

Database

CX Core ART

Exported on 10/27/2022

Table of Contents

1	Data Layer.....	3
1.1	Introduction	3
1.1.1	Layer 1: Environment Configuration / Product Data.....	3
1.1.2	Layer 2: Master/Static Data Configuration.....	4
1.1.3	Layer 3: Configuration Data	4
1.1.4	Layer 4: Transactional Data	4
1.1.5	Summary	4
1.2	Classification of Data Objects.....	5
1.3	Layering Concept on Keycloak side	8
1.4	Releasing of Configuration Changes	9
1.5	ERM draft	10
1.6	OpenAPI Specification draft	10
2	DB Model.....	11
2.1	Entity Relationship Model (ERM)	11
2.2	Legacy	12
2.3	Focus Entity "App"	12
2.3.1	Get to the Swagger Docu	12
2.3.2	DB Model - Focus Apps.....	13
2.4	Focus Entity "Company".....	14
3	Audit logging	15
3.1	1. PostgreSQL Session Logging (pgAudit)	15
3.2	2. Application Audit Logging.....	15

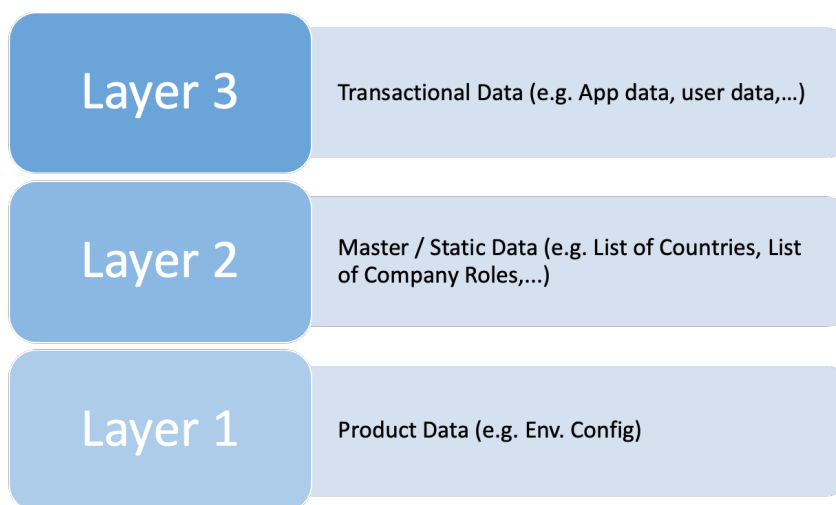
1 Data Layer

i Under creation - only a draft as starting point

- [Introduction](#)(see page 3)
 - [Layer 1: Environment Configuration / Product Data](#)(see page 3)
 - [Layer 2: Master/Static Data Configuration](#)(see page 4)
 - [Layer 3: Configuration Data](#)(see page 4)
 - [Layer 4: Transactional Data](#)(see page 4)
 - [Summary](#)(see page 4)
- [Classification of Data Objects](#)(see page 5)
- [Layering Concept on Keycloak side](#)(see page 8)
- [Releasing of Configuration Changes](#)(see page 9)
- [ERM draft](#)(see page 10)
- [OpenAPI Specification draft](#)(see page 10)

1.1 Introduction

The CX Portal & Network Services system stores different types of data in the same database: the master / static data, business base data, as well as the transaction data. Therefore, we need to classify all data / entities of CX into the following groups:



1.1.1 Layer 1: Environment Configuration / Product Data

The first layer contains the system configuration and holds all items that are required to run the application in an environment. The artifacts of layer 1 are automatically deployed to the different environments via the CI/CD pipeline. Layer 1 represents the base software, which is the same for all environments in all markets. However, the required system configurations may differ slightly from environment to environment and must therefore be managed by the CI/CD pipeline.

Layer 1 includes the following configuration items, but is not limited to these items: DB Setup, Network Paths, Authentication, Application Server Setup, Hostnames, Service Accounts, SSL Certificates.

1.1.2 Layer 2: Master/Static Data Configuration

The second layer contains all data that is required for the business base setup of the system.

Those data are part of the CI/CD deployment chain.

1.1.3 Layer 3: Configuration Data

??

1.1.4 Layer 4: Transactional Data

The fourth layer holds all transaction and master data, which is created on a daily basis in the system, such as users, apps and contracts. This data is always specific for one environment and is therefore not deployed by the CI/CD pipeline.

Transactional data is created by operating the system in a market or testing activities.

1.1.5 Summary

The table below summarizes the main aspects of the four data layers. Please note that the controls and approvals will be established for the the PROD / PRE-PROD environments after the initial go-live of TopGear.

Layer	Examples	Technical Controls & Approvals	Versioning	Technical Realization
1	DB Setup, Network Paths, Authentication, Application Server Setup, Hostnames, Service Accounts, SSL Certificates.	Requires a release. Control and Approvals are explained in ???	Version controlled by the platform's CI/CD pipeline	CI / CD pipeline is used for the transport. Items are configured and managed in Octopus Deploy.

Layer	Examples	Technical Controls & Approvals	Versioning	Technical Realization
2	Countries, currencies, vehicle types, roles, business units, GL accounts, financial product, workflows.		Version controlled by the platform's CI/CD pipeline	CI / CD pipeline is used for the transport. Items are configured in the Ambit Desktop Configuration Client.
3	--	-	-	-
4	Users, apps and contracts	Controls and approvals in context of daily business as part of the workflow configuration.	No, since changes are done directly in the production system, as part of daily operations. Audit trail will track the changes.	-

1.2 Classification of Data Objects

The following table classifies the main entities of the data dictionary into the layers 2-4. Layer 1 contains the software and plugins, which do not sit in the database. The table below features the following columns:

- **Core Data Objects:** Grouping of entities into the Core Data Objects.
- **Entities:** Entities of the Data Dictionary.
- **Data Layer:** Data Layer, according to the definition above. Only contains layers 2-4, since layer 1 does not sit in the database.
- **Creation / Configuration Strategy:** Explains, how the entity is created / configured.

Core Data Objects / Business Objects	Entities	Data Layer	Creation / Configuration Strategy
Company	Details	3	Created as part of the registration
	Address	3	Created as part of the registration
	BPN	3	Added along the registration process

	Status	2	Company Status Labels are deployed as Static Data
Consent	Consent Record	3	Whenever a consent is needed, a record will get stored
	Consent Status	2	Along with the consent record, the status is set
Application	Application Record	3	Created with the new CX Member invite
	Application Status	2	Status is set along with the application record
Agreement	Agreement	3	
	Agreement Category	2	Agreement Category ID/Labels are deployed as Static Data
Roles	App/User Roles	3	Created/Updated when a new service/app is getting registered
	Company Roles	2	Company Roles are deployed as Static Data (e.g. App Provider, Data Consumer, etc.)
	Company Role Description	2	Company Roles Description are deployed as Static Data (e.g. translation of the role and corresponding details)
	App/User Roles Description	3	Created as part of the app release process
Connector	Details	3	Created as part of the connector registration
	Status	2	Connector Status ID's/Labels are deployed as Static Data
	Type	2	Connector Type ID's/Labels are deployed as Static Data
Documents	Templates	3	
	Document	3	

	Status	2	Document Status ID's/Labels are deployed as Static Data
	Type	2	Document Type ID's/Labels are deployed as Static Data
Use Cases	-	2	Use Case ID's/Labels are deployed as Static Data
IAM	Clients	3	
	IdP Details	3	
	IdP Category	2	IdP Category ID's/Labels are deployed as Static Data
	Service Account	3	
	User	3	
Language	-	2	Languages are deployed as Static Data
Countries	-	2	Countries are deployed as Static Data
Service Account	Details	3	
	Roles	3	
	Status	2	Service Account Status IDs/Labels are deployed as Static Data
Invitation	Details	3	
	Status	2	Invitation Statis ID's/Labels are deployed as Static Data
Apps	Details	3	
	Images	3	
	Tags	3	

	Language	3	
	License	3	
	Description	3	
	Status	2	App Status ID's/Labels are deployed as Static Data
	Subscription Status	2	App Subscription Status ID's/Labels are deployed as Static Data
	Subscription Details	3	
User	Status	2	
	Account	3	
	Roles	3	User assigned roles
	App Favorites	3	
	User	3	

1.3 Layering Concept on Keycloak side

For keycloak, the concept is similar.

There is a base setup (provided as static data setting) as well as transactional data.

The difference between keycloak and portal is inside the load.

Its not planned to have soon any delta updates in keycloak regarding the base data. Off course there will changes in the initial load, but for a certain time those are planned to get added by the operator manually till an actual delta load solution is in place.

Initial data load in regards of the different instances

Central Keycloak Instance

- Realm: 'Master' & 'Operator'
- User: operator admin (master realm); CX Admins (operator realm)
- Clients:

ID	Name	Description	Grant Type
1	Cl1-CX-Portal	Portal client	Public
2	Cl2-CX-Registration	Registration client	Public
3	Cl3-CX-Semantic	Semantic Hub client	Public
4	Cl4-CX-DigitalTwin	Digital Twin client	Public
5	Cl5-CX-Custodian	Custodian wallet client	Confidential
6	Cl7-CX-BPDM	BPDM backend service (Golden Record)	Confidential
7	technical_roles_management	Client managing technical roles for central services	Bearer

- Technical Users:

ID	Name	Description	Service
1	sa-reg-1	Portal technical user to connect to central idp.	Central IdP
2	sa-reg-2	Portal technical user to connect to shared idp.	Shared IdP
3	sa-cl5-custodian-2	Portal technical user to connect to wallet.	Managed Wallets
4	sa-cl3-cx-1	Portal technical user to connect GitHub and Semantic Hub.	Semantic Hub

regarding the technical user setup as well as the function of an technical user, details are explained here [Technical User & Technical User Management](#)

1.4 Releasing of Configuration Changes

As explained above, there are different ways of controlling and approving changes. The following two mechanisms are applicable:

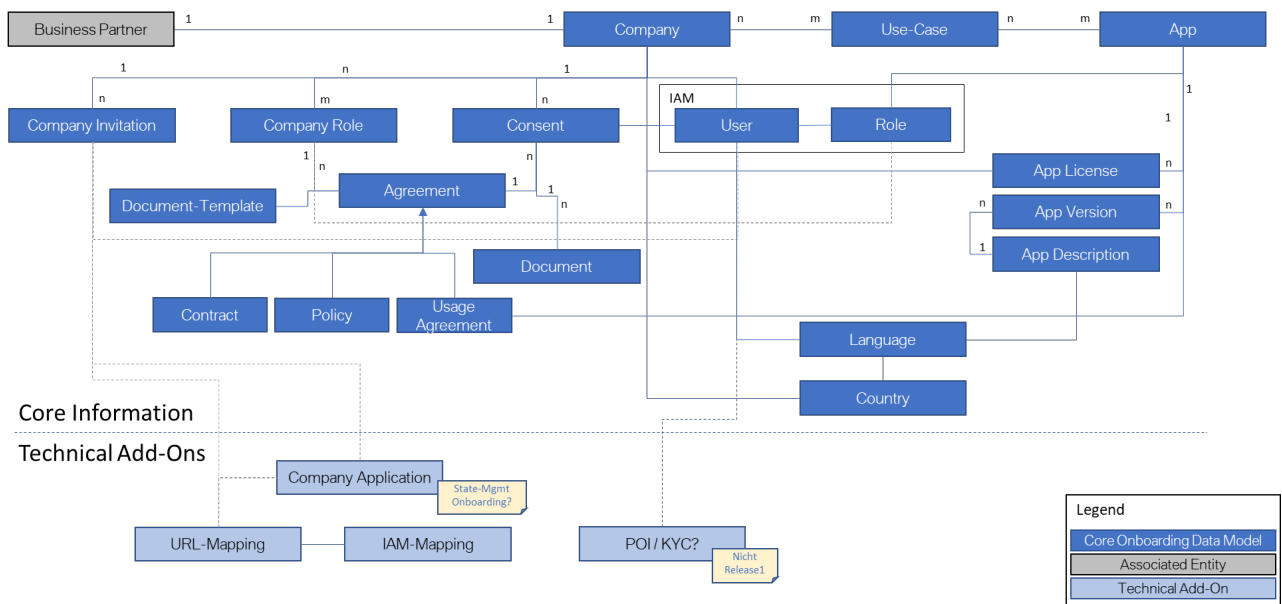
1. Regular Releases: Layer 1 & 2

2. Regular Operational Changes: Layer 4

1.5 ERM **DRAFT**

Current draft (work in progress) of the Entity Relationship Model (High Level)

Portal & Company Onboarding – ERM



1.6 OpenAPI Specification **DRAFT**

Specification of the data layer in OpenAPI

Swagger UI: <http://swagger servicedev.germanywestcentral.azurecontainer.io:8080/swagger-ui/#/Consent%20Entity%20Service>

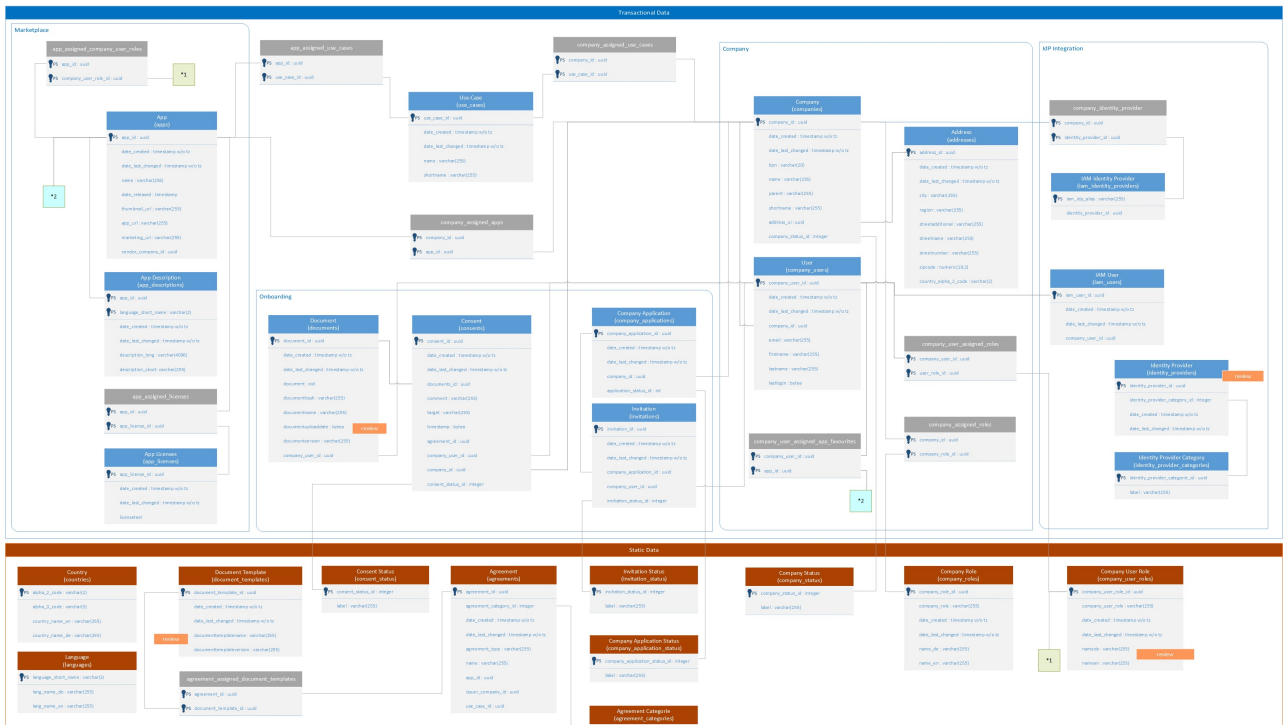
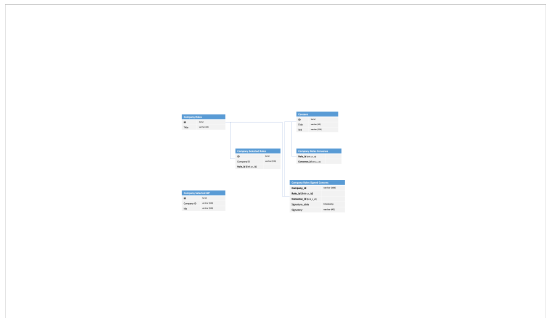
Version	Date	File
0.5	2022-02-17	CX_Onboarding_API_v0.5.yml
0.4	2022-02-17	CX_Onboarding_API_v0.4.yml

2 DB Model

2.1 Entity Relationship Model (ERM)

DEV (click to enlarge)

Old DB Model



Powerpoint (image)	Visio
CX_Portal_Release_1_ObjectModel_v0.1.pptx	CX_Portal_Realease1_ObjectModel_v0.1.vsd

Also see git for sql:

<https://github.com/tractusx-team-portal-onboarding/devenv/tree/master/postgres/init>

2.2 Legacy

DB dump catenaxdev003database:

- postgres
- iam
- iamcentralidp
- iamsharedidp
- gpdmdb
- tdmdb



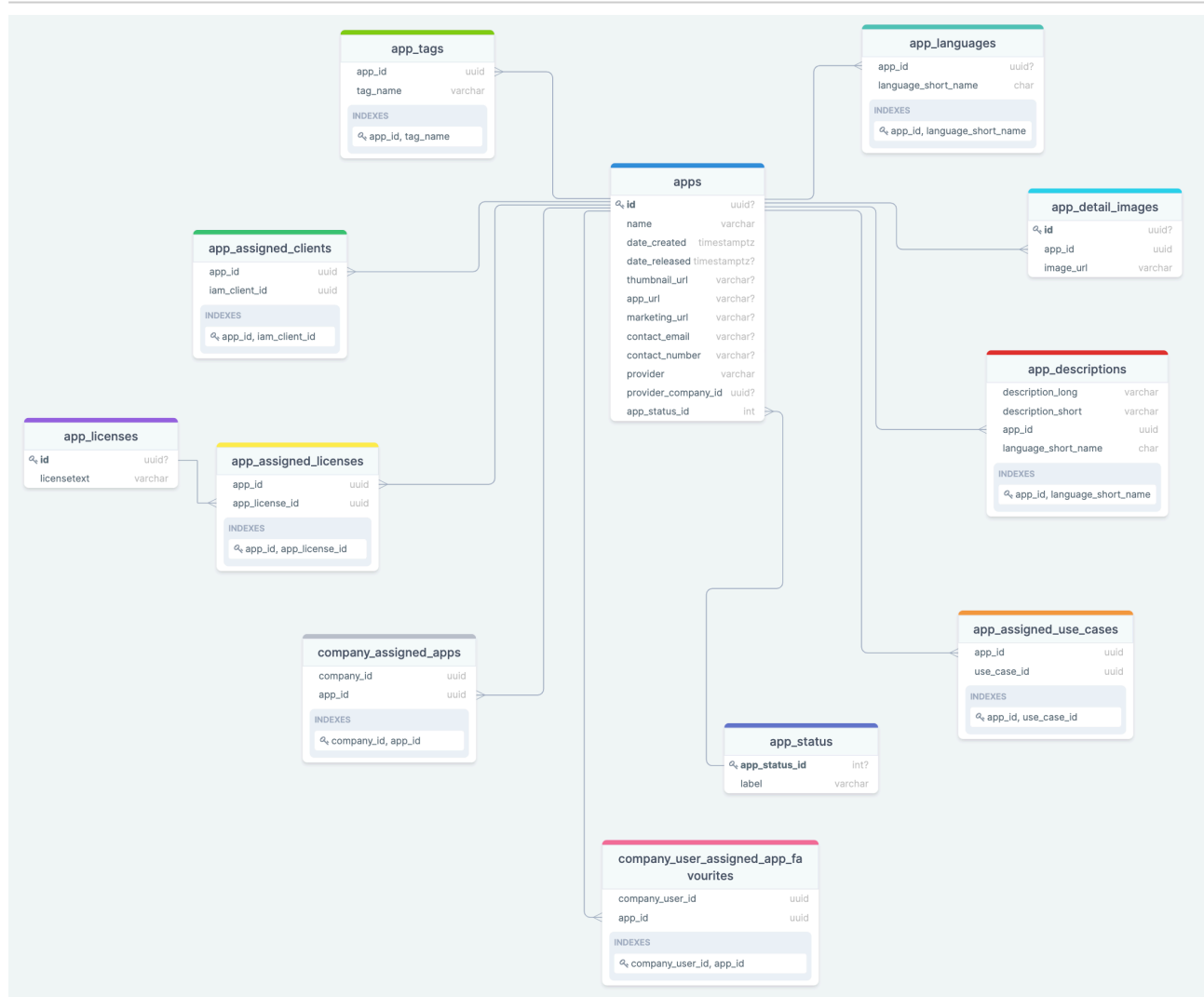
20220211_catena...003database.zip

2.3 Focus Entity "App"

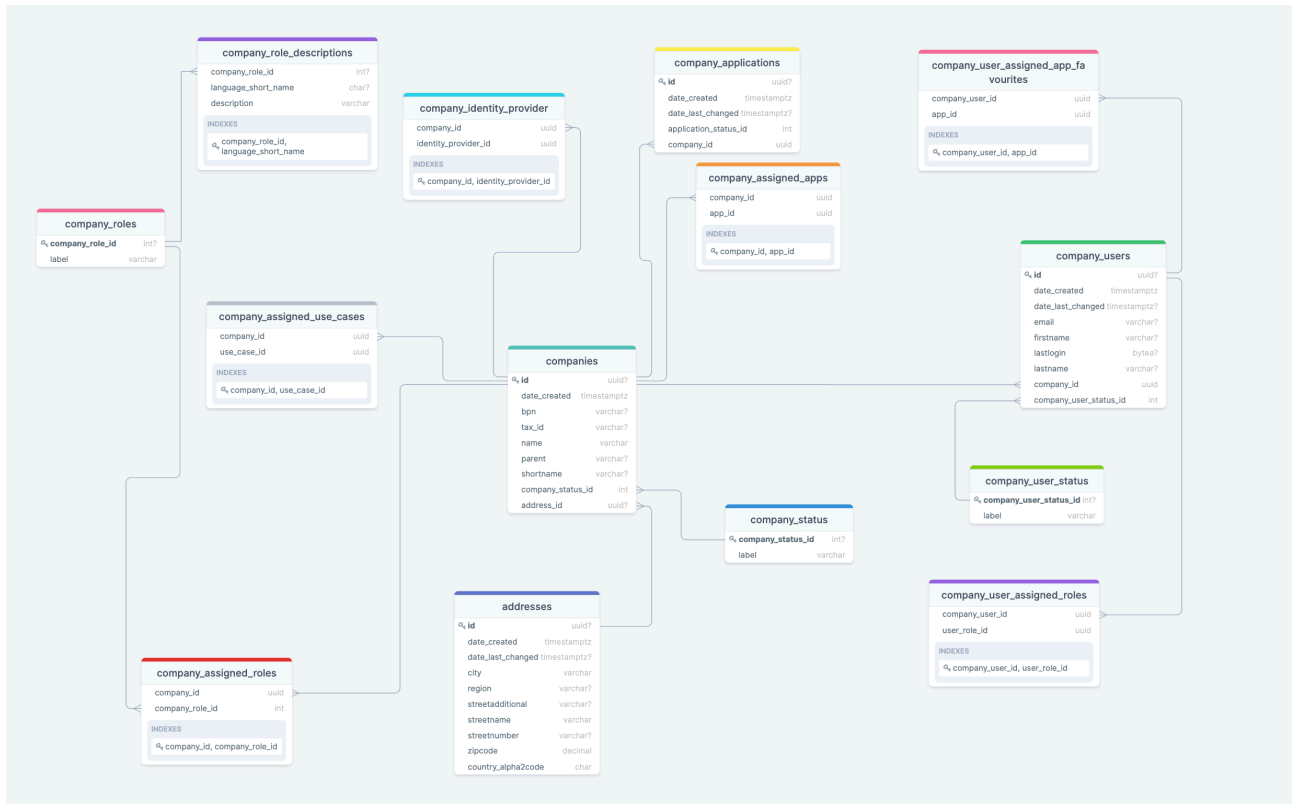
2.3.1 Get to the Swagger Docu



2.3.2 DB Model - Focus Apps



2.4 Focus Entity "Company"



3 Audit logging

For db audit logs, 2 generic approaches are used

1. PostgreSQL Session Logging (pgAudit)
2. Application Logging

3.1 1. PostgreSQL Session Logging (pgAudit)

PgAudit module is enabled for the portal db via the Bitnami PostgreSQL Image:

<https://github.com/bitnami/bitnami-docker-postgresql#auditing>

Following operations are audited: WRITE, DDL


WRITE: INSERT, UPDATE, DELETE, TRUNCATE, and COPY

DDL: CREATE, ALTER, and DROP schema objects

For a complete list of auditable operations see <https://github.com/pgaudit/pgaudit#pgauditlog>

In addition the parameters time stamp with milliseconds, user name and database name are configured as log line prefixes.

Future enhancements: Auditing logging on application side namely in the backend. The auditing on db level with pgAudit doesn't provide the information by which person specific user an operation was triggered because db operations are executed by our generic db users (for instance the 'portal').

 CPLP-884 - PostgreSQL DB Audit Logging **CLOSED**

3.2 2. Application Audit Logging

IN PROGRESS

Application Audit Logging is implemented to fulfill and enable extended audit log functionalities for the CX portal.

In the implementation the possibility of pgAudit extension and db table logging inside the postgresSQL was validated.

Due to the functional requirement to enable search, quick validations and audit traceability, the in-DB audit implementation got preferred.

Implementation: PI#5

What: for each audit relevant table, an audit_tablename_version table is created and storing any original table INSERT, DELETE or UPDATE records. Important, we always store:

- Who changed
- What
- When

Why we use versioning for the audit tables?

To ensure that an audit table is never modified, but only entries are added, we have introduced versioning for audit table.

Each audit table has version suffix in the name, this corresponds to the name of the migration in which the audit table was added.

See the example below for further information:

How to enable an entity to be auditable in the code (Example: CompanyUser)

1. The auditable entity needs to implement interface 'IAuditable'

```
public class CompanyUser : IAuditable
```

2. Create Audit entity which should derive from the auditable entity and implement `IAuditEntity`

```
public class AuditCompanyUser : CompanyUser, IAuditEntity
```

3. Add DbSet in `PortalDbContext` for the newly created Entity

```
public virtual DbSet<AuditCompanyUser> AuditCompanyUsers { get; set; } = default!;
```

4. Add `.ToTable` configuration to the `OnModelCreating` for the auditable entity

```
entity.ToTable("company_users")
```

5. Add `modelBuilder.Entity` configuration to the `OnModelCreating` for the audit table ignore all virtual properties of the base entity. !IMPORTANT! -> The tablename of the audit table must exactly match the name of the migration in snakecase and begin with `audit_`.

```
modelBuilder.Entity<AuditCompanyUser>(x => {
    x.HasBaseType((Type?)null);
    x.Ignore(x => x.Company);
    x.Ignore(x => x.Consents);
    x.Ignore(x => x.Invitations);
    x.Ignore(x => x.UserRoles);
    x.Ignore(x => x.CreatedNotifications);
    x.ToTable("audit_company_users_cp1p_1254_db_audit");
});`
```


6. Add Migration as described in the readme.md in

`CatenaX.NetworkServices.PortalBackend.Migrations` the name must match the table name of the audit table

```
20220802122302_CPLP-1254-db-audit
```

7. Add the AuditTrigger to the created migration. Add

`migrationBuilder.AddAuditTrigger<T>("versionName")` at the end of the Up method and

`migrationBuilder.DropAuditTrigger<T>()` to the beginning of the Down method. T should be the Audit Entity and the version should equal the backpart of the name of the migration without timestamp as snakecase.

Up:

```
migrationBuilder.AddAuditTrigger<AuditCompanyUser>("cplp_1254_db_audit");
```

Down:

```
migrationBuilder.DropAuditTrigger<AuditCompanyUser>();
```

Additional relevant information:

⚠ If not already existing in the original table, a `uuid` and `last_editor_id` attribute need to get added to the original id.

⚠ whenever the original table attributes are changed (e.g. adding an attribute or deleting an attribute) the already existing audit table will be set to frozen and a new audit table is getting created. All new audit relevant records will be stored in the new audit table

⚠ CPLP-1249 - DB Auditing - Enable Audit Management Concept & Implementation Rules **CLOSED**

⚠ CPLP-1394 - DB Auditing - Audit Table Implementation (User roles, Roles, Application) **CLOSED**

⚠ CPLP-1395 - DB Auditing - Audit Table Implementation (...) **OPEN**