CoMPAS Architecture

Interested in contributing? Please read carefully the CONTRIBUTING guidelines. Refer to GLOSSARY for technical terms and acronyms.

This page contains an overview overview of the ComPAS architecture. More details can be found on the ComPAS architecture website:

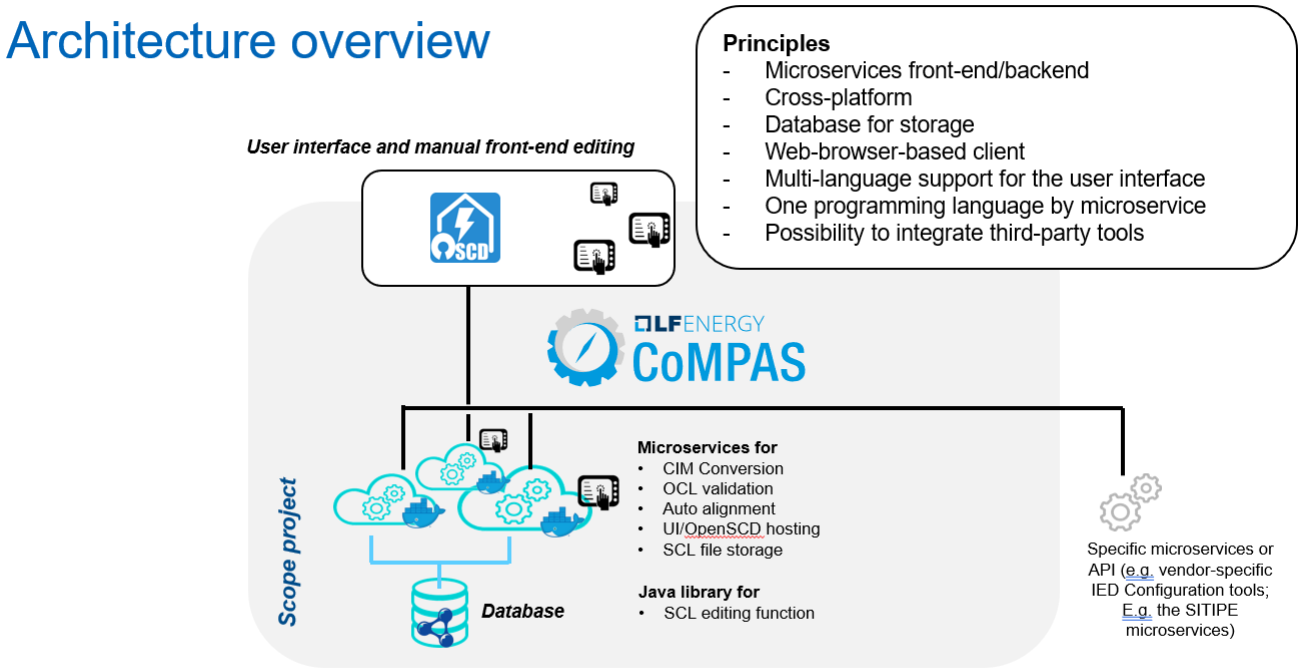
https://com-pas.github.io/compas-architecture/

Principles for Overall Architecture

The project will adopt the following principles for the overall software architecture:

- A **modular** architecture based on **microservices** facilitating a tailor-made integration of the building blocks in third-party IT systems
- A **cross-platform** software ensuring compatibility with various end-user environments.
- **Scriptable functional modules** to enable the customization/automatization of the configuration process according to the specificities of users
- **Light-weight web-browser-based client**
- **Multi-language support** for the user interface

Architecture Overview



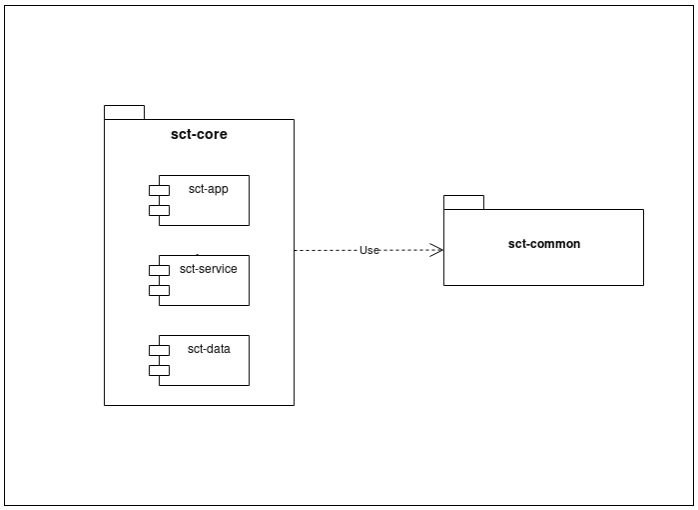
https://github.com/com-pas/contributing/blob/main/roadmap-docs/CoMPAS%20architecture%20overview%202023.pptx

Functional architecture

Overall functional block diagram

blocked URL

The diagram below shows the functional block diagram structure and their interactions with the deployment requirements. It is composed of two packages: sct-core and sct-common (which could be replaced by compas-core depending on the choice of code hosting)



The requirements are constructed into ease of implementation, deployment, integration and adaptability with respect to the tools.

System Configuration Tool (SCT) functional diagram

blocked URL

System Configuration Tool (SCT) is part of CoMPAS open source project. It goal is to bring a flexible and adaptative tool for configuring PACS (Power Automation and power Control System). It's an n-tiers architecture which combine reliability, flexibility, modularity and adaptability to allows users to choose their own database to implement the SCT.

The following architecture (package diagram) is divided in 3 major parts :

- sct-app : hosts the controller part which expose application resources on apis (API REST), it allows the sct application to be interfaced directly with web application or services which need to communicate with the SCT.
- sct-service : this part could be considered as the engine of the SCT. It computes all needed operations and uses sct-data for database access.
- sct-data : implements a generic abstract data service which should be extended by specific real data service for exchange with chosen database.
- database : this part allows user to store their data, for each database chosen sct-data should be implemented by real sct-data-typedb service in order to have compatible repositories for database exchange. It's not part of sct architecture.

Compas SCT (System Configuration Tool) is dedicated to manage automatization of binding signals for 61850 standard. It allows third parts to realize their system configuration. This tool plays a capital role on managing devices configuration and communication on power systems.

The architecture of the SCT is oriented microservices and present advanced abstraction level in order to facilitate usage by third parties.

It's design choice is to allow every user to re-implement the tool (data access part) according to his own needs. This architecture simplify usage and decrease contraints on database type to use and service aor controller re-adaptation for each need.

The component diagram shows two parts : sct-core and sct-common.

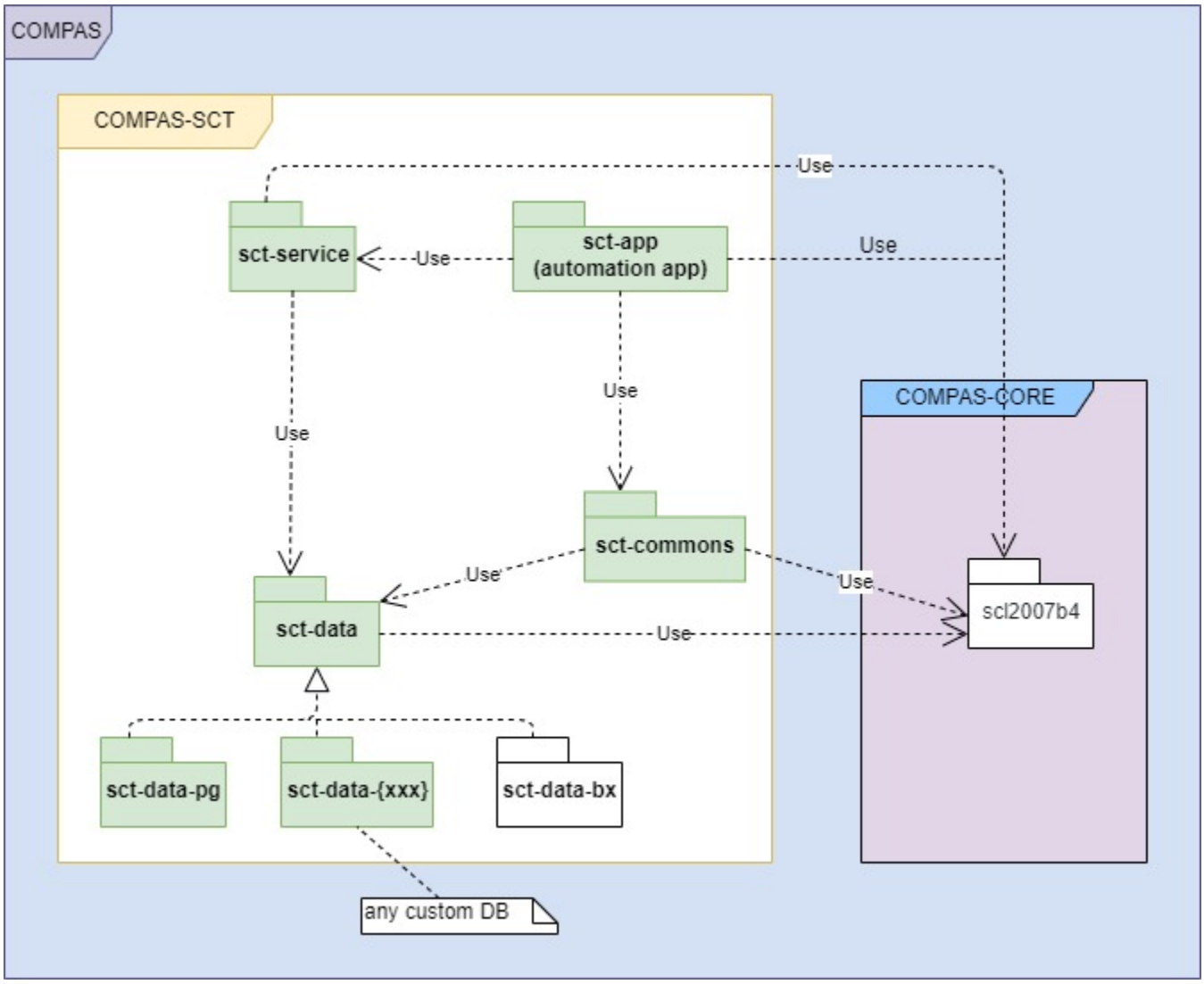
- sct-core : is the central part of the SCT. It implements all needed services to build and compute system files (SCD, SSD, etc). It also contains an abstract data module (implementation shown in code) that should be re-implemented by third part users in order to adapt the tool on their context. An application module is also available to allow communication with external components (HMI or services). In this way each vendor can use the tools with it's own tools to take plenty of SCT functionalities.
- sct-common : used to optimize and bring together all identified common or reusable parts of the code or functionality. It can be sct own common elements or directly common elements regrouped in compas-core as an external dependency (module) containing all common elements used in compas tool (sct, compas-cim-mapping, compas-scl-xsd, etc).

This model diagram describes the n-tiers architecture chosen for SCT implementation. It is composed by 2 parts : sct-core and sct-common. The sct-core part is devided into 3 layers : sct-app (controller), sct-service (service) and sct-data (data access abstract layer).

sct-app or controller layer and sct-service are concrete in opposite of sct-data which is abstract in order to allow it's re-implementation by concrete methods for user purpose and context.

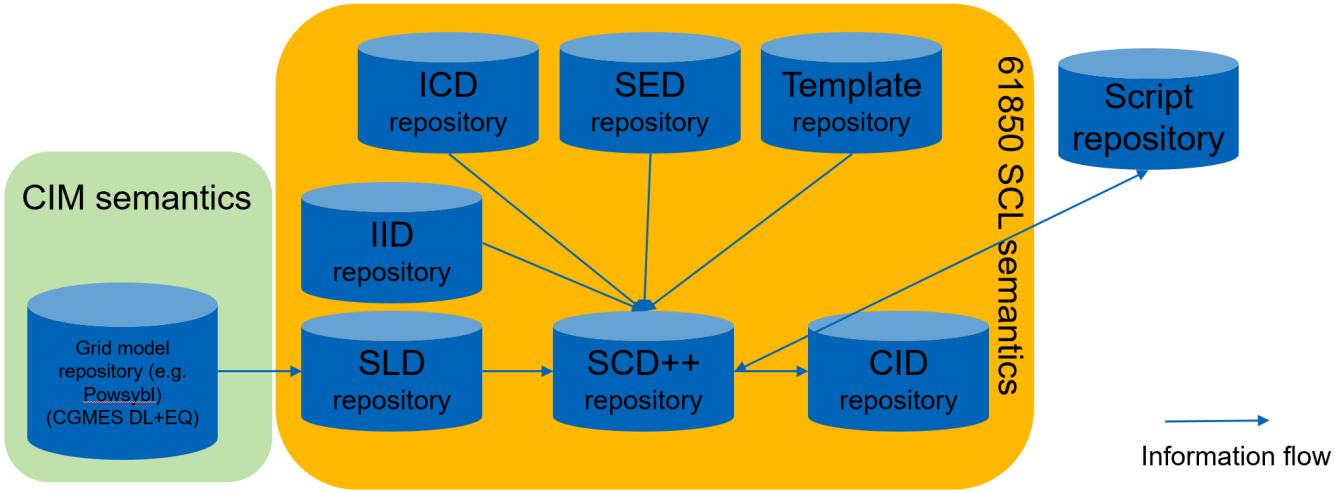
The source code is in Java, Spring Boot and hosts in compas space on GitHub. The build process and IC/DC are realized with Github Actions and allows contributor to construct a valuable product.

The component/package diagram allows to see the high modularity of the SCT architecture which gives a choice to build all components at the same time or build them one by one which may contribute off smart managing off package versions.



Data Architecture

Conceptual Overview of Data Architecture Repositories and Semantics



Tools

Fossoly is used by LF energy to scan on license compliance.

LFX security is used for security scanning in the code.