zP4GKsrMNICzqCROFosefAcnm-ht1-nBXhh0ULImQWk9_hiMtiSLmLCBr6hJTqhijSGxAyw
/cas	GET	Get certificate chain (root, backend, intermediate) for a part number Example: https://am-pki-backend-int.app.corpintra.net/api/etm/ca/v1/cas?externalId=1234567890

PKI Rate Limits

ETM PKI API	Rate Limit per CDIS client	Description
CDiS Emergency	500 requests / min	Each CDiS client is allowed to request up to 500 emergency requests per minute.
CDiS single request for one ECU certificate (Normal process)	1000 requests / min	Each CDiS client is allowed to request up to 1000 ECU certificates per minute.
CDiS request list of public keys within time range (Normal process)	200 requests / min	Each CDiS client is allowed to request up to 200 paginated lists of Public Keys per minute. Hint: The request is paginated. Each page currently contain up to10.000 Keys (It is planned to limit to just 1000). For each next page an additional request is needed. CDiS then requests the certificates for the returned Public Keys in single requests. Currently the time range is unlimited - Limiting to one day is the aim.
CDiS request certificate chain	20 requests / min	Each CDiS client is allowed to request up to 20 certificate chains per minute. (as CDiS only needs to request a certificate chain one time this is more than enough)

VDM

Endpoint	Type	Description
/tester-authentications	GET	Get the public key for a hostname (where ZenZefi is running and the signature created) Example: <baseurl>/tester-authentications?hostname=m054yisg3s00029

ZenZefi

Endpoint	Type	Description
/signpayload	POST	Create a signature over the data provided with the ZenZefi private key Example: <baseurl>/signpayload Body; HELLOWORLD

Support

Monitoring

Logs: To observe CDiS logs a solution provided by its host environment is needed. For IS-Server instances it is the "OVO-monitoring".

Default log level: INFO

The log level can be changed using the application.properties (section #logging). Only switch to DEBUG for issue analysis.

Log files are rotated and zipped when switching to a new one. Default values are 30 files with 30MB max size. This can also be changed in the application.properties.

Metrics: CDiS provides a metrics endpoint at <baseurl>/q/metrics. It also provides a Grafana dashboard that can be imported into the host Grafana instance. It currently shows:

- the amount of imported certificates by the update cronjob
- how often the emergency process was executed
- // general system status / information was broken by changing the metrics provider (smallrye).

Error Handling Checklist

How to handle the following issues:

Issue	Symptoms	Reaction
No connection to PKI	Errors when retrieving certificates with errors indicating an issue on PKI side	Firewall open on port 443? PKI down? Network issue? Contact the PKI support (see chapter "Contacts")
No connection to GAS-OIDC	Errors when obtaining authorization tokens indicating errors on GAS-OIDC side	Firewall open on port 443? Check the GAS-OIDC status at IAM Status Page (corpintra.net) Contact the GAS-OIDC support (see chapter "Contacts")
Connection to CDiS endpoints refused	<title>401 Authorization Required</title>	Check BasicAuth settings in application.properties. Maybe user and/or password are not set correctly

Certificates can't be stored in DB	Certificates are not added to the table "certificate" but "identifier"	Check user access rights to DB DB full?
Scheduler not running	No new certificates are imported. No calls to PKI are done.	Check the settings under /sysinfo via SwaggerUI. Last run. Next Run. Check the DB table "qrtz_triggers" and the field "trigger_state". If BLOCKED it needs to be set to WAITING otherwise the scheduler won't run. Logs need to be analyzed why it failed. If it fails again logs are needed to see why. If the scheduler is not running contact the CDiS support.
Emergency process: Signature verification fails	{"errorMessage":"Signature is not valid."}	Check the testers public key and compare it to the VDM database. The tester needs to be coupled correctly. Check BasicAuth settings to VDM endpoint /tester-authentications in application.properties (#vdm basic auth).
Too many certificates issued via the emergency processes	The emergency process is done way too often leading to slow response times of CDiS.	Inform PKI contacts to improve supplier reliability. Monitoring on PKI side required.
Certificates could not be imported due to PKI downtime	There are no new certificates in db table "certificate" but public keys in the "identifier" table.	During the next update run the identifiers are tried again and their certificates will be imported if the PKI is available again. So the situation should be solved automatically. After x retries an identifier is set to "don't try again" status (must be reset manually by SQL).
PKI responds with error 500	API calls are answered with 500 and an error message	Contact the PKI support (see chapter "Contacts")

AMS

The following CDiS service levels are available for each plant:

Support Level	**Team**	**Task**	Availability	**Contact:** **E-Mail / Servicenow**
1st level support	Frontdesk / Data center	Monitors CDiS log files and tries to fix standard issues according to the "Error Handling Checklist".	24/7	Individually for each plant. Centrally monitored by Sindelfingen as well.
2nd level support	Support team (IS-Server MBRDI)	Handles all issues and incidents the 1st level support was not able to fix.	24/7	MBRDI Servicenow group VDS_SUPPORT Servicenow Ticket
3rd level support	CDiS AMS team (ITP/IE MBRDI)	Handles all issues and incidents the 1st and 2nd level support were not able to fix.	Office hours (UTC+5:30)	Servicenow: Request Something --> Something is broken --> “Application Service” --> “CDiS– Production” Servicenow Group: VSM_CDiS_L3

PKI Incident Process

If an error is related to PKI please follow the following INC ticket process or forward existing tickets to PKI_AMPKI_Support-ext

1. Go to https://mercedes-benz.service-now.com/
2. Select "Something is broken"
3. Select "My issue is related to: Application Service"
4. Dependent on the environment you face the incident:
    a. ETM PKI Integration Environment:
       Select "Choose affected Application Service: "PKI_AMPKI - Test"
    b. ETM PKI Production Environment:
       Select "Choose affected Application Service: "PKI_AMPKI - Production"
5. Select "Choose a Template: ETM Template"
6. Please answer the questions in the ETM Template as much as possible before sumitting your incident.
7. Prioritization: You can let us know how urgent your incident is but please be aware that we will not create Prio 1 Incidents based on Incidents reaching us via ServiceNow (such will only be created based on our internal Monitoring).

The following support elements are available in ServiceNow:

Element	Support Group
Service Groups	1st level Support to be provided by operator 2nd level Support: VDS_SUPPORT 3rd level Support: VSM_CDiS_L3
Issue related to	Application_Service
Application Service	CDiS - Production
Template	-

Ticket Impact	Description
1 – High	Production is disturbed because of CDiS. MB is losing money. Needs to be fixed asap.
2 – Medium	Production is not impacted yet, but will soon. If not fixed immediately MB will lose money.
3 – Low	Production is not impacted, but the issue needs to be analyzed and fixed anyway

Ticket Urgency	Description
1 – High	Needs to be fixed instantly.
2 - Medium	Needs to be fixed asap
3 – Low	Needs to be fixed in the next days

Contacts

Component	Responsible Team	Contact: E-Mail
Production assembly plants - IS-Server	ITO/CS Vehicle Diagnostic	Jan Kratt: jan.kratt@mercedes-benz.com
CDiS	ITP/IE	Mithun Shivalingaiah: mithun_kumar.shivalingaiah@mercedes-benz.com
PKI	ITT/SE	Oliver Burgmaier oliver.burgmaier@mercedes-benz.com Oliver Killguss oliver.killguss@mercedes-benz.com
GAS-OIDC		gas-oidc@mercedes-benz.com IAM Support (corpintra.net)