



SAFETY ASSURANCE FRAMEWORK FOR CONNECTED, AUTOMATED MOBILITY SYSTEMS

D6.3

First European Test Case Library

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	D6.3: First European Test Case Library
Deliverable responsible	Marcos Nieto, Vicomtech
Status - Version	Final – V1.0
Submission date	23/07/2025
Keywords	SUNRISE Data Framework, Federation layer, Scenario Databases

Authors

Role	Name
Main authors	Yavar Taghipour Azar, Juan Diego Ortega (Vicomtech)
Contributing authors	Marcos Nieto (Vicomtech) Jobst Beckmann (IKA) Jose Diaz Mendoza (TNO) Eren Mungan (AVL) Doha Khtatba, Chaima TLILI (Vedecom)

Quality Control

	Name	Organisation	Date
Peer review 1	Stefan de Vries	IDIADA	09/07/2025
Peer review 2	Raul Ferreira	CAF	10/07/2025

Version history

Version	Date	Author	Summary of changes
0.1	19/05/2025	Marcos Nieto	First draft of document structure and content
0.2	17/07/2025	All	First revised version with content in all sections
0.3	22/07/2025	Yavar Taghipour, Juan Diego Ortega, Marcos Nieto	Final revised version

1.0	23/07/2025	Yavar Taghipour, Juan Diego Ortega, Marcos Nieto	Final version
-----	------------	--	---------------

Legal disclaimer

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

Copyright © SUNRISE Consortium, 2025.

TABLE OF CONTENTS

EXECUTIVE SUMMARY	9
1 INTRODUCTION	11
1.1 Project introduction	11
1.2 Purpose of deliverable	13
1.3 Intended audience	13
1.4 Deliverable structure and relation to other parts of project	14
1.5 Reformulation of deliverable nomenclature	15
2 SUNRISE DATA FRAMEWORK	17
2.1 Introduction	17
2.2 Architecture	18
2.3 Implementation and deployment	20
2.3.1 Back-end components	21
2.3.1.1 Auth Management	21
2.3.1.2 Data Management	24
2.3.1.3 Query Manager	25
2.3.1.4 SCDB Management	26
2.3.2 Front-end components	29
2.3.2.1 User Management	29
2.3.2.2 SCDB Management	31
2.3.2.3 Search UI	33
2.3.2.4 Dashboard	35
3 SCENARIO DATABASES	37
3.1 Introduction	37
3.1.1 Adscene	37

3.1.2	SafetyPool	37
3.1.3	Scenario Center	38
3.1.4	Scenius.....	39
3.1.5	Streetwise.....	40
3.2	Demo SCDB	42
3.2.1	Objectives of the Demo SCDB within SUNRISE	42
3.2.2	Architecture Summary	43
3.2.3	Backend Architecture	43
3.2.3.1	Cognito Authentication	44
3.2.3.2	DynamoDB	45
3.2.3.3	S3 Storage.....	46
3.2.3.4	Lambda Functions	46
3.2.3.5	API Gateway.....	47
3.2.4	DEMO SCDB Frontend	48
3.2.5	Scenario Generation	49
4	CONCLUSIONS	50
5	REFERENCES	52
	ANNEX 1. ONBOARDING TO SUNRISE DF	53
	ANNEX 2. SCENARIO DATABASE API REQUIREMENTS.....	60
	ANNEX 3. DATA MODELS FOR SUNRISE DF TABLES.....	72

LIST OF FIGURES

Figure 1: Safety Assurance Framework stakeholders	12
Figure 2: Workplan of the SUNRISE Project	13
Figure 3: Position of the SUNRISE Data Framework within the SAF	17
Figure 4 : SUNRISE DF Architecture	19
Figure 5: SUNRISE Data Framework cloud infrastructure deployment.....	21
Figure 6: SUNRISE DF User Pool viewed in AWS Cognito console	22
Figure 7: SUNRISE DF defined user groups to handle different permissions within the platform	23
Figure 8: SUNRISE DF tables created in AWS DynamoDB	25
Figure 9: Sequence diagram depicting the OAuth2 steps within the SUNRISE architecture	28
Figure 10: SUNRISE DF Sign-up and Login pages	30
Figure 11 : SUNRISE DF landing page	30
Figure 12: User profile page with option to request change role as SCDB Host.....	31
Figure 13 : Interface in SUNRISE for SCDB hosts to add and configure new databases	31
Figure 14: SUNRISE DF form to onboard a new SCDB	32
Figure 15 : SCDB management for a regular user	32
Figure 16: Scenario query based on ODD, Behavior and Road Users	33
Figure 17 : Query results page	34
Figure 18 : Scenario information page	34
Figure 19 : Download page and internal structure of compressed file	35
Figure 20 : Overview of metadata and quality metrics for Demo-SCDB	36
Figure 21 : Re-simulation of a scenario.center scenario.	39
Figure 22: AWS Resource Architecture and Service Connections for the Demo SCDB	44
Figure 23: Architecture of the lambda function in the AWS CDK	47
Figure 24 : SUNRISE Data Framework onboarding process.	54
Figure 25: SUNRISE User Registration Screen.....	57
Figure 26: Requesting SCDB Host Role	58
Figure 27: Manage SCDB dropdown in user menu and SCDB Configuration	58
Figure 28: Tabs related to add new SCDB	59

LIST OF TABLES

Table 1: OpenSCENARIO package content.....	70
Table 2 : Data model for SCDB table.....	72
Table 3: Data model for UserSCDBConnection table.....	73
Table 4: Data model for QueryManagerQuery table.....	74
Table 5 : Data model for UserNotification table	75

ABBREVIATIONS AND ACRONYMS

Abbreviation	Meaning
AD	Automated Driving
ADS	Automated Driving System
AEB	Autonomous Emergency Braking
API	Application Programming Interface
AWS	Amazon Web Services
CCAM	Connected, Cooperative, and Automated Mobility
CDK	Cloud Development Kit
CLI	Command Line Interface
CORS	Cross-Origin Resource Sharing
COTSATO	Concretizing Test Scenarios and Associating Test Objectives
DF	Data Framework
IAM	Identity and Access Management
ISMR	In-Service Monitoring and Reporting
JWT	Json Web Token
NATM	New Assessment/Test Method for Automated Driving
ODD	Operational Design Domain
ODR	OpenDRIVE forma
OEM	Original Equipment Manufacturer
OSC	OpenSCENARIO
OSG	OpenScenario Graph
PDF	Probability Density Function
RBAC	Role-Based Access Control
SAF	Safety Assurance Framework
SCDB	Scenario Database

SDL	Scenario Description Language
SOTIF	Safety Of the Intended Functionality
SUNRISE	Safety assUraNce fRamework for connected, automated mobility SystEms
SUT	System Under Test
UC	Use Case
UI	User Interface
UUID	Universal Unique Identifier
V&V	Verification and Validation
WP	Work Package

EXECUTIVE SUMMARY

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

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

This deliverable presents the **SUNRISE Data Framework (DF)** which is a cloud application implemented in the context of WP6 activities, specially T6.4 about European Test Case Library. The SUNRISE DF implements and operates as a federation layer that provides centralised and harmonised access to **SCDBs**.

The SUNRISE DF is a supporting element for the Store and Query & Concertise blocks of the SAF. Its application is highly recommended (though not mandatory to follow the SAF methodology) since it provides the following benefits:

1. It offers a centralized single portal to access all connected SCDBs through standardised interfaces
2. It guarantees data governance through user authentication and authorisation
3. It enhances and simplifies compliance with scenario querying processes
4. It centralises scenario retrieval for a variety of stakeholders (e.g., certifying entities, CCAM developers, test services, etc.)

Several components have been implemented which compose the SUNRISE DF, including a set of back-end functions (management, deployment, API services), and front-end user interfaces (Query UI, Dashboard, User access). Details on these components and other aspects of the standardised input and output interfaces of the SUNRISE DF with the SCDBs are provided in deliverable "**D6.2 Define and development of SCDB input and output standards and interfaces**".

As the SUNRISE DF operates as a **federation layer**, it does not offer storage services, so all scenario content remains at the SCDB repositories, controlled by SCDB owners. The SUNRISE DF is open for **onboarding new SCDBs** which desire to offer their scenarios. Procedures to onboard a user and a SCDB are described in **Annex 1 Onboarding**. In technical terms, the onboarding essentially implies first to register in SUNRISE DF and then provide an API which is prepared at the SCDB side compliant with the API requirements described in **Annex 2** of this deliverable.

As part of the documentation and the onboarding process, a **SUNRISE Demo SCDB** has been also built, as an example mock-up scenario database that fully complies with the API requirements and the SUNRISE SAF. The Demo SCDB is reported also in this deliverable and will serve as an example software documentation that interested SCDB owners can use as a reference to onboard the SUNRISE DF.

This document also summarizes the current status (as of July 2025) of SCDBs already onboarded into the SUNRISE DF, namely StreetWise, SafetyPool, Scenario.center, Adscene, and Scenius.

The SUNRISE DF is deployed in a cloud service and can be accessed at **ccam-sunrise.eu**

1 INTRODUCTION

1.1 Project introduction

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

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

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

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

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

Evolving from the achievements obtained in HEADSTART and taking other project initiatives as a baseline, it becomes necessary to move to the next level in the development and demonstration of a commonly accepted **Safety Assurance Framework (SAF)** for the safety validation of CCAM systems, including a broad portfolio of Use Cases (UCs) and comprehensive test and validation tools. This will be done in **SUNRISE**, which stands for **Safety assURaNce fRamework for connected, automated mobility SystEms**.

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

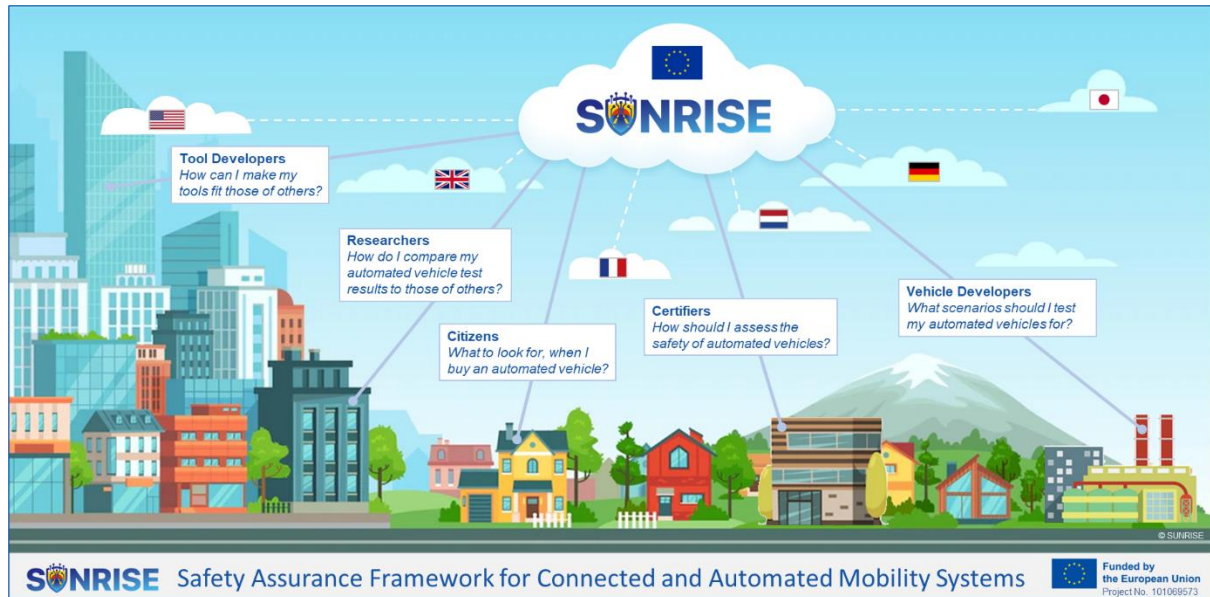


Figure 1: Safety Assurance Framework stakeholders

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

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

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

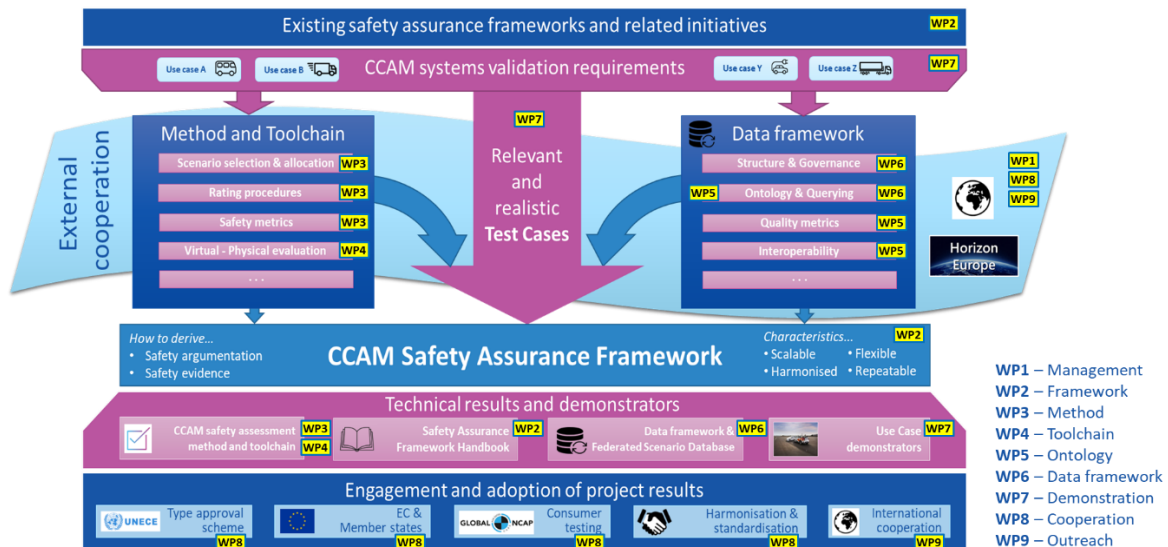


Figure 2: Workplan of the SUNRISE Project

1.2 Purpose of deliverable

The purpose of this deliverable is to explain the SUNRISE Data Framework (SUNRISE DF), as a federation layer under the SUNRISE Safety Assurance Framework, to provide access and interconnect to existing SCDBs. The nature of the SUNRISE DF described in deliverable D6.3 is of type DEMO (Demonstrator, pilot, prototype), as described in the Grant Agreement. The SUNRISE DF is a software platform that contains a few components to implement the required functionality.

The purpose of this document is to explain:

- The development and deployment of the SUNRISE Data Framework and its components
- The existing connection with existing SCDBs (Adscene, SafetyPool, Scenario.center, Scenius, Streetwise)
- The developed Demo SCDB to exemplify how SCDBs shall implement the interface
- The technical documentation of the SCDB API requirements.
- Definition of access rights
- Definition of user profile
- Definition of data management mechanism

These deliverable updates some aspects reported in the former “D6.1 Methodology for SCDB application for generic use cases” (submitted August 2024), including a revised architecture of the SUNRISE Data Framework, as can be found in section 2.

1.3 Intended audience

The intended audience of this deliverable is multiple: (i) SCDB hosts that may learn from this document what the SUNRISE Data Framework is and how to create the API interfaces required to connect their SCDBs, (ii) SCDB users that want to learn what the SUNRISE Data

Framework offers, and, (iii) guest users without specific SCDB access, and (iv) other entities and consortia that may want to extend the SUNRISE Data Framework with additional functionalities or features.

1.4 Deliverable structure and relation to other parts of project

The contents of this document are divided into the following chapters:

- **Chapter 2: SUNRISE Data Framework:** this section includes extensive documentation on the architecture of the SUNRISE Data Framework, along with implementation and deployment details of the built prototype. The back-end and front-end components are detailed.
- **Chapter 3: Scenario Databases:** this section summarizes relevant information about the external SCDBs connected to the SUNRISE Data Framework, with some details on the type of user access mechanisms. This section also explains the developed and deployed Demo SCDB.
- **Chapter 4: Conclusions:** this section includes the main findings of this deliverable.
- **Chapter 5: References:** this section contains the bibliography and other references used in this document.
- **Annex 1: Onboarding SCDBs:** this section includes high-level information for interested, external SCDBs about the onboarding process to connect with the SUNRISE Data Framework.
- **Annex 2: Scenario Database API requirements:** this section contains the technical requirements defined for the endpoints to be used for data exchange between SUNRISE Data Framework (DF) and individual scenario databases (SCDB).
- **Annex 3: Data Models for SUNRISE DF tables:** This section contains the structure of the SUNRISE DF storage database.

Relations to Other SUNRISE Tasks and Work Packages

Current deliverable is tightly connected with several other SUNRISE tasks and work packages, ensuring coherence and integration across the project. The key relationships include:

- **Task T3.4 (Subspace Creation Methodology):** Deliverable D3.4 leverages the subspace creation methodologies developed in task T3.4 for defining and refining queries in the SUNRISE DF Query Manager. These methodologies support the systematic selection of scenario subsets relevant for safety assessments [1].
- **Task T5.2 (Ontology):** The SUNRISE DF uses the ontology developed in task T5.2 as a harmonized taxonomy for scenario description and tagging. This ontology enables semantic queries and consistent scenario classification across different SCDBs [2].
- **Task T5.3 (Quality Metrics for SCDB Content):** Metrics defined in deliverable D5.3 influence how scenario data quality is assessed and presented in the SUNRISE DF. Quality indicators provided by deliverable D5.3 inform data exchange processes and support users in evaluating the reliability and relevance of SCDB content [3].

- **Tasks T6.1 and T6.2 (Standardized Input/Output Data and Interfaces):** Deliverable D6.3 aligns with tasks T6.1 and T6.2 in defining standardized formats and interfaces (e.g., APIs, data schemas) that ensure interoperability between the SUNRISE DF and external SCDBs [4].
- **Work Package 7 (Use Case Development and Execution):** The development and testing of use cases in WP7 rely in part on the SUNRISE DF for accessing and retrieving relevant scenarios.

1.5 Reformulation of deliverable nomenclature

This deliverable has been produced by the coordinated action of multiple tasks within WP6. Task T6.4 has contributed significantly to it. In the Grant Agreement, Task T6.4 is entitled “T6.4 Sample case of a European Test Case Library”. It is there described with the goal “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”.

The task has been executed without delays or deviations, following the key aspects present at the task description in the Grant Agreement, including the development of a software prototype, as an operational federation layer that provides access to existing databases. This software is the presented **SUNRISE Data Framework**, as reported in this deliverable D6.3.

For the sake of clarity, there is a nomenclature-related interpretation issue that requires clarification to avoid confusion.

The term “test case” was undefined at the time of writing the Grant Agreement and was used interchangeably with “scenario” and “test scenario”. During the first period of the SUNRISE project (M1-M12), the terminology was deeply discussed, and the terms have been clarified: in summary, a “scenario” is a piece of information that describes a driving situation, while “test case” is a more complex concept that not only includes a scenario, but also a “system under test”, and a “test environment”. This is a critical distinction, which avoids further confusion reading through SUNRISE documentation.

During the project-level discussions of SUNRISE, the goal of having a European **Test Case Library** was regarded as unfeasible from a technical and organisational point of view (impossibility to cover the “system-under-test” part), and it was substituted by the achievable goal of having a European **Test Scenario Library**, still aligned with the reading of the funding call, which can be understood as requesting the project to build a European-level repository of *scenarios*, but not *test cases*.

Consequently, task T6.4 (and consequently, this deliverable D6.3), **was re-interpreted to focus on a European Test Scenario Library**, which took the form of the SUNRISE Data Framework, that implements or works as a federation layer providing access to existing scenario repositories, the so-called SCDBs.

The rest of the document targets the description of this SUNRISE Data Framework, and the connected Scenario Databases.

2 SUNRISE DATA FRAMEWORK

2.1 Introduction

The SUNRISE Data Framework (SUNRISE DF), a key outcome of the technical activities in SUNRISE WP6, serves as the data management layer for the Safety Assurance Framework (SAF). Its primary purpose is to streamline and centralize access to diverse external SCDBs.

The SUNRISE DF provides a federated set of services that act as a harmonized and centralized entry point for interacting with existing external SCDBs. This approach ensures the governability of data owners and protects their business models while enabling seamless access to scenario data. As illustrated in Figure 3, the position of the SUNRISE DF within the overall SAF enables efficient validation workflows, advanced querying, and traceable scenario selection, in line with the SAF's core structure. Its role integrates directly with other SAF blocks, including Allocate and Execute, making it a critical enabler of end-to-end scenario-based testing.

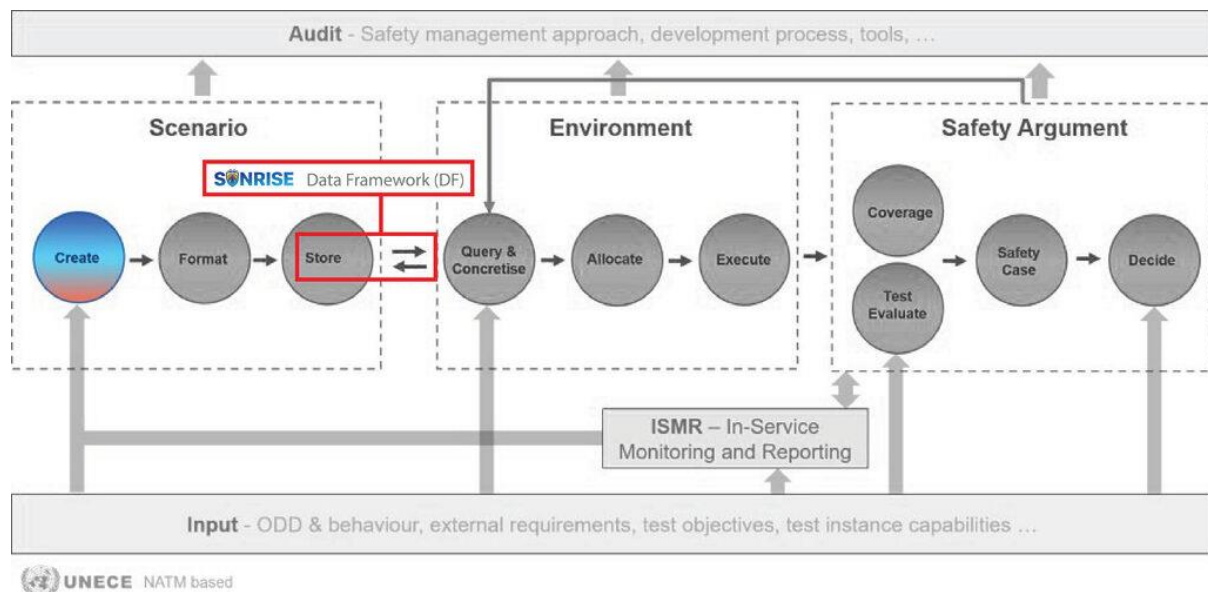


Figure 3: Position of the SUNRISE Data Framework within the SAF

Specifically, the SUNRISE DF addresses the SAF's requirements for SCDB interfacing, including data formatting, query capabilities, and access and authorization control. For data formatting, SUNRISE DF currently utilizes the ASAM OpenLABEL ontology. In future steps for the SUNRISE DF, this will be replaced with the more comprehensive SUNRISE Ontology, developed in Task T5.2 and described in Deliverable D5.2, which also covers INPUT and OUTPUT standardization as detailed in Deliverable D6.2. The concept of a federation layer means that data remains distributed across various SCDBs, managed by third-party entities connected to the SUNRISE DF.

To further enhance data utility, the SUNRISE DF includes data processing services. These services harmonize queries, facilitate exploration of different SCDBs, analyse search results,

and prepare scenario content for its direct use within other SAF blocks like Allocate and Execution.

The development of the SUNRISE Data Framework was guided by a set of structured storylines that represent typical interactions and processes involving various stakeholders. These storylines were instrumental in identifying functional requirements, defining system behaviour, and ensuring that the SUNRISE DF architecture supports all relevant use cases across the data management lifecycle.

Storyline-1 focuses on user onboarding, addressing the definition of user roles (such as consumer, producer, and owner), authentication procedures, and user access via the front-end dashboard. **Storyline-2** covers the onboarding of new SCDBs, including the responsibilities of the SUNRISE administrator, registration workflows, and configuration of input/output interfaces with SCDBs. **Storyline-3** details the process of querying connected SCDBs, including the use of the front-end interface for formulating search queries, the role of the query manager for semantic search, and the management of authorization and API-specific adaptations. It also includes optional visual analytics functionalities to support scenario exploration. **Storyline-4** addresses the uploading of scenarios to SCDBs, including editing or creation in the front-end, format harmonization through ontology-based conversion, authorization handling, and integration with SCDB-specific interfaces for content transfer. **Storyline-5** describes the application of selected scenarios to testing use cases, enabling the transfer of harmonized scenario data to external testing platforms and retrieval of corresponding results.

These storylines provided a structured basis for the design and implementation of the SUNRISE DF, ensuring consistency with the SAF's functional goals and technical requirements, and enabling alignment with stakeholder needs throughout the scenario data workflow.

2.2 Architecture

The SUNRISE DF is a software platform designed to facilitate user interaction with various SCDBs. It enables parallel queries across these databases and the retrieval of scenario packages for use in diverse testing activities.

The SUNRISE DF comprises three main parts:

1. Front end components (Web Application)

This user-friendly web application provides an intuitive interface for users to access the SUNRISE DF's services.

2. Backend Components

These components consist of various services that power the platform's functionality. They handle user credentials, data storage, process user queries, and manage interactions with SCDB APIs.

3. Data Storage Layer

The data storage layer serves as the central layer for various types of data managed within the SUNRISE DF. It securely stores:

- SCDB connection and authentication information.
- Saved search queries and configurations.
- Temporary storage for scenario metadata retrieved from SCDBs.
- System logs and operational data for monitoring and auditing.

This layer is essential for ensuring persistence, security, and integrity of data across multiple user sessions and system operations. While the SUNRISE DF operates as a federated system accessing distributed SCDBs, the internal data storage ensures that user-specific data and operational information remain consistent and secure.

The functional architecture of the implemented solution as described in deliverable D6.2 is the following:

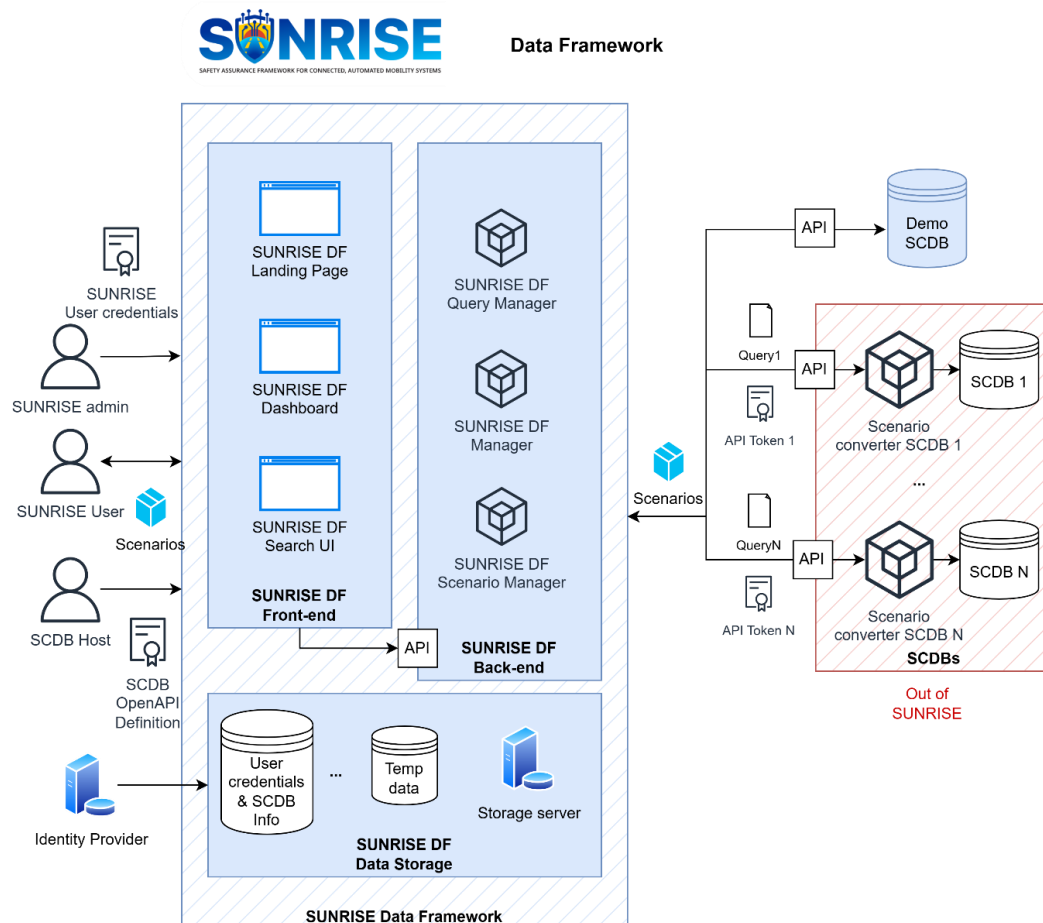


Figure 4 : SUNRISE DF Architecture

2.3 Implementation and deployment

The SUNRISE DF is implemented using a combination of modern technologies, which will be detailed in the subsequent sections. For deployment, the platform leverages Amazon Web Services (AWS) Amplify.

AWS Amplify is a comprehensive set of tools and services designed to accelerate the development and deployment of full-stack web and mobile applications on AWS. It significantly simplifies the process of building scalable, secure, and performant cloud-powered applications by abstracting away much of the underlying infrastructure management.

Key aspects of AWS Amplify that are particularly beneficial for the SUNRISE DF deployment include:

- **Accelerated Development and Deployment:** Amplify offers a streamlined workflow, enabling rapid prototyping and continuous deployment. It integrates seamlessly with popular Git repositories (like GitHub, GitLab, and Bitbucket), automating the build, test, and deployment process whenever changes are pushed to the codebase. This allows the SUNRISE DF development team to iterate quickly and deliver updates efficiently.
- **Managed Hosting:** Amplify provides fully managed hosting for web applications, including support for custom domains and automatic SSL certificates. This ensures that the SUNRISE DF web application is globally available, reliable, and secure, without requiring the team to manage servers or networking infrastructure.
- **Integration with AWS Services:** Amplify acts as a powerful abstraction layer, making it easy to integrate with a wide array of AWS backend services. This includes services for authentication (e.g., Amazon Cognito for user management), data storage (e.g., Amazon S3 for file storage, Amazon DynamoDB for NoSQL databases), serverless functions (e.g., AWS Lambda for backend logic), and API development (e.g., AWS AppSync for GraphQL APIs or AWS API Gateway for REST APIs). This deep integration allows the SUNRISE DF to leverage the full power of the AWS ecosystem.
- **Scalability:** Built on AWS's serverless architecture, Amplify automatically scales with traffic demands, ensuring consistent performance for the SUNRISE DF even during peak usage. This eliminates the need for manual scaling and infrastructure provisioning, allowing the team to focus on core application logic.
- **Developer-Friendly Tools:** Amplify provides a powerful Command Line Interface (CLI) and libraries that simplify the process of configuring and managing cloud resources directly from the command line. This allows developers to add functionalities like authentication, APIs, and storage without needing in-depth knowledge of the underlying AWS services.

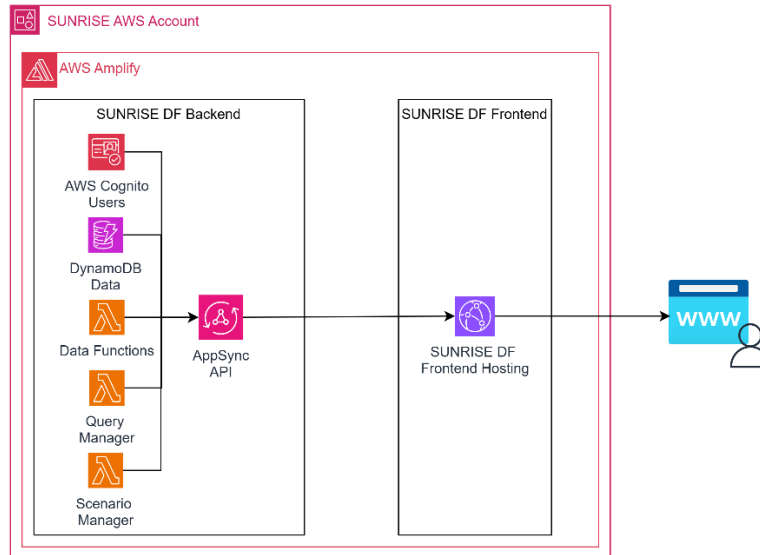


Figure 5: SUNRISE Data Framework cloud infrastructure deployment

2.3.1 Back-end components

The back-end components of SUNRISE DF consist of various services crucial for the platform's smooth operation. These services are implemented using the **AWS Amplify framework**, which enables seamless integration and deployment of server-side back-end components with front-end components. Leveraging TypeScript for back-end development, the same mechanism used for front-end development, has created a **harmonized approach** to implementing the entire platform.

By using a common programming framework with AWS Amplify, based on serverless components, the costs of using and maintaining the back-end components in AWS are optimized and allow for dynamic scalability, automatically adjusting resources based on demand, which translates to significant cost savings and improved performance for users.

The SUNRISE DF back-end components are the following:

- **SUNRISE DF Auth Management:** service that enables secure authentication flow and control access to data and files.
- **SUNRISE DF Data Management:** service that enables built secure, real-time APIs to interact with AWS databases quickly and easily
- **SUNRISE DF Query Manager:** a set of back-end functions deployed as AWS Lambda functions to prepare the queries that will be sent to the different external SCDBs
- **SUNRISE DF SCDB Management:** a set of back-end functions deployed as AWS Lambda functions which allow the connectivity and harmonize the requests to the external SCDBs.

2.3.1.1 Auth Management

The **SUNRISE DF Auth Management** component is a critical service designed to provide secure authentication and access control for the entire SUNRISE DF. This component was

robustly implemented leveraging the **AWS Amplify framework**, specifically utilizing **AWS Cognito** for its core functionalities.

At the heart of the authentication system is an **AWS Cognito User Pool**, which serves as the central directory for managing all user sign-up, sign-in, and authentication processes to the SUNRISE DF resources. This User Pool is configured to ensure comprehensive user profiles, requiring the following attributes upon user registration:

- **email**: The primary email address of the user, used for communication and account recovery.
- **given_name**: The user's first name.
- **family_name**: The user's last name.
- **accept_terms**: A boolean attribute confirming the user's acceptance of the platform's terms and conditions, ensuring compliance and user agreement.

This setup ensures that all users are properly identified and that their access is managed securely within the SUNRISE DF ecosystem.

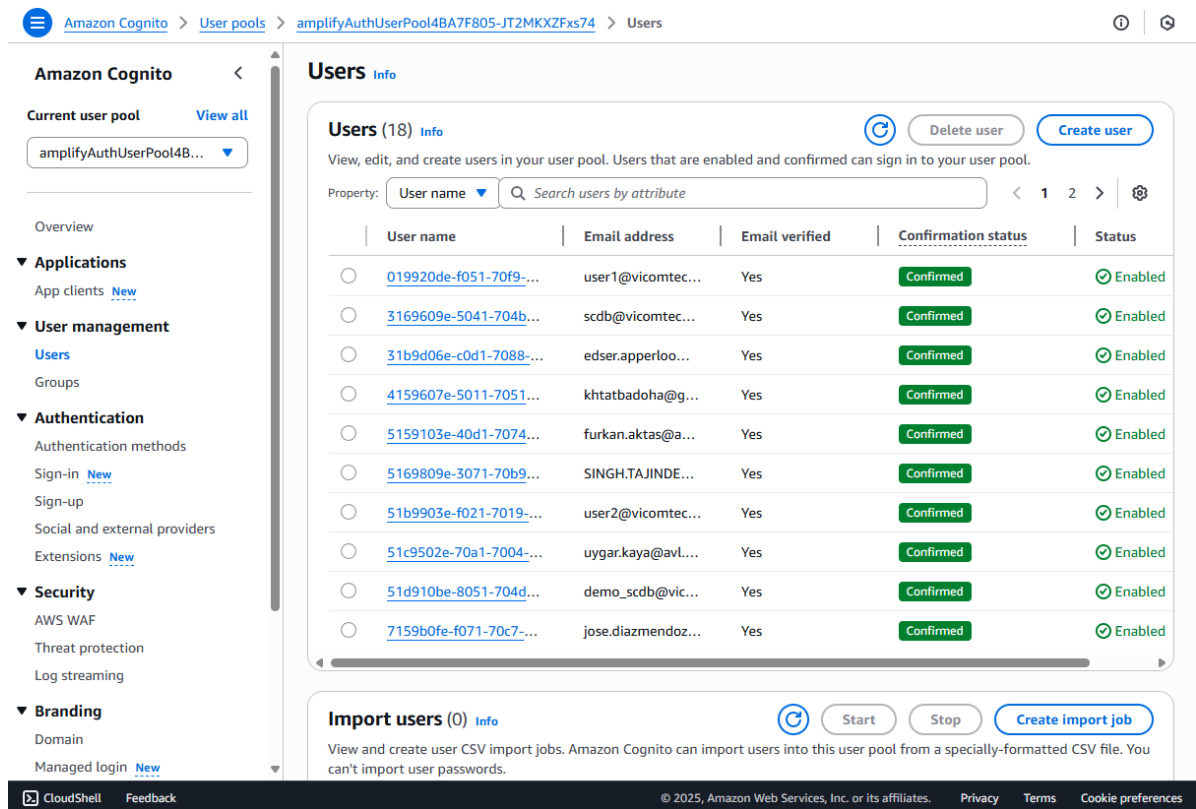


Figure 6: SUNRISE DF User Pool viewed in AWS Cognito console

Defined User Roles and Permissions

Within the SUNRISE DF Auth Management component, distinct user roles have been defined, each corresponding to specific **User Groups** within the AWS Cognito User Pool. These roles dictate the level of access, and the actions users are permitted to perform on the platform, ensuring a structured and secure operational environment. The defined user groups are:

- **Admins:**
 - **Role:** Administrators of the SUNRISE DF.
 - **Permissions:** Possess comprehensive rights to manage the platform, including the ability to delete users, modify user profiles, and change the roles (user groups) of other users. This group is responsible for the overall governance and maintenance of the user base.
- **SCDB Hosts:**
 - **Role:** Users responsible for integrating new SCDBs into the SUNRISE DF.
 - **Permissions:** Are specifically authorized to onboard new SCDBs, configure their integration details, and manage their availability within the system.
- **Users:**
 - **Role:** Standard, regular users of the SUNRISE DF.
 - **Permissions:** Can provide connection details to existing SCDBs, create new queries, and execute queries against the SCDBs they have access to. This group represents the primary end-users interacting with the data querying functionalities.

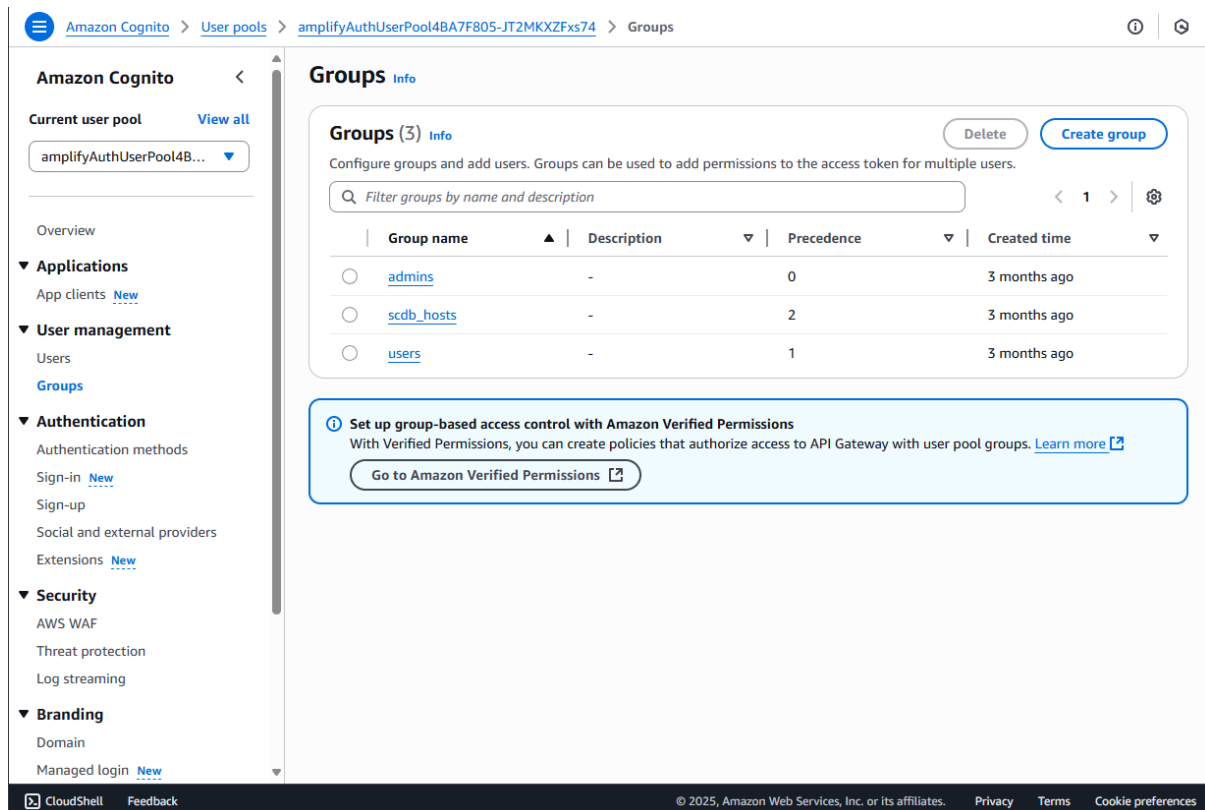


Figure 7: SUNRISE DF defined user groups to handle different permissions within the platform

This role-based access control mechanism ensures that users only have access to the functionalities necessary for their specific tasks, enhancing security and maintaining the integrity of the SUNRISE DF.

2.3.1.2 Data Management

The **SUNRISE DF Data Management** service is a core back-end component responsible for securely storing essential metadata and operational data for the platform. This service is designed to house critical information such as metadata for registered SCDBs, sensitive user connection details, and a comprehensive history of scenario queries.

This robust data service is implemented using **AWS Amplify**, with **AWS DynamoDB** serving as the underlying database.

AWS DynamoDB is a service that provides a fast, flexible NoSQL database service for applications that need consistent, single-digit millisecond latency at any scale. It is a fully managed, multi-region, multi-master database with built-in security, backup and restore, and in-memory caching for internet-scale applications. DynamoDB's key characteristics include:

- **Serverless:** It automatically scales capacity to meet traffic demands, eliminating the need for server provisioning or patching.
- **High Performance:** It offers consistent, low-latency performance, making it ideal for high-throughput, low-latency applications.
- **Scalability:** It can handle petabytes of data and millions of requests per second, scaling up or down with zero downtime.
- **Durability and Availability:** Data is replicated across multiple Availability Zones within an AWS Region, ensuring high durability and availability.
- **Flexible Data Model:** It supports both document and key-value data models, allowing for flexible schema design.
- **Security:** It provides encryption at rest and in transit, along with fine-grained access control through AWS Identity and Access Management (IAM).

The following tables have been created within this DynamoDB instance to store vital information of the SUNRISE DF:

- SCDB
- UserSCDBConnection
- QueryManagerQuery
- UserNotification

Detailed data models for the above tables are provided in Annex 3.

Name	Status	Partition key	Sort key	Indexes
QueryManagerQuery-tpsw4clrineofa6hcdahyp7me-NONE	Active	ModelId (S)	-	0
SCDB-tpsw4clrineofa6hcdahyp7me-NONE	Active	id (S)	-	0
UserNotification-tpsw4clrineofa6hcdahyp7me-NONE	Active	id (S)	-	0
UserSCDBConnection-tpsw4clrineofa6hcdahyp7me-NONE	Active	id (S)	-	1

Figure 8: SUNRISE DF tables created in AWS DynamoDB

2.3.1.3 Query Manager

The Query Manager is a core component of the SUNRISE DF that serves as the intelligent query orchestration engine for ODD-based scenario queries. It acts as the central hub where users define their scenario search requirements and manage their query collections within the SUNRISE DF.

When users need to find specific scenarios across multiple SCDBs, the Query Manager takes their ODD selections and transforms them into a standardized format that all connected databases can understand. This happens through automatic conversion to ASAM OpenLabel format [5], which ensures that a query created once, can be sent to any connected SCDB without compatibility issues.

The component maintains a persistent repository of user queries, allowing teams to build up libraries of reusable search criteria over time. Each query captures detailed ODD specifications including environmental conditions, road user types, and behavioral patterns. Users can return to previous queries, modify them as requirements evolve, or reuse them for different validation scenarios.

Query Management Workflow

The query management process follows a straightforward four-stage workflow:

1. Query Creation & Storage

Users build queries through the Search UI by selecting from organized ODD taxonomies. The system structures these selections into three main categories:

- **ODD Elements:** Weather conditions, road types, lighting, and other environmental factors
- **Road User Types:** Cars, trucks, pedestrians, cyclists, and other traffic participants
- **Behavioural Patterns:** Lane changes, overtaking, braking, and other driving behaviours

Each query gets a unique identifier (QUE-XXXXXX) and is stored in Amazon S3 with metadata that includes when it was created, who created it, and which databases it targets.

2. Query Retrieval & Management

The system provides standard database operations for query management:

- Browse existing queries or find specific ones by ID
- Update queries when requirements change or new databases become available
- Delete queries as needed

3. OpenLabel Format Conversion

A critical function of the Query Manager is the automatic conversion of internal query structures into ASAM OpenLabel format. This standardization process:

- Maps each ODD element to standardized ontology references
- Preserves the include/exclude logic that determines which scenarios match
- Ensures consistent query format across all connected SCDBs

4. SCDB Integration & Dispatch

Once converted, queries are distributed to appropriate SCDBs through the SCDB Management component. The system handles:

- Figuring out which databases are relevant for each query
- Sending queries to multiple databases simultaneously
- Managing different authentication requirements for each SCDB
- Collecting results and keeping track of which database provided which scenarios

2.3.1.4 SCDB Management

The SCDB Management component in the back-end consist of several functions which allows storing the information about the SCDB API securely in the configured AWS DynamoDB. Through the SUNRISE DF frontend the SCDB Host will upload an OpenAPI file with the specific details of the API. Once the OpenAPI file is uploaded, SUNRISE DF parses the specification and stores the SCDB's details in the AWS DynamoDB database, including endpoint paths and security requirements. This enables SUNRISE to automatically configure routing, validate expected payloads, and prepare the correct authentication flows when users attempt to access the SCDB's resources. As a result, when a user initiates a connection to a particular SCDB through SUNRISE, the platform knows precisely how to redirect the user into the appropriate authentication process defined by that SCDB. This foundational onboarding

workflow ensures that SUNRISE can support a diverse range of SCDBs with differing architectures and security models, while maintaining a seamless and consistent user experience across the federated data framework.

SCDB Authentication and Access Workflow via SUNRISE DF

In the SUNRISE DF, access to Scenario Database (SCDB) APIs is tightly controlled to ensure security, privacy, and compliance with each SCDB's specific requirements. A crucial architectural decision in SUNRISE is that **tokens and security credentials never leave the backend**. The SUNRISE frontend is intentionally designed to be **token-naive** — it does not store, process, or even see access tokens. Instead, all protected communication with SCDBs occurs through **proxy functions implemented in the SUNRISE backend**.

SUNRISE supports two security mechanisms for accessing SCDBs, each defined in the SCDB's OpenAPI specification:

1. OAuth2 Authorization Code Flow

For SCDBs secured with OAuth2:

1. User Login or Registration:

A user accesses the SUNRISE frontend and requests a resource that requires SCDB access. The SUNRISE backend detects the absence of an active session for that SCDB.

2. Redirection to SCDB Authorization Server:

The SUNRISE backend responds by redirecting the user to the SCDB's authorization server login page.

3. User Authenticates at SCDB:

The user logs in (or registers) directly on the SCDB's own authentication interface.

4. Authorization Code Returned:

After successful login, the SCDB authorization server redirects back to the SUNRISE backend with an **authorization code**.

5. Token Exchange:

The SUNRISE backend exchanges the authorization code for a set of tokens:

- **Access Token:** used to authenticate API calls
- **Refresh Token:** used to renew the access token when it expires
- **ID Token** (if issued): contains identity claims, but remains private to the backend

6. Token Storage in Backend:

Tokens are securely stored in the SUNRISE backend for the user's session context. **They are never exposed to the frontend or user's browser.**

7. API Requests via Proxy Functions:

When the frontend requests SCDB data, it calls SUNRISE backend proxy endpoints. Subsequently, the backend:

- Attaches the stored access token to the outbound request to the SCDB API
- Processes the SCDB API response
- Returns the data to the frontend in a secure, token-agnostic way

This architecture ensures the frontend remains entirely insulated from SCDB-specific security implementations, reducing attack surfaces and simplifying frontend code. The corresponding sequence diagram is depicted in Figure 9.

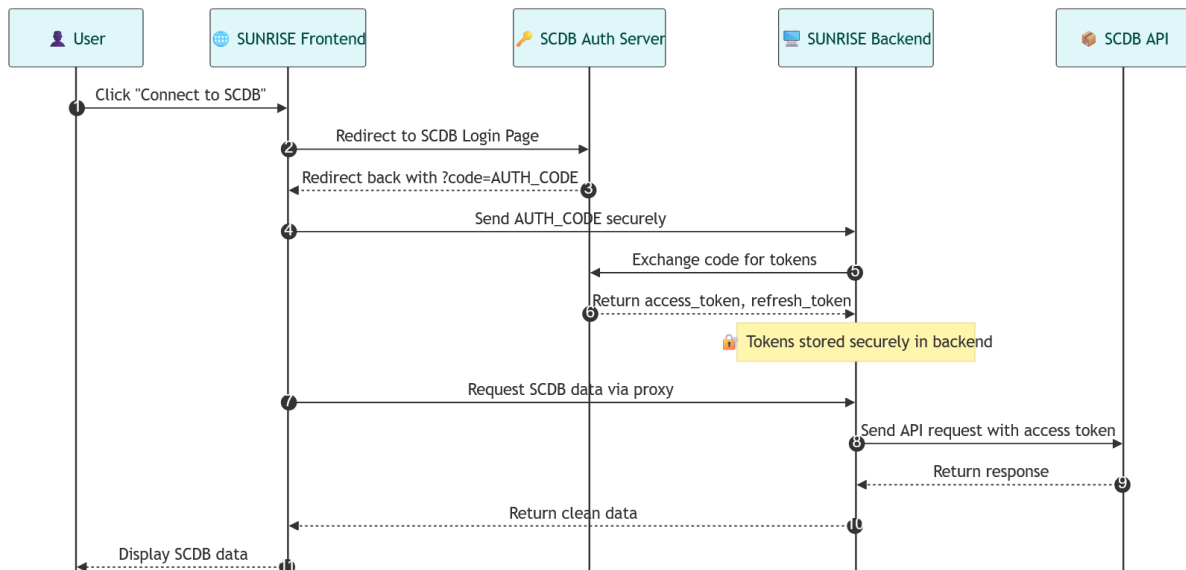


Figure 9: Sequence diagram depicting the OAuth2 steps within the SUNRISE architecture

2. API Key Security

For SCDBs secured with API keys:

- Users obtain an API key directly from the SCDB provider.
- During configuration or onboarding, the API key is stored securely in the SUNRISE backend.
- When a frontend operation triggers a SCDB API call, the SUNRISE backend proxy attaches the required API key in the HTTP headers as specified in the SCDB's OpenAPI definition.

- The front-end remains unaware of the key and simply consumes the result returned by the backend proxy.

This process provides a simpler alternative to OAuth2 while maintaining secure separation of credentials [6].

2.3.2 Front-end components

The web application, accessible at **ccam-sunrise.eu**, was developed using the **React** framework and is hosted on **AWS Amplify**. The front-end is comprised of the following key components:

- **User Management:** This component handles all user-related functionalities, including user sign-up, registration, and profile updates. It also enables users to request a change of role to become a `scdb_host`.
- **SCDB Management:** Based on the user's permissions (`admin`, `scdb_host`, or regular user), this section allows for the onboarding of new SCDBs and facilitates connections to existing ones.
- **SearchUI:** The SearchUI component empowers users to create new queries by selecting ODD tags. These queries are then dispatched to the connected SCDBs. Once a list of scenarios is retrieved, this component also allows for the download of scenario packages.
- **Dashboard:** This component provides users with a comprehensive overview of a selected SCDB. It displays relevant information such as the total number of scenarios, the different types of scenarios available, the taxonomy in use, and various quality metrics.

2.3.2.1 User Management

The User Management component of the SUNRISE DF front-end application is responsible for handling user authentication and profile administration. It consists of two primary elements:

1. **Accessing Components:** This first element provides an interface for users to register for a new account and log in to the SUNRISE DF. It interacts directly with the AWS Cognito User Pool configured in the back-end to ensure that all user credentials are securely stored and managed throughout the authentication process.

SUNRISE

Sign Up

Name: Enter your Name

Surname: Enter your Surname

Email: Enter your email

Password: Enter your password

Confirm Password: Confirm your password

☐ I agree with the [Terms & Conditions](#) ✓

[Sign Up](#) [Already have an account? Sign In](#)

Sign In

Email: user1@vicomtech.org

Password:

[Forgot Password?](#)

[Sign In](#) [Don't have an account? Sign Up](#)

Figure 10: SUNRISE DF Sign-up and Login pages

The **registration form** requires users to provide their name, email address, and a password. Upon submission, the application automatically sends a **verification link** to the provided email address.

The user must click this link to verify their account before they can log in. After the first successful login, the user is directed to the application's main **landing page**.

SUNRISE Home Dashboard Scenario Search Documentation Hi, User1 User1

Welcome to the Federated European Scenario Database Framework

The SUNRISE Data Framework (DF) implements the data management layer of the SUNRISE Safety Assurance Framework (SAF), with a particular focus on the governance, structuring, and utilization of Scenario Databases (SCDBs). It provides a federated set of services that allow unified access to distributed scenario data, while respecting the autonomy and data policies of individual SCDB providers.

The Data Framework ensures that scenario data can be harmonized, queried, analyzed, and utilized effectively within the SAF. Key concepts that structure the Data Framework include scenarios, the scenario database, and the Operational Design Domain (ODD). These elements are grounded in a shared ontology, ensuring consistent interpretation across tools and stakeholders.

Safety Assurance Framework

UNICE NATM based Harmonized approach for safety assurance of CCAM systems

Features	Methods	Tools	Metrics
Scenario based ODD based Multi-environment Future proof AI compatible Scalable Modular Flexible	Target setting Scenario extraction Scenario coordination Scenario selection Scenario allocation Virtual & physical tests Toolchain validation Risk assessment Safety argumentation	Scenario databases Scenario hub Data Framework Query manager Simulation Framework Online handbook Virtual assistant	Scenario quality Database quality Model quality OOD coverage Test run evaluation Safety performance Pass/Fail

Ontology Structured set of concepts and terms (taxonomy) ensuring accurate information exchange among different components

International alignment Type Approval entities, EC, Consumer testing agencies, Standardization bodies, Twin projects & initiatives

Standards UNECE NATM, UNECE R157, ISO 2450X, ISO 26262, ISO 21448, ASAM, ISO PWS 1883, ISO PWS 1889, FM, SAE J2016

SUNRISE Funded by the European Union

The SUNRISE project is funded by the European Union's Horizon Europe Research & Innovation Actions under grant agreement No. 101060723

About Us Terms of Service Contact

Figure 11 : SUNRISE DF landing page

2. **Profile and Role Management:** Once authenticated, this second element allows users to manage their profiles and update their information. It also facilitates role-based permissions, providing users with the functionality to request an upgrade of their role to scdb_host.

Figure 12: User profile page with option to request change role as SCDB Host

2.3.2.2 SCDB Management

In the SUNRISE DF, managing and connecting external SCDBs begins with a structured **onboarding** process. Each SCDB host is granted exclusive access to a dedicated SUNRISE administrative panel, distinct from the interfaces used by regular users. Through this panel, SCDB hosts can register their SCDB with SUNRISE by providing a fully compliant **OpenAPI** specification file. This file, typically validated using tools like Swagger, defines all available API endpoints, data models, and crucially, the security schemes (such as OAuth2 or API Key mechanisms) supported by the SCDB API. The onboarding process ensures that all technical details about the SCDB are formally captured, standardized, and integrated into SUNRISE.

Figure 13 : Interface in SUNRISE for SCDB hosts to add and configure new databases

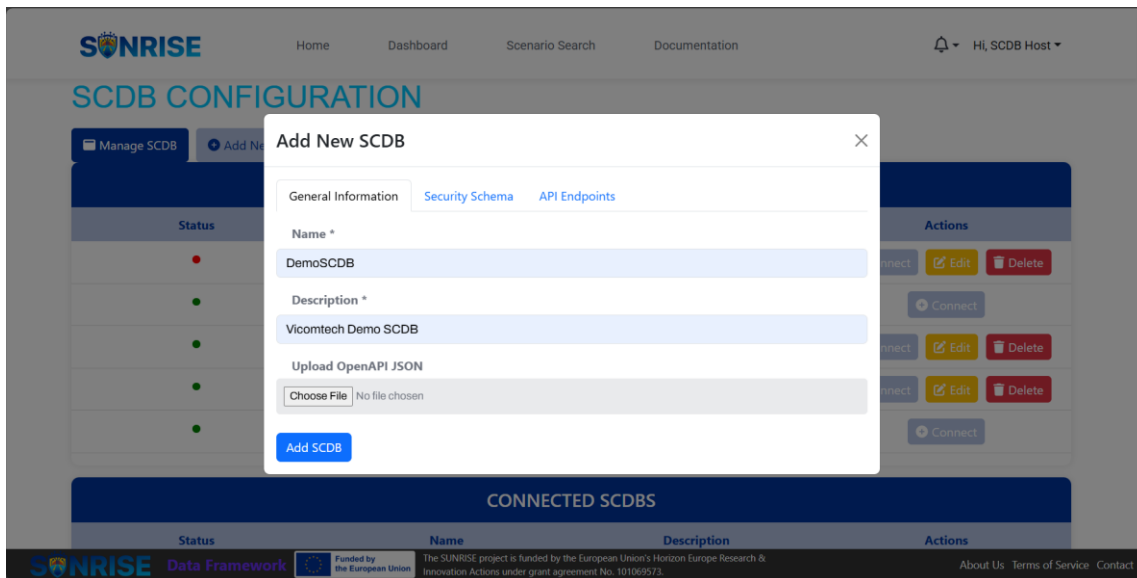


Figure 14: SUNRISE DF form to onboard a new SCDB

Additionally, the SCDB Management component provides functionality for regular users to establish connections with available SCDBs. The interface presents users with a comprehensive list of all accessible SCDBs, each accompanied by a "Connect" button. Upon initiating a connection, a popup window appears, detailing the required authentication steps. The connection mechanism displayed is dependent on the security method supported by the specific SCDB. Currently, the SUNRISE DF supports two security methods: **API-KEY** and the **OAuth 2.0 protocol**.

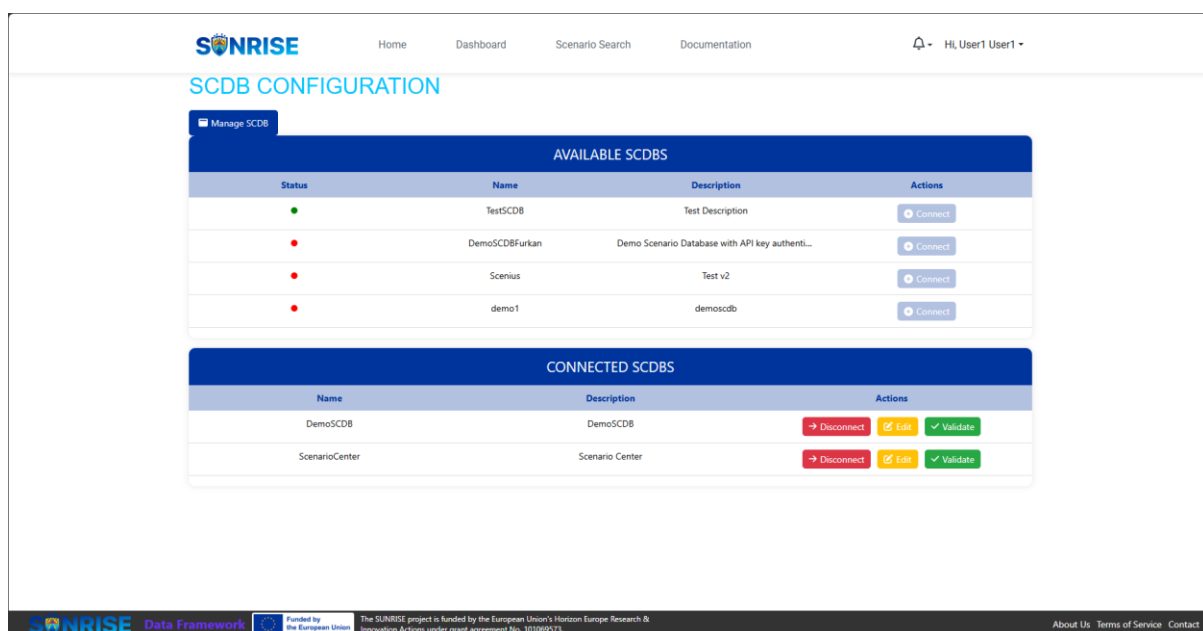


Figure 15 : SCDB management for a regular user

2.3.2.3 Search UI

The Search UI component of the SUNRISE DF serves as the primary interface through which users can explore and retrieve scenarios from connected SCDBs. Designed with usability and flexibility in mind, this component empowers users to construct semantically rich and targeted queries based on standardized ODD tags, submit these queries to SCDBs, and manage the returned results.

Search Workflow:

The search process in the SUNRISE DF is composed of the following stages:

- **ODD-Based Query Builder**

Users begin by selecting relevant ODD attributes from a structured and interactive taxonomy-based interface. These attributes typically include:

- ODD
 - Environmental Conditions
 - Scenery
 - Dynamic Elements
- ROAD USER
 - Animal
 - Human
 - Vehicle
- BEHAVIOUR
 - Motion
 - Communication

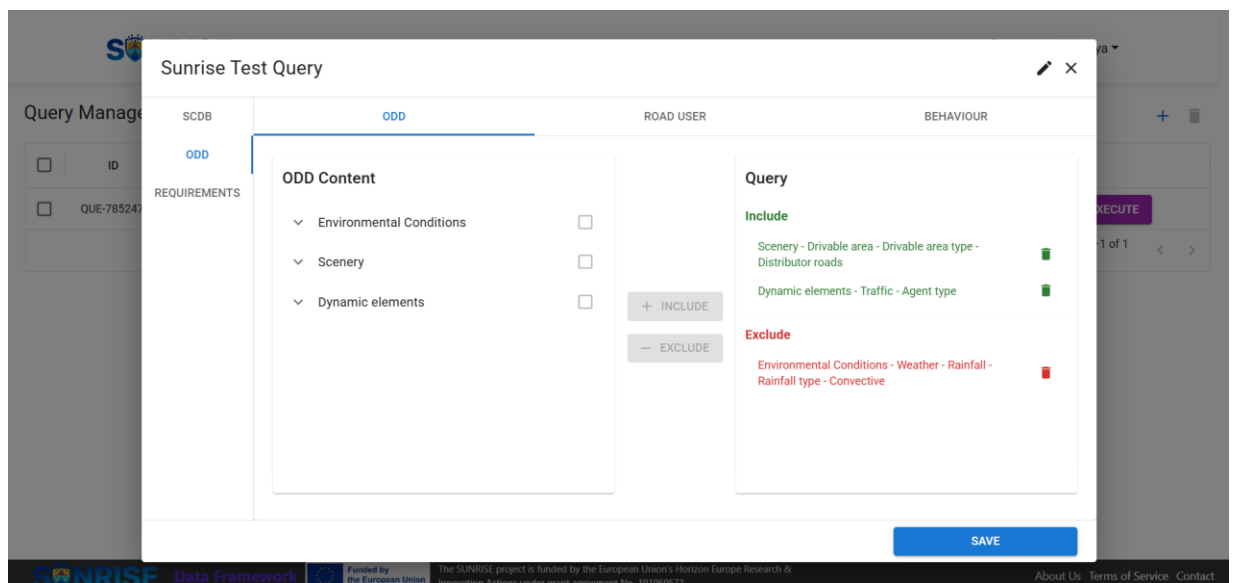


Figure 16: Scenario query based on ODD, Behavior and Road Users

- **Query Dispatch and SCDB Communication**

Once a query is formulated, it is dispatched in real-time to all connected SCDBs using a standardized query schema. Communication leverages the API definitions. Queries

are transmitted concurrently, and the platform handles asynchronous responses from each SCDB.

- **Query Execution & Results Management**

Returned scenarios are aggregated and presented to the user in a unified table or card-based format. Each scenario is accompanied by essential metadata such as:

- Scenario Database, Name, ID, Type and ASAM OpenX
- Scenario Description, Entities, Parameters (Value, Range & Concrete Variation), Statistics, Include and Exclude information's.

The screenshot shows the 'Scenario Search' interface. At the top, there are buttons for '+ NEW SEARCH', 'DOWNLOAD SETTINGS', and 'REQUEST PACKAGE'. Below is a table with columns: SCDB, SCENARIO NAME, SCENARIO ID, SCENARIO TYPE, and ASAM OSC. The table lists 10 scenarios, all of type 'Concrete'. The first scenario is selected. To the right, a sidebar shows details for 'Envelope RTS Scenario 1', including 'ASAM OPENX ATTACHMENTS' (OpenSCENARIO, OpenDRIVE, OpenSceneGraph) and a 'DESCRIPTION' section.

SCDB	SCENARIO NAME	SCENARIO ID	SCENARIO TYPE	ASAM OSC	
<input checked="" type="checkbox"/>	scenario-center	Envelope RTS Scenario 1	scenario-center-envelope-rtts-1	Concrete	✓
<input type="checkbox"/>	scenario-center	Envelope RTS Scenario 2	scenario-center-envelope-rtts-2	Concrete	✓
<input type="checkbox"/>	scenario-center	Envelope RTS Scenario 3	scenario-center-envelope-rtts-3	Concrete	✓
<input type="checkbox"/>	scenario-center	Envelope RTS Scenario 4	scenario-center-envelope-rtts-4	Concrete	✓
<input type="checkbox"/>	scenario-center	Envelope RTS Scenario 5	scenario-center-envelope-rtts-5	Concrete	✓
<input type="checkbox"/>	scenario-center	Envelope RTS Scenario 6	scenario-center-envelope-rtts-6	Concrete	✓
<input type="checkbox"/>	scenario-center	Envelope RTS Scenario 7	scenario-center-envelope-rtts-7	Concrete	✓
<input type="checkbox"/>	scenario-center	Envelope RTS Scenario 8	scenario-center-envelope-rtts-8	Concrete	✓
<input type="checkbox"/>	scenario-center	Envelope RTS Scenario 9	scenario-center-envelope-rtts-9	Concrete	✓
<input type="checkbox"/>	scenario-center	Envelope RTS Scenario 10	scenario-center-envelope-rtts-10	Concrete	✓

1 row selected

1-10 of 10

Envelope RTS Scenario 1

ASAM OPENX ATTACHMENTS

☒ OpenSCENARIO

☒ OpenDRIVE

☐ OpenSceneGraph

DESCRIPTION

Fits tags [] (Tags ['RoadTypeDistributor', 'TrafficAgentType', 'RainTypeConvective'] are not yet supported by this interface).

Figure 17 : Query results page

The screenshot shows the 'Scenario Information' page. On the left, there is a 3D visualization of a road with a car labeled 'Ego (1)'. Below the visualization is a 'Description' section. On the right, there is a form with fields for 'SCDB' (Scenius), 'Scenario ID' (SCEN-448535), 'Scenario Type' (Logical), and 'Modify Date' (6/13/2025). Below these fields are 'ASAM OPENX ATTACHMENTS' (OpenSCENARIO, OpenDRIVE, OpenSceneGraph) and a table of 'ENTITIES'.

Scenario Information

SCDB: Scenius

Scenario ID: SCEN-448535

Scenario Type: Logical

Modify Date: 6/13/2025

ASAM OPENX ATTACHMENTS

☒ OpenSCENARIO ☒ OpenDRIVE ☒ OpenSceneGraph

ENTITIES

Entity	Category
Ego	Car
TOF	Car

Description

Ego Vehicle performs a cut in maneuver behind the Target Vehicle

Figure 18 : Scenario information page

- **Scenario Package Download**

For each scenario (or for selected batches), users can initiate a package download. These scenario packages are provided in a standardized .zip format which contains scenario data (See Table 1, Annex 2 for details).

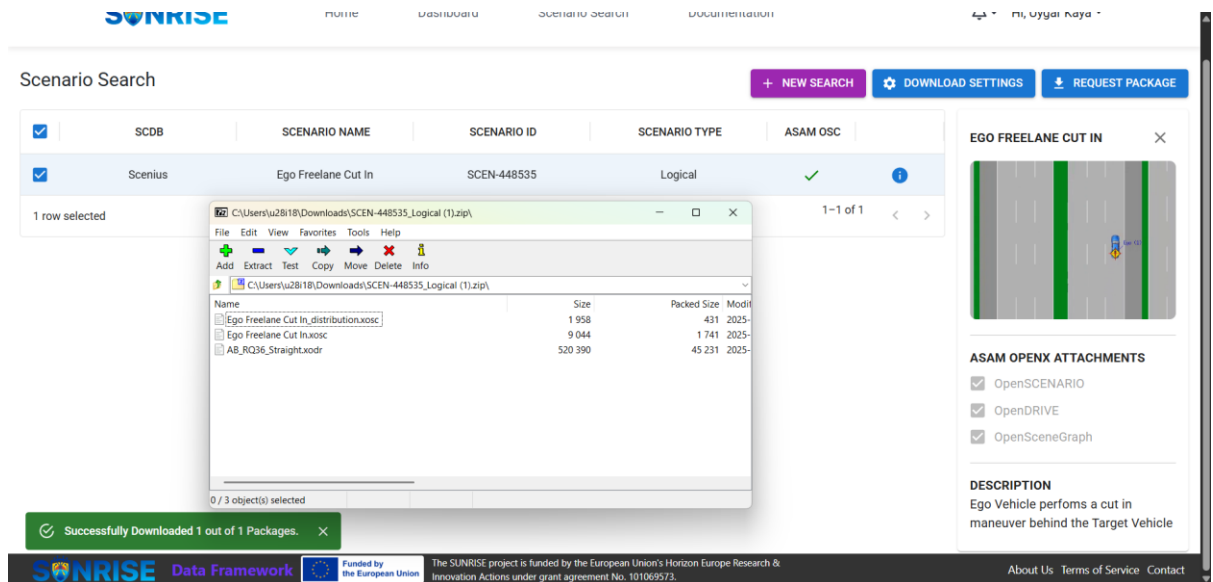


Figure 19 : Download page and internal structure of compressed file

Additional Features:

- **Query History and Reusability:** Users can view and re-execute their previous searches from the session history.
- **Download Settings:** Users can customize the desired OpenX format by modifying the download options.

2.3.2.4 Dashboard

The Dashboard section of the platform was developed to provide a comprehensive overview of a selected SCDB. It functions as a central hub for users to access detailed information about connected SCDBs and monitor their status.

On the left-hand side panel, users can view a list of all SCDBs they are connected to, along with their current health status, which are dynamically populated based on the SCDB Management configuration mentioned above.

Dashboard Structure :

The dashboard is organized into four key pages, each designed to present specific insights:

- **Common Page:**
Displays essential metadata about the selected SCDB, including its name, owner, version, available packages, and the total number of scenarios.

- **Scenarios Page:**
Provides a detailed analysis of SCDB's scenarios. It includes:
 - Filtering and distribution based on entities present in the scenarios, road types, weather conditions, time of day etc...
 - Scenario count by category: Functional, Concrete, and Logical.
 - A Scenario Complexity Score Heatmap to visually represent the complexity and diversity of the scenarios.
- **Taxonomy Page:**
Offers a structured view of the Operational Design Domain (ODD) taxonomy, helping users understand how scenarios are classified according to various driving conditions.
- **Quality Metrics Page:**
Displays critical data quality indicators, including:
 - **Data Completeness:** Measures whether all required data points are present without gaps or missing entries.
 - **Detection Accuracy:** Evaluates how precisely the system identifies objects, events, or features in the data.
 - **Data Freshness:** Indicates how up to date the data is, reflecting the most recent state or condition.
 - **Overall Accuracy:** Assesses how closely the data matches real-world or ground truth values.
 - **Data Consistency:** Ensures uniformity and logical coherence of data across time, systems, and formats.
 - **Distinct Scenarios:** Captures the variety of different environmental or operational conditions included in the data.
 - **Covered Kilometers:** Represents the total distance over which data has been collected, indicating coverage and scale.

This dashboard section enhances user experience by delivering both summaries and detailed metrics, enabling the user to have a clearer understanding of the SCDB.

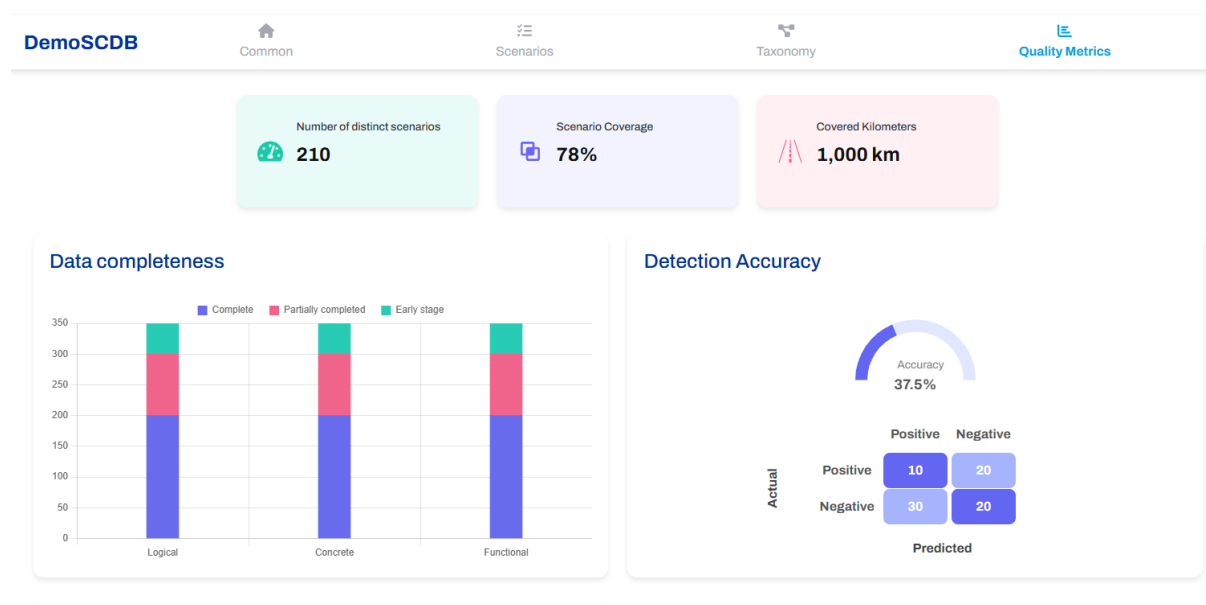


Figure 20 : Overview of metadata and quality metrics for Demo-SCDB

3 SCENARIO DATABASES

3.1 Introduction

Ensuring seamless access to and coordinated management of SCDBs is essential. Each SCDB acts as a well-crafted repository of driving scenarios—spanning real-world incidents to synthetically generated edge cases—encoded in standardized, interoperable formats. Federating these SCDBs through a unified framework gives stakeholders consistent access to scenario data while enabling individual databases to maintain and expand their own content autonomously. This integrated structure supports efficient scenario selection, retrieval, validation, and deployment across virtual, hybrid, and physical testing environments, all governed by harmonized policies and access controls.

Below, we present the key SCDBs available through the SUNRISE DF, highlighting their scenario collections, credential management methods, usage policies, and API interfaces.

3.1.1 Adscene

ADScene is a robust scenario database and description format jointly developed by Renault SAS and Stellantis. It enables the creation and management of digital traffic scenarios, whether manually authored or automatically extracted from real-world driving data. Key capabilities include browsing, versioned storage in access-controlled containers, comprehensive scenario history for traceability, and powerful ODD-based querying.

Key Features:

- Scenario creation & management – Supports both manual authoring and automated extraction from driving data.
- Metadata insights – Provides detailed statistics and parameter distributions to inform informed scenario selection.
- Rich scenario portfolio – Includes accident-derived scenarios and real-world driving events.
- External API access – Exposes secure APIs to enable querying from ecosystem tools.

3.1.2 SafetyPool

Safety Pool™ is a secure and collaborative scenario database designed to accelerate both simulation-based and real-world testing of Connected and Autonomous Vehicle (CAV) technologies. It offers a standardized repository of driving scenarios to support the development, validation, and certification of advanced driving systems [7].

Key Features

- **Extensive Scenario Library:** Contains over 250,000 diverse scenarios curated from expert-designed cases, accident reconstructions, and naturalistic driving data.

- **Dual-Level Description:** Offers human-readable abstracts along with formal, machine-readable definitions, catering to both engineers and automated systems.
- **ODD & Behavior Tagging:** Enables users to filter scenarios by operational domains and specific maneuvers.
- **Platform-Compatible Exports:** Scenarios are compatible with ASAM OpenSCENARIO and OpenDRIVE formats and can be used in various simulation tools.
- **Secure, Token-Based Access:** Includes a contribution-based credit system, granular access controls, version tracking, and full metadata for traceability.

3.1.3 Scenario Center

From RWTH Aachen's Institute of Automotive Engineering (IKA), scenario center is a research database focused on providing driving scenarios based on a structured scenario concept and a highly automated toolchain from original source data to usable driving scenarios. The structured nature of the database allows for more information on distributions over different scenario characteristics, as well as ODD coverage [8].

The original data source for all scenarios available within scenario center is data from inD. This is a naturalistic trajectory dataset with 8200 vehicles and 5300 vulnerable road users recorded. The dataset focuses exclusively on urban traffic. The trajectories from this data set were converted into the OMEGA format (The OMEGA format is depreciated in favour of the OmegaPrime format) ground truth data format.

Scenarios within the database follow the structure of the holistic driving scenario concept for urban traffic. This structure defines all possible bilateral actions between an ego vehicle and another road user. An example for such a base scenario would be "Left turn following a leading object". In total 273 base scenarios have been defined. These can in turn be combined to form more complex enveloping scenarios. These complex scenarios are extracted from the ground truth data.

The scenario.center database supports the extraction of these scenarios in the form of OpenSCENARIO and OpenDRIVE files. Restimulations of the scenarios with esmini are available in video form to help user understanding of the scenario dynamics, as can be seen in Figure 20.

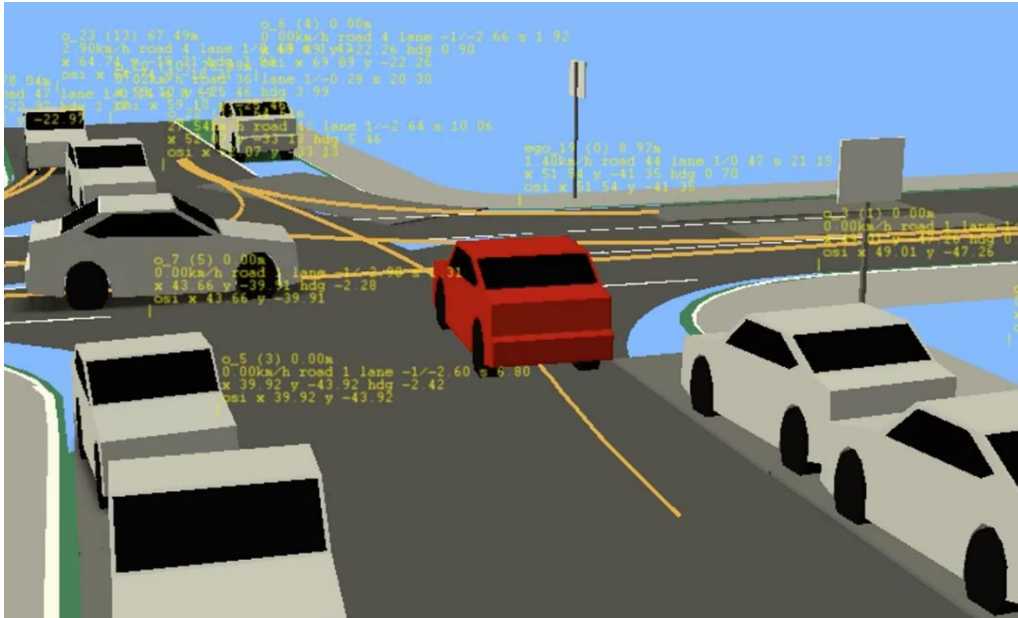


Figure 21 : Re-simulation of a scenario.center scenario.

Highlights:

- **Pipeline:** OMEGA input → automated event detection → logical/concrete scenario exports
- **Formats:** ASAM OpenSCENARIO/OpenDRIVE compatible
- **ODD awareness:** Filters via layer-based environmental and dynamic criteria
- **Generation modes:** Supports “Replay-to-Sim” (RtS), “Advanced Replay-to-Sim” (ARtS), and parametric generation
- Demo available at **scenario.center**

3.1.4 Scenius

AVL Scenius is a holistic solution for scenario-based ADAS (Advanced Driver Assistance System) and AD (Autonomous Driving) safety testing and validation – from requirement engineering, via scenario creation and management, to test case planning, execution, and reporting [9].

For integration with SUNRISE DF, AVL SCENIUS employs API Key-based authentication. Authorized users can access dedicated endpoints within SCENIUS using their assigned API keys. User credentials and licensing information are provisioned and managed by the SCENIUS team.

The SCENIUS platform is fully integrated with SUNRISE DF through a dedicated API controller. This integration includes all necessary endpoints to support operations such as system health checks, user validation, dashboard data retrieval, scenario querying, and scenario package downloads.

Scenarios within SCENIUS are stored in full compliance with the ASAM OpenSCENARIO standard, including both the XML packages and the associated metadata required by

SUNRISE DF. As a result, scenario requests are fulfilled with complete, standards-compliant packages.

Highlights:

- Formats supported: ASAM OpenSCENARIO [10], OpenDRIVE [11], OpenSceneGraph, OpenXOntology [12]
- ODD & KPIs: Users define ODDs and safety KPIs; linked traceability for SOTIF/ISO 26262 compliance
- Scenario Designer with live editor
- Supports MiL/SiL/HiL/ViL simulation, test planning, and reporting

3.1.5 Streetwise

From TNO (NL), **StreetWise** is a methodology developed by TNO to use driving data to extract real-world scenarios, determine the statistics of these scenarios, and use the scenarios for assessing automated driving systems. To support the research and development of StreetWise, it comes with a concept database that contains real-world scenarios.

StreetWise takes a **data-driven** approach. From object-level-trajectory data, it mines tagged, statistically rich scenarios ideal for ML validation, development, risk assessment and ODD coverage evaluation [13].

Highlights:

- **Inputs:** object-level trajectories, often from radar, camera, and CAN logs, or road side sensors.
- **Formats:** Exports to OpenSCENARIO/OpenDRIVE
- **ODD metrics:** Quantifies coverage and reveals “boring miles” vs. risky edge-case distribution
- **Tools:** GUI/API-driven export, compatible with Siemens PreScan and AVL ModelConnect
- Automatically mining scenarios from real-world driving data for creating scenario database content.
- Multivariate distributions of scenario parameter values are derived using statistical methods.
- The estimated multivariate distributions can be used to determine the exposure of concrete scenarios and generate test scenarios that are not necessarily observed in real-world data.
- Only concrete scenarios can be queried and downloaded, not logical scenarios.

Streetwise Scenario classes

The Streetwise Database, hosted on Azure Cosmos DB and accessible to SUNRIS, contains scenarios mined from real driving data. Each scenario is categorized into one of the following scenario classes:

1. Lead vehicle decelerating
2. Cut-in in front of ego vehicle
3. Ego vehicle performing lane change with vehicle behind
4. Lead vehicle cruising
5. Ego vehicle approaching slower lead vehicle
6. Ego vehicle driving in lane without lead vehicle
7. Cut-out in front of ego vehicle
8. Lead vehicle accelerating
9. Ego merging into an occupied lane

Streetwise API implementation for SUNRISE

For the SUNRISE project, Streetwise has implemented a new API. Built using Python's Fast-API framework, the Streetwise-SUNRISE API provides the endpoints required by the SUNRISE federation layer, enabling access to the underlying Streetwise database. The API is deployed as a Docker image on a TNO virtual machine.

API security credentials

The Streetwise-SUNRISE API uses an API key for user authentication. Any user wishing to access the API must possess a valid API key. Currently, only one API key is active for accessing the Streetwise-SUNRISE API. New users who require access should contact the TNO Streetwise team to request the key: Edser.apperloot@tno.nl or Jeroen.broos@tno.nl.

Implemented API endpoints

All SUNRISE API endpoints are implemented within the Streetwise-SUNRISE API:

- **/healthcheck** (GET): Checks whether the Streetwise database is online and accessible by SUNRISE. This endpoint does not require authentication.
- **/validateUserAuth** (POST): Validates the user's credentials using an API key. Returns the user's access_scope, which can be either read or write.
- **/getScdblInfo** (POST): Retrieves general information about the Streetwise database for display in the SUNRISE dashboard. This endpoint requires an API key and returns a JSON schema with relevant SCDB data. For demonstration purposes, it currently returns a static JSON with key SCDB metrics.
- **/getScenarioList** (POST): Returns a list of matching scenarios based on a SUNRISE query. Requires an API key and expects a JSON input based on the OpenLabel schema, containing include/exclude tags selected in the SUNRISE dashboard. Streetwise provides only concrete scenarios (individual scenario data points), not logical scenarios (scenarios' distributions). The response is a JSON object with general scenario information and parameters, formatted as expected by SUNRISE.
- **/getScenarioPackage** (POST): Generates an OpenScenario package for the scenarios selected by the user to download. Requires an API key. This endpoint uses Streetwise's test case generator library to create test cases by sampling from the same scenario distribution as the selected concrete scenarios. As a result, the returned test cases may differ from the exact queried scenarios but maintain the same core characteristics and

parameter distributions of the concrete scenario. The generated package includes both OpenScenario and OpenDrive files.

3.2 Demo SCDB

3.2.1 Objectives of the Demo SCDB within SUNRISE

The Demo SCDB is a mockup or example Scenario Database created as part of the SUNRISE project to demonstrate how an external SCDB can integrate with the SUNRISE Data Framework (DF). Its objectives are:

Demonstrating SCDB Onboarding

- Serve as a prototype database to exercise the full onboarding process of registering a new SCDB into the SUNRISE DF.
- Showcase required information such as SCDB metadata, API specifications, and credential management (e.g. OAuth, API Key).
- Enable demonstration of the UI flows and backend APIs supporting SCDB onboarding

Modeling Interface and API Specifications

- Provide a reference implementation of the expected APIs on the SCDB side, including RESTful endpoints, authentication mechanisms, and query/response structures.
- Illustrate compliance with required data formats and standards (e.g. ASAM OpenSCENARIO, OpenDRIVE, OpenXOntology) for seamless integration into the SUNRISE DF.

Validating Data Compatibility and Formats

- Ensure that scenario data—including metadata, tags, ODD information, and scenario packages—can be successfully exchanged with the SUNRISE DF.
- Provide a controlled environment to test SUNRISE DF components such as the Scenario Manager, Query Manager, and data validation workflows.

Illustrating the End-to-End Use Case Lifecycle

- Demonstrate the complete scenario data flow including:
- Query definition (manual or automated)
- Scenario retrieval from the SCDB
- Validation of retrieved scenarios
- Support traceability between use case requirements and scenario retrieval results.

Providing a Federated Testbed Example

- Serve as an example SCDB that participates in a federated setup within the SUNRISE DF.
- Showcase handling of multiple SCDB connections, harmonized queries, and secure data exchanges while preserving SCDB owners' governance.

- Highlight how session-based temporary storage and processing work within the SUNRISE DF.

The Demo SCDB contains a curated set of example scenarios, including:

- Highway traffic scenarios with simple manoeuvres (overtake, cut-in, ...)
- Highway lane change events
- Two-way roads overtake scenario with more than two entities

All scenarios are stored in standardized formats such as ASAM OpenSCENARIO and OpenDRIVE and include associated metadata compliant with the SUNRISE Ontology. While these scenarios are simplified compared to production-level SCDBs, they demonstrate the data structures, query capabilities, and interoperability envisioned by the SUNRISE Data Framework.

The primary users of the Demo SCDB are the SUNRISE project partners, who have used it extensively for testing and validating the SUNRISE Data Framework functionalities. In addition to its integration within the SUNRISE Data Framework, the Demo SCDB will also be made available through an independent web-based frontend, so that external stakeholder can also make use of it.

3.2.2 Architecture Summary

The Demo SCDB is deployed as a serverless, cloud-native architecture. Key design features include:

- AWS CloudStack-based deployment (CDK)
- Modular architecture with isolated constructs for each service (e.g., S3, DynamoDB, Cognito, Lambda)
- REST API gateway with defined resource routes
- Integration with federated SUNRISE DF via secure APIs and credential management
- Data stored in both object storage (S3) and structured metadata tables (DynamoDB)
- Scheduled cleanup for temporary storage to simulate SUNRISE DF session-based data retention

3.2.3 Backend Architecture

The main backend is defined in a CDK lib file which orchestrates the full cloud resources for the Demo SCDB, including:

- Cognito User Pool for authentication
- DynamoDB for metadata storage
- S3 for scenario package storage
- Lambda functions for business logic
- Scheduled cleanup Lambda for temporary data
- API Gateway for exposing REST APIs

All components are provisioned using AWS CDK (Cloud Development Kit) in TypeScript, ensuring infrastructure-as-code and easy reproducibility.

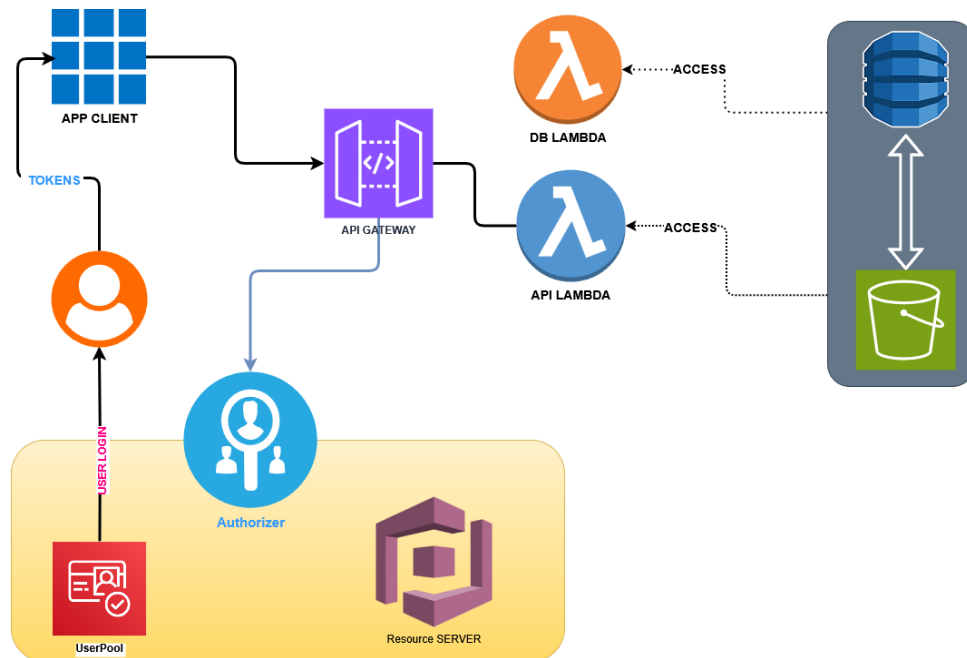


Figure 22: AWS Resource Architecture and Service Connections for the Demo SCDB

The following sections provide a brief overview of the internal structure of each of the components implemented in the CDK, to further clarify the data model and the workings of the backend of the Demo SCDB.

3.2.3.1 Cognito Authentication

The Demo SCDB creates a Cognito User Pool that provides:

- User sign-up and sign-in
- Token issuance (JWT) for secure authentication
- Integration with API Gateway for protecting endpoints

The Cognito user pool defines four distinct user groups, each with specific roles and permissions:

1. **Guest** — Has limited access, can view a list of scenarios and download a small number of scenarios.
2. **Standard** — Has unlimited access to view and download scenarios.
3. **Contributor** — Can view and download scenarios and is allowed to upload new scenarios.
4. **Admin** — Has full access to all data and functions, including user management capabilities.

3.2.3.2 DynamoDB

The Demo SCDB uses a modular and efficient DynamoDB schema to store and manage scenario metadata and relationships. The architecture relies on two primary tables:

1- Scenario

This single-table design ensures all scenario-related metadata, file references, and associated details are captured in a single DynamoDB item, optimizing retrieval and reducing the need for multiple table joins. The scenarios table stores comprehensive metadata for each scenario, including:

- Id: Unique scenario identifier
- scenarioName: Name of the scenario
- description: Scenario description
- scenarioDatabase: Indicates the source database (e.g., VICOMTECH_SCENARIO_DB)
- scenarioType: Scenario classification (e.g., Concrete)
- tags: Array of tag identifiers associated with the scenario
- openXAvailability: Flags indicating availability of formats like OpenSCENARIO (OSC), OpenDRIVE (ODR), or OSG
- entities: List of entities participating in the scenario
- scenarioParameters: Detailed parameter definitions relevant to the scenario
- odd: Operational Design Domain attributes
- files: List of file objects including filename, S3 storage key, content type, upload timestamp

2- Tag-scenarios

This table enables efficient reverse lookups, such as retrieving all scenarios associated with a specific tag. By separating tag associations into a dedicated table, the system avoids expensive scans of array attributes in the primary scenario table, significantly improving query performance. This table acts as a **many-to-many** mapping between tags and scenarios:

- tagId (Partition Key) – The tag identifier
- scenarioid (Sort Key) – References the linked scenario

Together, these tables support rapid, indexed retrieval of scenarios and ensure scalable storage for both metadata and tag relationships.

The Demo SCDB design intentionally avoids creating a separate table for tag definitions because tag metadata is centrally stored in S3 as a JSON ontology, ensuring consistency, versioning, and easier updates.

This DynamoDB structure models how SCDBs store scenario descriptors in a way that aligns with the SUNRISE DF architecture, providing both flexibility and performance for querying and managing large volumes of scenario data.

3.2.3.3 S3 Storage

The Demo SCDB uses Amazon S3 as the primary storage layer for all scenario-related files. This includes:

- Scenario Packages – Files in OpenSCENARIO (.xosc), OpenDRIVE (.xodr), or other formats used for defining traffic scenarios
- JSON Metadata – Machine-readable metadata describing scenarios
- Image Files – Visual assets linked to scenarios, such as thumbnails or diagrams

All files are organized in a structured key format (`{scenarioId}/{fileName}`) and Scenario file references, including S3 keys, content types, and upload timestamps, are stored in DynamoDB, ensuring seamless linkage between the metadata layer and the physical files in storage. This architecture ensures data consistency, security, and efficient retrieval of large binary assets required for simulation and validation workflows within the SUNRISE DF.

3.2.3.4 Lambda Functions

Two primary Lambda constructs have been developed within the Demo SCDB architecture, each serving distinct purposes. In this document, a “Lambda construct” means how AWS Lambda functions are grouped and defined as reusable components in the AWS CDK.

- i) The first set of Lambdas is designed for **API integration**. These functions are triggered by API Gateway requests and handle all necessary computations, including database interactions and business logic processing. They manage key operations such as:
 - Scenario uploads
 - Metadata retrieval
 - Data validation
 - Query processing

These Lambdas are tightly coupled to specific API routes through API Gateway integrations, enabling seamless request-response handling.

- ii) The second Lambda construct, defined in **Cleanup-construct**, is focused on **internal data management**. These functions are invoked either on scheduled intervals or in response to changes in the database. They are responsible for performing **cascade actions**, such as:
 - Synchronizing deletions or updates between the scenario table and related tag-scenario tables
 - Removing corresponding files from S3 buckets when an associated scenario record is deleted

Together, these Lambda functions implement the Demo SCDB's core backend logic, ensuring both external API services and internal maintenance processes operate efficiently and reliably.

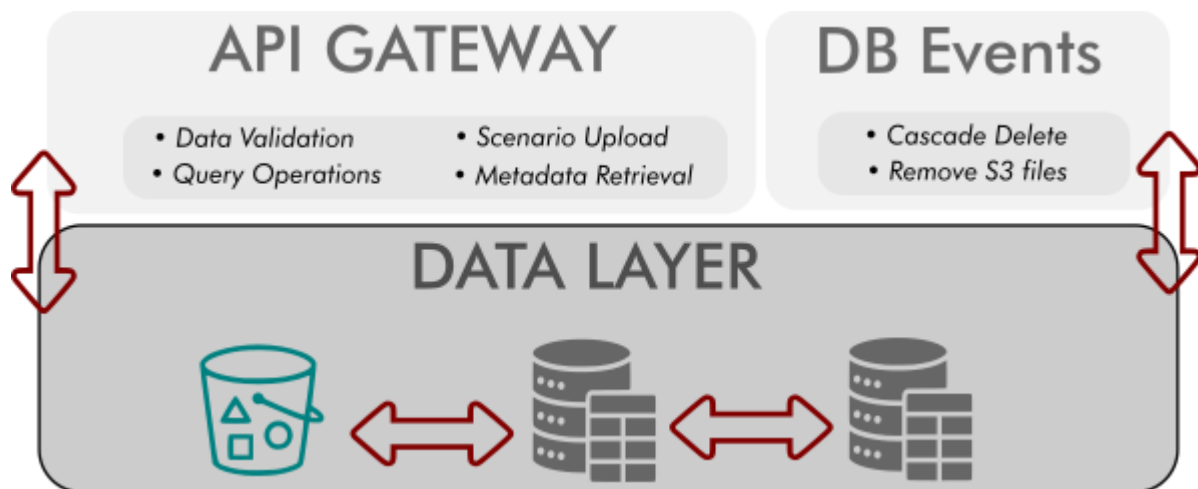


Figure 23: Architecture of the lambda function in the AWS CDK

3.2.3.5 API Gateway

Demo SCDB uses Amazon API Gateway to expose a comprehensive set of RESTful API endpoints, serving both the SUNRISE Data Framework requirements and the internal operational needs of the Demo SCDB.

This API layer is configured with:

- **Resource-based routing** — Endpoints are organized as REST resources with logical paths (e.g. /scenarios, /files, etc.) to model different operations and services.
- **Cognito integration** — All protected routes are secured using AWS Cognito, which enforces authentication and supports fine-grained authorization through OAuth 2.0 scopes. This ensures compliance with the SUNRISE federation security model.
- **CORS configuration** — Cross-Origin Resource Sharing (CORS) is fully enabled to allow secure requests from external front-end clients, such as the SUNRISE DF web interface.

The API allows:

- All HTTP methods (GET, POST, etc.)
- Common headers like Authorization, Content-Type, X-Amz-Date, and custom tokens
- Credentialed requests, supporting secure cookie or token-based authentication
- **Custom error handling** — Standardized API Gateway responses are defined for unauthorized and access-denied cases, ensuring consistent client-side error handling and seamless integration into front-end applications.

The Demo SCDB API Gateway implements **two main categories of endpoints**:

1. SUNRISE-Compliant Standard APIs

These are designed to fulfil SUNRISE DF's interoperability requirements. They include:

- **Five standard endpoints** that follow the data models and JSON schemas defined in Annex 2.

- OAuth-protected routes that support scopes for both read and write operations, ensuring secure integration with the federated SUNRISE ecosystem.

2. Demo SCDB Internal and Administrative APIs

In addition to the SUNRISE-standard endpoints, the Demo SCDB provides several **private APIs** for internal operations and administrative tasks. These include:

- File uploads and retrieval (e.g. presigned URL generation, direct uploads to S3)
- Scenario metadata population and validation
- Flexible methods for creating, retrieving, and updating scenarios
- Administrative or operational endpoints that mirror the types of APIs found in any production-grade Scenario Database

These internal APIs are designed for the day-to-day management of the Demo SCDB itself and are not part of the official SUNRISE federation interface. They are, however, representative of capabilities any real-world SCDB would implement for managing its data and content.

Security and Usage Controls

- **Authorization and Scopes:** The Demo SCDB uses Cognito authorizers and resource servers to manage fine-grained permissions through OAuth 2.0 scopes, ensuring that only authorized users or systems can invoke sensitive operations.
- **API Keys and Usage Plans:** Some internal APIs also enforce API key requirements to manage usage and apply throttling rules, which limit how many requests a client can make in a certain time period. This helps protect backend services and ensures fair usage among clients.
- **Binary Support:** The API Gateway is configured to support binary media types like ZIP archives and generic binary streams, enabling flexible file upload and download operations.

This architecture provides both **compliance** with SUNRISE federation standards and the **flexibility** to support the operational needs of a standalone scenario database, ensuring the Demo SCDB serves as a practical, real-world example for future SCDB implementations.

3.2.4 DEMO SCDB Frontend

To support the development and testing of the Demo SCDB, we created a dedicated frontend application using **React** and the modern **Vite** build toolchain. This frontend serves several critical purposes. Firstly, it enables comprehensive testing and demonstration of all backend APIs, including scenario uploads, retrieval of SCDB information, searching, and executing operations defined by the SUNRISE Data Framework. It also includes advanced administrative views for monitoring database tables, managing file uploads, and visualizing backend system health. Additionally, the frontend implements token inspection and decoding capabilities, allowing user information and group memberships directly within the UI for effective testing of role-based access controls (RBAC).

From a technical perspective, the frontend leverages **React Router** for navigation, **TypeScript** for type safety, and various libraries for UI components and animations, creating a modern, responsive user experience.

For production deployment, the compiled static assets from the Vite build are intended to be distributed via **AWS CloudFront**, providing global low-latency access and caching. The application is configured to read all necessary backend URLs and environment-specific settings from environment variables defined in the .env file, ensuring flexibility and secure separation of configuration from code.

This frontend not only accelerates development and integration testing with the SUNRISE DF but also serves as a reference implementation for how SCDB frontends can interact with the SUNRISE DF.

3.2.5 Scenario Generation

To streamline the generation and population of scenarios within the Demo SCDB, a dedicated Python toolkit was developed, that automates both scenario synthesis and metadata generation. This toolkit serves as a wrapper around the scenariogeneration library [14] and integrates ontology processing logic to produce meaningful, structured metadata. The metadata is compatible with both the OpenLABEL standard, and the internal JSON schemas defined by the deliverable D5.2 for queryability and semantic annotation.

Crucially, this metadata—generated alongside OpenSCENARIO and OpenDRIVE scenario files—is designed to align with the schema expected by our backend DynamoDB tables. When scenarios are uploaded via the frontend interface, the logic implemented there parses and transforms the metadata to seamlessly populate both the scenario and tag-scenario tables. Fields such as scenario ID, name, description, tags, ODD descriptors, scenario parameters, file references, and ontology-driven entity relationships are populated automatically, minimizing manual input and reducing integration errors.

Together, the Python toolkit and frontend logic form a cohesive pipeline that orchestrates the full workflow: from automated scenario generation, through standardized metadata creation, to secure, structured ingestion into the Demo SCDB backend. This ensures consistency, scalability, and full alignment with SUNRISE ontology.

4 CONCLUSIONS

This deliverable contains an explanation and documentation of the SUNRISE Data Framework (DF), a supporting tool for the SUNRISE Safety Assurance Framework (SAF). The SUNRISE DF implements a federated architecture that enables seamless, secure, and standardised access to multiple external scenario databases (SCDBs), addressing the critical need for a harmonized European-level scenario library. The SUNRISE DF is deployed in a cloud service and can be accessed at **ccam-sunrise.eu**

The main achievements of the deliverable are:

- **Design and deployment of the SUNRISE Data Framework:** a modular, **cloud application** developed, integrating several **back-end and front-end components as explained in Chapter 2**, to support **secure user access, scenario querying, and data exchange** with the external SCDBs that connect to it. The architecture ensures scalability, interoperability, and compliance with diverse security models (OAuth2, API keys).
- **Interconnection with external SCDBs:** the SUNRISE DF has successfully been connected with several major SCDBs: **ADSCENE, SafetyPool, Scenario.center, Scenius, and Streetwise**, demonstrating its ability to interoperate using the defined **SUNRISE input and output interfaces** as defined in tasks T6.1 and T6.2, and effectively provide data content from the external SCDBs in a harmonized manner. Details about each SCDB connected to SUNRISE DF could be found in **Chapter 3**.
- **Development of the Demo SCDB:** a fully functional reference implementation of a scenario database has been created to illustrate **onboarding procedures, API compliance, and standard interfaces**. This Demo SCDB serves as a blueprint for future external SCDB integrations and has supported validation of the SUNRISE DF components during the SUNRISE project. A complete description of the implementation of Demo SCDB is found in **Section 3.2**.
- **Definition of API requirements:** a comprehensive set of technical specifications established to guide external SCDBs owners to create a **programmatic interface** between applications using **OpenAPI interface** definition. More information is found in **Annex 2**.
- **Support for ontology-driven querying:** the SUNRISE DF enables **semantic scenario search** through harmonised tagging and ontology, facilitating harmonisation and standardisation (more information in deliverables D5.2 and D6.2).

The work presented in this deliverable, the SUNRISE Data Framework, paves the way for the harmonisation and centralisation of scenario databases among European practitioners of scenario-based testing, which in turn supports the adoption of the SUNRISE Safety Assurance Framework.

Looking ahead, several key areas for improvement have been identified for the **SUNRISE DF**. Immediate enhancements should focus on the **query generation** process by providing support for valued parameters and integrating the recently finalized **T5.2 ontology** [2] to enrich scenario querying.

A further consideration is the implementation of a mechanism to upload scenarios directly to SCDBs. However, the viability of this feature is contingent on future discussions and analysis to confirm whether external SCDBs can and will accept such uploads.

5 REFERENCES

- [1] SUNRISE, “D3.4 Report on the Initial Allocation of Scenarios to Test,” 2024. [Online].
- [2] SUNRISE, “D5.2 Harmonised descriptions for content of CCAM safety assessment data framework,” 2025. [Online].
- [3] SUNRISE, “D5.3 Quality metrics for scenario database,” 2025. [Online].
- [4] SUNRISE, “D6.1 Methodology for SCBD application for generic use cases,” 2024. [Online].
- [5] ASAM, “ASAM OpenLABEL®,” ASAM, 12 November 2021. [Online]. Available: <https://www.asam.net/standards/detail/openlabel/>. [Accessed 23 August 2024].
- [6] D. Hardt, “The OAuth 2.0 Authorization Framework,” Fremont, CA (optional – for IETF reports), 2012.
- [7] WMG, “Safety Pool™ Scenario Database,” 2024. [Online]. Available: <https://www.safetypool.ai/>. [Accessed 11 July 2024].
- [8] C. G. L. E. Michael Schuldes, “scenario.center: Methods from Real-world Data to a Scenario Database,” *IEEE Intelligent Vehicles Symposium (IV)*, pp. 1119-1126, 2024.
- [9] AVL, “AVL SCENIUS,” AVL, 2024. [Online]. Available: <https://www.avl.com/en-es/testing-solutions/automated-and-connected-mobility-testing/avl-scenius>.
- [10] ASAM, “ASAM OpenScenario,” 2024. [Online]. Available: <https://www.asam.net/standards/detail/openscenario/>. [Accessed 07 July 2024].
- [11] ASAM, “ASAM OpenDRIVE,” 2024. [Online]. Available: <https://www.asam.net/standards/detail/opendrive/>. [Accessed 07 July 2024].
- [12] ASAM, “ASAM OpenXOntology Concept,” ASAM, 2021. [Online]. Available: <https://www.asam.net/standards/asam-openxontology/>.
- [13] TNO, “StreetWise: scenario based safety assessment for automated driving,” TNO, 2025. [Online]. Available: <https://www.tno.nl/en/digital/smart-traffic-transport/safety-assessment-automated-driving/streetwise/>.
- [14] I. N. A. T. J. K. Mikael Andersson, “Github repository,” 2025. [Online]. Available: <https://github.com/pyoscx/scenariogeneration>.

ANNEX 1. ONBOARDING TO SUNRISE DF

Purpose and Scope

The SUNRISE Data Framework (DF) establishes a federated approach for scenario data sharing, enabling seamless integration of multiple scenario databases (SCDBs) while respecting data governance and ownership. Access to scenario data within each SCDB is managed according to the policies and requirements of the respective SCDB host¹, meaning users must obtain *credentials or authorization* directly from each SCDB host. The SUNRISE DF facilitates a harmonized interface and supports secure, traceable user access but does not override individual SCDB access controls. This ensures that SCDB owners maintain full control over their data and compliance with relevant agreements, while users benefit from a unified, ontology-driven query mechanism that spans diverse databases under common standards.

This document guides external SCDB owners through the four-phase onboarding process to connect their scenario metadata to the SUNRISE DF. It outlines mandatory requirements, API interface expectations, ontology alignment, and long-term responsibilities. Integration with the SUNRISE DF enables centralized access, semantic querying, and validation-ready scenario discovery.

The onboarding process is organized into four progressive phases, as depicted in Figure 24, each designed to facilitate smooth and sustainable integration:

- Phase 1: Readiness Assessment
- Phase 2: Initiation and Planning
- Phase 3: Implementation and Validation
- Phase 4: Operation and Lifecycle Management

¹ NOTE: in this document we refer to SCDB hosts, owners or providers, as synonyms, referring to entities that manages, owns or serves scenario databases services

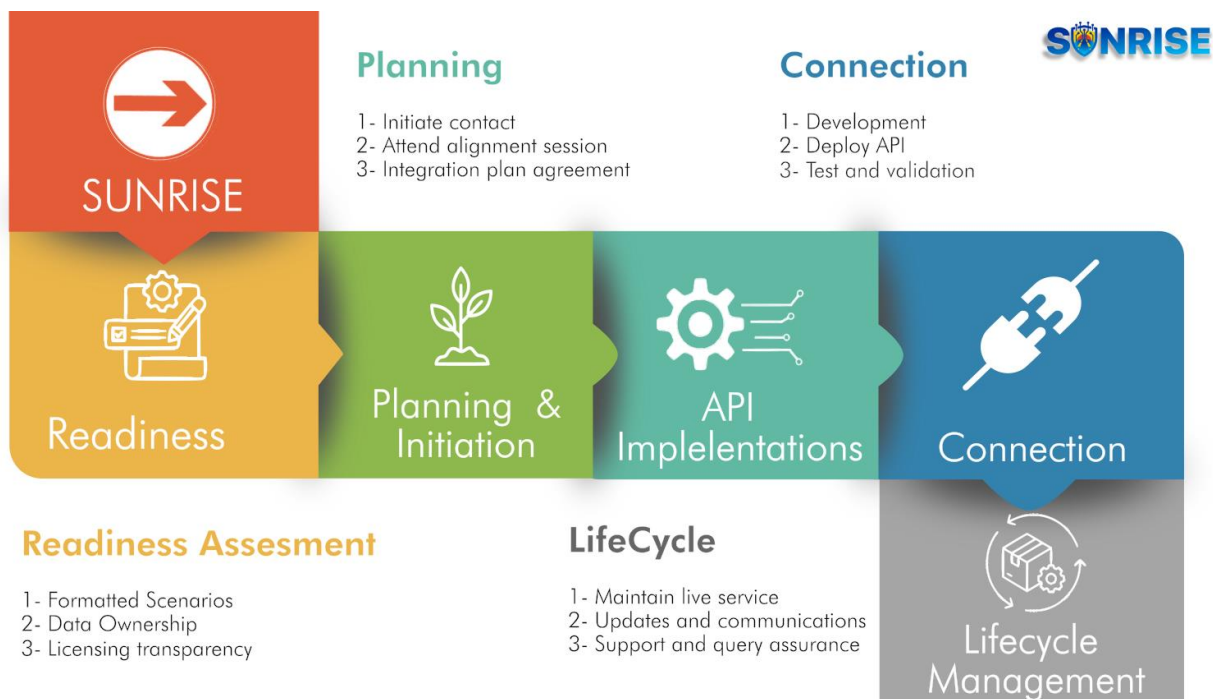


Figure 24 : SUNRISE Data Framework onboarding process.

Preparation & Requirements

Before registering as an SCDB Host in SUNRISE, providers must ensure:

Scenario components: Each scenario inside the SCDB includes a description file, a road network file, and all required scenario definition files (i.e., an OpenScenario XML-compliant .xosc file and an OpenDrive 1.X-compliant .xodr file). Additionally, optional files like scene graph files (e.g., osgb, .opt. osgb, or. ive) and OpenScenario distribution files may be provided and will be made available to users upon request.

Metadata structure: The metadata for each scenario in an SCDB should include general information—such as scenario ID, name, and description—as well as a comprehensive set of query-relevant tags describing scenario attributes (e.g., manoeuvrers, environmental conditions, participants, etc.). While these tags do not need to follow the SUNRISE Ontology² natively, they must be sufficiently detailed and complete to support mapping to the SUNRISE Ontology as needed for integration.

For example, consider a case where a user wants to find all scenarios involving a “cut-in” manoeuvre occurring in rainy weather. If the internal metadata includes appropriate tags (such as “cut in” and “rainy weather” or similar equivalents), it is possible to map these to the SUNRISE Ontology during integration. This ensures that such queries will yield all applicable scenarios, enabling seamless and consistent search functionality across different databases once connected via the SUNRISE Data Framework.

² SUNRISE Ontology – see SUNRISE deliverable “D5.2 Harmonised descriptions for content of CCAM safety assessment data framework”

Initiation & Planning

Once the above requirements are met, providers should:

- **Register and Request SCDB Host Role:**
 - Sign up via the SUNRISE sign-in page.
 - Navigate to your profile and request the SCDB Host role.
- **Alignment Session:**
 - Once the SCDB Host role is approved, a planning session is available on request with the SUNRISE DF team to discuss architecture, API integration, metadata expectations, ontology alignment, and the onboarding timeline.

Implementation & Validation

At this stage, the SCDB owner is expected to integrate with the SUNRISE framework by providing a stable API in the SUNRISE format, supporting ontology-based scenario queries, and participating in validation and testing. A query mapping layer may be required to handle SUNRISE-tagged queries. SUNRISE will provide support throughout the process. The main activities include:

API Implementation and Deployment: Develop and deploy a robust API for the SCDB, ensuring all endpoints are clearly documented using OpenAPI 3.0+ (see Annex 2). The API must support query and response structures defined by the SUNRISE-provided JSON schema, enabling consistent and standardized communication with the SUNRISE DF.

Ontology Alignment and Query Mapping: Adapt the SCDB's data model and query handling logic to align with the SUNRISE ontology and tagging structure. If necessary, implement a query mapping layer to translate SUNRISE-compliant queries into SCDB-specific filters and ensure that responses use the correct tags and formats. Reference examples and further guidance are available in the SUNRISE documentation.

Validation and Testing: Engage in a validation phase led by the SUNRISE team, which will include reviewing API documentation, test endpoints, and verifying that the SCDB correctly handles queries and returns accurate results. This process ensures full interoperability and compliance with SUNRISE requirements.

Operation & Maintenance

After successful validation, the SCDB is integrated into the SUNRISE DF. Ongoing participation is based on the following recommendations:

Availability: It is advisable to uphold robust operational reliability of metadata endpoints, aiming for sustained high availability, and to ensure that any service interruptions are communicated proactively.

Quality Assurance: SCDB Providers are encouraged to uphold high standards of accuracy, completeness, and consistency in all scenario metadata. Maintaining this level of quality supports reliable data integration and downstream usage.

Coordination: Ongoing, open communication is strongly encouraged. Providers should proactively inform the SUNRISE team of any upcoming structural changes and are encouraged to address feedback or support requests in a timely and collaborative manner.

Summary for Onboarding

- Read terms and conditions of SUNRISE DF
- Ensure scenario files and summaries are in place
- Confirm metadata structure and ontology readiness
- Maintain full data ownership and licensing transparency
- Initiate contact with SUNRISE DF
- Attend alignment session and agree on integration plan
- Develop and publish compliant metadata API
- Undergo testing and validation with SUNRISE
- Maintain live service, metadata quality, and communication

Contacts and References

- **Onboarding support:** contacts@ccam-sunrise-project.eu
- **Ontology and metadata guidance:** contacts@ccam-sunrise-project.eu
- **Templates and schemas:** *SUNRISE Documentation Portal*

Onboarding Step-by-Step Guide

1. Introduction

The SUNRISE Data Framework enables integration of external Scenario Databases (SCDBs) through a secure, standardized onboarding process.

This guide provides **step-by-step instructions with screenshots** to help SCDB providers:

- Register for the SUNRISE platform
 - Request and obtain the SCDB Host role
 - Configure and connect a new SCDB
-

2. Registration in SUNRISE

To begin, register for a SUNRISE account:

Visit the SUNRISE Sign-In Page³, and Click **Sign Up** and complete the registration form:

- Full Name
- Email Address
- Password

Once completed, you'll receive a confirmation email.

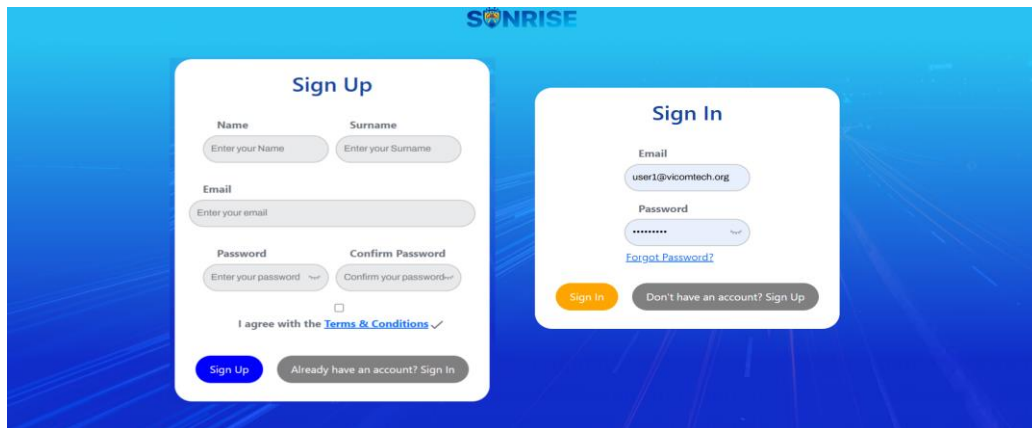


Figure 25: SUNRISE User Registration Screen

3. Requesting the SCDB Host Role

After registering and signing in:

- Click your profile icon in the top-right corner.
- Select **My Profile**.
- Click **Request Role change to SCDB Host** and submit your request.

Your request will be reviewed by the SUNRISE DF team. You will receive email confirmation once approved.

³ ccam-sunrise.eu

SUNRISE Home Dashboard Scenario Search Documentation Hi, User1 User1

PROFILE

First Name* User1 Last Name* User1 Email* user1@vicomtech.org

Save Change Password Request Role Change to SCDB Host Delete Profile

Figure 26: Requesting SCDB Host Role

4. Add a new SCDB

Once the SCDB Host role is approved:

You will continue to access **Manage SCDB** from the dropdown menu in the top-right corner.

However, an additional tab called **“Add new SCDB”** becomes visible within the **SCDB Configuration page**. This tab allows SCDB Hosts to create and configure new SCDB integrations.

SUNRISE Home Dashboard Scenario Search Documentation Hi, SCDB Host

SCDB CONFIGURATION

Manage SCDB Add New SCDB

AVAILABLE SCDBS			
Status	Name	Description	Actions
●	Streetwise	TNO Streetwise Scenario Database	Connect Edit Delete
●	Scenius	Added by AVL Team	Connect
●	ScenarioCenter	Scenario.Center SCDB	Connect Edit Delete
●	AdScene	AdScene Database	Connect Edit Delete
●	AVL Scenius	AVL Scenius	Connect

CONNECTED SCDBS			
Status	Name	Description	Actions

Figure 27: Manage SCDB dropdown in user menu and SCDB Configuration

Clicking “add new SCDB” Tab a pop-up window opens, containing three tabs for configuration:

Tab 1 – General Information

- Enter the SCDB **Name** and **Description**.
- Upload the **OpenAPI JSON specification** file for your SCDB.

Tab 2 – Security Configuration

Based on the uploaded OpenAPI file, select the **security schema** required for your SCDB integration (e.g. API key, OAuth).

Tab 3 – Endpoint Mapping

For each of the five core endpoints required by SUNRISE, choose the corresponding endpoint path from dropdown lists.

Example: Map the **HealthCheck** function to `/healthcheck` or your equivalent endpoint. After completing all tabs, click **Save** to finalize your SCDB setup. You can then test the connection to ensure successful integration.

The figure displays two screenshots of the 'Add New SCDB' form. The left screenshot shows the 'General Information' tab, which includes fields for 'Name *', 'Description *', and 'Upload OpenAPI JSON' (with a 'Choose File' button). Below these is an 'Add SCDB' button. The right screenshot shows the 'API Endpoints' tab, which features five dropdown menus for mapping API endpoints: 'API Endpoint - Health Check', 'API Endpoint - Validate User Connection', 'API Endpoint - Get Scenario List', 'API Endpoint - Get Scenario Package', and 'API Endpoint - Get SCDB Information'. Each dropdown has an 'x' icon and a downward arrow. An 'Add SCDB' button is also present at the bottom of this tab.

Figure 28: Tabs related to add new SCDB

ANNEX 2. SCENARIO DATABASE API REQUIREMENTS

Introduction

This document describes requirements for the endpoints that will be used for data exchange between SUNRISE Data Framework (DF) and individual scenario databases (SCDB).

Data Framework Components section contains information related to main components that take place in SUNRISE DF. More detailed information for each component can be found in the project folder.

API Endpoints section contains technical specifications related to endpoints that need to be implemented by individual SCDBs. Example request, response and schema files will be distributed with this document.

Data Framework Components

Dashboard

Dashboard is used to show information related to connected databases.

Search UI

Search UI is used to retrieve scenarios from SCDBs by using query criteria.

Scenario Manager

Scenario Manager is the component that applies validity checks on retrieved scenarios.

API Endpoints

SCDB Health Check - /healthCheck

Endpoint Objective

This endpoint reports the SCDB's online status, allowing the SUNRISE Data Framework to confirm that the SCDB is operational and ready to receive requests and facilitate interaction.

Use Cases

- When onboarding a new SCDB, the SUNRISE Data Framework will check the online status of the SCDB before uploading the SCDB API details to SUNRSIE Data Framework database.
- When a user wants to connect to a SCDB, the SUNRISE Data Framework will check the online status of the SCDB.

HTTP Method

HTTP GET

Expected Behaviour

Input Validation:

No input is required in this endpoint

Business Logic:

The SCDB will return its availability to perform API requests

Dependencies:

There is no SUNRISE DF related dependency defined for this endpoint.

Response Requirements

Expected Outputs

This endpoint will return a JSON message with the online availability of the SCDB. The returned message will follow a JSON schema which is attached with the complementary files of this document.

Response Status Codes

- **200:** OK - Dashboard information successfully retrieved.
- **400:** Bad Request – Request is not formed correctly
- **404:** Not Found – Requested endpoint does not exist
- **429:** Too Many Requests – Request rate is higher than SCDB can handle
- **503:** Service Unavailable – Service / Endpoint is temporarily unavailable

Error Handling

Error responses should include details about the issue

Non-Functional Requirements

Security

This endpoint will not have any security layer; the request should be user-agnostic. Meaning that no authentication or authorization is required.

Usability

- Error messages must be descriptive and provide actionable insights to users or developers.

- JSON responses should adhere to a clear and consistent schema, making it easier for frontend and backend developers to consume and debug the data.

Scalability

With the large number of users of SUNRISE DF, the SCDB API should be able to handle many requests.

Validate User Auth - /validateUserAuth

Endpoint Objective

This endpoint verifies user authentication against the SCDB. It enables the SUNRISE DF to validate provided user credentials, such as API keys or tokens, and confirm authorization for SCDB API requests. In addition, this endpoint should allow to receive the access scope of the user to the SCDB.

Use Cases

- This endpoint validates the provided API-key for SCDB implementations utilizing API-key authentication. Successful validation confirms the key's authorization to access SCDB API resources.
- SUNRISE DF utilizes this endpoint to verify user authentication against SCDB, supporting both OAuth 2.0 and API-key authentication methods.

HTTP Method

HTTP POST

Headers

Header	Description
Content-Type	application/json
Accept	application/json
X-API-Key	Optional API key for authentication
Authorization	Optional OAuth2 Bearer token

Expected Behaviour

Input Validation:

This endpoint does not mandate a request body. However, depending on the security mechanisms implemented by the respective SCDB API (API-KEY, token, OAuth2), certain parameters will be necessary for successful authentication. Examples include:

- API Key Authentication: The API-KEY can be provided within the request header, body, or as a query parameter.
- Bearer Token Authentication: The bearer token must be included in the Authorization header.
- OAuth2 Authentication: Following successful user authentication with the API's authentication server, the resulting access token must be supplied in the Authorization header of the request.

Business Logic:

The request should include the necessary authentication configuration to verify the identity and authorization scopes in the SCDB.

Dependencies:

There is no SUNRISE DF related dependency defined for this endpoint.

Response Requirements

Expected Outputs

The response of the endpoint may be a JSON message containing the following fields:

- Authorization: flag to indicate if the user is authorized to make queries to the SCDB.
- Access scope: list of access levels to the SCDB content

The schema and examples of the responses are included in the complementary files attached with these requirements specifications.

Status Codes

- **200:** OK – Authorization is correct
- **400:** Bad Request – Request is not formed correctly
- **401:** Unauthorized – Request does not contain proper authorization
- **404:** Not Found – Requested endpoint does not exist
- **429:** Too Many Requests – Request rate is higher than SCDB can handle
- **503:** Service Unavailable – Service / Endpoint is temporarily unavailable

Error Handling

Error responses should include details about the issue

Non-Functional Requirements

Performance

There is no performance requirement defined for this endpoint.

Security (authentication)

Authentication will be inherited from SUNRISE DF's authentication layer. Depending on the individual SCDB, API key or token can be used.

Usability

- Error messages must be descriptive and provide actionable insights to users or developers.
- JSON responses should adhere to a clear and consistent schema, making it easier for frontend and backend developers to consume and debug the data.

Reliability and Availability

There is no specific reliability requirement defined for this endpoint.

For the availability, it is expected that the corresponding endpoint to be available at all times in individual SCDB side.

Dashboard Info - /getScdblInfo

Endpoint Objective

This endpoint provides information for the Dashboard component by querying the selected SCDB. It enables users to retrieve database details, visualize scenarios, and analyze key metrics through an interactive and customizable dashboard.

Use Cases

- Users select an SCDB and initiate a request through the Dashboard UI (frontend). The backend processes the selection, retrieves SCDB information, and returns it in a structured JSON format. Information returned includes metadata, scenario breakdowns, statistics, taxonomy and quality metrics.
- Users can add interesting graphs to the summary page and download it as PDF

HTTP Method:

HTTP POST

Headers

Header	Description
Content-Type	application/json
Accept	application/json
X-API-Key	Optional API key for authentication
Authorization	Optional OAuth2 Bearer token

Expected Behavior

Input Validation:

Requests must include a valid SCDBID parameter. Input is validated against the API schema.

Business Logic:

SCDBID is used to query the relevant SCDB.

SCDB should provide detailed database information, including metadata, SCDB quality metrics, statistics and taxonomy details.

Dependencies:

- Responses rely on the structure and availability of data in individual SCDBs.

Response Requirements

Expected Outputs

The endpoint returns a JSON payload containing:

- SCDB metadata (e.g. name, owner, version, and license expiration).
- Health status
- SCDBs details, including parameters statistics, entities, odd, quality metrics etc. (Scenarios list if it's required in the taxonomy).

In case of an unavailable field in the SCDB the field should be null in the JSON.

Status Codes

- **200:** OK - Dashboard information successfully retrieved.

- **400:** Bad Request – Request is not formed correctly
- **401:** Unauthorized – Request does not contain proper authorization
- **404:** Not Found – Requested endpoint does not exist
- **429:** Too Many Requests – Request rate is higher than SCDB can handle
- **503:** Service Unavailable – Service / Endpoint is temporarily unavailable

Error Handling

Error responses should include details about the issue

Non-Functional Requirements

Performance

There is no performance requirement defined for this endpoint.

Security (authentication)

Authentication will be inherited from SUNRISE DF's authentication layer. Depending on the individual SCDB, API key or token can be used.

Usability

- Error messages must be descriptive and provide actionable insights to users or developers.
- JSON responses should adhere to a clear and consistent schema, making it easier for frontend and backend developers to consume and debug the data.

Scalability

SCDBs with higher data volumes (e.g., millions of scenarios) should not degrade overall query performance.

Query Scenarios - /getScenarioList

Endpoint Objective

This endpoint is used for querying scenario database (SCDB) by using ODD / Requirement criteria defined by the user in SUNRISE DF UI.

It is expected individual SCDBs to consume content provided in this request call, perform search in the database and provide information in defined output format.

Use Cases

- The user creates a query by using Search UI (Frontend component) and executes. Query Manager receives query selection from user via frontend-backend communication. Query Manager converts data framework internal data to OpenLabel

format. Generated OpenLabel content is sent to databases that are onboarded to SUNRISE DF.

HTTP Method

HTTP POST

Headers

Header	Description
Content-Type	application/json
Accept	application/json
X-API-Key	Optional API key for authentication
Authorization	Optional OAuth2 Bearer token

Expected Behavior

Input Validation:

Query Manager will have a validator to ensure that the generated OpenLabel content is valid against its schema. The same validation can also be applied by SCDBs while consuming request.

Business Logic:

- The query provided by Query Manager will contain set of taxonomy tags with their included / excluded subset. SCDBs must return scenarios that included tags are linked and excluded tags are not linked.
- The scenario might have more ODD element links than it is requested in the query. Scenario must be considered as suitable if it includes or excludes the tags in the query.

Dependencies

There is no SUNRISE DF related dependency defined for this endpoint.

Response Requirements

Expected outputs

The Query Manager expects a custom JSON content to be provided by SCDBs. This model contains scenario meta data, parameter info, etc. The list of content and its data model is presented in the appendix section with examples.

It is expected that SCDB might not have all the information required by SUNRISE DF. Each field in the data model is marked as required or optional. SCDBs can skip the optional information if unavailable.

Status Codes

- **200:** OK - The list of scenarios returned
- **400:** Bad Request – Request is not formed correctly
- **401:** Unauthorized – Request does not contain proper authorization
- **404:** Not Found – Requested endpoint does not exist
- **429:** Too Many Requests – Request rate is higher than SCDB can handle
- **503:** Service Unavailable – Service / Endpoint is temporarily unavailable

Error Handling

Error responses should include details about the issue.

Non-Functional Requirements

Performance

There is no performance requirement defined for this endpoint.

Security (authentication)

Authentication will be inherited from SUNRISE DF's authentication layer. Depending on the individual SCDB, API key or token can be used.

Reliability and Availability

There is no specific reliability requirement defined for this endpoint.

For the availability, it is expected that the corresponding endpoint to be available at all times in individual SCDB side.

Download Scenarios - /getScenarioPackage

Endpoint Objective

This endpoint is used for downloading scenarios as OpenSCENARIO packages.

It is expected individual SCDBs to use scenario related info in the request and provide scenario packages accordingly.

Use Cases

- **Concrete Scenario Download:** The user selects scenario and clicks on download scenario on SUNRISE DF UI. SUNRISE DF creates download request (with Scenario ID, package content, scenario type and parameter values) and calls relevant SCDB API. SCDB consumes this request and provides a scenario package where only one OpenSCENARIO file is present.
- **Logical Scenario Download:** The user selects scenario and clicks on download scenario on SUNRISE DF UI. SUNRISE DF creates download request (with Scenario ID, package content, scenario type and parameter values) and calls relevant SCDB API. SCDB consumes this request and provides a scenario package where a logical OpenSCENARIO file and its distribution file are present.

HTTP Method

HTTP POST

Headers

Header	Description
Content-Type	application/json
Accept	application/json
X-API-Key	Optional API key for authentication
Authorization	Optional OAuth2 Bearer token

Expected Behavior

Input Validation:

A custom request form will be created for download scenarios endpoint. Its schema will be shared in appendix. SCDBs can perform their own validation against this schema.

Business Logic:

A download request MAY contain one or more scenarios.

SCDB endpoints are expected to accept a JSON array of scenario requests in a single API call, enabling batch downloads. Each scenario is identified by its ScenarioId and associated metadata. This reduces the number of API calls required for multi-scenario downloads.

Note: In the current SUNRISE implementation (July 2025), the backend sends one scenario id per request to each SCDB host, processing multiple scenarios in a loop on the backend. However, the request schema and endpoint design already support request for multiple scenarios. Future SUNRISE versions may switch to sending the requests containing more than one scenario. SCDB hosts must ensure they support receiving and processing arrays of scenario requests as defined in the schema.

Dependencies:

- The request content of this endpoint will contain scenario IDs from the query results.

Response Requirements

Expected Outputs

The download scenario expects an OpenSCENARIO (OSC) content package to be provided by SCDBs. This package may contain the files defined in Table 1.

Table 1: OpenSCENARIO package content

File Name / Type	Description	Required / By request
OpenSCENARIO [.xosc]	Base OpenSCENARIO file	Required
OpenSCENARIO Distribution File [.xosc]	OpenSCENARIO file that contains parameter distributions	By request
OpenDRIVE [.xodr]	OpenDRIVE file linked with OSC	Required
Scene Graph File [.osgb, .opt.osgb, .ive, .fbx, etc.]	Scene Graph file linked with ODR	By request

Required files must always exist in the packages where “**by request**” ones must be provided when requested in the call.

The SCDB returns the scenario package as a binary ZIP archive. To support JSON-based data exchange and compatibility with web protocols, the ZIP file is encoded as a Base64 string in response. The frontend is responsible for decoding the Base64 data back into binary form to reconstruct the ZIP file for download. This mechanism allows users to download multiple scenario packages in one operation, even though each package is retrieved separately from the SCDB.

Status Codes

- **200:** OK - The scenario package is returned
- **400:** Bad Request – Request is not formed correctly
- **401:** Unauthorized – Request does not contain proper authorization
- **404:** Not Found – Requested scenario does not exist
- **429:** Too Many Requests – Request rate is higher than SCDB can handle
- **503:** Service Unavailable – Service / Endpoint is temporarily unavailable

Error Handling

Error responses should include details about the issue

Non-Functional Requirements

Performance

There is no performance requirement defined for this endpoint.

Security (authentication)

Authentication will be inherited from SUNRISE DF's authentication layer. Depending on the individual SCDB, API key or token can be used.

Reliability and Availability

There is no specific reliability requirement defined for this endpoint.

For the availability, it is expected that the corresponding endpoint to always be available on the individual SCDB side.

ANNEX 3. DATA MODELS FOR SUNRISE DF TABLES

This annex provides a detailed explanation of the data model tables utilized within the SUNRISE AWS environment in the DynamoDB service. It includes comprehensive descriptions of each attribute, outlining their type and (sub)structure, to support a thorough understanding of the system's data architecture.

SCDB – Data Model

Table 2 : Data model for SCDB table

Attribute	Type	Sub-structure	Description
Name	string		The unique name of the Scenario Database.
description	string		A brief summary of the SCDB's purpose or content.
securitySchema	json		JSON object defining the security and authentication method (e.g., OAuth2, API Key).
serverUrl	string		The base URL for the SCDB's server.
apiEndpoint	ApiEndpoints	healthCheck: json validateUserConnection: json getScenarioList: json getScenarioPackage: json getSCDBInformation: json	Defines the specific API endpoints for actions like health checks and data retrieval.
httpBearer	HTTPBearer	getToken: json	Configuration for obtaining an HTTP Bearer token.

oauth2Params	OAuth2Variables	clientId: string tokenEndpoint: string domainUrl: string refreshToken: string	Stores the specific parameters required for the OAuth2 authorization flow.
userSCDBConnections	hasMany	Relation to UserSCDBConnection	A link to all user connections associated with this SCDB.

UserSCDBConnection – Data Model

Table 3: Data model for UserSCDBConnection table

Attribute	Type	Sub-structure	Description
oauth2Token	string		The user's OAuth2 access token for the SCDB.
oauth2Params	customType	refreshToken: string scope: string type: string expiresIn: integer createdAt: string	Stores details about the OAuth2 token, such as its scope, type, and expiration.
apiKey	string		The user's API key for authentication, if used by the SCDB.
bearerToken	string		The user's bearer token for authentication, if used by the SCDB.
bearerTokenExpiration	string		The expiration timestamp for the bearer token.

owner	string		The unique ID of the user who owns this connection record.
scdbId	id		The foreign key ID that links this record to a specific SCDB.
scdb	belongsTo	Relation to SCDB	Establishes a relationship from this connection back to the parent SCDB.

QueryManagerQuery – Data Model

Table 4: Data model for QueryManagerQuery table

Attribute	Type	Description
ModelId	string	The unique identifier for the query, which serves as the primary key (e.g., QUE-123456).
Name	string	A user-friendly name for the query.
ScenarioDatabase	string[]	An array of SCDB names that this query will target.
Odd	json	A JSON object that defines the Operational Design Domain (ODD) for the query.
Requirement	json	A JSON object that defines the specific requirements the query must satisfy.
CreateDate	string	The timestamp marking when the query was first created.
ModifyDate	string	The timestamp marking the last time the query was modified.
owner	string	The unique ID of the user who created and owns this query.

UserNotification – Data Model

Table 5 : Data model for UserNotification table

Attribute	Type	Description
userId	id	The unique ID of the user who should receive the notification.
type	string	The category of the notification (e.g., 'alert', 'info', 'update').
message	string	The actual content or body of the notification message.
isRead	boolean	A flag (true or false) to track if the user has read the notification. Defaults to false.
owner	string	The unique ID of the user who owns this notification record.