D6.2

Define and development of SCDB input and output standards and interfaces

Project short name
SUNRISE

Project full name
Safety assUraNce fRamework for connected, automated mobIlity SystEms

Horizon Research and Innovation Actions | Project No. 101069573
Call HORIZON-CL5-2021-D6-01

Funded by the European Union

ccam-sunrise-project.eu/

Dissemination level	Public (PU) - fully open
Work package	WP6: Data framework design and usage definition
Deliverable number	D6.2: Define and development of SCDB input and output standards and interfaces
Deliverable responsible	Eren Mungan, AVL TR (AVL)
Status - Version	Final – V1.0
Submission date	22/07/2025
Keywords	Data Framework, Scenario Database, Input Output, Ontology

Authors/Contributors

Role	Name
Main author	Eren Mungan (AVL TR)
Contributing authors	Marcos Nieto, Juan Diego Ortega Valdivieso (Vicomtech) Jobst Beckmann (RWTH IKA) Doha Khtatba, Chaima Tlili (VEDECOM)

Quality Control

	Name	Organisation	Date
Peer review 1	Stefan de Vries	IDIADA	08/07/2025
Peer review 2	Eloy Collado	IDIADA	11/07/2025

Version history

Version	Date	Author	Summary of changes
0.1	06/07/2025	Eren Mungan	First draft of the document
0.2	08/07/2025	Stefan de Vries, Eloy Collado	Review of the first draft
0.3	16/07/2025	Eren Mungan	Second version after review
0.4	21/07/2025	Eren Mungan	Addition of new requirements

0.5	22/07/2025	Eren Mungan	Final version candidate after review
1.0	22/07/2025	Eren Mungan	Final version

Legal disclaimer

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Climate, Infrastructure and Environment Executive Agency (CINEA). Neither the European Union nor the granting authority can be held responsible for them.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS AND ACRONYMS

Abbreviation	Meaning
API	Application Programming Interface
ASAM	Association for Standardizations of Automation and Measuring Systems
CCAM	Cooperative Connected and Automated Mobility
D	Deliverable
DF	Data Framework
DTO	Data Transfer Object
EC	European Commission
EU	European Union
Euro NCAP	European New Car Assessment Programme
JSON	JavaScript Object Notation
ODD	Operational Design Domain
OSX	ASAM OpenSCENARIO Package
RE	Requirement
SAF	Safety Assurance Framework
SCDB	Scenario Database
SUNRISE	Safety assUraNce fRamework for connected, automated mobIlity SystEms
UI	User Interface
URI	Uniform Resource Identifier
WP	Work Package

EXECUTIVE SUMMARY

Safety assurance of Cooperative, Connected, and Automated Mobility (CCAM) systems is a crucial factor for their successful adoption in society, yet it remains a significant challenge. It is generally acknowledged that for higher levels of automation, the validation of these systems by conventional test methods would be infeasible. Furthermore, certification initiatives worldwide struggle to define a harmonized safety assurance approach enabling massive deployment of CCAM systems.

The SUNRISE project develops and demonstrates a CCAM Safety Assurance Framework (SAF). The overall objective of the SUNRISE project is to accelerate the large-scale and safe deployment of CCAM systems. In alignment with international twin projects and initiatives, the project aims to achieve this objective by providing a SAF consisting of three main components: a Method, a Toolchain and a Data Framework. The Method is established to support the SAF safety argumentation, and includes procedures for scenario selection, sub-space creation, dynamic allocation to test instances and a variety of metrics and rating procedures. The Toolchain contains a set of tools for safety assessment of CCAM systems, including approaches for virtual, hybrid and physical testing. The Data Framework provides online access, connection and harmonization of external Scenario Databases (SCDBs), allowing its users to perform query-based extraction of safety relevant scenarios, allocation of selected scenarios to a variety of test environments, and reception of the test results.

This deliverable presents the definition and development of input and output standards and interfaces. It provides a comprehensive overview of the design and implementation of the SUNRISE Data Framework (DF) and its integration with external Scenario Databases (SCDBs).

A key objective of this deliverable is to define the functional architecture and implementation details of the SUNRISE DF, including the standard file formats and methodologies employed. Additionally, it addresses the requirements established in previous deliverables and work packages, explaining their impact on the implementation. The deliverable also summarizes the alignment results and outcomes derived from various work packages and deliverables.

For SCDB hosts, this document serves as a guideline to develop the necessary functions for full integration with the SUNRISE DF. In particular, ontology-related information and the methodologies applied for scenario filtering are emphasized as critical components.

The deliverable acts as a foundational reference for partners and external stakeholders. It provides SCDB hosts with clear and detailed instructions on how to leverage the described methodologies and processes to integrate their existing scenario databases with the SUNRISE DF.

Overall, the SUNRISE project strives to establish a comprehensive Safety Assurance Framework and a federated scenario databases approach (by means of the SUNRISE DF) that will enhance the validation and safety assurance of CCAM systems, promote harmonisation, and facilitate the safe deployment of these technologies.

1 INTRODUCTION

1.1 Project introduction

Safety assurance of Connected, Cooperative, and Automated Mobility (CCAM) systems is a crucial factor for their successful adoption in society, yet it remains a significant challenge. CCAM systems need to demonstrate reliability in all driving scenarios, requiring robust safety argumentation. It is acknowledged that for higher levels of automation, the validation of these systems by means of real test-drives would be infeasible. In consequence, a carefully designed mixture of physical and virtual testing has emerged as a promising approach, with the virtual part bearing more significant weight for cost efficiency reasons.

Worldwide, several initiatives have started to develop test and assessment methods for Automated Driving (AD) functions. These initiatives already transitioned from conventional validation to a scenario-based approach and combine different test instances (physical and virtual testing) to avoid the million-mile issue.

The initiatives mentioned above, provide new approaches to CCAM validation, and many expert groups formed by different stakeholders, are already working on CCAM systems' testing and quality assurance. Nevertheless, the lack of a common European validation framework and homogeneity regarding validation procedures to ensure safety of these complex systems, hampers the safe and large-scale deployment of CCAM solutions. In this landscape, the role of standards is paramount in establishing common ground and providing technical guidance. However, standardising the entire pipeline of CCAM validation and assurance is in its infancy, as many of the standards are under development or have been very recently published and still need time to be synchronised and established as common practice.

Scenario Databases (SCDBs) are another issue tackled by several initiatives and projects, that generally tends to silo solutions. A clear concrete approach should be used (at least at European level), dealing with scenarios of any possible variations, including the creation, editing, parameterisation, storing, exporting, importing, etc. in a universally agreed manner.

Furthermore, validation methods and testing procedures still lack appropriate safety assessment criteria to build a robust safety case. These must be set and be valid for the whole parameter space of scenarios. Another level of complexity is added, due to regional differences in traffic rules, signs, actors and situations.

Evolving from the achievements obtained in HEADSTART and taking other project initiatives as a baseline, it becomes necessary to move to the next level in the development and demonstration of a commonly accepted **Safety Assurance Framework (SAF)** for the safety validation of CCAM systems, including a broad portfolio of Use Cases (UCs) and comprehensive test and validation tools. This will be done in **SUNRISE**, which stands for **S**afety ass**U**ra**N**ce f**R**amework for connected, automated mobIlity **S**yst**E**ms.

The SAF is the main product of the SUNRISE project. As the following figure indicates, it takes a central role, fulfilling the needs of different automotive stakeholders that all have their own interests in using it.

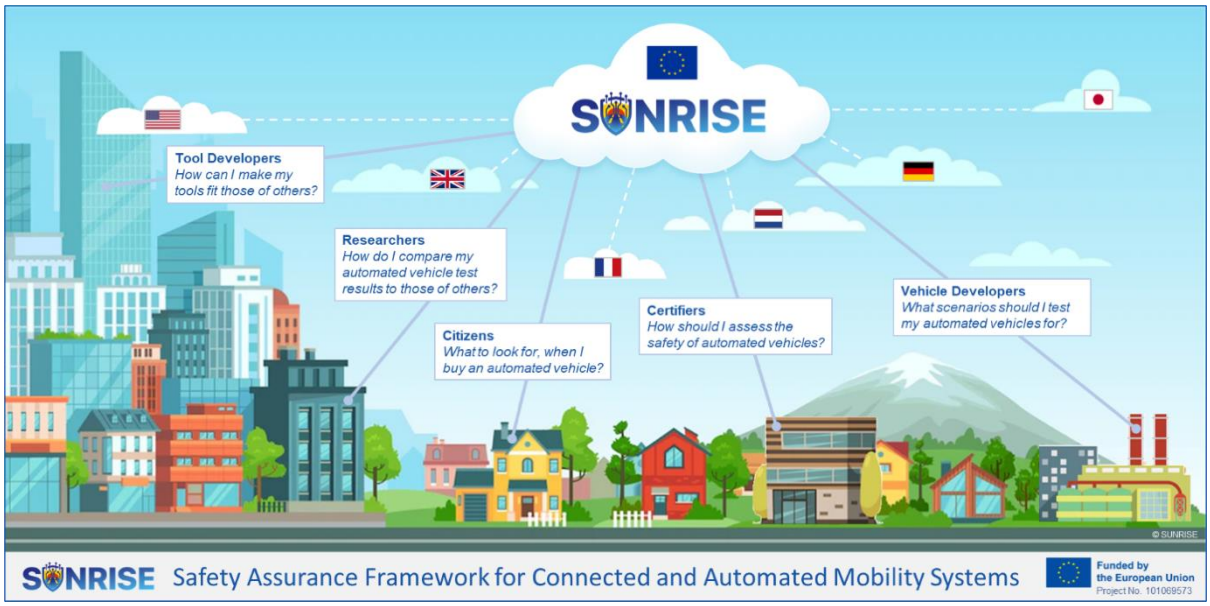


Figure 1: Safety Assurance Framework stakeholders

The **overall objective** of the SUNRISE project is to accelerate the safe deployment of innovative CCAM technologies and systems for passengers and goods by creating demonstrable and positive impact towards safety, specifically the EU's long-term goal of moving close to zero fatalities and serious injuries by 2050 (Vision Zero), and the resilience of (road) transport systems. The project aims to achieve this objective by providing a SAF consisting of three main components: a Method, a Toolchain and a Data Framework. The **Method** is established to support the SAF safety argumentation, and includes procedures for scenario selection, sub-space creation, dynamic allocation to test instances and a variety of metrics and rating procedures. The **Toolchain** contains a set of tools for safety assessment of CCAM systems, including approaches for virtual, hybrid and physical testing. The **Data Framework** provides online access, connection and harmonization of external Scenario Databases (SCDBs), allowing its users to perform query-based extraction of safety relevant scenarios, allocation of selected scenarios to a variety of test environments, and generation of the test results. The SAF will be put to the test by a series of **Use Cases demonstrations**, designed to identify and solve possible errors, gaps and improvements to the underlying methods, tools and data.

Following a common approach will be crucial for present and future activities regarding the testing and validation of CCAM systems, allowing to obtain results in a standardised way, to improve analysis and comparability, hence maximising the societal impact of the introduction of CCAM systems.

The following figure shows the general workplan of the SUNRISE project.

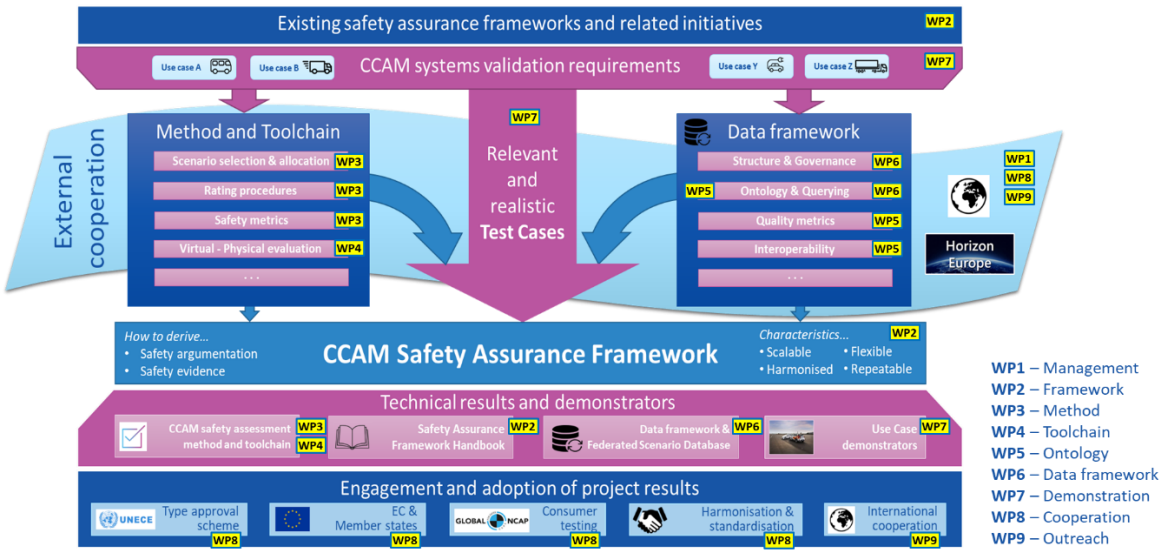


Figure 2: Workplan of the SUNRISE Project

1.2 Context within Work Package 6

This deliverable is part of the outcomes of the activities carried out in Work Package 6 (WP6) of the SUNRISE project.

WP6 focuses on the development of the SUNRISE Data Framework (DF), as the federation layer that permits access, connection, and harmonisation of external and distributed Scenario Databases (SCDBs).

In the broader context of the SUNRISE Safety Assurance Framework (SAF), the SUNRISE Data Framework (DF) implements the “Store” and “Query & Concretise” blocks, as can be seen in the following figure.

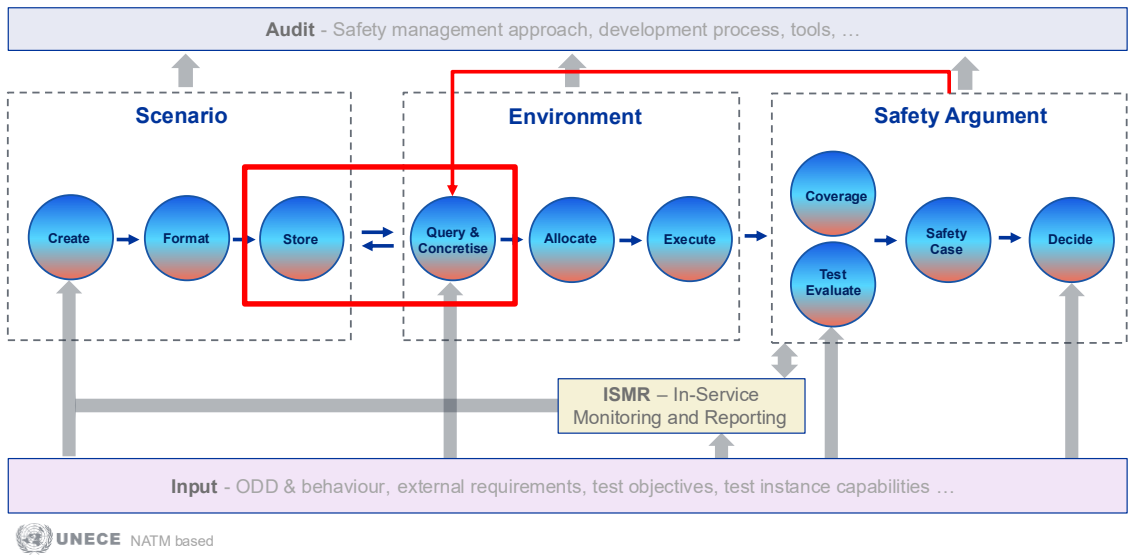


Figure 3: The SUNRISE SAF with the SUNRISE DF marked in red.

WP6 establishes the design to start the developments and implementation of the SUNRISE DF and thereby materialize the federation layer. This includes the temporary **storage** of scenarios retrieved from SCDBs through ontology-based **queries**.

It is of utmost importance to clarify the difference and relation between the concept of SCDBs and SUNRISE DF. According to the terminology of the SUNRISE project (included in the Grant Agreement, and the SUNRISE Glossary):

- **Scenario Database (SCDB):** An individual database of test scenarios (a description of a temporal and spatial traffic constellation). Existing examples of scenario databases are Safety Pool™, Streetwise, Scenius, Adscene, Scenario.Center.

- **SUNRISE Data Framework (SUNRISE DF):** The SUNRISE DF refers to the implementation of the data management layer of the Safety Assurance Framework (SAF), with special focus on management of access to SCDBs. The SUNRISE DF creates a federated set of services to interact with existing SCDBs, allowing accessing scenario data from a centralized and harmonized entry point. The interaction with the connected SCDB is done to guarantee governability of data owners and their business models. The SUNRISE DF addresses the requirements of the SAF regarding interfacing with SCDBs (formatting, query capabilities, access and authorization control, etc.). The concept of federation layer implies that data is stored and distributed in different SCDBs, which is managed by 3rd party entities connected to the SUNRISE DF. Data processing services will be developed in the SUNRISE DF to harmonize queries, explore different SCDBs, analyse search results, and prepare scenario content for its utilisation in the SAF Toolchain.

To better understand the scope of this deliverable, it is useful to review the structure of the activities of WP6. In a nutshell, WP6 is organized in four tasks groups:

- T6.1 Develop standards on scenario input to SCDB - focuses on input standards and data flows into SCDBs.
- T6.2 Develop standards output from SCDBs - focuses on output standards and data flows from SCDBs.
- T6.3 SCDB application for generic use cases - focuses on use cases and utilisation of SUNRISE DF with testing environments.
- T6.4 Sample case of a European Test Case Library - focuses on user access, federation layer, and implementation of the SUNRISE DF.

Therefore, within WP6, the activities carried out related to SCDB input and output standards are reported in this deliverable (D6.2 Define and development of SCDB input and output standards and interfaces).

The draft functional architecture of SUNRISE DF is proposed in D6.1 (Methodology for SCDB application for generic use cases) and the final architecture with defined functionalities, standard interfaces and data formats are described in this deliverable [1].

1.3 Purpose of deliverable

The purpose of this deliverable is to present and describe the "Define and development of SCDB input and output standards and interfaces ". Its aims to describe the process of SCDB integration into SUNRISE DF. This process includes integrating SCDBs, querying SCDBs based on defined search criteria and receiving dashboard information.

The key outcome of this deliverable is the defining a standardized process and data formats required to establish connection between SUNRISE DF and SCDBs. This outcome ensures that SCDBs are seamlessly incorporated, enabling consistent and efficient safety assurance across components of the SUNRISE SAF and CCAM systems in general.

This deliverable also outlines the use of ontology and its integration with scenario metadata to support data enrichment by completing and classifying it. The proposed ontology-based querying approach enables the identification of critical edge-case scenarios through the construction of comprehensive and semantically rich queries.

Although inputting scenarios into SCDBs was an original objective, it was deprioritized in favour of querying and retrieval, leaving the push-to-SCDB feature unimplemented and deferred to future work.

This deliverable serves as a reference for the developments made within the WP6 and interfaces for SUNRISE DF. D6.2 is being released at the same time as D6.3 which includes technical details of SUNRISE DF [2].

1.4 Intended audience

The intended audience of this deliverable is primarily the **WP5**, **WP6**, **WP7** as it will describe how a SCBD shall be integrated in SUNRISE DF for scenario query and dashboarding purposes. However, since this deliverable contains overall architecture of SUNRISE DF, it is also relevant for the rest of the consortium and WPs.

Importantly, this deliverable is also of significant interest to external stakeholders, particularly SCDB hosts. As these stakeholders are responsible for maintaining and managing scenario databases, the methodologies and processes outlined in this document offer them valuable guidance on how to align their systems with the SUNRISE DF. This alignment will enable SCDB hosts to ensure compatibility and interoperability within the SAF and with CCAM systems' safety assurance requirements, thus playing a vital role in the broader deployment of these technologies.

1.5 Deliverable structure and relation to other parts of project

The contents of this deliverable are divided in the following chapters:

Chapter 2: SUNRISE Data Framework Functional Architecture chapter describes the functional architecture resulting from all storylines and the user workflows, that are implemented in SUNRISE DF. These architectures include data flows and shows the main principles of the SUNRISE DF.

Chapter 3: Data Formats chapter describes the data formats used to exchange information between SUNRISE DF and SCDBs. Usage of these formats and modifications made on standards are also explained in detail.

Chapter 4: Conclusions. It describes the summary of the main findings of the deliverable.

Annex 1: *User Workflow for Querying Scenario Databases*. This workflow describes how a user can query scenario databases by using ontology via Search UI.

1.5.1 Link to Task T4.4

Deliverable D4.4, produced under Task T4.4, presents a comprehensive analysis of Validation and Verification (V&V) frameworks across various projects and stakeholders. The study highlights that open-source standards developed by ASAM are widely adopted by SUNRISE partners and the other stakeholders.

Based on these findings, it was concluded that ASAM standards, such as OpenLabel, OpenSCENARIO, and OpenDRIVE, should be utilized within the proposed harmonized V&V Simulation Framework. Accordingly, the design and implementation of the SUNRISE Data Framework (DF) are based on these formats for querying SCDBs and retrieving scenario packages.

Specifically, OpenLabel is used to exchange ontology information between the SUNRISE DF and SCDBs, while OpenSCENARIO and OpenDRIVE are employed to obtain the content of scenario packages.

1.5.2 Link to Task T5.1

The deliverable D5.1 resulting from task T5.1, presents a set of comprehensive requirements for the SUNRISE DF and federated scenario databases [3]. For this deliverable D6.2, the requirements related to the SUNRISE DF implementation have been used on how to define user operations, Query Manager, Search UI and Dashboard. Requirements related to SUNRISE DF implementation is listed below. **D5.1 Description** refers to original requirement taken from D5.1 while **D6.2 Remark** explains its detail and usage.

- **RE1 D5.1 Description:** The data system shall allow to create, read, update and delete user roles and profiles.
  RE1 D6.2 Remark: The data system shall support the creation, reading, updating, and deletion of user roles and profiles, including access rights and action history.
- **RE2 D5.1 Description:** The data system shall apply user rights to scenarios and other content as defined in the underlying scenario databases.
  RE2 D6.2 Remark: The data system shall enforce user rights, ensuring users can only access their licensed scenarios and databases.
- **RE3 D5.1 Description:** The data shall allow to record, inspect and delete user history.
  RE3 D6.2 Remark: The data system must allow for the recording, inspection, and deletion of user history for quality control and to comply with the right to be forgotten.

- **RE4 D5.1 Description:** The Data Framework shall allow querying of the various underlying SCDBs for retrieval of scenario data.
  RE4 D6.2 Remark: The Data Framework shall be capable of querying various underlying scenario databases to retrieve scenario data.
- **RE5 D5.1 Description:** The API of the Data Framework shall be designed so that implementation at consumer side takes less than a week of work.
  RE5 D6.2 Remark: The API of the Data Framework must be designed for easy and rapid implementation by consumer-side software engineers, taking less than a week.
- **RE6 D5.1 Description:** The API shall be based on REST-API.
  RE6 D6.2 Remark: The API shall be based on modern standards, specifically a REST-API, to ensure state-of-the-art and standardized communication.
- **RE7 D5.1 Description:** The Data Framework shall provide a graphical user interface (GUI) that allows searching of scenarios.
  RE7 D6.2 Remark: The Data Framework shall provide a graphical user interface (GUI) to allow users to search for scenarios.
- **RE8 D5.1 Description:** The Data Framework shall provide a GUI that allows exporting scenarios.
  RE8 D6.2 Remark: The Data Framework shall provide a GUI that enables users to export selected scenarios.
- **RE9 D5.1 Description:** The Data Framework shall provide data visualisations.
  RE9 D6.2 Remark: The Data Framework shall include easily understandable data visualizations, such as statistics and graphs, for both contained and extracted data.
- **RE10 D5.1 Description:** The Data Framework shall provide data statistics.
  RE10 D6.2 Remark: The Data Framework will provide detailed statistics on accessible and exported data, including scenario counts, parameter distributions, and ODD coverage.
- **RE11 D5.1 Description:** The Data Framework shall include a data characterisation system.
  RE11 D6.2 Remark: The Data Framework shall include a data characterization system to provide an overview of the connected databases, including data size and contributor information.
- **RE12 D5.1 Description:** Scenarios should be characterised based on their quality.
  RE12 D6.2 Remark: Scenarios should be characterized by a quality metric based on the quality of their parameter ranges and transferability to test cases.
- **RE13 D5.1 Description:** The Data Framework shall allow revisions in the data processing chain.
  RE13 D6.2 Remark: All components of the data processing chain within the Data Framework must be designed to accommodate future updates and revisions.
- **RE16 D5.1 Description:** The Data Framework shall have a check for ensuring all data of the scenario elements are present during download.
  RE16 D6.2 Remark: The Data Framework shall include a check to ensure that all essential data for scenario elements is present during the download process.
- **RE17 D5.1 Description:** The data framework shall facilitate the retrieval of various metrics of a scenario (category) if available in scenario database across different data sources within a specific ODD.

RE17 D6.2 Remark: The data framework shall facilitate the retrieval of various available metrics for a scenario category from different data sources within a specific ODD.

- **RE18 D5.1 Description:** The Data Framework shall provide an option for users to store their query and query results.
  RE18 D6.2 Remark: The Data Framework will provide an option for users to store their queries and the corresponding results for traceability and reproducibility.
- **RE19 D5.1 Description:** The Data Framework may contain scenarios for testing connectivity and cyber-security aspects.
  RE19 D6.2 Remark: The Data Framework may contain scenarios specifically designed for testing vehicle connectivity and cybersecurity aspects.
- **RE21 D5.1 Description:** The Data Framework shall track with which version data of the Data Framework and related toolchain was created/modified.
  RE21 D6.2 Remark: The Data Framework must track the specific versions of the framework and toolchain that were used to create or modify any data.
- **RE22 D5.1 Description:** The Data Framework shall track the version of the scenario database.
  RE22 D6.2 Remark: The Data Framework shall track the version of the scenario databases to ensure compatibility and accurate data retrieval through any changes.
- **RE23 D5.1 Description:** The Data Framework shall assign the corresponding revision version for the copied scenario.
  RE23 D6.2 Remark: When a scenario is copied, the Data Framework shall assign it a corresponding revision version to maintain a clear version history.
- **RE24 D5.1 Description:** The Data Framework shall allow different types of files attachments.
  RE24 D6.2 Remark: The Data Framework will allow for the attachment of various unprocessed file types, such as pictures and videos, to scenarios.
- **RE25 D5.1 Description:** The data framework should present the scenarios to the user by type, application and certain characteristics.
  RE25 D6.2 Remark: The data framework should present scenarios to the user organized by type, application, and other characteristics, potentially using a hierarchical or tagged structure.
- **RE26 D5.1 Description:** The input format (from SCDBs to the data framework) shall be compatible with agreed standardised formats.
  RE26 D6.2 Remark: The input format for data from scenario databases into the framework must be compatible with agreed-upon standardized formats.
- **RE27 D5.1 Description:** The output format of the data framework shall be compatible with other relevant formats including ASAM.
  RE27 D6.2 Remark: The output format of the data framework must be compatible with other relevant formats, including industry standards like ASAM OpenDRIVE and OpenSCENARIO.
- **RE28 D5.1 Description:** The Data Framework shall allow querying of scenarios via API and GUI.
  RE28 D6.2 Remark: The Data Framework shall support the querying of scenarios from underlying databases through both a GUI and an API.

- **RE29 D5.1 Description:** The Data Framework shall enable querying similar scenarios from different databases using the same query.
  RE29 D6.2 Remark: The Data Framework will enable users to query for similar scenarios across different databases using a single, consistent query.
- **RE30 D5.1 Description:** The Data Framework shall be able to query and filter based on agreed taxonomy (meta data.).
  RE30 D6.2 Remark: The Data Framework shall be able to query and filter scenarios based on an agreed-upon taxonomy of metadata.
- **RE31 D5.1 Description:** The Data Framework shall filter scenarios based on ODD and OD.
  RE31 D6.2 Remark: The Data Framework will provide the capability to filter scenarios based on the Operational Design Domain (ODD) and Operational Domain (OD).
- **RE32 D5.1 Description:** The Data Framework shall filter based on omissions (NOT statement).
  RE32 D6.2 Remark: The Data Framework shall allow users to filter scenarios by excluding certain elements using a "NOT" statement in their queries.
- **RE33 D5.1 Description:** The result of a query with the Data Framework shall be reproducible.
  RE33 D6.2 Remark: The results of any query performed with the Data Framework must be reproducible, possibly by refining the query with additional parameters like a date range.
- **RE34 D5.1 Description:** The Data Framework shall respect the usage of raw personal information according to national and international privacy laws.
  RE34 D6.2 Remark: The Data Framework will adhere to national and international privacy laws like GDPR and will only accept anonymized scenario data.
- **RE35 D5.1 Description:** Copyrights and licenses must be present and visual for the user with respect to all data sources in the pipeline.
  RE35 D6.2 Remark: Copyright and license information for all data sources must be clearly visible to the user in the GUI and included in any exported files.
- **RE36 D5.1 Description:** Data framework shall comply with relevant cybersecurity best practices and standards.
  RE36 D6.2 Remark: The Data Framework must comply with relevant cybersecurity best practices and standards, including access control for authentication and authorization.
- **RE37 D5.1 Description:** The Data Framework shall be hosted in a secure cloud environment.
  RE37 D6.2 Remark: The Data Framework shall be hosted in a secure cloud environment to ensure reliability, flexibility, and manageable costs.
- **RE38 D5.1 Description:** The data framework shall contain scenario meta information, based on agreed taxonomy (meta data)
  RE38 D6.2 Remark: The data framework's output data must include scenario meta-information, such as data source and timestamp, based on an agreed taxonomy.
- **RE39 D5.1 Description:** Copyright shall be part of the output data, based on agreed taxonomy.
  RE39 D6.2 Remark: Copyright and license information for the data source shall be included as part of the output data, potentially as comments in output files.

- **RE40 D5.1 Description:** The data framework should include the definition of mandatory and optional parameters based on agreed taxonomy/ontology.
  RE40 D6.2 Remark: The data framework should define mandatory and optional parameters, along with their naming conventions, based on an agreed taxonomy or ontology.
- **RE47 D5.1 Description:** The Data Framework shall be compatible with the following databases: Safety PoolTM Scenario Database, Scenius, Moove, ADScene, VVMethods, and Streetwise.
  RE47 D6.2 Remark: The Data Framework must be compatible with and able to read data from specified databases, including Safety Pool™ Scenario Database and AVL Scenius™.
- **RE48 D5.1 Description:** The required and optional parameters/fields shall be defined.
  RE48 D6.2 Remark: The required and optional parameters or fields for a scenario shall be clearly defined to accommodate varying relevance across different scenarios.
- **RE50 D5.1 Description:** The Data Framework shall allow definition of parameter ranges and/or parameter value sets.
  RE50 D6.2 Remark: The Data Framework will allow for the definition of parameter ranges and/or sets of parameter values to enable the creation of logical scenarios.
- **RE51 D5.1 Description:** The scenarios shall include the probabilities for the different parameter ranges/sets, if available
  RE51 D6.2 Remark: Scenarios shall include, if available, the probabilities for their different parameter ranges or sets to allow for prioritized testing.
- **RE53 D5.1 Description:** Scenarios shall contain a human-readable description.
  RE53 D6.2 Remark: All scenarios must contain a brief, human-readable description so users can understand their content.
- **RE54 D5.1 Description:** All scenarios shall have the required fields filled with relevant data.
  RE54 D6.2 Remark: All required fields for scenarios must be filled with relevant data to ensure that they can be queried in a consistent manner.
- **RE55 D5.1 Description:** Logical scenario shall cover a variety of concrete scenarios.
  RE55 D6.2 Remark: Logical scenarios should be defined in a way that they cover a variety of concrete scenarios, either through parameter ranges or multiple parameter sets.
- **RE58 D5.1 Description:** The output file(s) shall satisfy the requests of the user for specific version and content and must be valid regarding version and content.
  RE58 D6.2 Remark: The output files must be valid and satisfy the user's request for specific versions and content, adhering to specified OpenSCENARIO versions.
- **RE59 D5.1 Description:** The data framework shall check if the output of scenario of individual databases are in agreed format/language(s) Any scenario description language may be acceptable at SCDB level as long as the SCDB owner provides means to convert/translate or adapt to the harmonised SUNRISE scenario language(s).
  RE59 D6.2 Remark: The data framework shall ensure that the output of individual databases is in an agreed-upon standardized format, with database owners providing conversion means if necessary.

- **RE60 D5.1 Description:** The data framework shall enable scenarios with different formats that are: human-readable, executable and retrievable from database, if scenarios have these format.
  RE60 D6.2 Remark: The data framework will enable scenarios to be available in different formats that are human-readable, executable, and retrievable from the database.
- **RE61 D5.1 Description:** The data framework should enable scenario definition to have optional capabilities to handle V2X, if scenarios have these format.
  RE61 D6.2 Remark: The data framework should enable scenario definitions to have optional capabilities to handle V2X (Vehicle-to-Everything) communications if available.
- **RE62 D5.1 Description:** All data should be traceable to its origin (original input data).
  RE62 D6.2 Remark: All data processed by the Data Framework must be traceable back to its original source scenario database.
- **RE63 D5.1 Description:** The data framework shall allow feeding of data coming from various data sources.
  RE63 D6.2 Remark: The data framework shall allow for the feeding of data from various sources to enable data enrichment.
- **RE64 D5.1 Description:** The data Framework shall allow flexibility depending on end user of “results”.
  RE64 D6.2 Remark: The Data Framework shall be flexible to accommodate different end-user needs and safety assessment requirements.
- **RE65 D5.1 Description:** The data framework shall contain a labelled scenario database. SCDBs connected to the data framework shall contain labelled scenarios
  RE65 D6.2 Remark: The data framework will contain a labeled scenario database, with connected databases also containing scenarios with descriptive tags and labels.
- **RE66 D5.1 Description:** The data framework may be able to auto validate scenarios based on ODD tags and behaviour tags (for scenario parameters) defined by ASAM OpenLabel tags, based on agreed ontology
  RE66 D6.2 Remark: The data framework may have the capability to automatically validate scenarios based on ODD and behavior tags defined by an agreed-upon ontology like ASAM OpenLabel.
- **RE71 D5.1 Description:** The hosting method of the data framework shall be defined.
  RE71 D6.2 Remark: The hosting method for the data framework must be defined, whether it will be in a cloud, on specific servers, or another method.
- **RE72 D5.1 Description:** Incentives for stakeholders shall be considered.
  RE72 D6.2 Remark: The incentives for stakeholders to use the data framework instead of directly accessing specific databases should be considered and made clear.
- **RE73 D5.1 Description:** The usage for scientific stakeholder shall be defined.
  RE73 D6.2 Remark: The access rights and usage policies for scientific stakeholders shall be clearly defined, specifying if they differ from industrial users.
- **RE74 D5.1 Description:** The legal form of the data framework shall be defined.
  RE74 D6.2 Remark: The legal form under which the data framework will be operated, such as a company or an association, must be defined.

Although D5.1 proposes additional requirements, some were not considered during the design phase, either due to their limited relevance to the SUNRISE DF implementation or because

they were revised during the implementation phase based on technical discussions between stakeholders and SCDB owners.

These requirements are referenced throughout the deliverable wherever they are satisfied.

1.5.3 Link to Task T5.2

The SUNRISE Ontology, developed as part of Task T5.2, provides a structured classification and specification system for scenario tags. These tags are used to describe the scenario's design domain, participating actors, and their behaviours. In addition to defining the tags themselves, the ontology also captures the relationships and constraints between them, effectively serving as a design guideline for the SUNRISE Data Framework (DF).

The content and methodology established in Task T5.2 have been extensively integrated into the development of the SUNRISE DF, particularly in the design of workflows used to query Scenario Databases (SCDBs).

Based on an analysis of existing and previous ontologies, it was determined that the application should not be rigidly tied to a single ontology. Instead, the ontology should be treated as an interchangeable component. As a result, the ontology used in the SUNRISE DF has been implemented as a modular and replaceable element.

1.5.4 Link to Task T5.3

Deliverable D5.3, an output of task T5.3, provides a comprehensive report on scenario database quality metrics. This document encompasses extensive research findings pertaining to these metrics, serving as a foundational guideline for their implementation.

The SUNRISE DF implementation leverages these scenario database quality metrics to extract critical information regarding scenario content from designated databases. Consequently, the insights presented in D5.3 are instrumental in the design and implementation phases of SUNRISE DF. Additionally, onboarded Scenario Database (SCDB) owners utilize this information to ensure data quality and relevance.

Deliverable D5.3 categorizes and defines scenario database quality metrics across various bases, acknowledging that metric applicability varies depending on the specific database content. Therefore, SUNRISE DF is engineered not to ensure the perpetual availability of quality metrics, but rather to present existing and relevant metrics to the user.

1.5.5 Link to Deliverable D6.1

Deliverable D6.1, produced under Task T6.3, presents the foundational research and design principles for the SUNRISE Data Framework (DF). This deliverable elaborates on key concepts and proposes detailed storylines and workflows corresponding to the various user operations within SUNRISE DF.

Storylines #1, #2, and #3 are directly related to the SUNRISE DF and provide a solid foundation for its implementation, whereas Storyline #5 is indirectly related. An overview of these storylines is provided in Chapter 2 of this deliverable.

Storyline #4 concerns the process of pushing scenarios to the SCDBs. While this storyline is described in detail in Deliverable D6.1 and its workflow is presented, it was ultimately deprioritized. After thorough evaluation and discussions within the working group, querying scenario databases and retrieving scenario packages were identified as having greater value and priority. As a result, the functionality for pushing scenarios into SCDBs was not implemented and has been designated as future work.

2 SUNRISE DF FUNCTIONAL ARCHITECTURE

2.1 Introduction

This chapter describes the functional architecture of SUNRISE Data Framework, its components and connection to individual SCDBs. Figure 4 below shows the overall architecture of SUNRISE DF components.

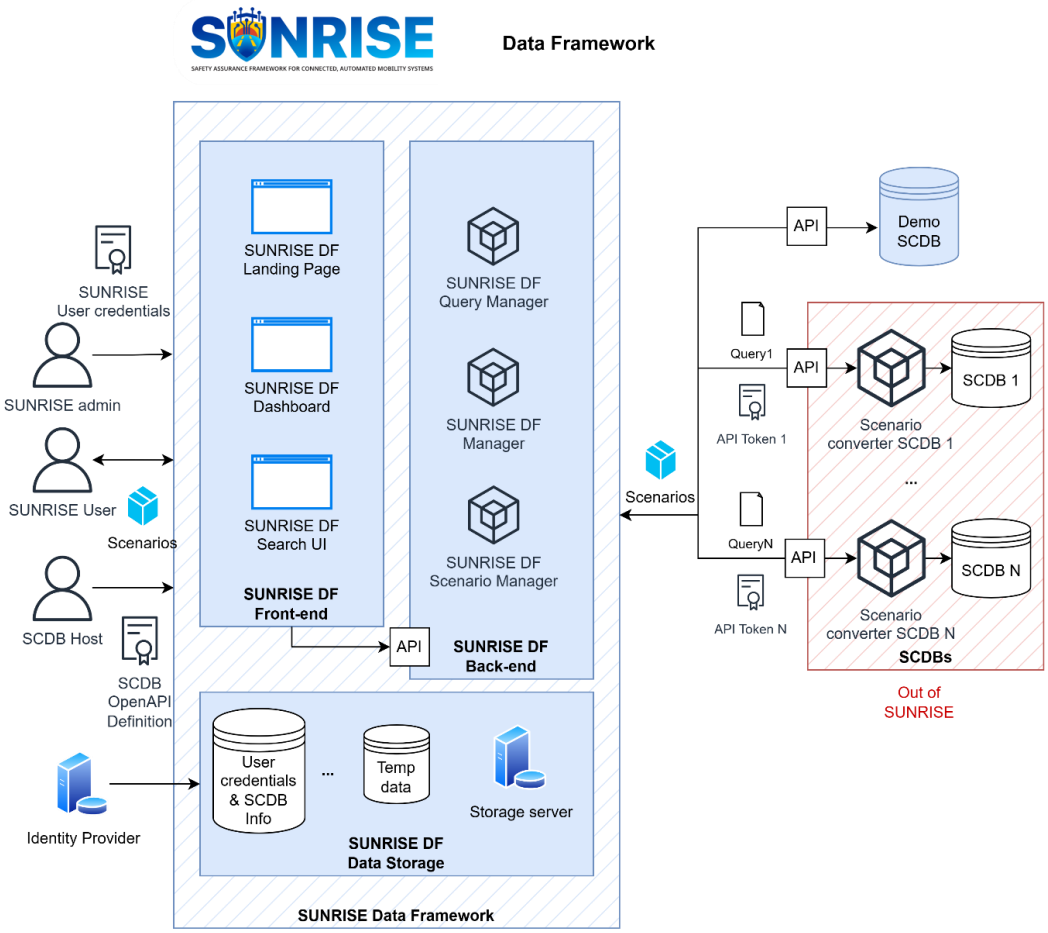


Figure 4: SUNRISE DF Architecture

The SUNRISE DF system is structured into three primary layers: frontend, backend, and data storage. These layers are interconnected through a GraphQL API, which is implemented within the backend layer to facilitate efficient communication and data exchange across the system. This architecture fulfils **RE6** which ensures that SUNRISE DF has state-of-the-art standard.

The SUNRISE DF architecture is composed of multiple integrated sub-components, enabling seamless interaction with users through frontend interfaces and with individual SCDBs via backend APIs. User credentials and database models are securely stored in a centralized database.

The frontend layer comprises components such as the Landing Page, Dashboard, Search UI, and User/SCDB Manager, primarily facilitating user interaction. The backend layer includes the Query Manager, DF Manager, and Scenario Manager, and is responsible for managing communication with the SUNRISE DF database and individual SCDBs. The data storage layer consists of multiple collections used to securely store user credentials, SCDB connection and authentication details, and query models. It also serves as temporary storage for scenario-related information.

The Scenario Manager was initially intended to serve as a comprehensive validation component, ensuring that all received packages comply with the relevant standards. However, due to the diversity of file types, the functionality was modularized and distributed across multiple functions within the Query Manager.

More detailed information for the components is described in the following sub-sections.

2.2 Search UI & Query Manager

The Search UI and Query Manager are integral components facilitating the retrieval of data from individual SCDBs using ontology-based tagging which has been developed in T5.2 and described in D5.2 as SUNRISE Ontology [4]. The Search UI serves as the frontend interface, enabling users to define and configure search parameters. In parallel, the Query Manager operates as the backend service, translating user-defined criteria into standardized data formats and executing queries across the relevant SCDBs.

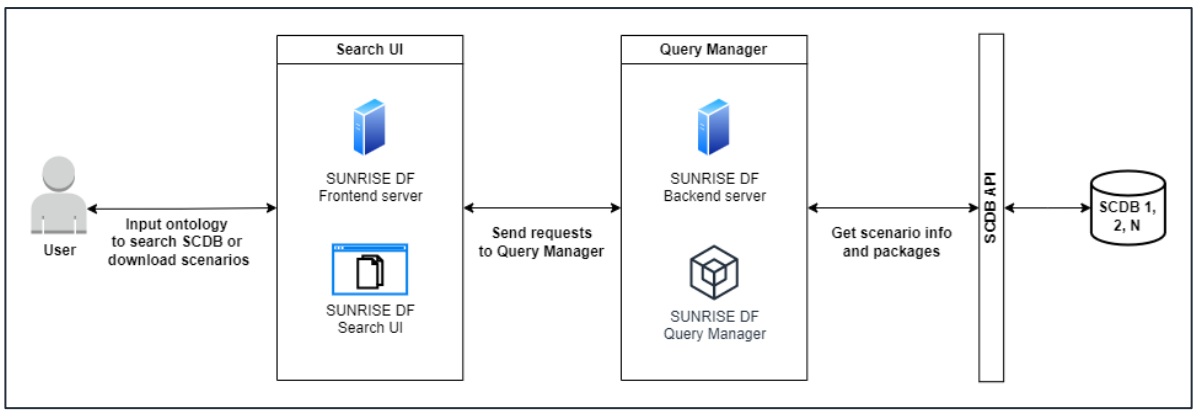


Figure 5: User workflow for retrieving and downloading scenarios from SCDBs

Figure 5 shows the main user workflow for searching SCDBs and downloading scenario packages based on selection.

The combination of the Search UI and Query Manager components fulfils **RE7, RE8, RE28**, which mandates the presence of both a graphical user interface and an API layer for querying and downloading scenarios.

2.2.1 Search UI

The Search UI consists of a series of user interface pages that enable users to create and modify queries, search SCDBs and view detailed information about the retrieved scenarios. It

is designed around two core functionalities: scenario search and scenario download. The search functionality supports the formulation and management of queries based on ontology-driven metadata, whereas the download functionality facilitates the retrieval of complete scenario packages for further analysis or use in subsequent blocks of the SAF such as **Allocate** or **Execute**.

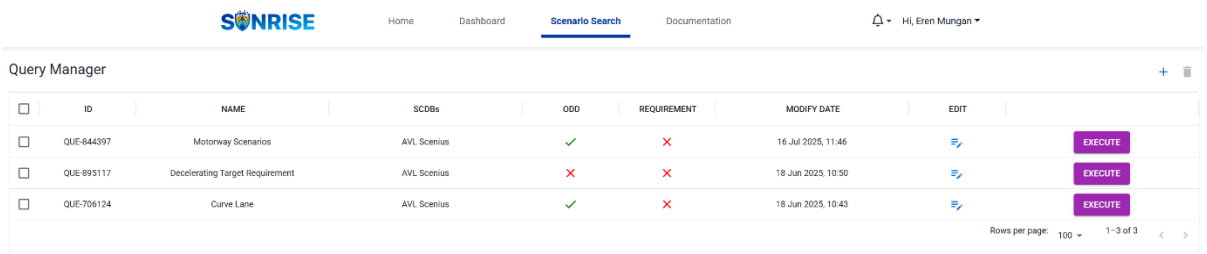


Figure 6: Search UI - Query Screen

Figure 6 illustrates the query management interface within the Search UI. This screen enables users to manage existing search queries or create new ones. Additional filtering options are provided to help users locate specific queries or modify previous ones. The SUNRISE DF retains user-defined queries persistently until they are explicitly deleted by the user. This feature satisfies **RE18** and **RE33** in D5.1.

The SUNRISE DF stores created or executed queries in the database in query model format. Each query model comprises the following elements:

1. The target SCDBs to be searched
2. Associated ontology tags
3. Relevant requirement tags

Ontology tags for a search query are defined under 3 main categories: ODD, Behaviour and Road. **ODD** category contains environmental, scenery and dynamic elements in the scenario. **Road User** category includes different actor types that may take an action in the scenario. And finally, **Behaviour** category contains motion and communication related tags. These main categories and their sub-tags define the content of the ontology. This categorization fulfils the **RE17**, **RE31** focusing on filtering scenarios based on ODD. And **RE19** is fulfilled with Behaviour category which contains cybersecurity related communication tags.

In addition to content, each ontology also proposes a methodology to use these tags. The Search UI and Query Manager uses ISO34503 ontology creation methodology even though content belongs to OpenLabel Ontology [5] [6]. Therefore, the user must either include or exclude chosen ontology tags in the query.

It is expected that individual SCDBs use included and excluded information while searching their database for suitable scenarios. The logic behind include / exclude mechanism can be explained with the following rules:

- If a tag is included in the query, the retrieved scenarios must have this tag linked
- If a tag is excluded in the query, the retrieved scenarios must not have this tag linked.

Please note that the retrieved scenario may include additional ontology tags beyond those specified in the query. As a result, the scenario may contain more tag associations than the requested subset.

Excluded tags are designed to behave as NOT statement and used as a filtering based on omissions. This functionality fulfils **RE32**. By using this logic, the user can create specific query criteria that targets edge-case scenarios. In addition to ODD tags, Search UI also accepts external test requirements. Test requirements support to maintain the traceability of retrieved scenarios. These requirements are stored as requirement tags within queries based on user selection. Requirements are provided as a separate JSON file, following the schema defined in deliverable D6.1. Accordingly, the Search UI accepts JSON file inputs, which are processed by the Query Manager to generate corresponding requirement tags. The code snippet below illustrates an example requirement expressed in JSON format. The Query Manager utilizes this input to extract the relevant tags from the ontology and incorporates them into the search query.

```
{
    "ODD": {
        "IlluminationArtificial": ["any"],
        "DaySunPosition": ["SunPositionFront", "SunPositionLeft"],
        "RainType": ["RainTypeDynamic", "RainTypeConvective"]
    },
    "Behaviour": "MotionCross"
}
```

To ensure semantic consistency, all requirement tags specified in the JSON file must be compatible with the existing ontology tree used in Search UI which is OpenLabel ontology. Any tags that do not align with the ontology structure are disregarded during this step.

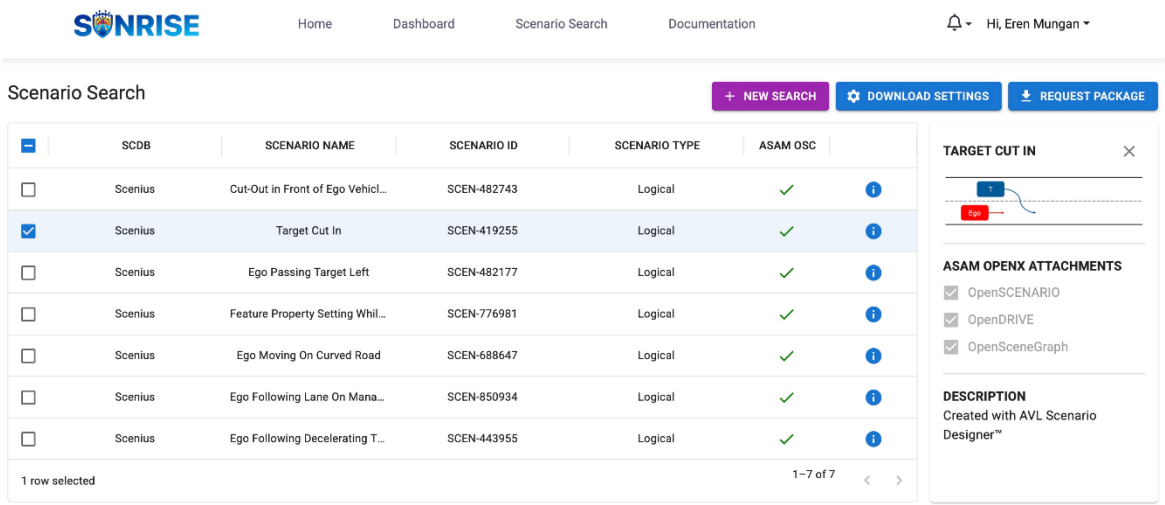


Figure 7: Search UI - Retrieved scenarios

The Search UI transmits query models to the Query Manager, which executes the search across all connected SCDBs. The aggregated results from the different SCDBs are then presented to the user. Figure 7 illustrates an example search result, displaying a list of retrieved scenarios from the connected SCDBs. This list presents only summary-level information for each scenario in a structured tabular format.

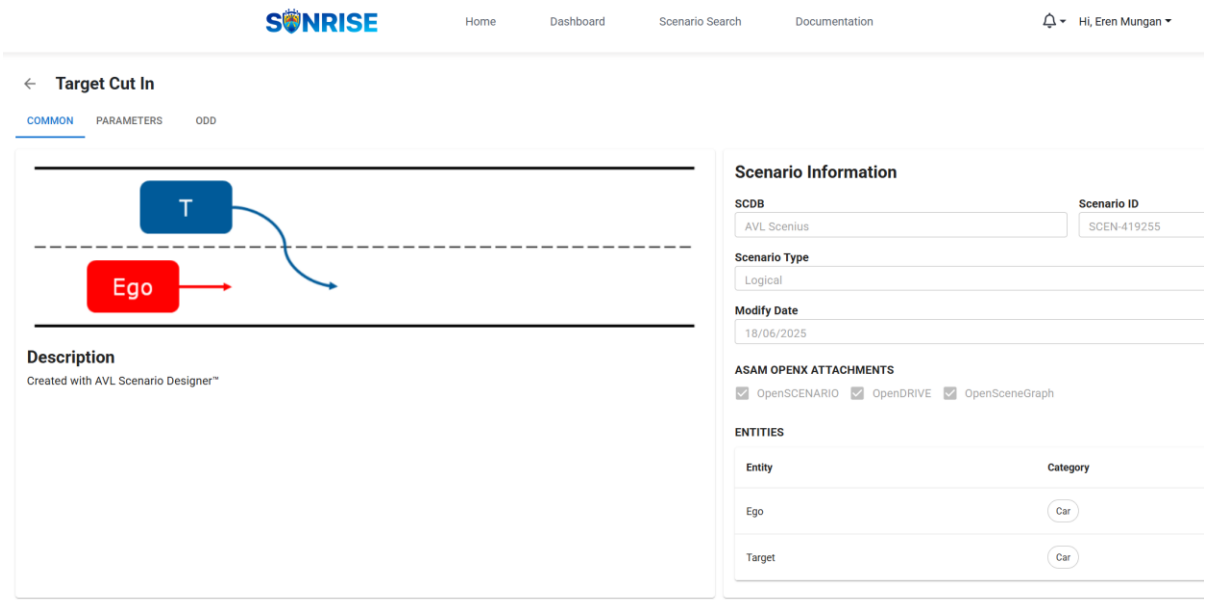


Figure 8: Scenario details UI - Common tab

Users can access more detailed information for each scenario through the Search UI (see figure above). A dedicated scenario detail page is available for every retrieved scenario and includes the following components:

- Scenario Illustration

- Metadata – including unique scenario ID, scenario type, and database information
- OSX Availability – availability of OpenSCENARIO XML XML, OpenDRIVE, etc.
- Scenario Parameters – with corresponding values, ranges, and statistical data
- Ontology Tags – listing all ontology tags associated with the scenario

Each SCDB is expected to provide this set of information for every scenario in a standard format. This output format satisfies **RE59**. The specific content may vary depending on the scenario type, which can be either logical or concrete. Search UI will simply show no data info if there is no data available for any property. Providing logical or concrete scenarios are in the SCDB responsibility. SCDBs support the grouping of concrete scenarios into logical scenarios through their formatting capabilities. This functionality ensures compliance with **RE25**.

The received scenario includes metadata containing the scenario identifier, its source within the SCDB, brief description, media and relevant ontology information. This metadata satisfies **RE38, RE53** and **RE54**.

Scenario parameters include default values as well as defined variations based on the scenario type. Variations can be specified either as ranges, with minimum, maximum, and step size values, or as concrete values. Using parameter ranges, logical scenarios can be defined to encompass the parameter values of corresponding concrete scenarios. These capabilities ensure compliance with **RE51** and **RE55**.

As part of its second core functionality, the Search UI enables users to download scenario packages. For this purpose, each SCDB is expected to provide scenario files in the ASAM OpenSCENARIO XML package format. Depending on the scenario type and its complexity, the package may include the following files:

- **OpenSCENARIO XML:** The scenario file that describes the dynamic content of a scenario, including maneuvers, events, and actions of entities within a simulated environment
- **OpenSCENARIO XML Distribution file (included only for logical scenarios):** The file that contains parameter variations with their reference to base OpenSCENARIO XML file.
- **OpenDRIVE:** The file format that provides a detailed description of the road network, including road geometry, lanes, and signs, serving as the static layer for simulations.
- **Scene Graph File:** The file that defines the visual and spatial arrangement of objects within a 3D environment, establishing the hierarchical structure of a simulated scene.

Since the base OpenSCENARIO XML file does not inherently contain parameter variations, the distribution file is used to define the logical parameter set associated with a logical scenario. The use of OpenSCENARIO XML package fulfils **RE58** in this context.

The Search UI provides a configuration interface that allows users to customize the content of the scenario package prior to download. This customization is designed to help users to only download required files but not the whole package which might have huge size. The user can customize package content according to execution environment needs. For instance, scene graph files could be skipped if the execution environment has low fidelity. This interface contains the following options:

- OpenSCENARIO XML – Mandatory
- OpenDRIVE – Optional
- Scene Graph File – Optional

Download requests are initiated via the Query Manager, based on the user-defined download settings. The retrieved files are then saved locally on the user's machine. Although users can select multiple scenarios for download in a single operation, the system sends a separate request for each scenario to prevent memory overload. Once all individual requests are completed, the Search UI aggregates the results and stores them temporarily for initiating browser download activity.

All requests for querying scenarios and downloading scenario packages are executed through user-specific connections and authentication mechanisms. As a result, users can only access data authorized for their accounts. This approach ensures compliance with Requirements **RE34**, **RE35, RE39**, **RE72, RE73** and **RE74** which address copyright and licensing constraints.

Detailed workflow that shows querying SCDBs for suitable scenarios and downloading scenario packages are shown in **Annex 1**.

2.2.2 Query Manager

The Query Manager is designed as the backend component of the Search UI, responsible for interfacing with both the SUNRISE DF database and the connected SCDBs. It incorporates dedicated functions to:

- Convert user-defined queries into the OpenLabel format required for querying SCDBs [6]
- Process and transform retrieved results into a format compatible with the Search UI
- Manage the download workflow for packages based on user-selected settings

This modular architecture enables efficient query translation, seamless data exchange between components, and consistent scenario retrieval across heterogeneous SCDB implementations.

Converting queries into the OpenLabel format and execute:

The Query Manager receives query models from the Search UI and encodes all associated ontology and requirement tags into the ASAM OpenLabel file format. In addition to the tags themselves, relevant ontology metadata, such as the ontology Uniform Resource Identifier (URI) and related properties, is also embedded within the OpenLabel file. This ensures that connected SCDBs can interpret the tags accurately in the context of the correct ontology tree, enabling consistent and semantically valid query execution across different databases.

Once the conversion is completed, the Query Manager sends requests to each connected and selected SCDB to retrieve relevant scenario information. This functionality fulfills **RE29**, which mandates that the Data Framework must support querying similar scenarios from different databases using a unified query.

Process and transform retrieved results:

Query Manager receives scenario Data Transfer Object (DTO) that contains the metadata, parameter and ontology information which will be shown to user on scenario detail page. As part of the request handling process, the Query Manager validates each retrieved Scenario DTO against a predefined schema to ensure that the scenario data conforms to the expected structure. For each query model, the Query Manager issues individual requests to each relevant SCDB, allowing for parallel and isolated query execution. This functionality demonstrates that **RE4** has been fulfilled.

Once valid responses are received, the Query Manager maps the scenario data into an internal representation model, which is then transmitted to Search UI. This enables the UI to render the results in a structured and user-friendly format. The data received is also validated at this step. This functionality demonstrates that **RE16** has been fulfilled.

Managing the download workflow:

The Query Manager manages the downloading process of scenario packages by orchestrating requests and retrieved contents. Once download request is received from Search UI, the Query Manager, groups scenarios by their SCDB and creates valid download requests for each scenario. Each request prepared for a scenario contains the following properties:

- Scenario Database Name
- Scenario ID – Unique ID provided by SCDB
- Scenario Type – (Logical / Concrete) identifier if the scenario has multiple versions
- Requested OSX package content

During the search process, SCDBs indicate the availability of the ASAM OpenSCENARIO XML (OSX) package for each scenario. If the user requests specific files that are not available in SCDB, the Query Manager automatically adjusts the request to include only the files that

are available. For example, if the user selects the Scene Graph File as part of the download package, but SCDB has explicitly reported this file as unavailable, the Query Manager excludes it from the request. The user is informed via pop-up about this action.

SCDBs are expected to provide scenario content as a ZIP package containing the requested files. The Query Manager temporarily stores these packages and transfers them to the user's local machine once the download process is complete.

2.3 Dashboard

The Dashboard component is designed to provide an overview of each SCDB connected to the SUNRISE DF platform. This overview aims to give users a clear understanding of the content and capabilities of each individual SCDB. In addition to summarizing the types of scenarios and data available, the Dashboard also displays the current operational status of each SCDB and relevant administrative information, such as planned maintenance schedules or known availability issues.

The dashboard info contains:

- Database Info – Name, owner, version
- Service Info – License and maintenance information
- Scenario Counts – Logical, Concrete and Functional scenarios
- Taxonomy / Ontology Tree
- SCDB Quality Metrics

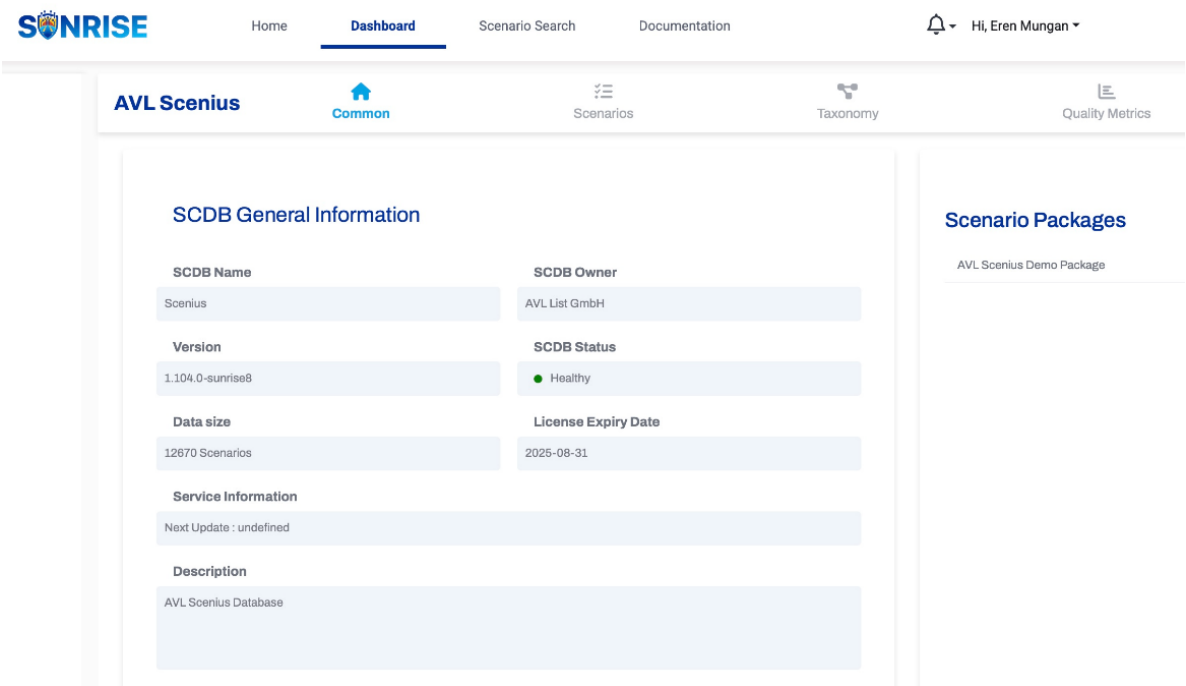


Figure 9: Dashboard - Common Tab

The Dashboard component consists of a set of read-only screens designed to present key information to the user. While the data itself is not editable, the component supports user interaction through various features such as data visualization, view customization, and

filtering. These capabilities enable users to explore and interpret the available information in different formats, facilitating a better understanding of the status and content of each connected SCDB. This functionality fulfils **RE11** which mentions that the user must see connected databases, total scenario counts.

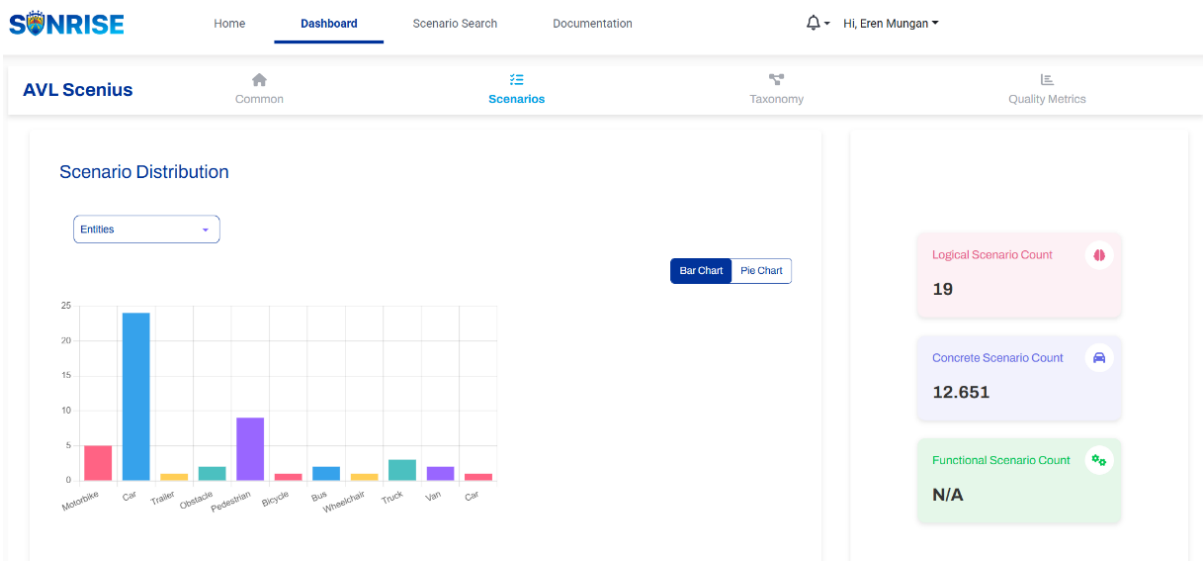


Figure 10: Dashboard - Scenarios Tab

Figure 9 and Figure 10 illustrate the Common and Scenario tabs within the Dashboard component. These tabs allow users to explore the contents of each SCDB across various categories through simple selection mechanisms. Information contain in these tabs fulfils **RE9** and **RE10**.

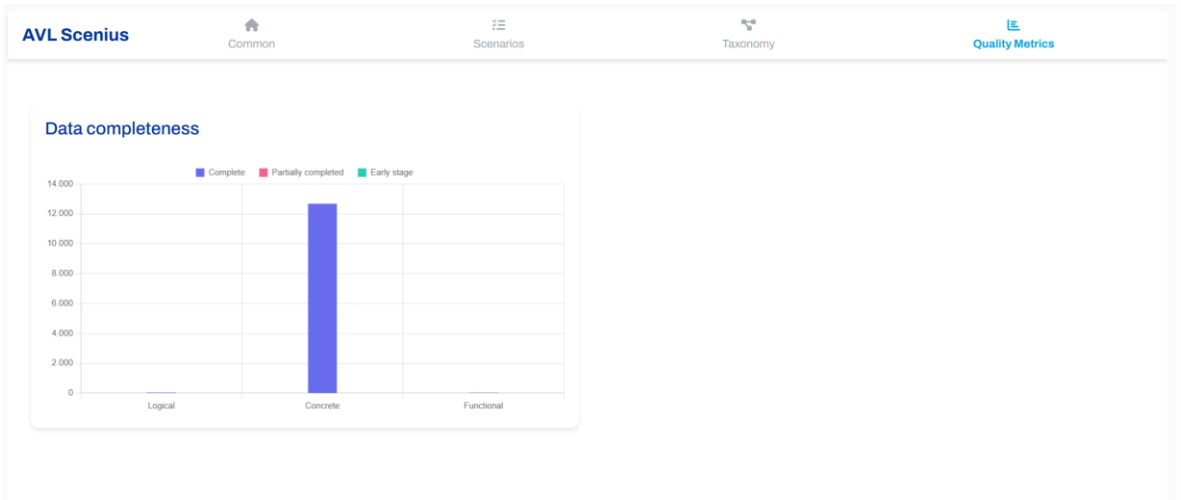


Figure 11: Dashboard - Quality Metrics tab

Figure 11 presents the quality metrics view within the dashboard. This screen displays plots and charts based on the provided quality metrics, offering users a visual representation of the data. The quality metrics are defined in detail in Deliverable D5.3 and are calculated by the SCDB [7]. These metrics are then supplied to the SUNRISE DF as part of the dashboard information. This capability of showing quality metrics, fulfils **RE12**.

The Dashboard relies on SCDBs to supply the required data. When a user selects an SCDB, a corresponding request is sent to that SCDB. The retrieved data is then mapped to the frontend component and rendered in a user-friendly format to support intuitive exploration and interpretation.

2.4 User Credentials & SCDB Connection Management

User credential management and SCDB connection handling are additional responsibilities managed by the SUNRISE DF platform. It is essential that each user has secure access to their own data and maintains a persistent, stable connection to the relevant SCDBs. This capability fulfils the **RE2**.

SUNRISE DF operates within a cloud infrastructure provided by a third-party service provider, which ensures a secure and resilient environment against cybersecurity threats. This setup fulfils Requirements **RE36**, **RE37** and **RE71**.

SUNRISE DF is implemented using a modular architecture, where each component, such as the Search UI and Query Manager, has its own dedicated functional class and communicates via the API layer. This modular structure facilitates future development and maintenance, fulfilling **RE13**.

SUNRISE DF includes a dedicated database for securely storing user credentials and SCDB connection information, with all sensitive data encrypted. A user interface is provided to facilitate the management of this information. This functionality satisfies **RE3**.

During operations such as searching, downloading scenarios, or accessing dashboard data, SUNRISE DF utilizes this user-specific, tagged information to ensure authenticated and personalized interactions with the connected SCDBs.

The SUNRISE DF platform defines two user roles: **Standard User** and **Database Owner**.

- Standard Users are authorized to perform all available operations except SCDB connection management.

- Database Owners have extended permissions, allowing them to create and modify SCDB connections. However, database owners are restricted to modifying only the connections they have created and cannot alter connections configured by other users.

By default, newly registered users are assigned the Standard User role. Users can request an upgrade to the Database Owner role via the User Management interface. This capability fulfils the **RE1.**

To establish new SCDB connections, database owners must provide a valid OpenAPI schema that is a JSON file containing meta information for the endpoints and the corresponding authentication method. This file is used to prevent users from entering incorrect information and to shorten the integration time.

All necessary details regarding endpoint implementations, business logic, and request/response schemas and examples are documented in deliverable **D6.3**. This document enables SCDB owners to implement the required endpoints accurately and ensures proper communication with SUNRISE DF. The content and associated work presented herein fulfil **RE5** and **RE47** that ensures compatibility of partner databases: SafetyPool™, AVL Scenius™, Moove, ADScene, VVMethods, Scenario.Center and Streetwise.

SUNRISE DF utilizes dedicated database models for each SCDB connection and user. All connections and associated data operations (retrieval and usage) include unique identifiers that ensure traceability and linkage to the corresponding user and database instance. This approach satisfies Requirements **RE21**, **RE22** and **RE62**.

SCDB Authentication:

SUNRISE DF supports two authentication mechanisms for onboarding and accessing SCDBs:

- API Key
- OAuth 2.0

SCDB providers may choose either method based on their system requirements. Authentication credentials must be provided once for each SCDB connection.

SUNRISE DF implements a centralized authentication service, accessible to all internal components. Modules such as Query Manager and Dashboard, which require communication with individual SCDBs, leverage this service to ensure secure and authorized access during all interactions.

2.5 Other Components

SUNRISE DF contains implementation of some custom functions / sub-components to be used as helpers for main components. These sub-components are generally used to convert data formats or to perform a pre-check.

SCDB Health Check: This component gets connected SCDBs health status. It uses a simple request and response body and checks if SCDB is healthy and can be used for scenario search and dashboarding purposes.

User Validation: This component checks if the connected user is a valid user and have access to perform SUNRISE DF related operations.

Requirement Parser: This component parses requirements JSON files and creates tag data to be included in the search query. Requirement parser component is originally developed within T6.3 as part of automated query generation tool and explained in D6.1 in detail. SUNRISE DF uses requirement parser part of this implementation in Search UI and Query Manager components.

3 DATA FORMATS

3.1 Introduction

The SUNRISE DF employs a variety of data and file formats to facilitate data exchange both between internal components, with connected SCDBs and with other components within the SAF (such as the harmonized V&V Simulation Framework developed in WP4). Some of these data formats are based on established industry standards, while others have been specifically developed as custom formats to support the unique requirements of the SUNRISE DF ecosystem.

To establish a stable and fully functional connection, SUNRISE DF requires each SCDB to implement five distinct endpoints. Each endpoint adheres to a specific data format, designed to ensure consistent and reliable communication between SUNRISE DF and the SCDB.

This chapter provides a detailed explanation of the ontology structure and the associated data formats utilized by these endpoints.

3.2 SUNRISE Ontology

Ontology is a structured and formal representation of concepts, entities, and their relationships within a specific domain, enabling consistent understanding and communication between humans and machines. In the context of virtual validation for autonomous vehicles, ontology plays a critical role by standardizing terminology across simulation tools and test scenarios, supporting intelligent reasoning about traffic elements and behaviours, and facilitating systematic scenario generation and coverage analysis. It also enhances interoperability between different validation tools and standards, such as OpenSCENARIO and OpenDRIVE, by providing a shared conceptual framework. Overall, ontology significantly improves the efficiency, reliability, and comprehensiveness of virtual validation processes, which are essential for the safe and robust development of CCAM systems.

Ontology serves as a powerful framework for representing the relationships between scenarios and their associated domain context, particularly with respect to the Operational Design Domain (ODD). In SUNRISE DF, ontology is the core concept used to structure, tag, and interpret scenario metadata in a semantically meaningful way.

This ontology-driven approach powers an ongoing data-enrichment process, seamlessly integrating new contextual attributes, such as roadway geometry, environmental conditions, and actor behaviours, into every scenario record. Leveraging ontology in this way enables systematic data enrichment by standardising, completing, and classifying scenario metadata.

By anchoring each test scenario to a shared ontology, execution results inherit the same semantic identifiers and tags. Building on this ontology-based scenario model, the deliverable proposes using the ontology itself to provide full traceability of test-execution results. Recording KPIs and artefacts as ontology-referenced entities establishes an end-to-end chain linking requirements to scenarios, test runs and outcomes. This linkage unlocks automated

coverage dashboards across ODD and functional requirements, instantly revealing validation gaps and guiding targeted test planning.

Several ontologies have been developed in the field of automated driving, including ISO 34503, OpenLABEL, and OpenXOntology [5] [6] [8]. While these ontologies differ in structure and content, they share a common goal: to provide a formalized representation of concepts relevant to scenario modelling and the Operational Design Domain (ODD).

SUNRISE DF uses a simplified version of OpenLabel ontology to search SCDBs. This version contains all tags except the ones that require numerical values to be assigned. Due to complexity of logic required to define numerical tag values, a simplified version of OpenLabel ontology is used as a content for queries. Incorporating numerical values introduces an additional layer of complexity to the query commands and necessitates a more extensive search process on the SCDB side. Furthermore, designing a corresponding UI component that supports numerical input with include/exclude logic becomes increasingly complex and less user-friendly.

This simplified version contains only the tags without numerical values. Examples and rules for this ontology and its representation on OpenLabel is described in section 3.3.3. This content has been agreed upon by the work packages, stakeholders, and partners. Therefore, the use of the OpenLabel ontology fulfills Requirement **RE30** and **RE40** that focus on using the same naming convention. Additionally, it satisfies **RE66** since OpenLabel contains ontology tags.

On the other hand, SUNRISE Ontology, developed within Work Package 5 (D5.2), builds upon the strengths of these existing ontologies. It integrates key features and methodologies from established standards while also incorporating the specific requirements and domain needs identified throughout the SUNRISE project. This tailored approach ensures semantic consistency, project alignment, and extensibility for advanced scenario definition and validation.

The SUNRISE ontology is designed not only to comprehensively cover all project-defined requirements but also to explicitly represent the relationships between ontology tags. This enhanced semantic structure facilitates more precise definitions of Operational Design Domains (ODD) and scenarios, including complex and edge cases.

Considering these characteristics, the SUNRISE Ontology was utilized as a design guide during the development of SUNRISE DF, particularly for its capabilities in tag classification and the establishment of relationships between them.

3.3 Input / Output Data Formats

SUNRISE DF relies on API-based communication to exchange data with connected SCDBs. Most of this communication is conducted using JSON-structured Data Transfer Objects (DTOs), which are used for operations such as retrieving content to be displayed on the dashboard and scenario metadata resulting from search queries.

However, certain operations require the use of specialized file formats, particularly for executing searches and retrieving complete scenario packages. The following data and file formats are employed within SUNRISE DF:

- ASAM OpenSCENARIO XML
- ASAM OpenDRIVE
- ASAM OpenLabel
- Scene Graph

The OpenSCENARIO XML, OpenDRIVE, and Scene Graph formats are used to deliver complete scenario packages. In contrast, the OpenLabel format is utilized to represent search criteria, including ontology-based tags, during query execution.

3.3.1 ASAM OpenSCENARIO XML

ASAM OpenSCENARIO XML is a standardized scenario description format published by ASAM, primarily used for virtual scenario execution in simulation environments [9]. It provides a structured, machine-readable syntax to model complex and realistic interactions among traffic participants such as vehicles, pedestrians, and infrastructure elements.

The specification supports a wide range of features, including time-based and event-based triggers, conditions, maneuvers, and actions, enabling the definition of both deterministic and stochastic scenarios. OpenSCENARIO XML also supports parameterization through its native syntax, allowing for flexible scenario definitions.

To define sets of related scenarios, the OpenSCENARIO XML file can be accompanied by a distribution file, which contains the parameter variations—either deterministic or stochastic. In this context:

- The standalone OpenSCENARIO XML file represents a concrete scenario
- The combination of the XML file with its associated distribution file represents a logical scenario

SUNRISE DF uses OpenSCENARIO XML file format to download scenario files from individual SCDBs. It requires only base OpenSCENARIO for concrete scenarios and needs distribution file for logical ones. This use of OpenSCENARIO XML and other coupled files ensures that both SCDBs and SUNRISE DF uses standardized files for input and output. This functionality fulfils **RE26** and **RE27**.

3.3.2 ASAM OpenDRIVE

ASAM OpenDRIVE is a standardized XML-based file format developed to describe the static road network used in driving simulation environments [10]. It defines the geometric and topological structure of roads, including elements such as lanes, intersections, signals, elevations, and road markings.

Within the SUNRISE DF framework, OpenDRIVE serves as a complementary file format to OpenSCENARIO during the retrieval of scenario packages.

3.3.3 ASAM OpenLabel

ASAM OpenLabel is a standardized format developed by ASAM for the annotation and labelling of data used in the development, training, and validation of automated driving systems. Although originally intended for labelling sensor and scenario data, the structure of OpenLabel, particularly its support for ontology-based categorization, makes it well-suited for representing query criteria.

Within the SUNRISE DF framework, OpenLabel is repurposed to store and exchange search parameters, leveraging its ontology-aligned design to ensure semantic consistency and interoperability across connected SCDBs.

Figure 12 presents an example of an OpenLabel file generated by SUNRISE DF for querying connected SCDBs. The file is structured into three main sections:

1. **Metadata:** Contains schema version and custom comment. For SUNRISE DF, include/exclude tags are implemented with a Boolean tag value. Therefore, its explanation is described in comment section.

2. **Ontologies:** This section contains master ontology related info. URI shows the master ontology where boundary list and mode shows which tags will not be in consideration.

3. **Tags:** This section contains ontology tags that will be used during the scenario search process. Excluded ontology tags are marked with Boolean tag value, while included ones does not have this identifier.


```
{
    "openlabel": {
        "metadata": {
            "schema_version": "1.0.0",
            "comment": "IsExcludedInQuery boolean tag value is used to define if the tag is excluded in the query or not."
        },
        "ontologies": {
            "0": {
                "uri": "https://openlabel.asam.net/V1-0-0/ontologies/openlabel_ontology_scenario_tags.ttl",
                "boundary_list": [
                    "DrivableAreaSigns",
                    "DrivableAreaEdge",
                    "DrivableAreaSurface"
                ],
                "boundary_mode": "exclude"
            }
        },
        "tags": {
            "0": {
                "type": "RoadTypeMinor",
                "ontology_uid": "0"
            },
            "1": {
                "type": "HorizontalStraights",
                "ontology_uid": "0",
                "tag_data": {
                    "boolean": [
                        {
                            "name": "IsExcludedInQuery",
                            "val": true
                        }
                    ]
                }
            },
```


Figure 12: An example OpenLabel file for scenario search purposes

Figure 13 illustrates an ontology tag associated with a numerical value, specifically a value range. Both OpenLabel and the ontology content developed within WP5 utilize ontology tags that can carry such values. To accommodate this, ontology tags capable of holding values are represented using tag data structures, enabling precise definition and filtering based on quantitative parameters.

Figure 13: Ontology tag with range values

Although OpenLabel is inherently a file format with a JSON extension, within the SUNRISE DF architecture it is utilized as the body of API requests rather than as a standalone file exchanged between systems. Consequently, no physical OpenLabel files are transferred between SUNRISE DF and connected SCDBs; instead, all exchanged information conforms to the OpenLabel data structure and format within the API communication.

3.3.4 Scene Graph

Scene graph files serve as mesh definitions for 3D environments, structuring the spatial and visual representation of simulation elements. They organize the hierarchy of objects (nodes) and their interrelationships in a tree-like structure. Each node typically includes information such as 3D models, spatial transformations, lighting properties, collision boundaries, and other visual or physical attributes.

Scene graph files are extensively used in virtual simulations to model environments, objects, and road infrastructures. They play a critical role in applications such as ADAS/AD testing, sensor simulation, and vehicle modelling. As a key component of high-fidelity simulation environments, scene graph files are commonly used in conjunction with ASAM OpenDRIVE files, which provide road network definitions.

Various scene graph formats are supported across different simulation platforms, enabling flexibility in integration and deployment.

Since scene graph files are required during the execution of scenario files, SUNRISE DF can request these files as part of the scenario package, based on user preferences.

4 CONCLUSIONS

The main actionable outcome of this deliverable is the SUNRISE Data Framework (DF) implementation. The SUNRISE DF provides a robust and standardized framework for **querying, managing, and retrieving scenario data from SCDBs**. By leveraging established standards such as ASAM OpenSCENARIO, OpenDRIVE, OpenLabel and SUNRISE Ontology, the DF ensures interoperability, semantic consistency, and extensibility across all internal components as well as components within the wider SUNRISE SAF. While OpenSCENARIO and OpenDRIVE enable detailed scenario and road network representation, OpenLabel has been adapted within SUNRISE DF to serve as a structured medium for ontology-based query definition, supporting both inclusion and exclusion semantics and value-bound parameters.

The SUNRISE DF includes an overview of the SCDB content via its Dashboard, thereby providing users with direct insight into connected databases. It uses methodologies implemented within the project such as **Quality Metrics** which are defined in D5.3, which help users to judge content quality and completeness. The detailed information about dashboard and quality metrics can be found in **chapter 2**.

The SUNRISE ontology defined and provided in deliverable D5.2, plays a central role by not only covering project-wide requirements but also modelling the relationships between tags. The details of ontology can be found in **chapter 3**. This capability significantly enhances the expressiveness and precision in **defining ODDs and complex or edge-case scenarios**. Using this ontology for designing Search UI, makes it possible to **query SCDBs to find suitable scenarios** for test allocation and execution and plays a crucial role in the harmonized V&V Simulation Framework developed in WP4.

The design fundamentals and implementation of SUNRISE DF can be used by both internal and external stakeholders **for further development in research and development projects, or development of new SCDBs**. The application itself supports user to manage multiple SCDBs by using standard file formats, therefore it demonstrates a working concept in this context.

The ontology content used and its application in the SUNRISE DF provide users with enhanced means for **safety validation of CCAM system**. Nevertheless, there is potential for further improvement and development. Using complex ontology contents **including numerical values with their range and querying scenarios based on these values** can be identified as an improvement point. Additionally, SUNRISE DF has room for design improvement and implementation of the scenario import storyline defined in deliverable D6.1. This workflow might help users not only receiving scenarios but also **pushing new scenarios into SCDBs**. This could also help to exchange scenarios between databases via query / import functionalities.

In summary, the SUNRISE Data Framework (DF) represents a key technological achievement within the SUNRISE project, **offering a scalable, standards-compliant, and semantically enriched web application for scenario data management**. By integrating leading open

standards and a domain-specific ontology, SUNRISE DF ensures interoperability, precision, and usability across the project's ecosystem. As both a demonstrator and a practical tool, it supports current project goals while also enabling future research, SCDB development, and broader collaboration within the automated driving validation community. Its modular design and extensible architecture make it a foundational asset for advancing scenario-based testing methodologies, even beyond the scope of SUNRISE.

5 REFERENCES

[1] SUNRISE, “D6.1 Methodology for SCBD application for generic use cases,” 2024. [Online]. Available: https://ccam-sunrise-project.eu/deliverable/d6-1-methodology-for-scbd-application-for-generic-use-cases/.

[2] SUNRISE, “D6.3 First European Test Case Library,” 2025. [Online].

[3] SUNRISE, “D5.1 Requirement for CCAM safety assessment data framework content,” 2024. [Online]. Available: https://ccam-sunrise-project.eu/deliverable/d5-1-requirements-for-ccam-safety-assessment-data-framework-content/.

[4] SUNRISE, “D5.2 Harmonised descriptions for content of CCAM safety assessment data framework,” 2025. [Online].

[5] ISO, “ISO 34503:2023 - Road Vehicles — Test scenarios for automated driving systems,” 2023. [Online]. Available: https://www.iso.org/standard/78952.html. [Accessed 2023].

[6] ASAM, “ASAM OpenLABEL®,” ASAM, 12 November 2021. [Online]. Available: https://www.asam.net/standards/detail/openlabel/. [Accessed 23 August 2024].

[7] SUNRISE, “D5.3 Quality metrics for scenario database,” 2025. [Online].

[8] ASAM, “ASAM OpenXOntology Concept,” ASAM, 2021. [Online]. Available: https://www.asam.net/standards/asam-openxontology/.

[9] ASAM, “ASAM OpenScenario,” 2024. [Online]. Available: https://www.asam.net/standards/detail/openscenario/. [Accessed 07 July 2024].

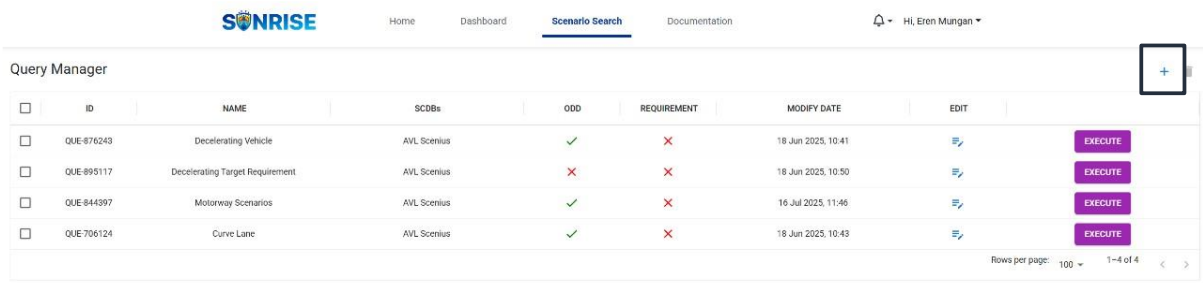
[10] ASAM, “ASAM OpenDRIVE,” 2024. [Online]. Available: https://www.asam.net/standards/detail/opendrive/. [Accessed 07 July 2024].

ANNEX 1: USER WORKFLOW FOR QUERYING SCENARIO DATABASES

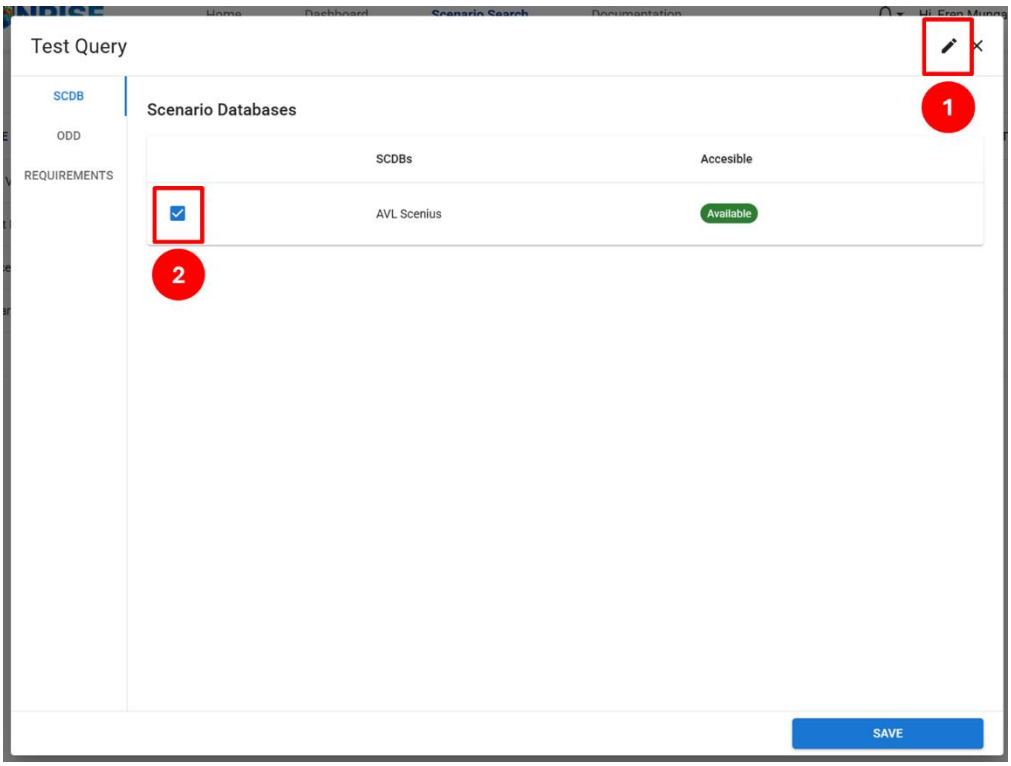
This section represents user workflow (guidance) to query SCDBs by using SUNRISE DF Search UI. The users can follow step by step guide below to use SUNRISE DF to query scenario databases.

Creating the query:

Navigate to **Scenario Search** page and click on **plus (+)** button located top-right. This will open a pop-up menu.

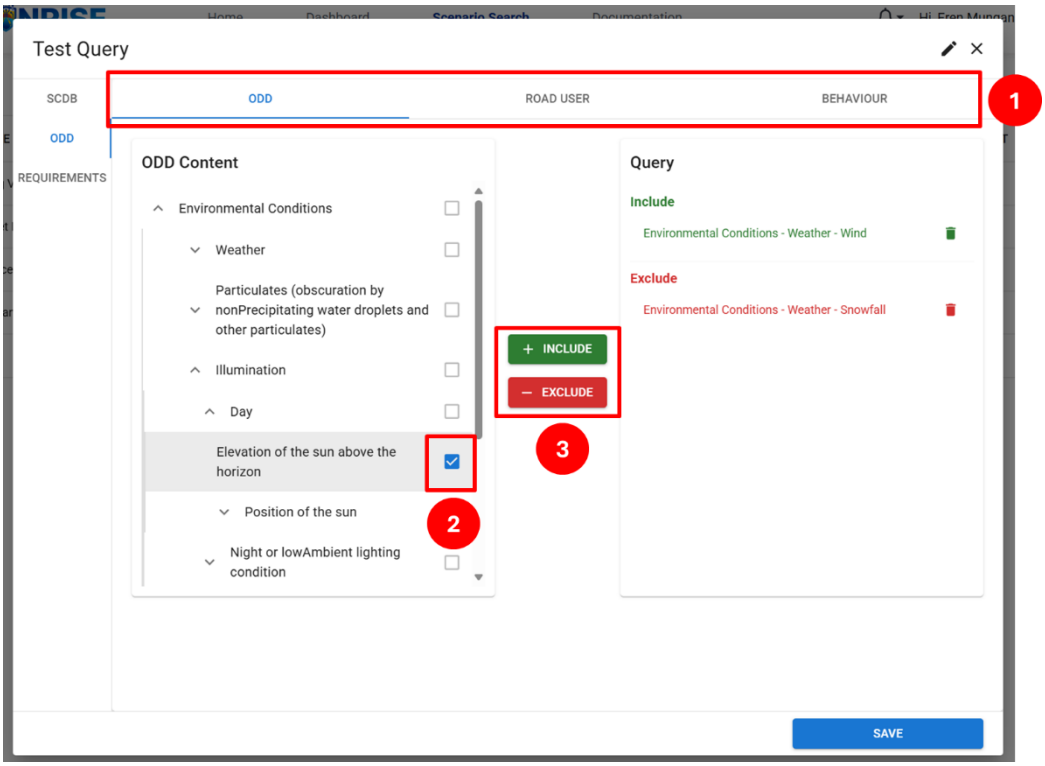


Rename query accordingly and **select SCDBs** to conduct search on.

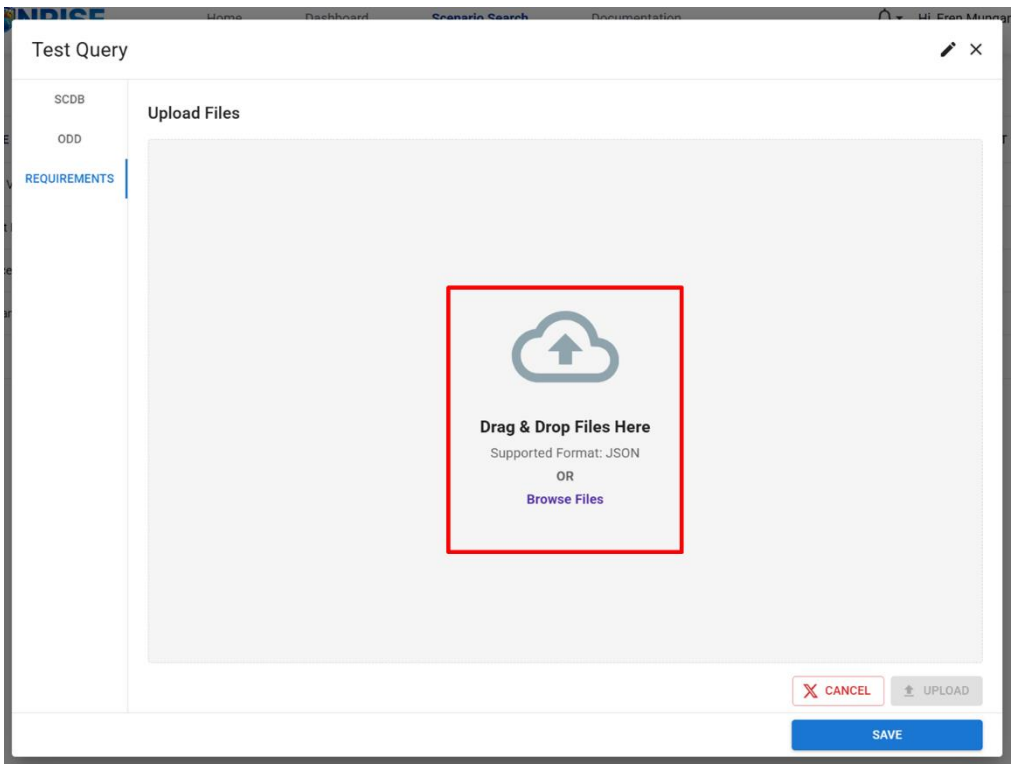


Navigate to **ODD menu**. Select category of ontology to see its content. Then, use content menu on the left-hand side and make selections. The user can only select leaf nodes. Parent

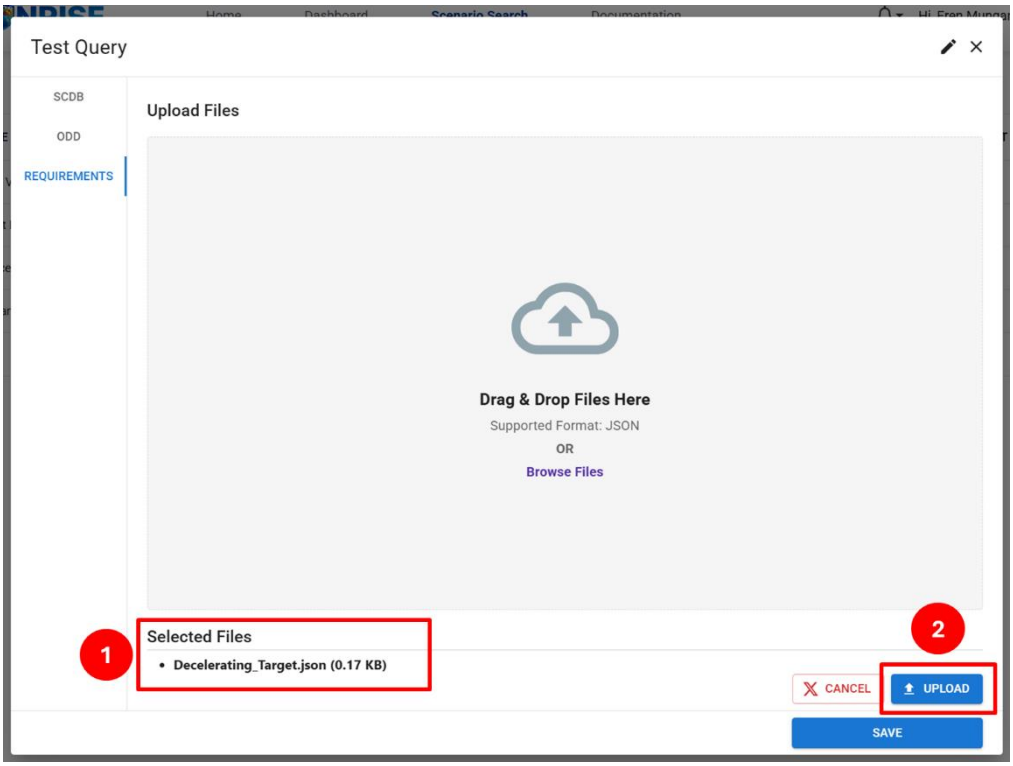
nodes are disabled for selection. Once selection is finished, the user can click either include or exclude buttons. Search UI will place user's selections on the query side (right).



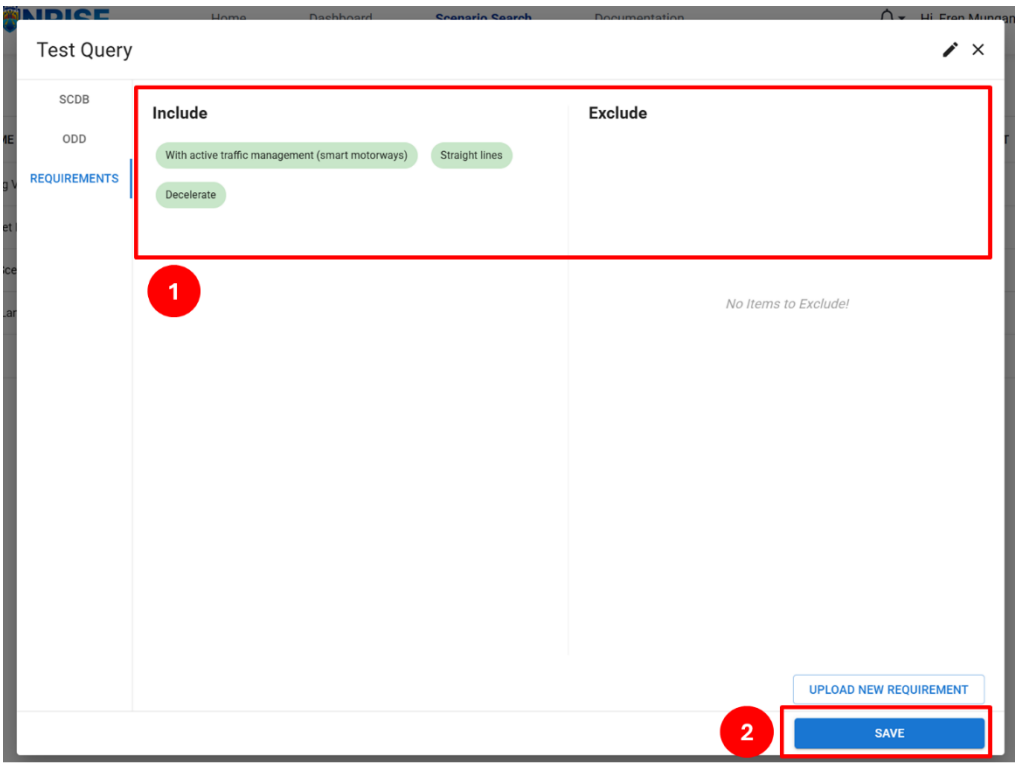
Navigate to **requirements menu** and click on browse files or drag and drop requirements JSON file. Once requirement file is provided, **click on upload**.



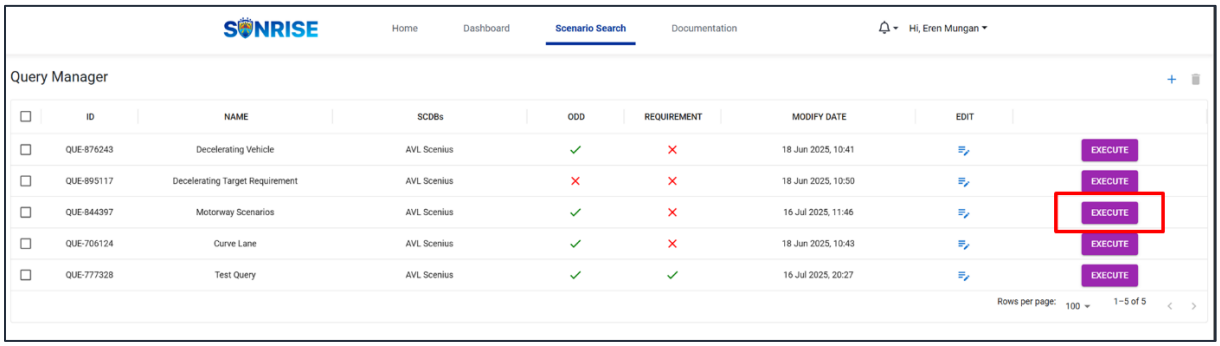
Used requirement file details will be shown on the left bottom. Then **click on upload** to start the parsing requirement process.



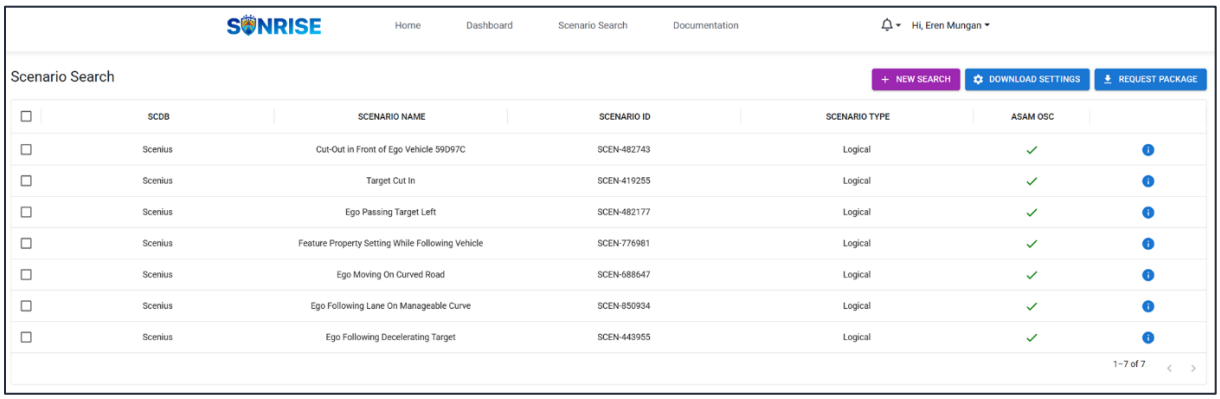
SUNRISE DF shows consumed requirements once the process is over. Observe the tags and **click on save** button to save query and close the pop-up window.



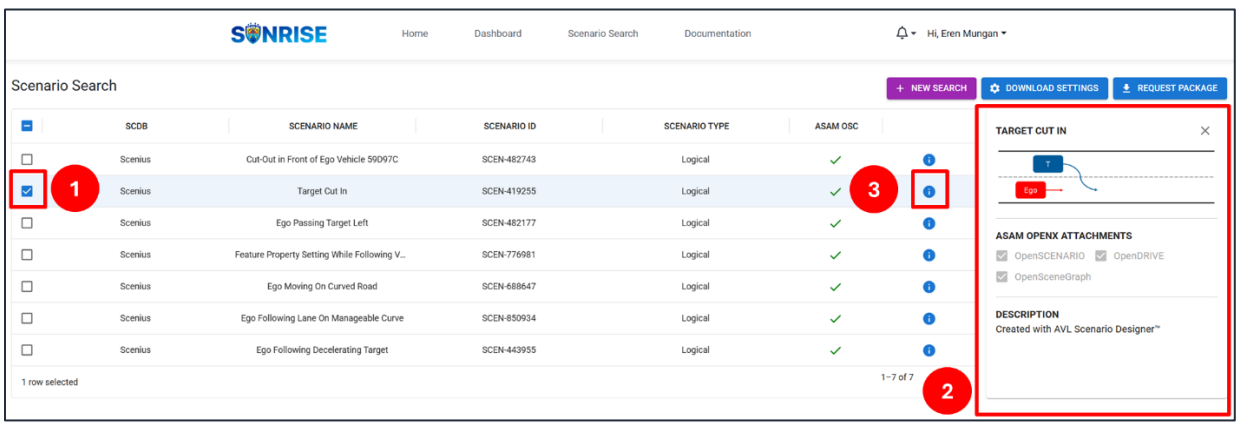
Click on **execute** button at the end of the query line. This will initiate SCDB search process.



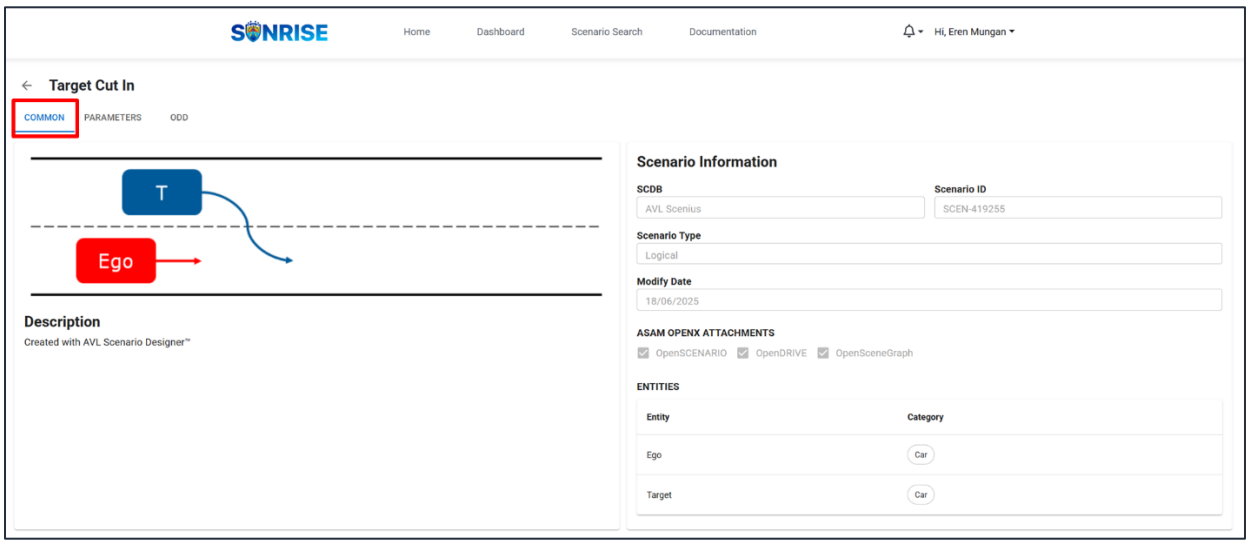
Search UI will show scenario search results once query process is completed. The user will have the full list of scenarios from selected databases.

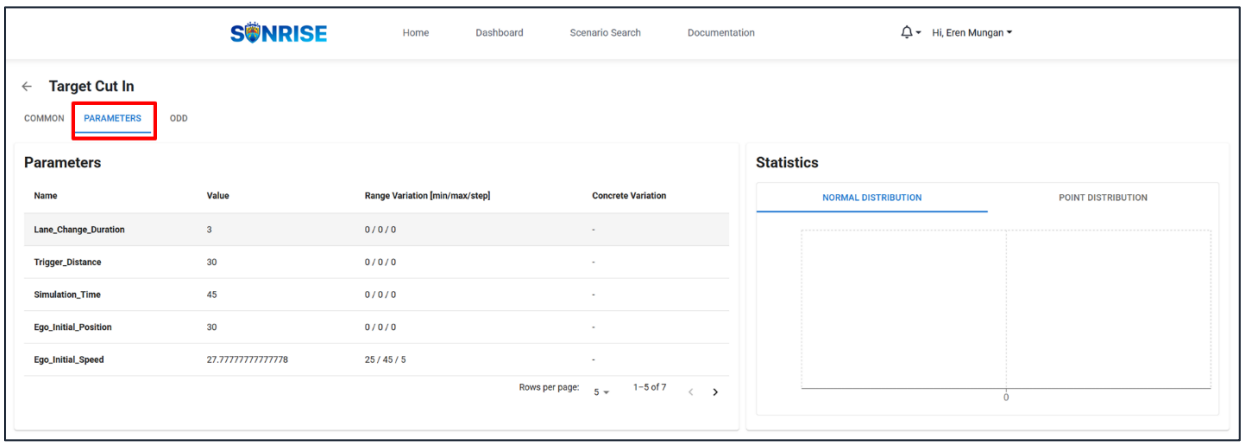


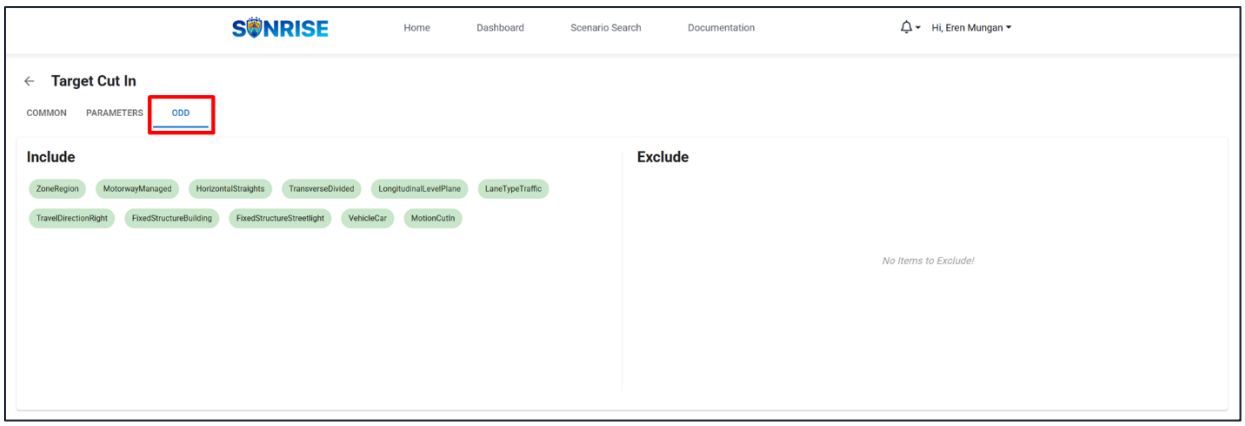
Click on a scenario to see its **summary info** on the right side. This view shows brief information about the retrieved scenario. **Click on info icon** to see more details.



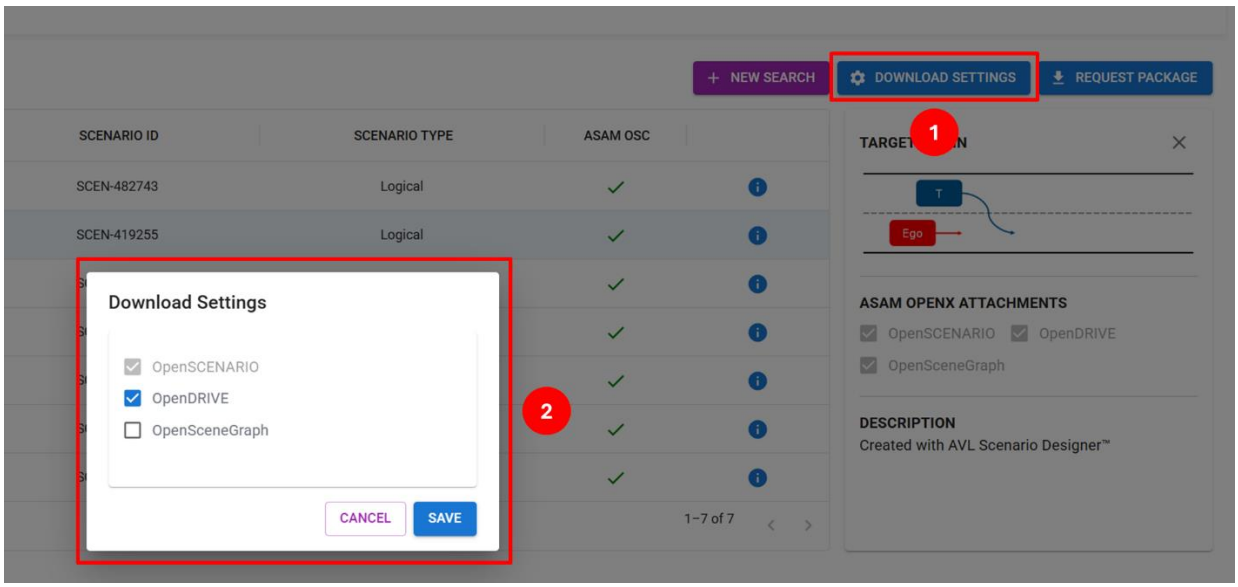
The user can navigate through the tabs to see common, parameters and ODD related information.



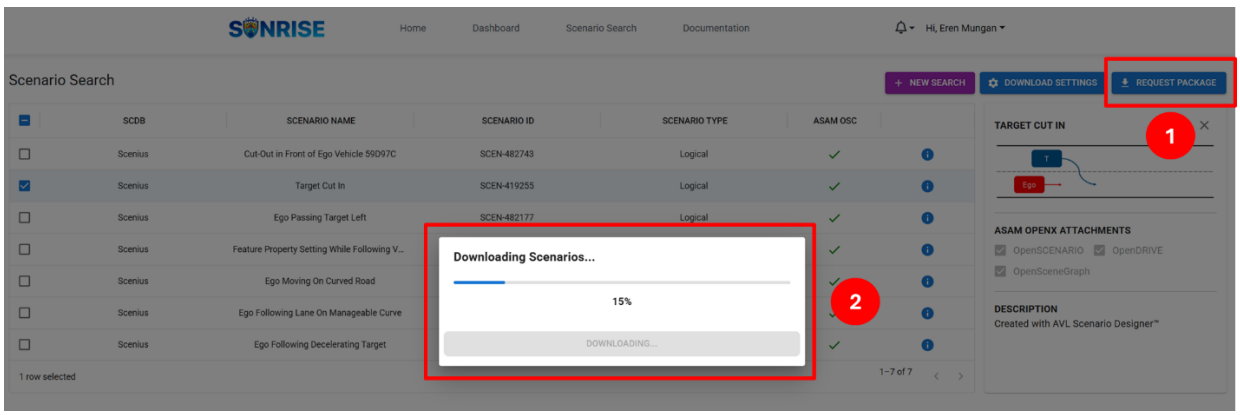




The user can set the content package for downloaded scenarios by using the **Download Settings** button. This will open a pop-up window where the user can select files to be included in the package.



Once this selection is done, the user can **select scenarios** to be downloaded and click on **Request Package** button. This will initiate scenario download process and the progress will be shown to user via progress bar.



Finally, the requested package will appear in the user's browser as below.