



SAFETY ASSURANCE FRAMEWORK FOR CONNECTED, AUTOMATED MOBILITY SYSTEMS

D6.1

Methodology for SCDB application for generic use cases

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	D6.1: Methodology for SCBD application for generic use cases
Deliverable responsible	Fadi Alakkad, WMG, University of Warwick (UoW)
Status - Version	Final – V1.0
Submission date	30/08/2024
Keywords	SUNRISE, Data Framework, SCDB, SAF, CCAM

Authors/Contributors

Name	Organisation
Fadi Alakkad, Xizhe Xhang, Siddartha Khastgir	WMG, University of Warwick (UoW)
Jobst Beckmann	RHEINISCH-WESTFAELISCHE TECHNISCHE HOCHSCHULE AACHEN (ika)
Thaddaeus Menzel, Xavier Boabén, Serg Vidal	IDIADA (IDI)
Eren Mungan	AVL List GmbH (AVL)
Tajinder Singh, Edwin van Hassel	SIEMENS INDUSTRY SOFTWARE NETHERLANDS BV (SISW)
Maximilian Grabowski	BUNDESANSTALT FUER STRASSENWESEN (BAST)
Jeroen Broos	NEDERLANDSE ORGANISATIE VOOR TOEGEPAST NATUURWETENSCHAPPELIJK ONDERZOEK TNO (TNO)
Marcos Nieto, Juan Diego Ortega Valdivieso	VICOMTECH (VICOM)
Athanasios Ballis	INSTITUTE OF COMMUNICATION AND COMPUTER SYSTEMS (ICCS)
Ankur Lodha	DEEPEN AI (DPEN)

Quality Control

	Name	Organisation	Date
Peer review 1	Emmanuel Arnoux	AMPERE	13/08/2024
Peer review 2	Stefan de Vries	IDIADA	16/08/2024

Version history

Version	Date	Author	Summary of changes
0.1	11/07/2024	All	First draft of document structure.
0.2	13/08/2024	Emmanuel Arnoux	Peer review to the document.
0.3	16/08/2024	All	Address first reviewer's comments.
0.4	23/08/2024	All	Addressing second reviewer's comments.
0.5	23/08/2024	All	Addressing second reviewer's comments.
1.0	30/08/2024	Fadi Alakkad	Final version ready for submission

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.

Copyright © SUNRISE Consortium, 2024.

TABLE OF CONTENTS

EXECUTIVE SUMMARY	10
1 INTRODUCTION.....	11
1.1 Project Introduction.....	11
1.2 Context within Work Package 6	12
1.3 Purpose of deliverable	13
1.4 Intended audience	14
1.5 Structure of the deliverable and its relation with other work packages/deliverables	14
1.5.1 Link to task 5.1	15
1.5.2 Link to task 5.2	15
1.5.3 Link to task 6.1	16
1.5.4 Link to task 6.2	16
1.5.5 Link to task 6.4	16
2 SUNRISE DATA FRAMEWORK.....	17
2.1 Introduction.....	17
2.2 Requirements for SUNRISE DF.....	17
2.3 SUNRISE DF architecture	18
2.4 Storylines.....	21
2.4.1 Storyline #1 Onboarding a new user	21
2.4.2 Storyline #2 Onboarding a new SCDB.....	23
2.4.3 Storyline #3 Searching scenarios in the SCDBs	24
2.4.4 Storyline #4 Pushing scenario to the SCDBs.....	25
2.4.5 Storyline #5 Apply scenarios to specific test cases	27
2.5 SCDBs Access	28
2.6 Data format.....	29

3	METHODOLOGY FOR SCDB APPLICATION FOR GENERIC USE CASES...	32
3.1	Introduction.....	32
3.2	Application Pipeline	32
3.2.1	SUNDRISE DF Landing page	32
3.2.2	Scenario Search.....	33
3.2.3	Automated Query Criteria Generation (AQCG).....	35
3.2.3.1	Introduction.....	35
3.2.3.2	Scope	35
3.2.3.3	Data formats and ontology	36
3.2.3.4	Assumptions.....	37
3.2.3.5	Design	38
3.2.3.6	Validation strategy.....	41
3.2.3.7	Limitations and Future work.....	45
3.2.4	Retrieving Scenarios	45
3.2.5	Scenario Validation	46
3.2.6	Integrating test environment.....	47
3.2.7	Concept and Development.....	48
4	CONCLUSIONS.....	51
5	REFERENCES.....	53
	ANNEX 1: VALIDATION EXAMPLES FOR AUTOMATED QUERY GENERATION TOOL	54
	ANNEX 2: EXAMPLES OF SCENARIO DATABASES FUNCTIONS.....	67

LIST OF FIGURES

Figure 1: The SUNRISE Safety Assurance Framework with its main parts.	12
Figure 2: SUNRISE Data Framework architecture.	19
Figure 3: SUNRISE DF Storyline #1 – onboarding a new user	22
Figure 4: SUNRISE DF Storyline #2 – onboarding a new SCDB	24
Figure 5: SUNRISE DF Storyline #3 – search content in SCDBs.....	25
Figure 6: SUNRISE DF Storyline #4 – input content into SCDBs.....	26
Figure 7: SUNRISE DF Storyline #5 – apply scenarios to specific test cases.	27
Figure 8: Integration of SCDB into SUNRISE DF via Automatic IdP Method.....	29
Figure 9: Integration of SCDB into SUNRISE DF via Manual API credential method	29
Figure 10: SUNRISE DF Dashboard	33
Figure 11: SUNRISE DF Search UI	34
Figure 12: A high-level conceptual view of the developed tool.....	35
Figure 13: An example of a possible workflow and user interactions when using the query generation tool.....	36
Figure 14: Functional architecture of the tool (extended with some data formats).....	39
Figure 15: Test environment and its interaction with framework	48
Figure 16: Concept test environment and its interface for testing scenarios.....	49
Figure 17: An example of RESTful API end points of executing scenarios in Esmini	50
Figure 18: Scenario Center	67
Figure 19: Query concept of Scenario Center	68
Figure 20: Scenius.	69
Figure 21: Safety Pool™ Scenario Database	70
Figure 22: Safety Pool™ Scenario Database’s query criteria.....	71
Figure 23: TNO Streetwise tool chain and workflow	71
Figure 24: High level components and services of the StreetWise tool chain	72

LIST OF TABLES

Table 1: Different approaches to assess the quality of the retrieved scenarios	41
--	----

ABBREVIATIONS AND ACRONYMS

Abbreviation	Meaning
ADAS	Advanced Driver Assistance System
ADS	Automated Driving System
API	Application Programming Interface
ASAM	Association for Standardizations of Automation and Measuring Systems
AQCG	Automated Query Criteria Generation
CCAM	Cooperative Connected and Automated Mobility
CSV	Comma-separated Values
D	Deliverable
DDT	Dynamic Driving Task
DF	Data Framework
DTO	Data Transfer Object
EC	European Commission
EU	European Union
Euro NCAP	European New Car Assessment Programme
GUI	Graphical User Interface
JSON	JavaScript Object Notation
ODD	Operational Design Domain
ODR	OpenDrive
OEMs	Original Equipment Manufacturers
OSC	OpenScenario
RE	Requirement
RESTful API	REpresentational State Transfer Application Programming Interface

SAE	Society of Automotive Engineers
SAF	Safety Assurance Framework
SCDB	Scenario Database
SDL	Scenario description language
SiL	Software-in-the-Loop
SUT	System Under Test
WP	Work Package

EXECUTIVE SUMMARY

The SUNRISE project aims to accelerate the safe deployment of Cooperative, Connected, and Automated Mobility (CCAM) technologies by developing and demonstrating a Safety Assurance Framework (SAF). The project addresses the challenges of safety assurance in CCAM systems by focusing on a mixture of physical and virtual testing, scenario databases, validation methods, and harmonisation of standards.

This deliverable, presents the "Methodology for SCDB application for generic use cases." It provides a comprehensive methodology for integrating the external Scenario Databases (SCDBs) within the SUNRISE Data Framework for testing and validation operations, incorporating discussions and details relevant to the entire SUNRISE Safety Assurance Framework.

A key outcome of this deliverable is a well-defined process for integrating external SCDBs into the testing and validation operations using the SUNRISE Data Framework. This process ensures that SCDBs can be effectively utilized within the framework, promoting consistency and efficiency in the safety assurance of CCAM systems. Additionally, the deliverable addresses key aspects for the implementation and application of external SCDBs, such as query process definition, data formats, and specific use case requirements as defined in WP7.

Furthermore, this deliverable includes a comprehensive overview of the SUNRISE Data Framework, highlighting its unique features and the role it plays in providing access to external SCDBs. It also emphasizes the importance of inter-task discussions within WP6 to support ongoing and future work in the project.

The deliverable serves as a fundamental source for various work packages within the SUNRISE project and will be used in tasks related to data framework harmonisation, architecture development, and the development of required standards on database interfaces. Moreover, the results presented in this deliverable are of significant interest to Scenario Database (SCDB) hosts.

For SCDB hosts, this document offers a clear and detailed explanation of how the methodologies and processes outlined can be leveraged to enhance their existing scenario databases systems and operations. Specifically, it provides actionable insights on how to integrate their databases within the SUNRISE Data Framework, ensuring compatibility and consistency with CCAM systems' safety assurance requirements. The deliverable also addresses key aspects such as the definition of query processes, adherence to standardized data formats, and alignment with specific use case requirements as outlined in WP7.

Overall, the SUNRISE project strives to establish a comprehensive Safety Assurance Framework and a federated scenario databases approach 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 Cooperative, Connected, 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 already 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 in this mixture for cost efficiency reasons.

Worldwide, several initiatives have started to develop test and assessment methods for automated driving functions. These initiatives have already moved 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 safety assurance. Nevertheless, the fact that there is a lack of a common European validation framework and homogeneity regarding validation procedures to ensure safety of these complex systems, hampers 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 whole 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 are another issue tackled by several initiatives and projects, providing silo solutions. A clear concrete approach should be used (at least at the 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 in order 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 initiatives as a baseline, it becomes necessary to move to the next level in the concrete specification and demonstration of a commonly accepted Safety Assurance Framework (SAF) for the safety validation of CCAM systems, including a broad portfolio of use cases and comprehensive test and validation tools. This will be done in SUNRISE, which stands for Safety assUraNce fRamework for connected, automated mobility SystEms.

The Safety Assurance Framework is the main product of the SUNRISE project. This framework takes a central role, fulfilling the needs of different automotive stakeholders that all

have their own interests in using it. 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 by creating and sharing a European federated Data Framework centralising detailed scenarios for testing of CCAM functions and systems in a multitude of relevant test cases with standardised, open interfaces and quality-controlled data exchange.

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 Data framework design and usage definition.

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 implements the “Store” part, as can be seen in the following figure.

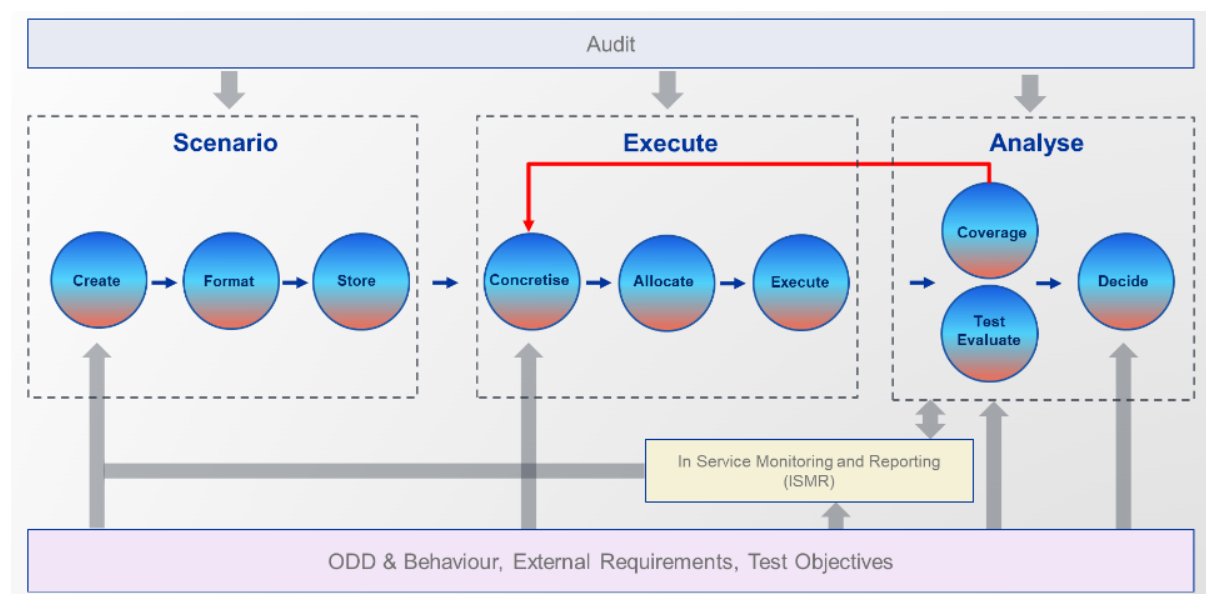


Figure 1: The SUNRISE Safety Assurance Framework with its main parts.

WP6 establishes the design to start the developments and implement a prototype of the SUNRISE DF and thereby materialize the federation layer.

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):** A database of scenarios (a description of a temporal and spatial traffic constellation). Existing examples of SCDBs are Safety Pool™, Streetwise, Scenius, Adscene (see more details on these SCDBs in Annex 2).
- **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 actually 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:

- 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 prototype.

Therefore, within WP6, the activities carried out in task “T6.3 SCDB application for generic use cases” are reported in this deliverable (D6.1 Methodology for SCDB application for generic use cases).

At the time of writing this deliverable (M24), the architecture of the SUNRISE DF has been proposed and implementations of its components are ongoing, with defined functionalities, input/output interfaces, and data flows; also, several storylines have been proposed, that exemplify the most valuable types of actions users can execute with the SUNRISE DF. See section 2 for more details on the architecture, and the storylines.

1.3 Purpose of deliverable

The purpose of this deliverable is to present and describe the "Methodology for SCDB application for generic use cases". Its aim is to describe the process flow using the **SUNRISE DF** for integrating the SCDBs into testing and validation operation. This process includes aspects of query process definition, data format, and specificities of use cases (as defined in WP7).

A key outcome of this deliverable is the establishment of a clear and structured process for integrating external Scenario Databases (SCDBs) into the testing and validation operations within the SUNRISE Data Framework. This outcome ensures that SCDBs are seamlessly incorporated, enabling consistent and efficient safety assurance across CCAM systems

This deliverable is the first one being released from work package 6 (WP6), in month 24 (August 2024). Consequently, inter-task related contents are also reported in this document, to serve as a reference for the rest of the work in WP6, and also to address the need to explain the SUNRISE Data Framework and its features.

Therefore, additionally to the scope of D6.1 to address the methodology for the application of external SCDBs, this deliverable also includes, in **Section 2.1**, a general overview of the concept of the SUNRISE DF.

Further work and implementation details will follow in tasks T6.1, T6.2 and T6.4, and reported in full detail in deliverables D6.2 and D6.3.

1.4 Intended audience.

The intended audience of this deliverable is primarily the rest of WP6 and WP7 as it will describe how a SCBD shall be integrated in testing and validation processes. However, as it presents the backbone for SCBD integration into SUNRISE DF, it should also be relevant for the rest of the project consortium as well as for readers outside the consortium.

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 with CCAM systems' safety assurance requirements, thus playing a vital role in the broader deployment of these technologies.

1.5 Structure of the deliverable and its relation with other work packages/deliverables

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

Chapter 2: SUNRISE Data Framework. This chapter introduces the SUNRISE DF and explains how it provides access to SCDBs and provides guidance on the process of integrating the SUNRISE DF into test and validation operations. It then focuses on the application of SCDBs, detailing their role in streamlining and enhancing the scenario selection process for various testing and validation purposes. Additionally, this chapter outlines the framework's architecture, including both back-end and front-end components, and details its key functionalities, such as user management, scenario searching, and data governance. Moreover, it introduces different user roles and provides specific storylines to illustrate how users interact with the platform, from onboarding to scenario selection and application. The chapter also emphasizes the importance of adhering to established standards to ensure interoperability and ease of integration with various validation tools.

Chapter 3: Methodology for SCDB application for generic use cases. This chapter outlines the methodology for developing and implementing the SCDB application within a generic use case framework. It details the application pipeline, which includes the steps for user interaction with the SUNRISE DF to test systems under test (SUT). The chapter covers the processes for defining query criteria, retrieving and validating scenarios, and managing the integration of these scenarios into the testing environment. Additionally, it introduces tools such as the Automated Query Criteria Generation (AQCG) and Scenario Manager, which streamline scenario retrieval and ensure the accuracy and relevance of the testing data. Finally, it presents the test environment and its interaction within SUNRISE DF.

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

Chapter 5: References. This section provides a list of all references cited in this deliverable.

Annex 1: Validation Examples for Automated Query Generation Tool. This annex provides a detailed evaluation of the Automated Query Criteria Generation (AQCG) tool. It focuses on a small subset of requirements from use case 1.1, selected to validate the quality of the generated query criteria. The full Operational Design Document (ODD) definition for UC1.1, along with the relevant requirements provided to the tool, is presented in Annex 1 - A. This section discusses the generated query criteria, showcasing only the portions necessary for discussion. The complete outputs of the query criteria are available in Annex 1 - B for further reference.

Annex 2: Examples of Scenario Databases Functions. This annex provides examples illustrating the functions of various Scenario Databases (SCDBs). It covers aspects related to querying, searching, and retrieving scenarios, showcasing how these functions are implemented and utilized within SCDBs.

1.5.1 [Link to task 5.1](#)

The deliverable D5.1 presents a set of comprehensive requirements for the SUNRISE DF and federated scenario databases. For this deliverable, the requirements related to the scenario manager have been used on how to validate the retrieved scenarios, format of the scenarios and scenarios query process. Examples of the requirements are listed below:

- RE26: The input format (from SCDBs to the SUNRISE DF) shall be compatible with agreed standardised formats
- RE52: The SUNRISE DF should ensure output scenarios in format that allows automated execution in simulation environments.
- RE59: The SUNRISE DF shall check if the output of scenario of individual databases are in agreed format/language(s)
- RE58: The input format shall include dynamic objects, map data and weather information and represent changing road network information

1.5.2 [Link to task 5.2](#)

The task T5.2 within the SUNRISE project is concerned with the harmonization of standards and ontologies for the description of scenario database content. These developments are especially relevant for this deliverable as they have a direct impact on the query process being developed here. In order to be able to define a query, a robust ODD ontology description based on agreed on standards that is compatible with the content available in the databases to be queried, is necessary. Such an ontology has been developed within task T5.2 based on requirements defined within task T6.3.

1.5.3 Link to task 6.1

The task 6.1 focuses on developing the necessary standards for database interfaces and processing to facilitate the input of scenario data. As part of this task, a dashboard will be created to link the SCDBs (Scenario Databases) to the federated interface, making it easier for users to search for scenarios. Additionally, methods for data enrichment will be developed, enabling the combination of different data sources. This will be enabled by a harmonized domain ontology that ensures the interoperability of various scenario formats. This development will be directly related to D6.1, helping to understand the design concept of communication between the SUNRISE DF and the connected SCDBs.

1.5.4 Link to task 6.2

The task 6.2 focuses on development of required standards on database interfaces and processing to enable the output of scenario data for testing and validation activities. This development includes creating necessary components for querying databases by using the query and requirement criteria defined by the user.

Development in T6.2 is directly related to this deliverable since it plays a role in querying databases to find relevant content for testing. The aim is to use the Search UI and Query Manager components (shown in figure 4) within task T6.3 and its deliverable D6.1.

1.5.5 Link to task 6.4

The task 6.4 is entitled “T6.4 Sample case of a European Test Case Library”. It’s goal to “to develop the First European Test Case Library Prototype as an operational example of the European level Scenario Data Base initiative [...]”. The description also mentions that a “federation layer will be designed and developed to integrate Scenarios from a variety of existing DDBB”. T6.4 has been re-interpreted focussing on a SUNRISE DF, that implements a federation layer providing access to existing scenario repositories, the so-called Scenario Databases (SCDBs).

Task T6.4 focuses on the actual implementation of the SUNRISE DF, which includes the process flow described in D6.1 about the main topic of task T6.3 “SUNRISE DF generic use cases”.

2 SUNRISE DATA FRAMEWORK

2.1 Introduction

This chapter introduces the SUNRISE Data Framework (DF) architecture, which plays a critical role in supporting Cooperative, Connected, and Automated Mobility (CCAM) validation. It details the requirements for integrating the SUNRISE DF into testing and validation processes, ensuring that users can efficiently access, manage, and utilize scenarios stored in various SCDBs. The chapter also outlines the architecture and components of the SUNRISE DF, emphasizing its role as a centralized platform that enhances scenario selection, compliance, and validation efficiency across multiple stakeholders, including certifying entities, developers, and test engineers. Subsequent sections will provide an in-depth look at the designed storylines which clarify the expected actions that end users can take while interacting with SUNRISE DF.

2.2 Requirements for SUNRISE DF

The requirements for integrating the SUNRISE DF into test and validation activities have been derived from deliverable D5.1 [12]. In D5.1, these requirements are organized into distinct clusters, each representing a specific set of related criteria. The requirements relevant to SUNRISE DF and this deliverable are presented below. These requirements will guide the development of the application pipeline for this task and will be incorporated into the developed storylines. Although we haven't used the exact wording from the requirements in D5.1, we have extracted the core concepts from the requirement titles and descriptions, referencing the relevant cluster numbers. This approach has been taken specifically for the purposes of this deliverable, ensuring that the requirements are tailored to the context and objectives of the work being carried out.

Requirements related to Scenario Filtering/Searching

These requirements are crucial for the efficient search and retrieval of scenarios from the database. They guided the development of the Automated Query Criteria Generation (AQCG) tool, which was utilized in Storyline #2 and will be detailed in section 3.2.3. Additionally, the filtering requirements played a key role in the development of Storyline #3. The following requirements have been derived from D5.1 (Cluster 9) and are informed by the expertise of partners involved in the validation of CCAM systems. The key requirements include:

1. SUNRISE DF should allow users to locate and retrieve scenarios based on various criteria according to the following sub requirements:
 - a. Query scenarios by scenario category such as cut-in, cut-out, lane changing, turning left/right, etc.
 - b. Query scenarios by CCAM system such as ABS, ACC, LKA, LDW, etc.
 - c. Query scenarios by its source: real-world (FOT, proving ground, NDS), artificial, accident/insurance data, ...
 - d. Query scenarios by region (country, continent, etc.).
 - e. Query scenarios by certain regulations and standard/norms.

- f. Query scenarios by a specific ODD and OD (e.g., type of traffic area, traffic conditions, environment)
2. Sort scenarios in ascending or descending order by ordering parameters such as scenario quality metrics (exposure, probability, criticality, ...), name, date, number of fatalities/injured people, etc.
3. Select the maximum number of scenarios to display per page.
4. Store user queries and queries results for traceability and post-processing analysis of multiple scenario quality metrics, such as scenario coverage in the defined ODD.

Requirements related to standards alignment

The alignment with standards played a crucial role in retrieving and validating scenarios from the SUNRISE DF. As outlined in D5.1 (Cluster 8 : RE26 and RE27), the SUNRISE DF must support exporting scenarios to widely recognized standards formats, such as ASAM OpenSCENARIO and OpenDRIVE.

This requirement has been applied across various storylines, including Storyline #3 and #5, to ensure that the retrieved scenarios adhere to the correct standard format. Additionally, the scenario manager component (see section 3.2.5) has been utilized to validate the integrity of the exported OpenSCENARIO and OpenDRIVE files, ensuring compliance with these standards.

Requirements related to Database data processing/analytics

The representation of scenarios and metrics is an integral part of the SUNRISE dashboard (see section 3.2.1) and is also utilized in the validation process by the scenario manager to verify and count the returned scenarios.

As derived from D5.1 (Cluster 4: RE9 and RE10), SUNRISE DF should allow for a representation of the scenario and metrics for multiple scenarios according to the following sub requirements:

1. Display the coverage of parameter space of the exported scenarios.
2. Provide data statistics of the scenarios, number of scenarios, parameters distributions, ODD coverage, etc.

2.3 SUNRISE DF architecture

The SUNRISE DF implements the federation layer of SUNRISE to connect to external SCDBs. In summary, the following principles have been used as design guidelines:

- Enable connectivity with heterogeneous SCDBs
- Guarantee data governance, managing user authentication and authorization respecting SCDBs access policies

- Propose harmonised set of standards for scenario content, including data formats, and semantics
- Enhance the efficiency, accuracy, and compliance of scenario selection processes for testing and validation operation
- Implement user applications to effectively access SCDBs through the single SUNRISE DF interface

Figure 2 represents SUNRISE Data Framework architecture.

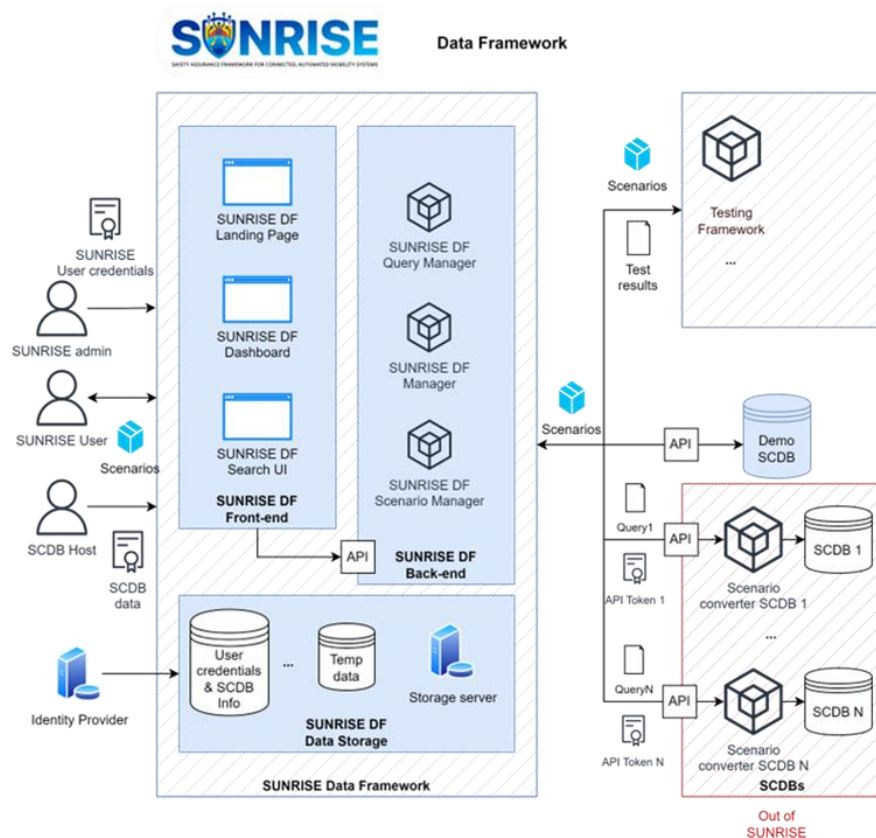


Figure 2: SUNRISE Data Framework architecture.

The SUNRISE DF is envisioned as a cloud application composed by a back-end framework that runs services, exposes APIs (Application Programming Interfaces) and various functionalities, interconnected with a front-end framework that serves web applications to users, to interact with the platform. A set of interfaces connect the back-end with external SCDBs, in order to establish the necessary user access mechanisms. Different user roles are defined in the storylines:

- SUNRISE admin: administrates the platform and manages users and registered SCDBs.
- SUNRISE user: generic user that uses SUNRISE DF to access SCDB content and consumes SUNRISE DF functionalities (e.g., scenario format validation)
- SCDB host: registers SCDB and monitors SCDB connection status and utilisation.

The components of the SUNRISE DF are summarized as follows:

- SUNRISE DF Dashboard – a front-end UI (User Interface) to visualize status of connected SCDBs and other statistics. It operates as the main landing page for all roles.
- SUNRISE DF Search UI – a front-end UI to trigger searches on SCDBs.
- SUNRISE DF Query Manager – a back-end component that manages queries (expansion, verification, formatting).
- SUNRISE DF Scenario Manager – a back-end component to parse query results and check/validate format.
- SUNRISE DF SCDB connection – user access management to SCDBs.
- SUNRISE DF back-end deployment framework – orchestration of back-end components in a cloud environment. Includes the SUNRISE DF API, as the application programming interface that exposes the functions SUNRISE DF expose to its components and front-end application.
- SUNRISE DF front-end deployment framework – integrated front-end platform to organise front-end components.
- SUNRISE DF Data Storage – set of internal databases of the SUNRISE DF to store temporal data, and user data, like the User/SCDB database, which contains information about registered users and SCDBs (see Storylines #1 and #2)
- SUNRISE Demo SCDB – sample mock-up SCDB to exercise storylines and showcase onboarding processes.

Considering the complexity of the SUNRISE DF platform, it is important to highlight the following principles to understand its scope:

- SUNRISE DF offers access to external SCDBs.
- SCDB hosts register their SCDB into SUNRISE, specifying details about the SCDB, specifications of the API, etc.
- Users can access SCDBs for which they already have user credentials (SUNRISE DF does not provide user credentials for SCDBs).
- SUNRISE DF manages the credentials via automated processes (can be seen as a gateway or bypass of credentials).
- SUNRISE DF is not a SCDB, and as such, does not store scenario content.
- SUNRISE DF stores temporarily content from SCDBs while a user session is active, to process query results, or to produce visual output for the front-end applications.
- The SUNRISE Demo SCDB is a mock-up SCDB created for the purpose of illustrating how SCDBs should connect to the SUNRISE DF. The motivation of this Demo SCDB is to serve as an example on: (1) what automated interfaces at SCDB side should look like, (2) how data should be formatted to be compatible with SUNRISE DF, and (3) how to onboard a new SCDB into the registered list of the SUNRISE DF.

The SUNRISE Data Framework (DF) is designed to support all SAF users involved in Cooperative, Connected, and Automated Mobility (CCAM) validation. Its purpose is to centralize scenarios, streamline scenario selection, improve compliance, and enhance validation efficiency. For example, the SUNRISE DF can be used by Euro NCAP test engineers who work with specific test protocols and by homologation test engineers who focus

on regulatory requirements. The SUNRISE DF provides a central repository for a wide array of scenarios from multiple sources, ensuring that users across various sectors such as certifying entities, CCAM developers (OEMs, TIER1s), and other stakeholders can efficiently access and utilize scenarios relevant to their testing and validation processes.

2.4 Storylines

The following storylines have been defined to clarify and exemplify the expected actions users can take when using the SUNRISE DF. These storylines have been discussed and developed during discussions carried out in the broader context of the WP6 activities.

As a whole, the storylines aim to identify the users involved, the expected functionalities, exchanged data packages, and also governance principles such as onboarding processes, authorization and authentication of users.

The storylines are as follows:

- **Storyline #1 – Onboarding a new user**
- **Storyline #2 – Onboarding a new SCDB**
- **Storyline #3 – Searching scenarios on the SCDBs**
- **Storyline #4 – Pushing scenarios to the SCDBs**
- **Storyline #5 – Apply scenarios to specific test cases**

Storyline #5 is particularly relevant for **Task T6.3**, as it covers the application of scenarios to specific test cases. This storyline, along with the others, provides a comprehensive view of how users interact with the SUNRISE DF, from onboarding to advanced scenario management.

Along the storylines, the following type of data is exchanged between components:

- **Credentials/API Keys:** digital artifacts and tokens used to transfer user credentials or other secrets over the network, to get access to specific content or to enable the utilisation of specific API functions.
- **Queries:** reading or writing statements intended to be received by SCDBs to operate actions on the database content.
- **Scenario packages:** set of files containing scenario content, e.g., OSC and ODR files.
- **Scenario Data Transfer Objects (DTO):** an object that defines how the data retrieved from a database will be sent over the network. I.e., a summary of the scenario content, in the form of a metadata object (e.g., a JSON file) to facilitate its light transfer between the network, visualize statistics or other administrative or tag information.

2.4.1 Storyline #1 Onboarding a new user

This storyline focuses on the process to create a new user for the SUNRISE DF. The user must provide credentials to access specific SCDBs.

NOTE: This storyline is being implemented in activities carried out under task T6.4. At the time of writing this deliverable, the implementation is not finished, and changes in the technical details of the storyline can be expected.

Pre-requisites

- The user has access rights to specific SCDBs
- The SCDBs have been registered and connected to the SUNRISE DF

Actors

SUNRISE admin – activates user accounts and validates user requests to onboard

SUNRISE user – registers and logs into SUNRISE DF; provide credentials to specific SCDBs

Diagram

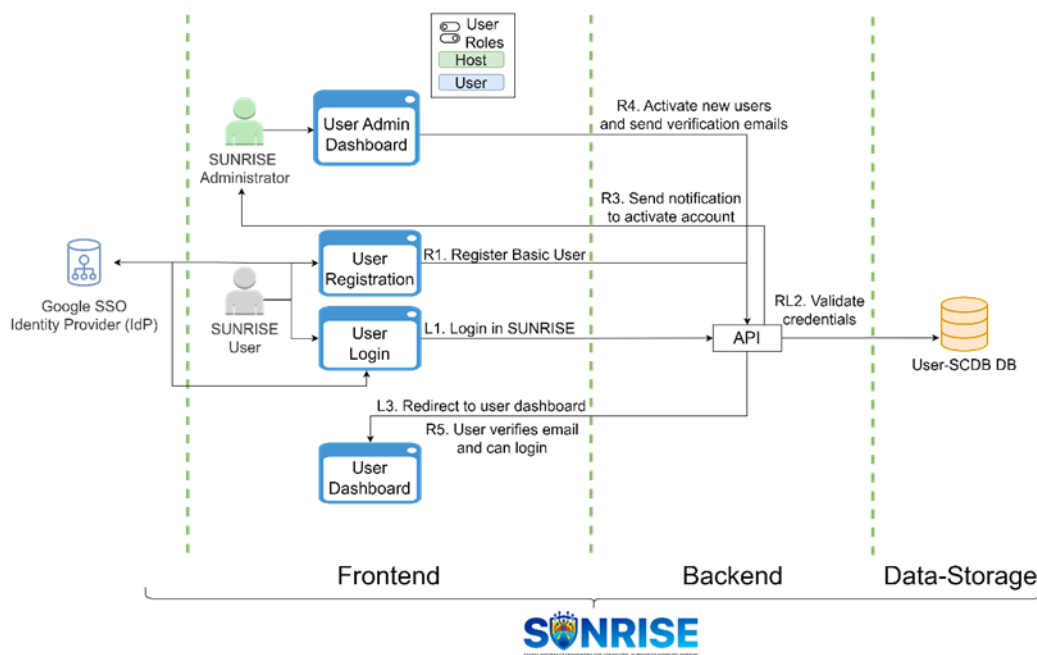


Figure 3: SUNRISE DF Storyline #1 – onboarding a new user

Components

SUNRISE DF Dashboard, User-SCDB DB, external SCDBs

Steps

User Registration

1. Register user.
 - User access SUNRISE DF Dashboard (landing page) to the registration page.
 - There are two ways of register a user:
 - o User fills a registration form with contact details and credentials (name, surname, email, password, company, role in company)
 - o User uses and **Identity Provider (IdP)** to register and completes later the rest of the necessary information for the account (name, surname, company, role in company)

- The registration request is sent to the backend API.
- 2. Validate credentials. The SUNRISE API connects to the SUNRISE DF Data Storage to make sure there is no user with those credentials and information. Then, creates the new account.
- 3. Send notification to activate account. A notification to the SUNRISE administrator will be sent to activate the new account.
- 4. Activate new users and send verification emails.
 - The SUNRISE administrator will activate the new accounts.
 - The SUNRISE administrator will assign the user roles.
 - o Tentative user roles are **User and Owner**
 - Owner: is allowed to register/unregister a SCDB in the system to be accessible by the SUNRISE users.
 - User: is like the normal role, they can access the SCDB where they have access permissions and get data.
 - SUNRISE sends a verification email to the user.
- 5. User verifies email and can login.

User Login

1. SUNRISE Login with credentials. User provides the credentials inside the SURISE login page or uses the IdP to login.
2. Validate credentials. SUNRISE API makes sure the user exists, has its account activated and the email validated in the User-SCDB DB and sends the necessary data to the frontend to perform the login
3. Redirect to user dashboard. The User is redirected to the SUNRISE DF Dashboard user dashboard to perform actions.

2.4.2 Storyline #2 Onboarding a new SCDB

This storyline focuses on the onboarding process of a new SCDB that wants to connect to SUNRISE DF.

NOTE: This storyline is being implemented in activities carried out under task T6.4. At the time of writing this deliverable, the implementation is not finished, and changes in the technical details of the storyline can be expected.

Pre-requisites

None.

Actors

SUNRISE admin – activates user accounts and validates user requests to onboard.

SCDB host – owner or responsible of the SCDB that wants to register the SCDB into the SUNRISE DF.

Diagram

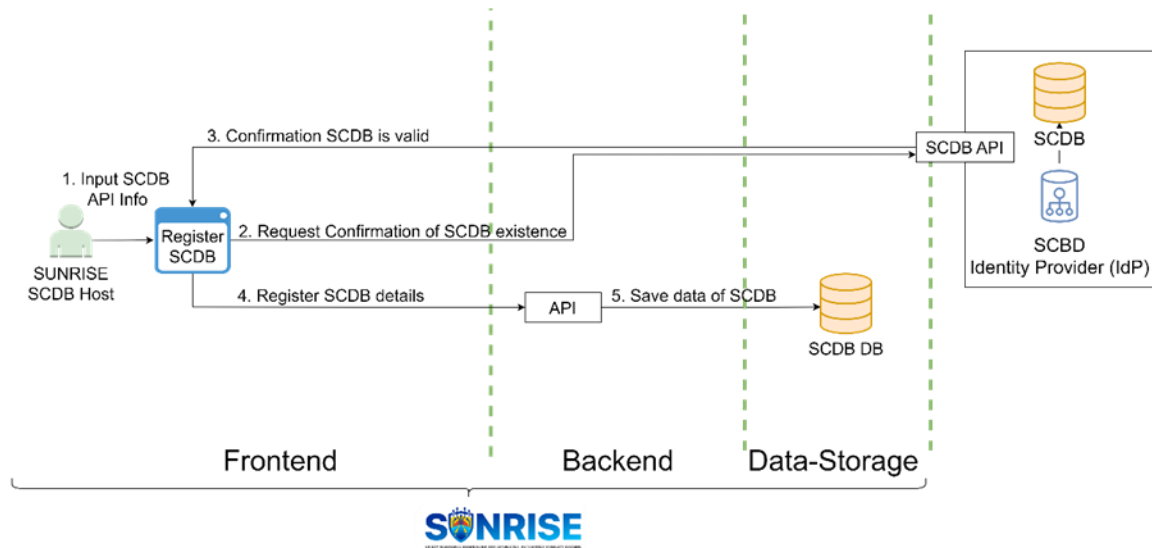


Figure 4: SUNRISE DF Storyline #2 – onboarding a new SCDB

Components

SUNRISE DF Dashboard (Register SCDB in Figure 4), SCDB database (as part of the SUNRISE DF Data Storage), external SCDB

Steps

1. The SCDB host provides information about the SCDB API and details of the SCDB via the SUNRISE DF Dashboard.
2. The SUNRISE DF Dashboard uses the information to connect to the SCDB API and request confirmation of existence.
3. Confirmation of existence is received from the SCDB via the SCDB API.
4. The SUNRISE DF Dashboard forwards details of the SCDB to the SUNRISE DF API.
5. The SUNRISE DF API saves the provided details of the SCDB into the SCDB database (part of the SUNRISE DF Data Storage) to finalize the registration process

2.4.3 Storyline #3 Searching scenarios in the SCDBs

This storyline focuses on the process of searching for scenario content already stored in SCDBs, and the process to find it creating queries using the SUNRISE DF.

Pre-requisites

- The user is registered as a user of the SUNRISE DF.
- The user has reading access rights to specific SCDBs.
- The SCDBs have been registered and connected to the SUNRISE DF.

Actor

SUNRISE DF user. In particular, this user can be understood as “Consumer” of scenario content.

Diagram

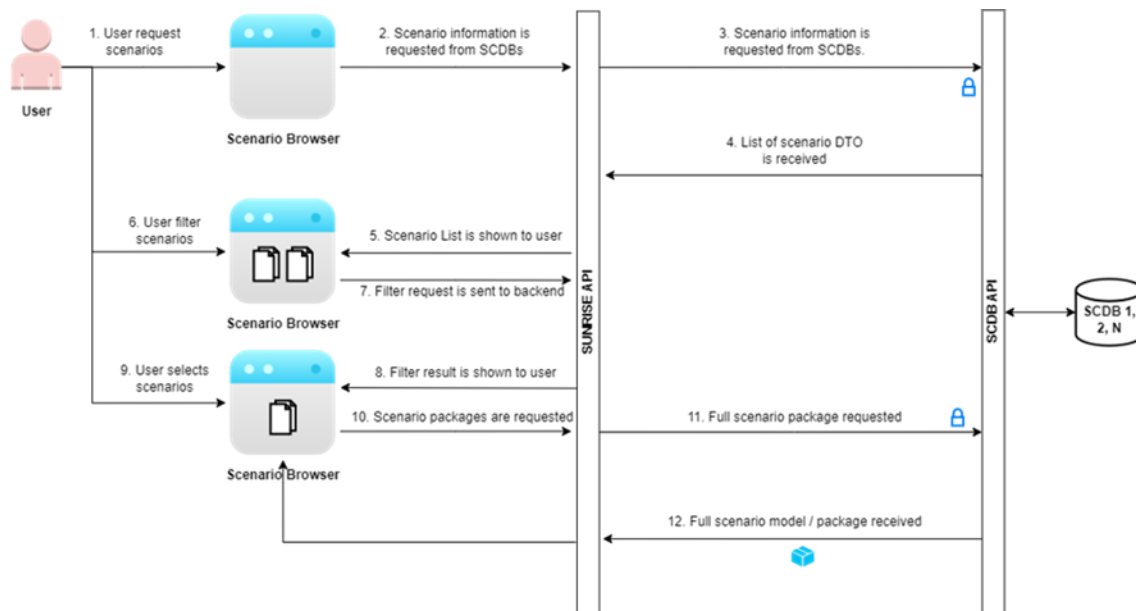


Figure 5: SUNRISE DF Storyline #3 – search content in SCDBs.

Components involved

SUNRISE DF Dashboard (Scenario Browser in Figure 3), SUNRISE API, external SCDBs.

Steps

1. An authorized user requests scenarios that are stored in specific SCDBs using the SUNRISE DF Dashboard. User selects which databases will be queried. NOTE: user has access to these SCDBs as described in Storyline #1.
2. SUNRISE API receives this request from the user and creates proper query commands for each database. Received list of Scenario DTOs, grouped by SCDB and unique ID assigned.
3. Grouped list of scenarios are shown to user in the SUNRISE DF Dashboard. Scenario DTO metadata is visualized.
4. User sends filter request by using filtering pane available in the SUNRISE DF Dashboard. Filtering might include simple tag-based or complex logic.
5. Backend filters scenarios by tags defined by the user. Filter output is shown to the user in the SUNRISE DF Dashboard.
6. User selects scenarios to include in its test planning and send request to retrieve full scenario package. User can select between logical scenario and concrete scenario, and other actions.
7. SUNRISE API receives this request from the user and creates proper commands to interface with the selected SCDBs
8. Scenario packages are received from the responding SCDBs and are downloaded by the user

2.4.4 Storyline #4 Pushing scenario to the SCDBs

This storyline focuses on the upload process of existing scenario content into specific SCDBs connected to the SUNRISE DF.

Pre-requisites

- The user is registered as a user of the SUNRISE DF
- The user has writing access rights to specific SCDBs
- The SCDBs have been registered and connected to the SUNRISE DF
- The SCDBs offer upload/input functions in their API (i.e., the SCDB allows an authorize user to upload content)

Actor

SUNRISE DF user. In particular, this user can be understood as a “Producer” of scenario content.

Diagram

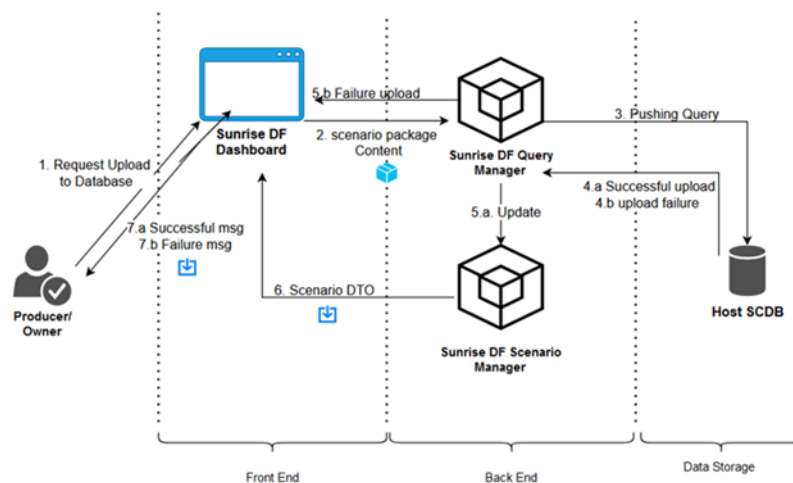


Figure 6: SUNRISE DF Storyline #4 – input content into SCDBs.

Components involved

SUNRISE DF Dashboard, SUNRISE DF Query Manager, SUNRISE DF Scenario Manager, external SCDBs.

Steps

1. A SUNRISE user requests upload to specific SCDB using the SUNRISE DF Dashboard
2. The user uploads a scenario package via the SUNRISE DF Dashboard, reaching the SUNRISE DF Query Manager
3. The SUNRISE DF Query Manager prepares a query to the specifically selected SCDB and runs it against the SCDB API
4. A) The upload is successful, and a success response is obtained from the SCDB API
B) The upload is not successful
5. A) The uploaded scenario package is sent to the SUNRISE DF Scenario Manager for validation
B) Failure upload message is sent to the SUNRISE DF Dashboard

6. A Data Transfer Object (DTO) is sent from the SUNRISE DF Scenario Manager to the SUNRISE DF Dashboard
7. A successful message is displayed to the user, with the DTO information visualized in the SUNRISE DF Dashboard
8. A failure message is displayed to the user, with information about the cause of the failure to upload the scenario package to the SCDB

2.4.5 Storyline #5 Apply scenarios to specific test cases

This last storyline focuses on the practical utilisation of the SUNRISE DF for specific test cases. I.e., this storyline assumes that users are registered, SCDBs have been connected, and mechanisms to search content in SCDBs via the SUNRISE DF are present.

This is the storyline closely related to the activities in T6.3 and this deliverable. Therefore, more details about the content exchanged, the interconnection between the SUNRISE DF and the testing environments are provided in section 3.

Pre-requisites

- The user is registered as a user of the SUNRISE DF
- The user has reading access to specific SCDBs
- The SCDBs have been registered and connected to the SUNRISE DF

Actors

SUNRISE user – this might be either “Consumer” or “Producer” type, as described in Storylines #3 and #4.

Diagram

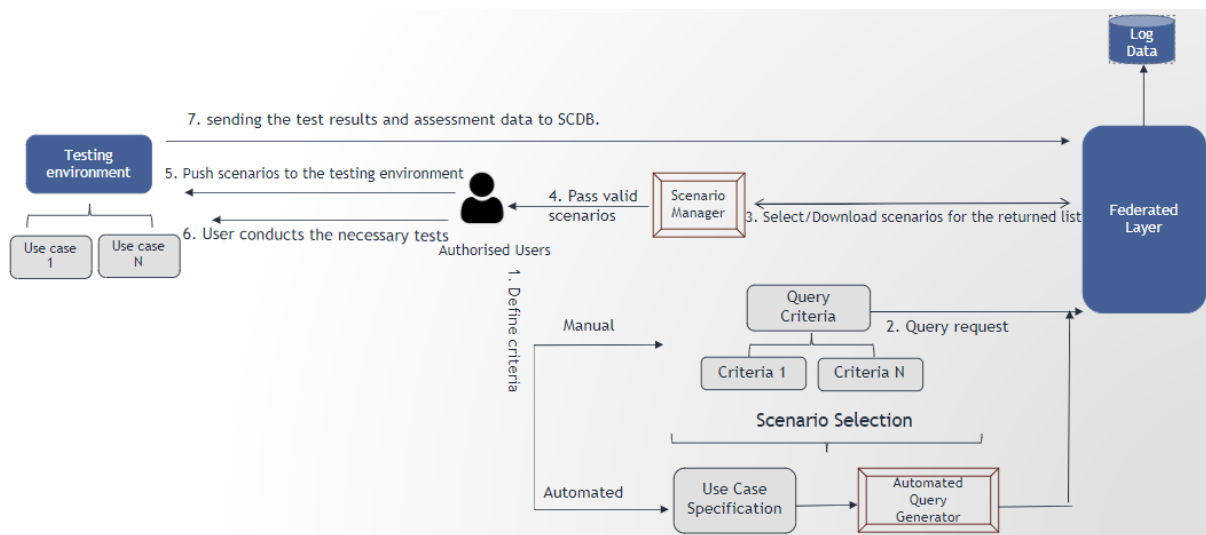


Figure 7: SUNRISE DF Storyline #5 – apply scenarios to specific test cases.

Components

SUNRISE DF Dashboard (omitted in Figure 7 for simplicity, assumed to be the component that provides the user access to the SUNRISE DF Scenario Manager), SUNRISE DF Scenario Manager, external SCDBs (depicted as “Federated Layer” in Figure 7).

Steps

1. Define criteria:
The users access the SUNRISE DF Dashboard to define a query criterion or alternatively provide the use case specifications for an automated query generation.
2. Query request:
The SUNRISE DF Dashboard allows the user to run the scenario selection feature, which convert the defined criteria into a query request for the federated layer.
3. Select/Download the scenarios for the returned list
The SUNRISE DF accesses the SCDBs and return the results, containing metadata and data packages about the scenarios that match the query.
4. Pass valid scenarios:
The SUNRISE Scenario Manager examines the results and validate/checks the format and completeness of OpenX files retrieved from the SCDBs, and then presents them to the user.
5. Push scenarios to the testing environment:
The user initiates the process to push the scenarios into the testing environment.
6. User conducts all the necessary tests:
The user interacts with the testing environment to conduct all the necessary tests
7. Sending the test results and assessment data to SCDB:
If the source SCDB can receive test results, the user can retrieve the results from the testing environment and forward them into the SCDB.

2.5 SCDBs Access

One of the main functionalities of the SUNRISE DF is the access to content that exists in external SCDBs that are connected to SUNRISE DF. According to Storyline #2, different and heterogeneous SCDBs can be connected to the SUNRISE DF. That implies that different possible mechanisms to retrieve content need to be implemented.

At the time of writing this deliverable, the following SCDBs have been studied and are under the process of connection with the SUNRISE DF. These activities are being carried out in the context of T6.4 and will be reported in D6.3.

- Scenius (SCDB host: AVL)
- Streetwise (SCDB host: TNO)
- SafetyPool (SCDB host: UoW/Deepen)
- ADScene (SCDB host: Renault)
- Scenario.Center (SCDB host: IKA)
- SUNRISE Demo SCDB (SCDB host VICOM)

NOTE: The SUNRISE Demo SCDB is a mock-up SCDB created for the purpose of illustrating how SCDBs should connect to the SUNRISE DF. The motivation of this Demo SCDB is to serve as an example on: (1) what automated interfaces at SCDB side should look like, (2) how

data should be formatted to be compatible with SUNRISE DF, and (3) how to onboard a new SCDB into the registered list of the SUNRISE DF.

There are currently two possible ways to integrate a SCDB into the SUNRISE DF (more details will be provided in D6.3 when the implementation is completed):

- Automatic IdP Method:** a SUNRISE user makes use of any of the SUNRISE DF functionalities to access SCDBs (e.g., to retrieve scenario content, as defined in Storyline #2). The SUNRISE DF sends a request to the Identity Provider (IdP) of the SCDB which then replies a request to provide credentials to the user. The user then introduces credentials and the SCDB IdP transfers a token to SUNRISE DF, to be used to perform the queries against the SCDB API. This is the preferred method in terms of security, as it implies no credentials are stored inside the SUNRISE DF, only temporal tokens. The easiness of integration is not ideal, as the SCDB hosts need to implement the SCDB IdP component.
- Manual API Credential Method:** a SUNRISE user makes use of any of the SUNRISE DF functionalities to access SCDBs. SUNRISE DF requests the user to provide an API Key, which the user provides and SUNRISE DF stores in a internal database within its SUNRISE DF Data Storage. SUNRISE DF requests a temporal token to the SCDB API, which is used to interoperate with the SCDB. This approach is easier to implement as it does not imply any changes or additions to the SCDB host, but it is suboptimal in terms of security, as the API Key is stored inside the SUNRISE DF Data Storage.

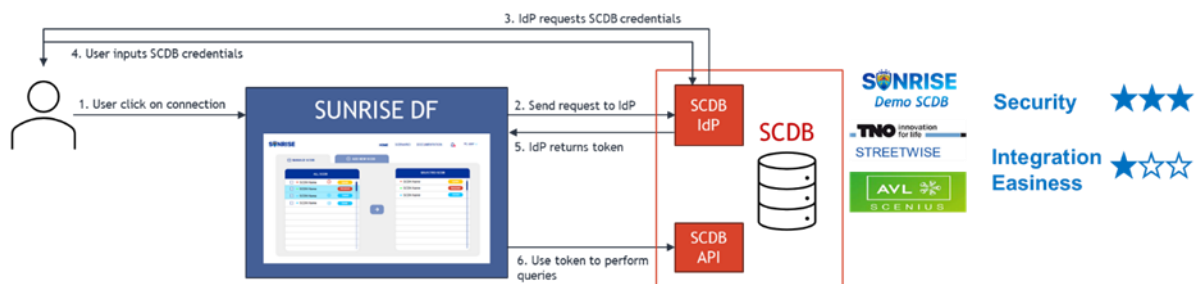


Figure 8: Integration of SCDB into SUNRISE DF via Automatic IdP Method.

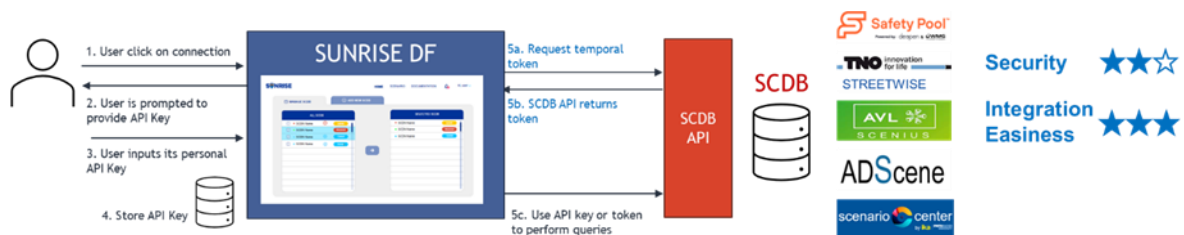


Figure 9: Integration of SCDB into SUNRISE DF via Manual API credential method

2.6 Data format

A variety of data formats are supported to ensure compatibility with different validation tools and systems, facilitate easy access, interpretation, and integration, and meet diverse user needs. The key data formats supported include:

OpenSCENARIO

OpenSCENARIO is a widely recognised ASAM standard for defining the behaviour of dynamic actors in driving simulations [1]. It enables users to create complex traffic scenarios that involve multiple vehicles, pedestrians, and other objects, detailing their behaviours, interactions, and triggers based on specific conditions.

The OpenSCENARIO standard is divided into two distinct branches: OpenSCENARIO DSL and OpenSCENARIO XML. OpenSCENARIO DSL offers a higher-level language for describing dynamic scenario content, including Key Performance Indicators (KPIs) and coverage metrics. This makes it particularly useful for scenarios descriptions requiring in an abstract form.

For this deliverable, however, we focus on OpenSCENARIO XML, which provides a detailed XML schema for specifying scenarios. This format is known for allowing exchange and re-use of scenario descriptions, ensuring interoperability with various partner tools, as well as open-source simulation platforms like CARLA and esmini. By using OpenSCENARIO XML, users can seamlessly execute scenarios across different platforms, enhancing the consistency and reliability of simulation-based testing and development processes.

OpenDRIVE

OpenDRIVE is an ASAM standard for defining road networks in driving simulations [2]. It provides a comprehensive framework for describing the geometric and logical aspects of roads, enabling users to create detailed and accurate virtual environments for vehicle simulations.

OpenDRIVE allows for the precise specification of various road attributes, including lane geometry, road markings, signage, intersections, and more. This detailed representation is needed for accurately modelling real-world driving conditions and supporting CCAM development.

The standard defines roads in terms of a network, where each road segment is described by its shape, lanes, and connections to other segments. This network-based approach ensures that the road layout is consistent and can be easily shared across different simulation platforms. For this purpose, the standard does also specify an XML schema.

BSI Flex 1889

BSI Flex 1889 is a standard for defining a structured natural language format for Level 3+ Automated Driving System (ADS) test scenarios [3]. It provides a clear framework for creating, categorizing, and communicating test scenarios to enhance confidence and safety in automated driving. This standard aids manufacturers, developers, and auditors in compiling and evaluating ADS evidence effectively.

OpenLabel

ASAM OpenLABEL establishes the standards for annotation formats and labelling methods for objects and scenarios [4]. It provides a framework for describing the structure and semantics of labels. This includes a standardized hierarchical taxonomy and a base ontology, allowing for customization and extensions, that ensures interoperability between different tools and platforms.

OpenLABEL also offers detailed guidelines on the correct application of these labelling methods and definitions. It has been used alongside the ISO34503 to describe the dynamic elements of scenarios.

3 METHODOLOGY FOR SCDB APPLICATION FOR GENERIC USE CASES

3.1 Introduction

In order to streamline the use of the SUNRISE DF testing and validation operations, an application pipeline has been developed which consists of modules which facilitate the user interaction with SUNRISE DF for testing their system under test (SUT). This includes defining the query criteria, requesting scenarios, retrieving scenarios, validating them, and pushing the scenarios to the testing environment. It will also cover sending the test results and assessment data to the SCDB. Additionally, as part of this deliverable, we will cover the scenario manager concept, which will check the validity of the retrieved scenario descriptions.

3.2 Application Pipeline

This application pipeline assumes that the user is registered with the SUNRISE DF, has read access to specific SCDBs, and that these SCDBs are registered and connected to the SUNRISE DF.

The steps of the application pipeline are as follows :

- 1- **SUNRISE DF Landing Page:** This is where users select their preferred SCDBs to search for scenarios.
- 2- **Scenario Search:** Users can perform a manual or automated query to search for scenarios that meet their specific use case.
- 3- **Retrieving Scenarios:** This step describes how the SUNRISE DF retrieves scenarios from the SCDBs.
- 4- **Scenario Validation :** This process involves a scenario manager responsible for validating the retrieved scenarios from the SCDBs and verifying the presence and correctness of the required scenario files.
- 5- **Integrating test environment:** This step is crucial for sending the scenario test files to the test environment, executing the tests, and sending the results back to the SUNRISE DF.

More details about each step will be discussed in the next section.

3.2.1 SUNDRISE DF Landing page

As illustrated in Figure 7, the storyline #5 captures the journey of the user when interacting with the data framework to conduct SUT testing. The first step for a user is to login to SUNRISE DF and interact with SUNRISE DF Dashboard. An example of SUNRISE DF landing page has been developed in T6.1, as seen in Figure 10.

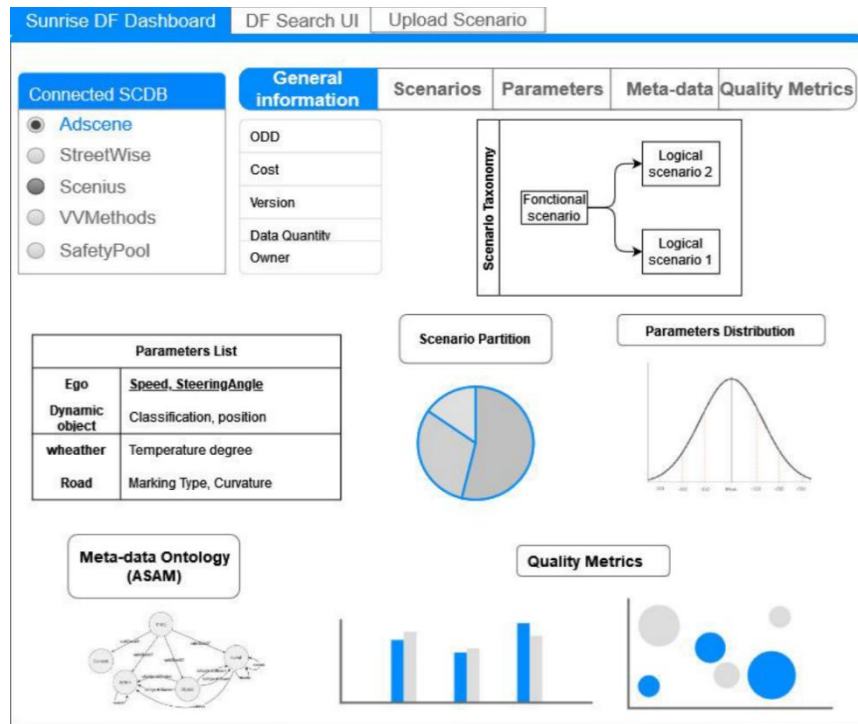


Figure 10: SUNRISE DF Dashboard

The user will be able to select the preferred SCDBs, and other information.

NOTE: This landing page concept is being implemented in activities carried out under task T6.1. At the time of writing this deliverable, the implementation is not finished, and changes in the technical details of the landing page can be expected.

3.2.2 Scenario Search

After selecting the scenario database, users need to navigate to SUNRISE DF search UI in order to search for scenarios. This can be done by two ways manual query and automated query

Manual Query:

This workflow allows the users to select different ODD and behaviour tags that represent the use case through a graphical user interface. As seen below a search UI Mock-up for manual query has been developed in T6.2 and is presented in the following Figure 11.

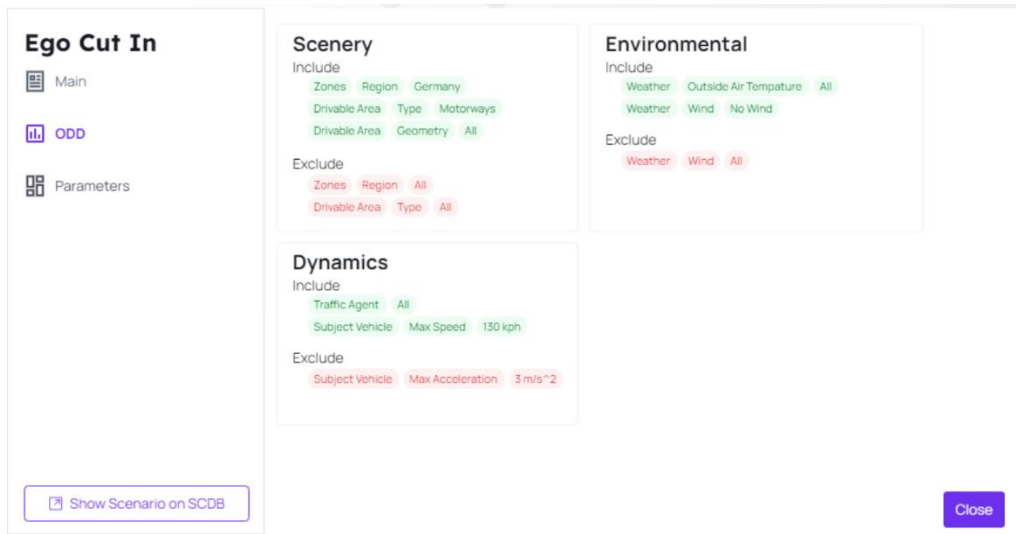


Figure 11: SUNRISE DF Search UI

The query will include scenery, environmental and dynamic elements that represents the scenario. Users can drag/drop or select tags. This process also includes “including” or “excluding” specific tags from the query criteria.

The querying process of the SUNRISE DF is designed to facilitate efficient and standardized access to driving scenario data across multiple partner databases. This process leverages the OpenLabel JSON format, which serves as the primary means of defining queries that are sent through the database framework.

In this framework, queries are structured using the OpenLabel JSON format, which allows users to specify precise tags that correspond to various driving scenarios. These tags are grounded in a common harmonized ontology defined by the SUNRISE project. This ontology ensures consistency and standardization across different databases, making it easier to retrieve relevant data. Additionally, if a target database supports other ontologies, users have the flexibility to incorporate these into their queries, further expanding the scope and specificity of the data retrieval process.

Once a query is defined in the OpenLabel JSON format, it is sent through the SUNRISE DF to the relevant partner databases. The framework acts as an intermediary, managing the interaction between the user's query and the partner databases. Each partner database interprets the query based on the provided tags and ontologies, returning the corresponding driving scenarios that match the query criteria.

NOTE: This search UI Mock-up concept is being implemented in activities carried out under task T6.2. At the time of writing this deliverable, the implementation is not finished, and changes in the technical details of the search UI can be expected.

Automated Query:

Another way to search for scenarios is based on inputs such as the ODD definition, test requirements, dynamic driving task (DDT). For this, an automated query process has been conceptually developed, and will be discussed extensively in the next section.

3.2.3 Automated Query Criteria Generation (AQCG)

3.2.3.1 Introduction

This section presents the development of an automated query criteria generation (AQCG) tool for SUNRISE Data Framework which supports the searching of scenarios in SCDBs for a given use case. The AQCG tool aims at automatic generation of query criteria for searching of scenarios, based on inputs such as the use case ODD, test requirements, dynamic driving task (DDT), etc. The high-level concept of the proposed tool is shown in **Figure 12**. Besides generating query criteria based on user input, the tool also provides feedback to user or the SAF about the resultant query criteria or any missing or inconsistent information in the inputs.

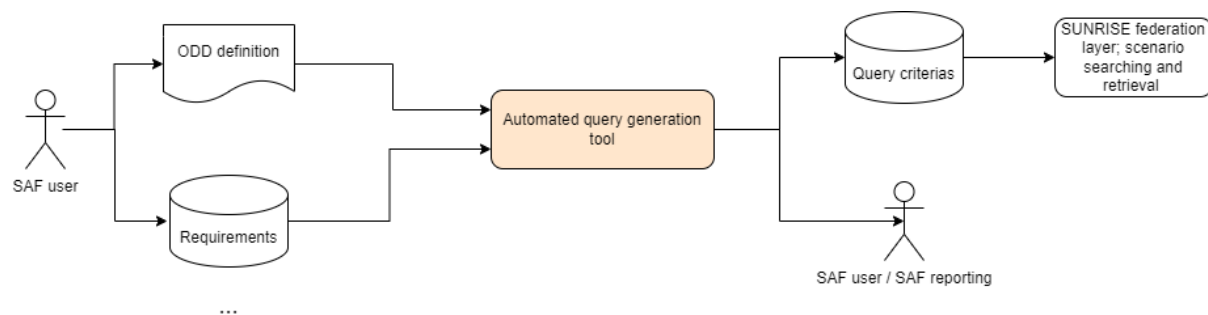


Figure 12: A high-level conceptual view of the developed tool.

The tool significantly reduces manual effort for querying and retrieval of scenarios, given the large number of test requirements for automated vehicles. Furthermore, this improves usability of the federation layer when searching for scenarios; as manual query criteria definition process may be quite cumbersome, for example, due to large number of ODD dimensions. Finally, user errors may occur when manually defining query criteria, therefore the tool can support a robust safety argument. However, while this tool can greatly support and improve the scenario extraction process, it is not necessary for conducting scenario queries and retrieval. Users can still perform scenario extraction without it using the GUIs of the SUNRISE DF. The tool is designed to facilitate and optimize this manual process.

3.2.3.2 Scope

When defining the scope, *UC1.1: urban perception testing*, defined in Sunrise D7.1 was used as an example to understand how use case requirements could be considered within the design of the tool as well as the outputs which may be expected from the tool given certain requirements.

From the analysis, it is clear that meaningful query criteria cannot be automatically generated for all use-case requirements. To facilitate the query criteria generation, a requirement must

be described precisely and must include some information related to *dynamic behaviours* or *ODD elements*. For example, the following requirements provide little information about how the requirement should be tested:

1. *UC1.1_REQ_SA_013: perception system shall run in real time* – it is not clear what should be tested. Should the perception algorithms be stress tested, e.g., in highly-cluttered scenes, or should the system be tested under high computational load, e.g., when many other modules or active, or must the system performance be tested in highly urgent situations, e.g., a pedestrian suddenly entering host path from an occluded position?
2. *UC1.1_REQ_SA_002: Apply ISO21448 and analyse safety in use (SOTIF)* - a very high-level requirement, not directly testable. Instead, this must be treated as a process requirement.
3. *UC1.1_REQ_SA_005: Perception system shall detect and track vehicles and pedestrians* – As this is again a high-level requirement, very little information can be extracted for the query, for example, the query criteria must define relevant actors to be vehicles and pedestrians. However, such a query would lead to extracting a very large number of scenarios from the SCDBs and does not lead to an effective search.

Thus, from these examples, it is clear that an input requirement must be well-defined to facilitate a precise query criteria output from the tool. It may be beneficial for the tool to reject requirements such as (1) and (2) above, since no query criteria can be defined for such requirements. Such feedback could be provided to the SAF. For requirements such as (3) above, a query criterion can be generated but lead to an enormous number of possible scenarios. Thus, it may be beneficial for the user of the federation layer to have the possibility to view and refine the query criteria, once generated, if desired. **An example user workflow** when engaging with the tool is shown in **Figure 13**.

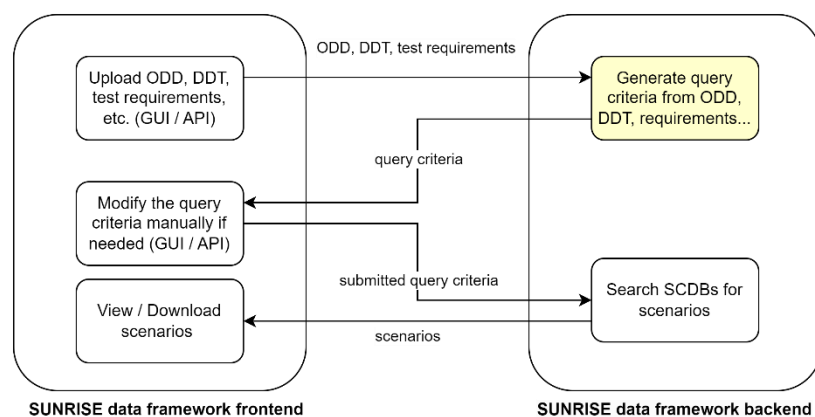


Figure 13: An example of a possible workflow and user interactions when using the query generation tool.

3.2.3.3 Data formats and ontology

The AQCG tool requires inputs from the user including the ODD definition and requirements. The data formats and ontologies for these inputs must be aligned with the rest of the SAF, such that there is a seamless data exchange and interfacing with different SAF modules.

Likewise, the tool will output a query criterion to be used by the federation layer for searching of scenarios. Thus, the output format and ontology must match the expected format by the federation layer.

Data formats and ontology for the following **inputs** need to be aligned with SAF:

- **ODD definition:** Although initially the ASAM OpenODD standard seemed the perfect fit given its simulation-oriented purpose; the standard is not yet released. Therefore, in alignment with D5.2 which aligns on data formats across the SAF, the ODD definition is expected in the ISO 34503 ODD definition language and ontology. Although the ODD definition language in this standard is natural language based, it has a simple grammar and only a few types of statements. Therefore, it can be parsed in a straightforward manner.
- **Requirements:** For requirements, it was agreed that a JSON file format would be used, with the ODD elements covered in the requirements to be described with the ISO 34503 ontology and the dynamic behaviour elements to be described using the ASAM OpenLabel standard. As the OpenLabel standard allows for adding custom tags, these can be added if the requirements relate to behaviour which cannot be described using the existing tags.

Data formats and ontology for following **outputs** need to be aligned with SAF:

- Query criteria: In alignment with D6.2, the ASAM OpenLabel standard was chosen for the query criteria output. A JSON file format is agreed. One important note is that instead of BSI 1883 ontology which Openlabel currently supports, the ISO 34503 ontology will be used for the query criteria. This is motivated due to the planned revisions of both OpenLabel and BSI 1883 to be more in line with ISO 34503. Thus, with ISO 34503 as the ontology for the query criteria, the tool is made more future-proof.

3.2.3.4 Assumptions

A key assumption made in the tool development was that the contents which are required in the query criteria by the federation layer would be standardized and known at an early stage of the tool development. However, this information is not available during the timeline of the tool development and therefore assumptions must be made about the contents of the query criteria which are expected by the federation layer.

Two main inputs are considered to formulate the assumptions; the needs for querying of scenarios as identified in **section 2.2** and an investigation into how scenarios are queried in one of the partner databases, the Safety Pool™ database [5].

Based on needs as identified in **section 2.2** the query criteria must include dynamic behaviour of both the SUT and target actor and cover common test behaviours (mentioned in **section 2.2** as "scenario category"), the ODD (in **section 2.2** querying by region is highlighted), the dynamic driving task (DDT) and system under test (SUT) (in **section 2.2** CCAM system), and regulations and standards. These requirements on the query criteria output are found to be similar as that of the Safety Pool database, where scenarios can be searched by scenario tags, dynamic behaviours, and are also organized into libraries by DDT / SUT / protocols.

3.2.3.5 Design

In this subsection, the specifications and functional architecture of the tool are presented.

Specifications:

1. **Query criteria is defined per test requirements:** In order to maintain traceability of the retrieved scenarios with respect to the test requirements, there must be a query criteria per test requirement. In addition, creating a combined query criteria would lead to many unnecessary scenarios, as test conditions from each requirement would multiply with others.
2. **Combining of ODD related attributes from ODD and test requirements:** Where an ODD attribute is explicitly covered in a test requirement, the overlap of the attribute values in the requirement is done with the values of the attribute from the ODD definition. When it is not explicitly covered, then all values of an attribute are relevant for testing and are to be included in the query criteria. For example, consider the following requirement: *UC1.1_REQ_SA_011 Perception system robustness: Crossing traffic shall be detected by the radar system in the presence of dense fog.* Based on the ODD ontology, the following requirement specifies the *particulates_type: "fog"*, and the *particulates_intensity: "dense"*. The corresponding UC1.1 ODD definition includes *particulates_type: "fog"* and *particulates_intensity: "clear, moderate, medium, dense"*. Thus, in this case only *particulates_intensity: "dense"* is relevant for the query criteria. For all other ODD attributes in the UC1.1 ODD definition, all values of the attributes are considered in the query criteria, since they are not explicitly specified in the test requirement.
3. **The query criteria are returned as empty when it is unclear how the requirement should be tested and it is reported to the user / SAF –** as described when outlining the scope for the tool, unclear requirements cannot be handled in a meaningful way.
4. **Knowledge-based scenarios as an input (optional):** Within the UC1.1 meetings in WP7, test scenarios have been created at a functional level for the use-case. Such test scenarios, when described using OpenLabel and ISO 34503 in a similar way to requirements, could also be considered as an additional, but optional, input to the tool. This has two motivations; one to search the scenario databases for similar scenarios, and the other to retrieve scenarios at a logical / concrete level from databases in a standardized format. It is noted that the scenarios could also be prescribed within test requirements but can be considered separately as done here in the design.
5. **SUT / DDT / Protocol as inputs (optional):** As scenario databases can contain libraries pertaining to a specific SUT / DDT / test protocol, this information could also be optionally provided as part of the query. These inputs must be considered together with the ODD definition, but do not need to be evaluated together with test requirements. Thus, when these inputs are provided, a query criterion containing these inputs can be formulated together with the ODD definition to search for relevant scenarios, given the SUT, DDT, and relevant protocols. The terminology for the DDT may consider the SAE J3016 standard. For terminology with respect to the SUT, no standards currently exist. In both cases, the terminology used is to be aligned with the

terminology considered in the SUNRISE data framework. This is still to be defined in T6.2.

Functional architecture:

The functional architecture (with data formats) is shown in **Figure 14**. For a given use-case, the user provides an ODD definition, requirements to be tested, and optionally provide functional knowledge-based scenarios, keywords describing the SUT / DDT, for example: urban perception level 4 system (as in UC1.1), and any relevant protocols for the SUT. Both requirements and knowledge-based scenarios must be described using OpenLabel ontology for behaviour tagging and ISO 34503 ontology for ODD related attributes.

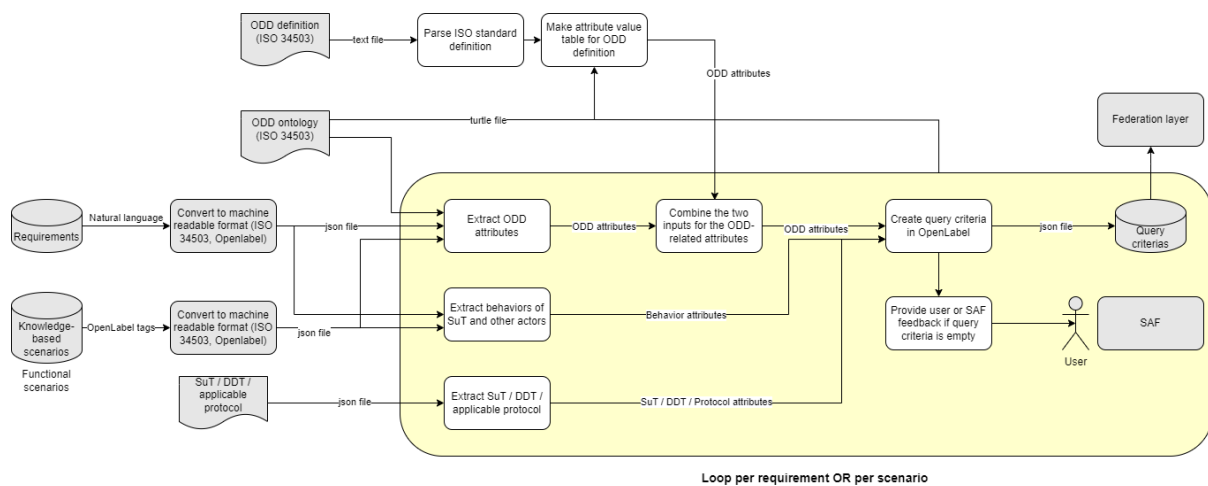


Figure 14: Functional architecture of the tool (extended with some data formats)

A short explanation of the different functional modules is presented as follows:

1. **Parse ISO standard definition:** Parse and extract the statement modifiers, ODD attributes, and the values for the ODD attributes. There are three main modifiers in ISO 34503 language: *include*, *exclude*, and *conditional*. These determine whether a corresponding ODD attribute is part of the ODD or not and when. The ODD attributes and values are as per the ODD ontology.

Example

UC1.1 ODD definition (one line): *Excluded rainfall intensity is [violent rain, cloudburst]*

Parsed: `{"modifier": "excluded", "attribute": "rainfall intensity", "value": ("violent rain, cloudburst")}`

2. **Make attribute value table for ODD definition:** In this step, the modifiers are interpreted to define a table with values for all ODD attributes relevant for the query criteria. To do this, an ODD ontology is needed which lists all possible values of each attribute. This was provided by task T5.2 (D5.2).

Example

Parsed: {"modifier": "excluded", "attribute": "rainfall intensity", "value": ("violent rain, cloudburst")}

ODD taxonomy: {"rainfall_intensity": ("no_rain", "moderate_rain", "light_rain", "heavy_rain, violent_rain, cloudburst")}

ODD attribute table: {"rainfall_intensity": ("no_rain", "moderate_rain", "light_rain", "heavy_rain)}

3. **Extract ODD attributes, behaviors of SUT and other actors, SUT / DDT / applicable protocol:** Reads the json files to extract the relevant information. For improved usability, the ODD ontology is referred to while extracting ODD attributes. This is useful to evaluate attribute values like "any", whereby all possible attribute values of the ODD can be selected. Furthermore, the ODD ontology could also be used to validate the user input to ensure ODD attributes and values are properly specified.

Example

Requirement (natural language): UCI.I_REQ_SA_011: Crossing traffic shall be detected by the radar system in the presence of dense fog.

ODD attributes (as per ISO 34503): {"particulates_type": "fog", "particulates_intensity": "dense"}

Dynamic behaviour tags (as per Openlabel ontology): {"behaviour": "cross", "road user": "vehicle"}

SUT: {"Radar"}

4. **Combine inputs for ODD-related attributes:** Here, the ODD attributes from the ODD definition and that from a requirement or a scenario are combined to create the list of values for each ODD attribute to be part of the query. The combination is done as per **specification 2**.

Example:

ODD attributes from definition: {"particulates_type": ("fog", "non precipitating water droplets"), "rainfall_intensity": ("no_rain", "moderate_rain", "light_rain", "heavy_rain)}

ODD attributes from requirement: {"particulates_type": "fog", "particulates_intensity": "dense"}

Combined ODD attributes: {"particulates_type": "fog", "particulates_intensity": "dense", , "rainfall_intensity": ("no_rain", "moderate_rain", "light_rain", "heavy_rain)}

5. **Create query criteria:** A json file is prepared for each requirement or scenario with data formats as discussed in the data formats subsection. In addition, a query criterion is generated containing the DDT, SUT, protocol information (when provided) together with the ODD.
6. **Provide user or SAF feedback:** If the query criterion is empty for a given requirement, the requirement may be under-specified, unclear, or incorrect with respect to the specified ontologies. This feedback is provided to the user and reported as part of the overall assessment of the system.

3.2.3.6 Validation strategy

The main validation step for the AQCG tool is to test its use for one of the use cases defined in WP7. The querying mechanism in the SUNRISE DF is not developed yet and therefore the tool cannot be integrated and tested with the data framework yet. Therefore, a different approach is needed for validation. Two possible validation approaches are listed as follows:

1. Manually assess the quality of the automatically generated query criteria for a small set of examples. The automatically generated query criteria are compared with manually generated query criteria for each ODD and test requirement combination.
2. Apply the generated query criteria for a small set of examples to one of the partner databases and assess the quality of the retrieved scenarios. Here, scenario quality metrics, such as those being investigated in D5.3, shall be used to assess properties such as coverage, diversity, and representativeness of the scenarios against the specified ODD and requirement.

The pros and cons for each approach are presented in **Table 1**.

Table 1: Different approaches to assess the quality of the retrieved scenarios

Name	Pros	Cons
1. Assess quality of generated query criteria	<ol style="list-style-type: none"> 1. Simple approach 2. Analysis focuses directly on the output of the tool, the generated query criteria. 	Subjective assessment of the test engineer.
2. Assess quality of retrieved scenarios using the query criteria.	<ol style="list-style-type: none"> 1. Objective metrics may be applied for quality assessment. 2. Analysis focuses on end result for user (retrieval of scenarios). 	Indirect assessment, since the quality of the retrieved scenarios depends also on the availability of scenarios in the partner database.

As validation approach 2 suffers from a major disadvantage that the assessment results depend additionally on the availability of scenarios in a given database, approach 1 has been chosen as the main validation strategy. However, it is to be noted, that additional validation may be performed also with respect to the quality of the retrieved scenarios, once the tool can be integrated in the federation layer. A small subset of requirements from UC1.1 are selected to validate the quality of the generated query criteria. The full ODD definition for UC1.1 which is also provided to the tool, together with the requirements, is shown in **Annex 1 - A**. In this section, we discuss the generated query criteria. Only small portions of the output query criteria which are necessary for discussion are shown below. The complete query criteria outputs are provided for reference in **Annex 1 - B**.

1. UC1.1_REQ_SA_005: Perception system shall detect and track vehicles and pedestrians.

ODD and behavior attributes extracted from requirement (input to tool):

ODD: {agent type: motor vehicles, vulnerable road users},

Behavior: []

Generated query criteria (extract which is discussed below only):

```
"Agent_type": [  
    "Motor_vehicle",  
    "Vulnerable_road_users"  
],
```

Validation of generated query criteria:

For this requirement, values for only the agent type attribute are specified. Since the specified values for the attribute are same as the ODD definition, the output query criteria also include the complete set of values in the ODD for attribute agent type. Furthermore, since no other attributes are explicitly mentioned, the rest of the ODD is also part of the query criteria.

2. UC1.1_REQ_SA_009: Detection of semi-occluded pedestrian in front or aside of crossing vehicle.

ODD and behavior attributes extracted from requirement (input to tool):

ODD: {agent type: vulnerable road users},

Behavior: ["MotionCross", "Occluded"]

Note that "occluded" is not part of the OpenLabel ontology, thus the ontology must be extended and aligned with the federation layer.

Generated query criteria (extract which is discussed below only):

```
"Agent_type": [  
    "Vulnerable_road_users"  
  ],  
"Behavior": [  
    "MotionCross",  
    "Occluded"  
  ]
```

Validation of generated query criteria:

Based on the test requirement, only vulnerable road users are made part of the query, although motor vehicles are also part of the ODD. It is challenging to capture implicit needs of the requirement. For example, often times, pedestrians may be occluded by other road users such as motor vehicles. However, as the requirement concerned itself with pedestrians only, motor vehicles although part of the ODD, were excluded from the query criteria. This limitation may however be addressed by the behaviour tags, as these may prompt selection of scenarios with different types of entities occluding the vulnerable road user.

3. UC1.1_REQ_SA_010: Crossing traffic need to be detected fast enough to avoid collisions when entering/passing over the intersection.

ODD and behavior attributes extracted from requirement (input to tool):

ODD: {agent type: any, junctions: intersection},

Behavior: ["MotionCross"]

Generated query criteria (extract which is discussed below only):

```
"Agent_type": [  
    "Motor_vehicle",  
    "Vulnerable_road_users"  
  ],  
"Junctions": [  
    "Intersection"  
  ],
```

Validation of generated query criteria:

In this example, the agent types required to test the requirement cover the entire ODD set for the agent type attribute. For junctions, the query is restricted to junctions, as required for testing the requirement.

4. UC1.1_REQ_SA_011: Crossing traffic shall be detected by the radar system in the presence of dense fog.

ODD and behavior attributes extracted from requirement (input to tool):

ODD: {agent type: any, particulates type: "fog", particulates intensity: dense particulates},

Behavior: ["MotionCross"]

Generated query criteria (extract which is discussed below only):

```
"Agent_type": [  
  "Motor_vehicle",  
  "Vulnerable_road_users"  
],  
"Particulates_type": [  
  "Fog"  
],  
"Particulates_intensity": [  
  "Dense_particulates"  
],
```

Validation of generated query criteria:

The generated query criteria exclude irrelevant values of the particulates attribute w.r.t. to the requirement; here, non-precipitating water droplets. Furthermore, particulates intensity is unspecified in the ODD definition. Based on the requirement, only dense intensity must be considered. Finally, all road users may be considered part of "crossing traffic," thus all values of the attribute in the ODD definition are included in the query criteria. The rest of the ODD definition is passed as is to the query criteria.

5. UC1.1_REQ_SA_014: Perception system shall work in adverse weather conditions

ODD and behavior attributes extracted from requirement (input to tool):

ODD: {particulates type: any, particulates intensity: any, rainfall intensity: any, Snowfall: any},

Behavior: {}

Generated query criteria (extract which is discussed below only):

```
"Particulates_type": [  
  "Fog",  
  "Non_precipitating_water_droplets"  
],  
"Particulates_intensity": [  
  "No_particulates",  
  "Few_particulates",  
  "Medium_particulates",  
  "Dense_particulates"
```

```
],
  "Rainfall_intensity": [
    "Light_rain",
    "No_rain",
    "Moderate_rain",
    "Heavy_rain"
  ],
  "Snowfall": [
    "Heavy_snow",
    "Light_snow",
    "Moderate_snow",
    "No_snow"
  ],
],
```

Validation of generated query criteria:

The query criteria are populated with all different weather conditions which are part of the ODD.

3.2.3.7 Limitations and Future work

A few limitations of the AQCG tool are noted in this section. Firstly, requirements are usually specified in natural language format. At this moment, they must be manually converted by the user to OpenLabel format, using OpenLabel and ISO34503 taxonomies. However, for ease-of-use, a more automated solution would be desired and may consider technologies such as large language models.

Secondly, more advanced error handling for underspecified or incorrect inputs is desired for the tool. For example, if the ODD definition does not conform to the ISO 34503 standard, the user could be made aware of this error. Therefore, a conformity check module could be implemented for each provided input to the tool.

It was already made clear in the scope of the tool that there may be certain requirements of the use cases as described in D7.1, which are not suitable for automated query generation. This is due to the requirements being at a very high-level, such that no ODD or behavior related information is apparent, or that the requirements are under-specified; unclear how they should be tested. It may be that safety requirements for ADS are typically formulated at a high-level. For example, if we consider the *Guidelines for Regulatory Requirements and Verifiable Criteria for Automated Driving System Safety validation* [6], drafted by the GRVA working group of the UN-ECE, several safety requirements seem to be formulated at a high-level, e.g., 5.10.16: *The ADS shall interact safely with other road users*. It is not trivial to understand the comprehensive set of scenarios which should be tested for such a requirement, and therefore, also challenging to formulate an automated query. Thus, some intermediate steps from the user may be needed when using the tool, such as refining of the requirements.

3.2.4 Retrieving Scenarios

After the defined query criteria, whether manual or automated, are passed on to the DF, the backend modules of the SUNRISE DF pass the query criteria to individual SCDBs and search

for relevant scenarios. The relevant scenarios are then retrieved by the DF from the individual SCDBs and can be provided to the user.

The SUNRISE DF returns a comprehensive JSON file that encapsulates all the necessary elements for reconstructing and analysing the requested scenarios. This JSON file includes encoded versions of the corresponding OpenSCENARIO and OpenDRIVE files, along with metainformation and all the relevant tags present in the scenario based on the user specified ontologies. The metainformation may contain direct links to multimedia recourses, such as photos and videos associated with the scenario. Additionally, the metainformation contains a unique identifier for the database of origin for the scenario, as well as the specific identifier for this scenario in the database. This format is designed with modularity and adaptability in mind, allowing for extensions of the format for specific use cases and applications.

For detailed mechanisms and workflow of the searching and subsequent retrieval of scenarios by the SUNRISE DF from the individual SCDBs, the reader is referred to SUNRISE D6.2.

Before the retrieved scenarios are passed to the user, the scenario format and description shall be checked and validated in the application pipeline. This is important as the SUNRISE DF is a harmonized interface to scenarios in many SCDBs and ensuring the quality of the scenarios being provided to the user is an important responsibility of the DF. This task is performed by the **scenario manager**, which will be discussed in the next section.

3.2.5 Scenario Validation

After the scenarios have been returned from the SCDBs through SUNRISE DF, it is important to ensure the validation of scenarios. The scenario manager component handles this validation.

The Scenario Manager is a back-end API component within the data framework responsible for validating and managing scenarios retrieved from connected SCDBs based on user-defined query criteria. Its primary functions include ensuring the validity of OpenSCENARIO and OpenDrive files, handling errors, and providing processed scenario counts. The Scenario Manager verifies the presence and correctness of the required scenario files, validates them against their respective schemas, and ensures the existence of necessary logic files. It communicates valid scenarios to users, returns failed scenarios with appropriate error messages to the data framework. The main tasks of the scenario Manager are:

- Pass the valid scenarios to the users.
- Return the failed scenarios to the users with appropriate error messages.
- Count the processed scenarios.

The functionalities include:

1. **Validation:**

- Check if both OpenSCENARIO and OpenDrive are present.
- Verify that both OpenSCENARIO and OpenDrive are valid XML.

- Validate OpenSCENARIO against its schema (OpenSCENARIO 1.1).
- Validate OpenDrive against its schema (OpenDRIVE 1.6).
- Ensure the logic file of OpenSCENARIO exists (e.g., RoadNetwork).

2. Error Handling

- Handle errors effectively. The Scenario Manager identifies and manages a wide range of errors, including but not limited to:
 - File Not Found: Reports missing OpenSCENARIO or OpenDrive files
 - Schema Validation Errors: Indicates discrepancies between files and their respective schemas.
 - XML Parsing Errors: Notifies users if the XML structure is invalid.
 - Missing Logic Files: Flags scenarios missing required logic files.
- Return the scenarios count.

The SUNRISE Data Framework facilitates querying scenarios across different SCDBs. The Scenario Manager component will analyse the results of these queries to validate and address any issues within the scenario content. As a key component, the Scenario Manager ensures that the scenario content is harmonized and accurate, making it essential for the activities in Task 6.4.

Additionally, Task 6.4 involves defining the interfaces and API specifications necessary for integrating the Scenario Manager with other components and the Front-End web application. This integration ensures seamless communication and functionality across the entire SUNRISE Data Framework.

3.2.6 Integrating test environment

The integration of test environment is a crucial part for completing the pipeline. The main aim is to transfer test scenario files to test environment and retrieve test results. The process is as follows:

1. Send received and checked scenario files to test environment.
2. Execute tests.
3. Retrieve the test measurement files and send them back to the SUNRISE DF from which they were received.

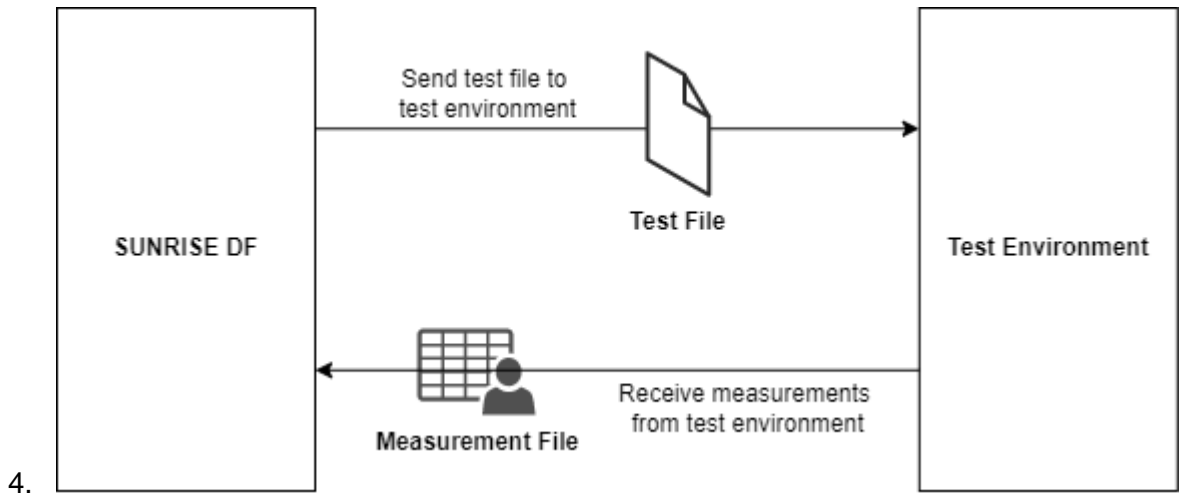


Figure 15: Test environment and its interaction with framework

The Figure 15 above shows the test environment integration and pipeline's interaction by using test files. The test environment's structure may be defined according to planned tests. It might include a 3D environment simulator for pure simulation or simulator coupled with a controller and vehicle dynamics for software-in-the-loop (SiL) tests, etc.

In this context, a concept SUNRISE DF simulation interface is created for testing scenarios queried from SCDBs. The main aim of this concept implementation is to demonstrate the use cases. This test environment concept is explained in the next section.

3.2.7 Concept and Development

A concept SUNRISE DF test integration interface is created for testing scenarios queried from SCDBs.

This concept implementation includes an ASAM OpenSCENARIO / OpenDRIVE compatible simulator and a RESTful API for receiving and providing files.

The Figure 16 shows overall picture of the test environment where simulation files are consumed, and measurement files produced via API.

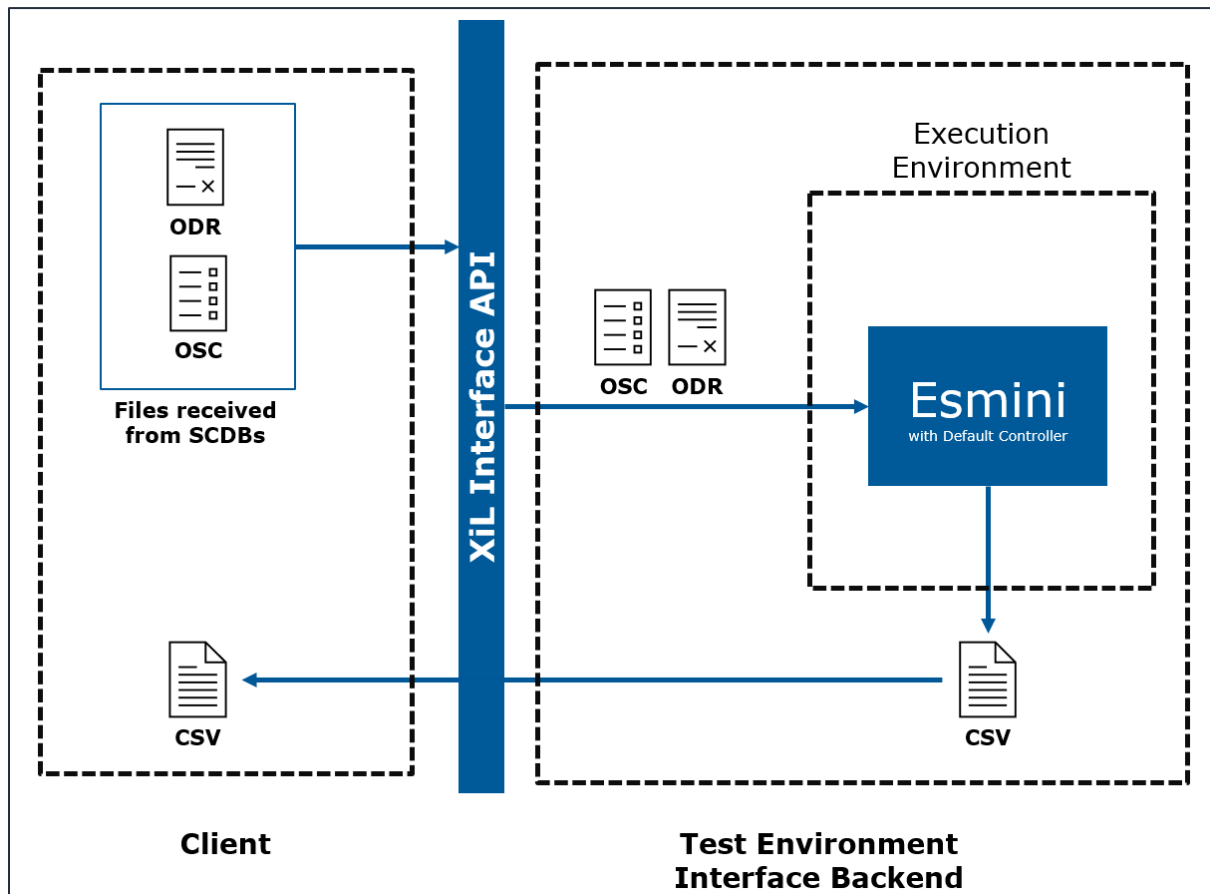


Figure 16: Concept test environment and its interface for testing scenarios.

A RESTful API (Representational State Transfer) is a type of web service that follows a set of principles and constraints designed to enable efficient and scalable communication between systems. RESTful APIs use standard HTTP methods, such as GET, POST, PUT, and DELETE, to allow different software applications to interact over the web. In the context of the test interface, the RESTful API facilitates the exchange of simulation files (such as ASAM OpenSCENARIO and ASAM OpenDRIVE) and returns measurement files in a CSV format, ensuring seamless integration and communication within the execution environment.

Esmini simulator [7] is preferred as simulator in this concept work due to its ease of use and good coverage of ASAM OpenSCENARIO (OSC) and OpenDRIVE (ODR) support. Esmini is also capable of providing simulation measurement files (time series data that contains object states and sensor info) in CSV format. This file can be retrieved after simulation is completed.

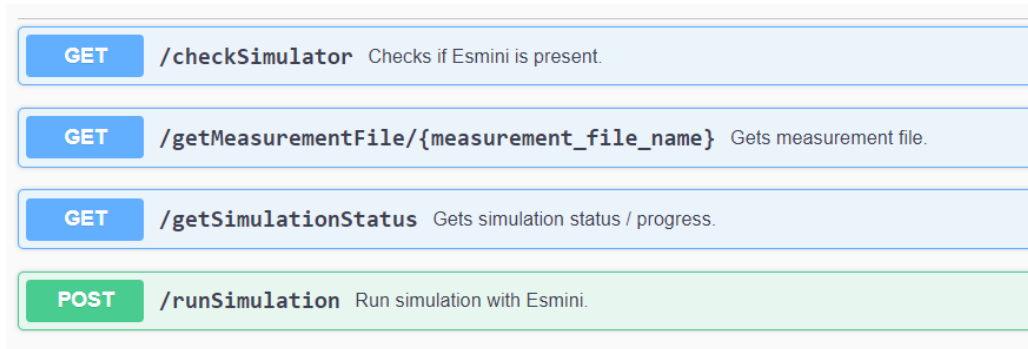
Esmini simulator is chosen as an example simulator for integration, however test environment is not limited with this simulator. The users can use different simulators according to test setup required.

Test interface has a RESTful API that has the following functionalities:

1. Checks if simulator is correctly configured.
2. Start simulations with provided OSC and ODR files and outputs measurement file.

3. Returns simulation status at any time.
4. Retrieving measurement file and providing a name.

Figure 16 shows an example of RESTful API end points (URLs for specific functions) of executing scenarios in the simulator.



GET	/checkSimulator	Checks if Esmini is present.
GET	/getMeasurementFile/{measurement_file_name}	Gets measurement file.
GET	/getSimulationStatus	Gets simulation status / progress.
POST	/runSimulation	Run simulation with Esmini.

Figure 17: An example of RESTful API end points of executing scenarios in Esmini

Test environment and its integration can be considered as the execution component of the workflow and planned to be used for executing scenarios retrieved from SCDBs via queries created in automated query creation tool.

The XiL platform does not include any result management system, but measurement files retrieved from the test environment will be sent to the individual SCDB from which the scenario is queried. The whole process, including sending measurement files back to corresponding SCDB, is being controlled by SUNRISE DF automatically.

4 CONCLUSIONS

This document details the processes and methodologies crucial for leveraging the SUNRISE Data Framework (DF) to improve safety assurance in Cooperative, Connected, and Automated Mobility (CCAM) systems. It presents key findings and methodologies for integrating Scenario Databases (SCDBs) into the SUNRISE DF, which will support subsequent tasks in the SUNRISE project and ensure effective implementation of CCAM systems.

The deliverable provides guidance on incorporating the SUNRISE DF into testing and validation operations. This includes scenario filtering and searching, alignment with standards, and visualization. Additionally, it illustrates the SUNRISE DF architecture, explaining its main components such as user operations, dashboards, and search functionalities. Five storylines are included to clarify and illustrate the potential actions users can take when interacting with the SUNRISE DF. This work will benefit different SUNRISE tasks, such as tasks running in WP5, WP6 and WP7, by providing a comprehensive workflow for integrating SUNRISE DF into a validation and test activities.

Furthermore, the document describes methods for integrating SCDBs into the SUNRISE DF and outlines the various data formats required to ensure compatibility with different validation tools and systems. This will benefit various stakeholders specifically SCDBs host that want to integrate their Databases within SUNRISE DF and will also benefit T6.4 by building a scalable and secure mechanism for connecting SCDBs with SUNRISE DF.

Moreover, the deliverable presents the application pipeline developed to facilitate user interaction with the SUNRISE DF, specifically addressing Storyline #5. This includes the SUNRISE DF landing page where users can select preferred SCDBs and search for scenarios using either manual or automated queries. This will be useful for T5.2, T6.1, T6.3 and T6.4.

An automated Query Criteria Generation (AQCG) tool is introduced to support scenario searching in SCDBs for specific use cases. The AQCG tool will play a significant role in Task 6.4. This deliverable outlines specific formats and queries that will be adopted across the SUNRISE DF, ensuring consistency and reliability in data handling. The deliverable also discusses post-search processes, such as scenario retrieval and validation, and introduces the concept of a scenario manager. The scenario manager concept will benefit T6.4 by developing the tool to validate scenarios. Additionally, the deliverable covers the integration testing environment for sending test scenarios to the test environment, executing tests, and returning results to the SUNRISE DF. This will benefit all SAF users and SUNRISE DF users included in WP4, WP5 and WP6.

The integration methodology developed in this deliverable is critical for various stakeholders, which focuses on establishing standardized data interfaces. By providing a clear pathway for linking SCDBs with the SUNRISE DF, this work enables seamless data exchange and interoperability, benefiting external SCDB hosts and several work packages such WP7 (T7.3) and WP6 (T6.1, T6.2 and T6.4).

Overall, this deliverable marks a significant milestone in the SUNRISE project, laying the groundwork for the continued development and implementation of the SUNRISE DF. Future deliverables, D6.2 and D6.3, will build upon this foundation, detailing further work and implementation specifics in the subsequent phases of the project.

5 REFERENCES

- [1] ASAM, “ASAM OpenScenario,” 2024. [Online]. Available: <https://www.asam.net/standards/detail/openscenario/>. [Accessed 07 July 2024].
- [2] ASAM, “ASAM OpenDRIVE,” 2024. [Online]. Available: <https://www.asam.net/standards/detail/opendrive/>. [Accessed 07 July 2024].
- [3] ISO, “BSI FLEX 1889 - Natural Language Description for Automated Driving Systems,” 2024. [Online]. Available: <https://www.bsigroup.com/en-GB/insights-and-media/insights/brochures/bsi-flex-1889-natural-language-description-for-automated-driving-systems/>. [Accessed 07 July 2024].
- [4] ASAM, “ASAM OpenLABEL®,” ASAM, 12 November 2021. [Online]. Available: <https://www.asam.net/standards/detail/openlabel/>. [Accessed 23 August 2024].
- [5] WMG, “Safety Pool™ Scenario Database,” 2024. [Online]. Available: <https://www.safetypool.ai/>. [Accessed 11 July 2024].
- [6] UNECE, “(GRVA) - Guidelines for Regulatory Requirements and Verifiable Criteria for Automated Driving System Safety Validation,” Geneva, 2023.
- [7] Esmini, “Esmini,” 2024. [Online]. Available: <https://github.com/esmini/esmini>. [Accessed 07 July 2024].
- [8] H. G. C. S. M. W. N. & E. L. Weber, “Holistic Driving Scenario Concept for Urban Traffic. In 2023 IEEE Intelligent Vehicles Symposium (IV). IEEE. <https://doi.org/10.1109/iv55152.2023.10186385>,” *IEEE Intelligent Vehicles Symposium (IV)*, 2023.
- [9] M. G. C. & E. L. Schuldes, “scenario.center: Methods from Real-world Data to a Scenario Database,” *arXiv*, 2024.
- [10] J. K. R. M. T. R. S. V. L. & E. L. Bock, “The inD Dataset: A Drone Dataset of Naturalistic Road User Trajectories at German Intersections,” *IEEE Intelligent Vehicles Symposium (IV)*, 2020.
- [11] S. K. a. P. J. X. Zhang, “Scenario description language for automated driving systems: a two level abstraction approach,” *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 973-980, 2020.
- [12] F. Alakkad, X. Xhang, S. Khastgir, E. d. Gelder, S. v. Montfort, J. L. Mallada, E. v. Hassel, M. Grabowski, o. Beckmann, X. Boabén, A. Nogués, S. Vidal, T. Menzel, I. Panagiotopoulos and A. Ballis, “SUNRISE D5.1 Requirement for CCAM Safety Assessment Data Framework Content,” 2023.

ANNEX 1: VALIDATION EXAMPLES FOR AUTOMATED QUERY GENERATION TOOL

A- ODD definition of UC1.1 (with slight modifications compared to D7.1 to conform to ISO 34503 ontology)

Base state: Restrictive

#Composition statements

Scenery

Included drivable area type is [minor roads, outdoor parking]

Excluded drivable area type is [motorways, radial roads, distributor roads]

Included lane type is [traffic lane]

Excluded lane type is [bus lane, cycle lane, tram lane, emergency lane]

Included direction of travel is [right hand travel]

Excluded direction of travel is [left hand travel]

Included drivable area surface conditions are [dry, wet road]

Excluded drivable area surface features are [cracks, swells]

Included road surface type is [segmented, uniform]

Included horizontal plane is [straight roads, curved roads]

Included vertical plane is [up-slope, down-slope, level plane]

Included transverse plane type is [undivided, pavements]

Excluded transverse plane type is [barriers on road edges]

Included types of lanes together are [traffic lanes]

Included drivable area surface type is [asphalt, concrete]

Excluded drivable area surface type is [cobblestone, gravel, granite setts]

Included drivable area signs are [regulatory, warning, information]

Included traffic information signs are [traffic lights full-time]

Included intersections are [T-junctions, Y-junctions, crossroads, roundabouts]

Included special structures are [tunnels, bridges, toll plazas, pedestrian crossings]

Environmental conditions

Included wind is [no wind, calm, light air, light breeze, gentle breeze]

Excluded rainfall intensity is [violent rain, cloudburst]

Included particulates are [non precipitating water droplets, fog]

Included illumination is [day, night, cloudiness, artificial illumination]

Dynamic elements

Included agent type is [vulnerable road users, motor vehicle]

Excluded special vehicles are [ambulance, police vehicle]

B- Generated query criteria

1. UC1.1_REQ_SA_005: Perception system shall detect and track vehicles and pedestrians.

{

```

"ODD": {
  "Agent_type": [
    "Motor_vehicle",
    "Vulnerable_road_users"
  ],
  "Drivable_area_type": [
    "Minor_or_local_roads",
    "Parking_space",
    "Primary_roads",
    "Shared_space",
    "Slip_roads_or_off-ramps"
  ],
  "Lane_type": [
    "Shared_lane",
    "Special_purpose_lane",
    "Traffic_lane"
  ],
  "Direction_of_travel": [
    "Right_hand_travel"
  ],
  "Drivable_area_surface_conditions": [
    "Dry",
    "Wet_road"
  ],
  "Road_surface_type": [
    "Segmented",
    "Uniform"
  ],
  "Horizontal_plane": [
    "Straight_roads",
    "Curved_roads"
  ],
  "Vertical_plane": [
    "Up-slope",
    "Down-slope",
    "Level_plane"
  ],
  "Transverse_plane_type": [
    "Divided",
    "Pavement",
    "Undivided"
  ],
  "Types_of_lanes_together": [
    "Traffic_lanes"
  ],
  "Drivable_area_surface_type": [
    "Asphalt",
    "Cement_concrete",
    "Pavers"
  ]
}

```

```

],
"Drivable_area_signs": [
    "Regulatory",
    "Warning",
    "Information"
],
],
"Traffic_information_signs": [
    "Traffic_lights_full-time"
],
],
"Intersections": [
    "T-junctions",
    "Y-junctions",
    "Crossroads",
    "Roundabouts"
],
],
"Special_structures": [
    "Tunnels",
    "Bridges",
    "Toll_plazas",
    "Pedestrian_crossings"
],
],
"Wind": [
    "No_wind",
    "Calm",
    "Light_air",
    "Light_breeze",
    "Gentle_breeze"
],
],
"Particulates": [
    "Non_precipitating_water_droplets",
    "Fog"
],
],
"Illumination": [
    "Day",
    "Night",
    "Cloudiness",
    "Artificial_illumination"
],
],
"Drivable_area_surface_features": [
    "Potholes",
    "Ruts"
],
],
"Rainfall_intensity": [
    "Heavy_rain",
    "Light_rain",
    "Medium_rain",
    "Moderate_rain",
    "No_rain"
],
],

```



```

"Special_vehicles": [
    "Traffic_management_vehicle",
    "Work_vehicle",
    "appliance_vehicle"
]
},
"Behavior": ""
}

```

2. UC1.1_REQ_SA_009: *Detection of semi-occluded pedestrian in front or aside of crossing vehicle.*

```

{
  "ODD": {
    "Agent_type": [
      "Vulnerable_road_users"
    ],
    "Drivable_area_type": [
      "Minor_or_local_roads",
      "Parking_space",
      "Primary_roads",
      "Shared_space",
      "Slip_roads_or_off-ramps"
    ],
    "Lane_type": [
      "Shared_lane",
      "Special_purpose_lane",
      "Traffic_lane"
    ],
    "Direction_of_travel": [
      "Right_hand_travel"
    ],
    "Drivable_area_surface_conditions": [
      "Dry",
      "Wet_road"
    ],
    "Road_surface_type": [
      "Segmented",
      "Uniform"
    ],
    "Horizontal_plane": [
      "Straight_roads",
      "Curved_roads"
    ],
    "Vertical_plane": [
      "Up-slope",
      "Down-slope",

```

```

    "Level_plane"
  ],
  "Transverse_plane_type": [
    "Divided",
    "Pavement",
    "Undivided"
  ],
  "Types_of_lanes_together": [
    "Traffic_lanes"
  ],
  "Drivable_area_surface_type": [
    "Asphalt",
    "Cement_concrete",
    "Pavers"
  ],
  "Drivable_area_signs": [
    "Regulatory",
    "Warning",
    "Information"
  ],
  "Traffic_information_signs": [
    "Traffic_lights_full-time"
  ],
  "Intersections": [
    "T-junctions",
    "Y-junctions",
    "Crossroads",
    "Roundabouts"
  ],
  "Special_structures": [
    "Tunnels",
    "Bridges",
    "Toll_plazas",
    "Pedestrian_crossings"
  ],
  "Wind": [
    "No_wind",
    "Calm",
    "Light_air",
    "Light_breeze",
    "Gentle_breeze"
  ],
  "Particulates": [
    "Non_precipitating_water_droplets",
    "Fog"
  ],
  "Illumination": [
    "Day",
    "Night",

```

```

        "Cloudiness",
        "Artificial_illumination"
    ],
    "Drivable_area_surface_features": [
        "Potholes",
        "Ruts"
    ],
    "Rainfall_intensity": [
        "Heavy_rain",
        "Light_rain",
        "Medium_rain",
        "Moderate_rain",
        "No_rain"
    ],
    "Special_vehicles": [
        "Traffic_management_vehicle",
        "Work_vehicle",
        "appliance_vehicle"
    ]
},
"Behavior": [
    "MotionCross",
    "Occluded"
]
}

```

3. UC1.1_REQ_SA_010: *Crossing traffic need to be detected fast enough to avoid collisions when entering/passing over the intersection.*

```

{
  "ODD": {
    "Agent_type": [
      "Motor_vehicle",
      "Vulnerable_road_users"
    ],
    "Junctions": [
      "Intersection"
    ],
    "Drivable_area_type": [
      "Minor_or_local_roads",
      "Parking_space",
      "Primary_roads",
      "Shared_space",
      "Slip_roads_or_off-ramps"
    ],
    "Lane_type": [
      "Shared_lane",
      "Special_purpose_lane",

```

```

    "Traffic_lane"
  ],
  "Direction_of_travel": [
    "Right_hand_travel"
  ],
  "Drivable_area_surface_conditions": [
    "Dry",
    "Wet_road"
  ],
  "Road_surface_type": [
    "Segmented",
    "Uniform"
  ],
  "Horizontal_plane": [
    "Straight_roads",
    "Curved_roads"
  ],
  "Vertical_plane": [
    "Up-slope",
    "Down-slope",
    "Level_plane"
  ],
  "Transverse_plane_type": [
    "Divided",
    "Pavement",
    "Undivided"
  ],
  "Types_of_lanes_together": [
    "Traffic_lanes"
  ],
  "Drivable_area_surface_type": [
    "Asphalt",
    "Cement_concrete",
    "Pavers"
  ],
  "Drivable_area_signs": [
    "Regulatory",
    "Warning",
    "Information"
  ],
  "Traffic_information_signs": [
    "Traffic_lights_full-time"
  ],
  "Intersections": [
    "T-junctions",
    "Y-junctions",
    "Crossroads",
    "Roundabouts"
  ],
],

```

```

"Special_structures": [
  "Tunnels",
  "Bridges",
  "Toll_plazas",
  "Pedestrian_crossings"
],
"Wind": [
  "No_wind",
  "Calm",
  "Light_air",
  "Light_breeze",
  "Gentle_breeze"
],
"Particulates": [
  "Non_precipitating_water_droplets",
  "Fog"
],
"Illumination": [
  "Day",
  "Night",
  "Cloudiness",
  "Artificial_illumination"
],
"Drivable_area_surface_features": [
  "Potholes",
  "Ruts"
],
"Rainfall_intensity": [
  "Heavy_rain",
  "Light_rain",
  "Medium_rain",
  "Moderate_rain",
  "No_rain"
],
"Special_vehicles": [
  "Traffic_management_vehicle",
  "Work_vehicle",
  "appliance_vehicle"
]
},
"Behavior": "MotionCross"
}

```

4. UC1.1_REQ_SA_011: *Crossing traffic shall be detected by the radar system in the presence of dense fog.*

```

{
  "ODD": {

```

```

"Agent_type": [
    "Motor_vehicle",
    "Vulnerable_road_users"
],
"Particulates_type": [
    "Fog"
],
"Particulates_intensity": [
    "Dense_particulates"
],
"Drivable_area_type": [
    "Minor_or_local_roads",
    "Parking_space",
    "Primary_roads",
    "Shared_space",
    "Slip_roads_or_off-ramps"
],
"Lane_type": [
    "Shared_lane",
    "Special_purpose_lane",
    "Traffic_lane"
],
"Direction_of_travel": [
    "Right_hand_travel"
],
"Drivable_area_surface_conditions": [
    "Dry",
    "Wet_road"
],
"Road_surface_type": [
    "Segmented",
    "Uniform"
],
"Horizontal_plane": [
    "Straight_roads",
    "Curved_roads"
],
"Vertical_plane": [
    "Up-slope",
    "Down-slope",
    "Level_plane"
],
"Transverse_plane_type": [
    "Divided",
    "Pavement",
    "Undivided"
],
"Types_of_lanes_together": [
    "Traffic_lanes"
]

```

```

],
"Drivable_area_surface_type": [
    "Asphalt",
    "Cement_concrete",
    "Pavers"
],
],
"Drivable_area_signs": [
    "Regulatory",
    "Warning",
    "Information"
],
],
"Traffic_information_signs": [
    "Traffic_lights_full-time"
],
],
"Intersections": [
    "T-junctions",
    "Y-junctions",
    "Crossroads",
    "Roundabouts"
],
],
"Special_structures": [
    "Tunnels",
    "Bridges",
    "Toll_plazas",
    "Pedestrian_crossings"
],
],
"Wind": [
    "No_wind",
    "Calm",
    "Light_air",
    "Light_breeze",
    "Gentle_breeze"
],
],
"Particulates": [
    "Non_precipitating_water_droplets",
    "Fog"
],
],
"Illumination": [
    "Day",
    "Night",
    "Cloudiness",
    "Artificial_illumination"
],
],
"Drivable_area_surface_features": [
    "Potholes",
    "Ruts"
],
],
"Rainfall_intensity": [
    "Heavy_rain",

```

```

        "Light_rain",
        "Medium_rain",
        "Moderate_rain",
        "No_rain"
    ],
    "Special_vehicles": [
        "Traffic_management_vehicle",
        "Work_vehicle",
        "appliance_vehicle"
    ]
},
"Behavior": "MotionCross"
}

```

5. UC1.1_REQ_SA_014: Perception system shall work in adverse weather conditions.

```

{
  "ODD": {
    "Particulates_type": [
      "Fog",
      "Non_precipitating_water_droplets"
    ],
    "Particulates_intensity": [
      "No_particulates",
      "Few_particulates",
      "Medium_particulates",
      "Dense_particulates"
    ],
    "Rainfall_intensity": [
      "Medium_rain",
      "Light_rain",
      "No_rain",
      "Moderate_rain",
      "Heavy_rain"
    ],
    "Snowfall": [
      "Heavy_snow",
      "Light_snow",
      "Moderate_snow",
      "No_snow"
    ],
    "Drivable_area_type": [
      "Minor_or_local_roads",
      "Parking_space",
      "Primary_roads",
      "Shared_space",
      "Slip_roads_or_off-ramps"
    ],
    "Lane_type": [

```



```

    "Shared_lane",
    "Special_purpose_lane",
    "Traffic_lane"
  ],
  "Direction_of_travel": [
    "Right_hand_travel"
  ],
  "Drivable_area_surface_conditions": [
    "Dry",
    "Wet_road"
  ],
  "Road_surface_type": [
    "Segmented",
    "Uniform"
  ],
  "Horizontal_plane": [
    "Straight_roads",
    "Curved_roads"
  ],
  "Vertical_plane": [
    "Up-slope",
    "Down-slope",
    "Level_plane"
  ],
  "Transverse_plane_type": [
    "Divided",
    "Pavement",
    "Undivided"
  ],
  "Types_of_lanes_together": [
    "Traffic_lanes"
  ],
  "Drivable_area_surface_type": [
    "Asphalt",
    "Cement_concrete",
    "Pavers"
  ],
  "Drivable_area_signs": [
    "Regulatory",
    "Warning",
    "Information"
  ],
  "Traffic_information_signs": [
    "Traffic_lights_full-time"
  ],
  "Intersections": [
    "T-junctions",
    "Y-junctions",
    "Crossroads",

```

```

        "Roundabouts"
    ],
    "Special_structures": [
        "Tunnels",
        "Bridges",
        "Toll_plazas",
        "Pedestrian_crossings"
    ],
    "Wind": [
        "No_wind",
        "Calm",
        "Light_air",
        "Light_breeze",
        "Gentle_breeze"
    ],
    "Particulates": [
        "Non_precipitating_water_droplets",
        "Fog"
    ],
    "Illumination": [
        "Day",
        "Night",
        "Cloudiness",
        "Artificial_illumination"
    ],
    "Agent_type": [
        "Vulnerable_road_users",
        "Motor_vehicle"
    ],
    "Drivable_area_surface_features": [
        "Potholes",
        "Ruts"
    ],
    "Special_vehicles": [
        "Traffic_management_vehicle",
        "Work_vehicle",
        "appliance_vehicle"
    ]
    ],
    "Behavior": ""
}

```

ANNEX 2: EXAMPLES OF SCENARIO DATABASES FUNCTIONS

This section represents examples of some scenario databases for the general process to understand how different SCDBs support querying, searching and retrieving scenarios, it's also details the supported scenario definitions, formats, and query definitions. Specifically, these databases are provided as examples to illustrate the general process. The outlined methods and interactions can be applied broadly to other scenario databases (SCDBs) not mentioned here. By following the described process.

Scenario Center

The scenario center database developed by ika provides access to a variety of different scenarios based on a holistic driving scenario concept supporting urban traffic [8]. This scenario concept models complex traffic situations through a combination of simple basis scenarios that describe bi-lateral interactions of traffic participants. The available scenarios are automatically extracted from real-world trajectory data [[9], [10]].

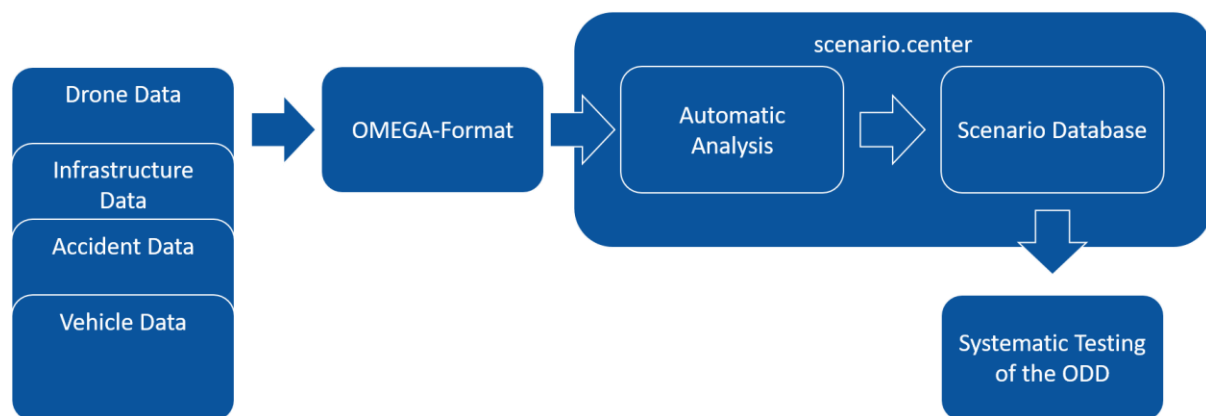


Figure 18: Scenario Center

There are two different ways to interact with the database. The first method involves interacting with the database directly using the database website. Alternatively, the scenario database API can be used to retrieve scenarios. The available API is a GraphQL API, which means that queries are defined in a JSON format and can dynamically specify the content to be returned. The main benefit of a GraphQL API is that the content returned to the user is exactly what was requested. Additionally, the database supports an export of the scenario in the OpenSCENARIO format for further use in the testing toolchain.

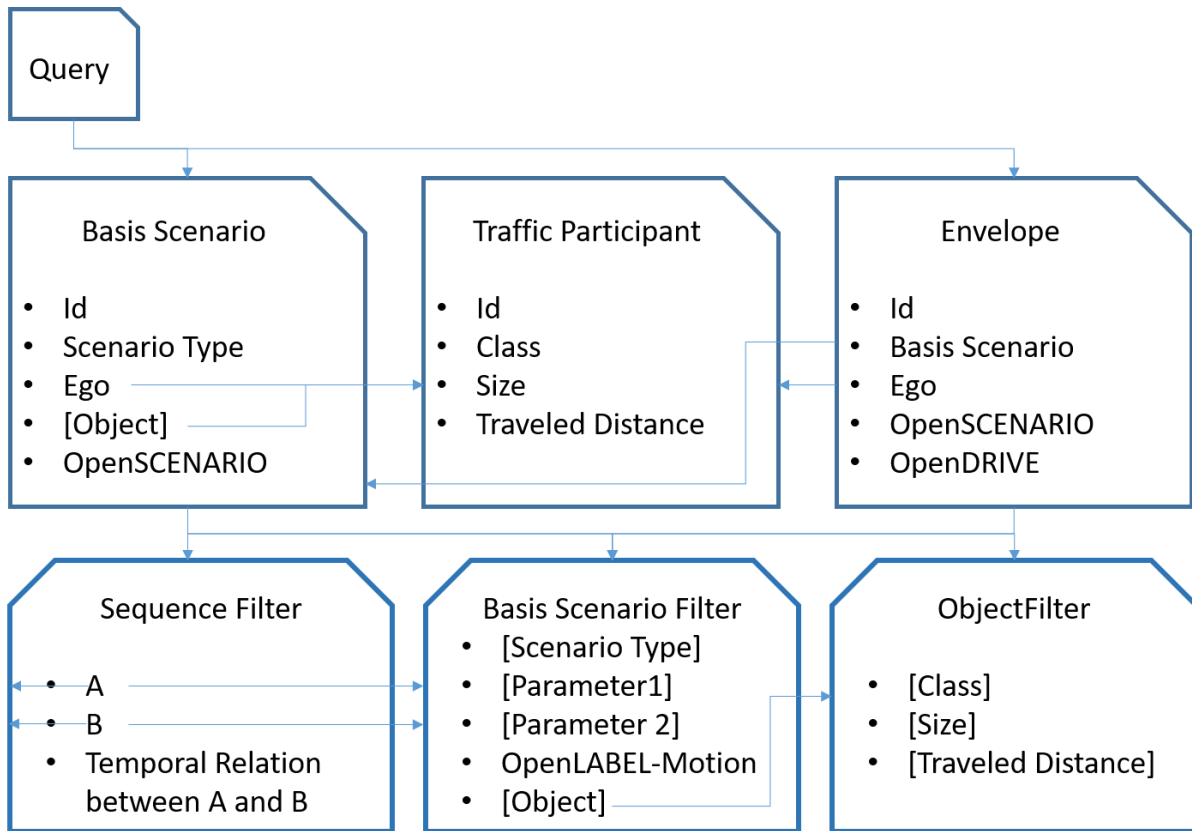


Figure 19: Query concept of Scenario Center

Scenarios can be requested based on the following criteria:

1. Type of scenario
2. Characteristics of traffic participants
3. Parameter Value ranges of scenarios
4. Sequence of scenarios

Scenius

Scenius is a toolchain that is being developed by AVL. AVL Scenius toolchain contains multiple components where Scenario Data Manager and Test Case Generator are the important components for this use case.

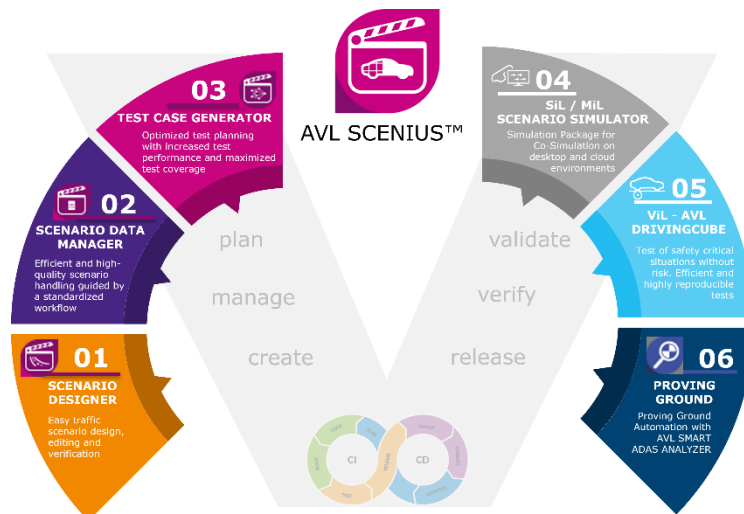


Figure 20: Scenius.

AVL Scenius supports ASAM OpenSCENARIO and can store scenario files with ontology and custom tags. It is possible to query scenarios by their:

1. ASAM OpenSCENARIO version
2. Dynamic ontology tags
3. Static / Environmental ontology tags
4. Custom tags

AVL Scenius has ASAM OpenXOntology tree integrated. However, it is possible to modify / add new content or modify existing taxonomies.

AVL Scenius has a REST API that enables automation. With related endpoints, it is possible to query scenarios in SCDB with the properties defined above. It is planned to use AVL Scenius' API to retrieve scenarios based on defined query criteria for demonstrations in this use case 1.1.

Safety Pool™ Scenario Database

The Safety Pool™ Scenario Database (SPSD) [5] is a secure repository for test scenarios related to Connected and Autonomous Vehicle (CAV) technologies. Developed to aid in the development, verification, and validation of CAV technologies, the SPSP is a collaborative effort between WMG at the University of Warwick,



Figure 21: Safety Pool™ Scenario Database

Safety Pool™ users can interact with the tool using either a graphical user interface (GUI) or via API. Scenarios in Safety Pool™ are grouped based on use cases, such as ALLS, EURO-NCAP - AEB Car to Car System, STPA, etc. These groups are referred to as libraries. Users can retrieve scenarios in various ways:

1- Search for Scenarios:

- Use ODD and behavior tags, supported in OpenLabel format.
- All labels comply with the ISO 34503 taxonomy.
- Perform complex logical operations like AND, OR, Include, Exclude, and Groups See **Figure 15**.
- The search query returns a list of scenarios, from which users can download either individual scenarios or bulk scenarios.

2- Navigate Libraries:

- Users can navigate to the preferred library and download scenarios either in bulk or by selecting specific scenarios.

3- API Access:

- Users can search for scenarios via API using JSON format.
- They can compile desired scenarios into a test suite and download all scenarios in the suite.

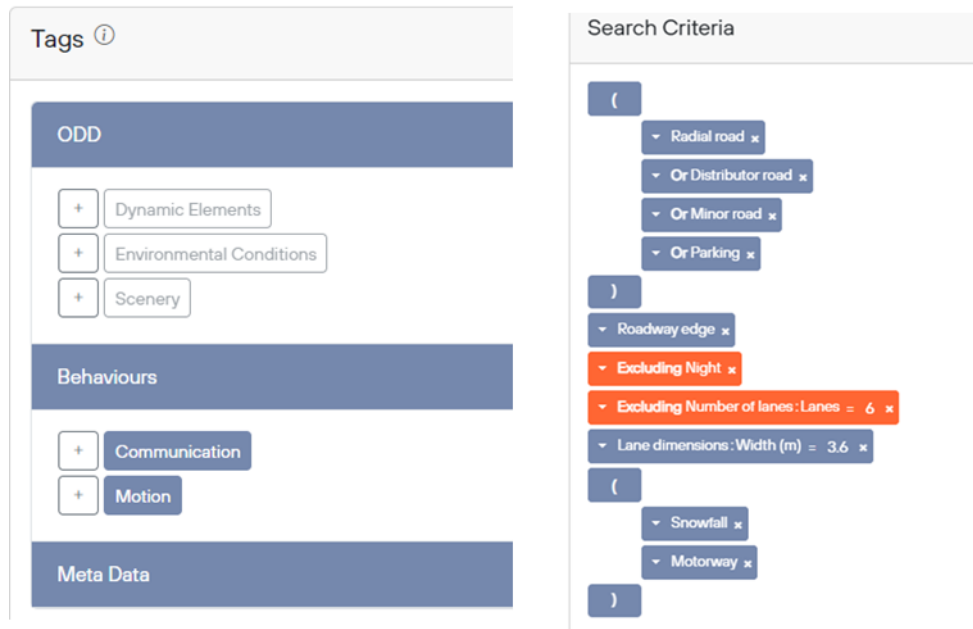


Figure 22: Safety Pool™ Scenario Database's query criteria

In terms of scenario formats and definitions, the retrieved scenarios can be returned in various formats that align with international standards, including:

- ASAM Open Scenario [1] and Open Drive [2]
- BSI Flex 1889 [3]
- SDL Level 2 (developed by UoW) [11]

Streetwise

Streetwise is a scenario-based assessment tool chain developed by TNO.

With the StreetWise tool chain, scenarios can be mined from vehicle driving data, the mined scenarios are stored in a database (Streetwise Database) including their statistics, and based on these statistics test cases can be generated for testing proposes.

The Streetwise toolchain is conceptualized as a scenario mining pipeline and consists of the following steps:



Figure 23: TNO Streetwise tool chain and workflow

The StreetWise tool chain is located at a Microsoft Azure-based platform that can be accessed by partners through an API and GUI to support the automatic interaction with partners for the exchange of data, scenarios, and test cases.

The StreetWise tool chain algorithms are all written in Python and are currently running on Azure Databricks. The driving data that has to be processed as well as intermediate results of data processing are stored on Datalake. The resulting scenarios are stored in a COSMOS database.

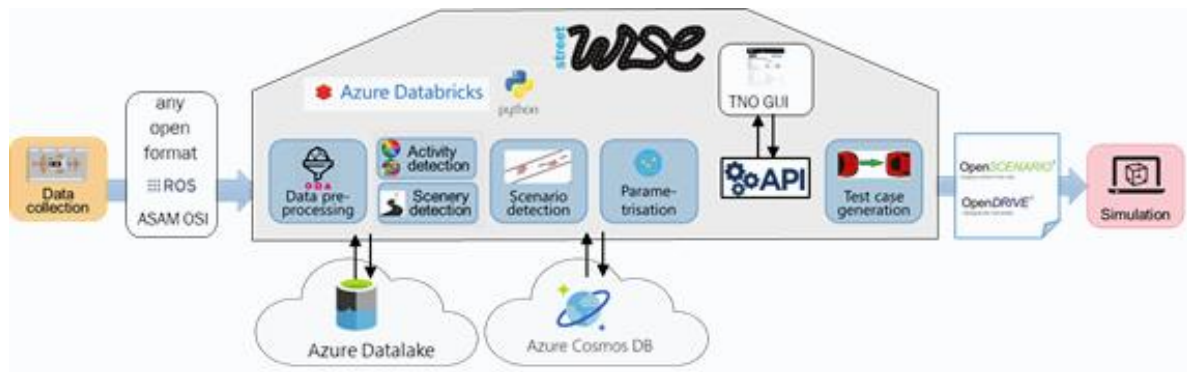


Figure 24: High level components and services of the StreetWise tool chain

The statistics of the scenarios stored in the Streetwise Database can be reached by using an API. The API can also be used for test case generation, in which test cases are generated in OpenDRIVE and OpenSCENARIO format, based on the scenarios stored in the database.

In addition to the API, the Streetwise GUI allows the user to visualize scenario properties and generate test cases:

1. The user can query different scenarios based on characteristics of the scenario such as vehicle type, road type, geographic area, tags.
2. Based on the scenarios selected, statistics of the scenario parameters are displayed, including parameters distributions. These statistics are computed from the scenarios stored in the Streetwise Database.
3. Test cases of the scenario(s) selected are sampled from the scenario parameters distribution within the parameters ranges selected by the user. The test cases are generated in OpenDRIVE and OpenSCENARIO format.

To access the Streetwise GUI and API:

1. The account needs to belong to the TNO Azure Active Directory
2. The account is verified as belonging to an active Streetwise License in the Azure Cosmos DB which also sets the access restrictions of that particular license.

3. A JSON web token (JWT) is issued by the Active Directory. Each request to the API needs to have this token in the header to authenticate the user.

The Streetwise API is implemented in JavaScript making use of Azure Functions to implement the business logic, and follows the REST format.

Each entity of the API is denoted as an endpoint which can be accessed by a GET/POST request. The majority of endpoints allow GET requests to retrieve data from the Streetwise Database. Some endpoints require parameters which can be filled in into the URL of the endpoint.

There is also one endpoint that allows a POST request in order to store the test cases generated by the user in the Streetwise Database.