OpenArchitectureModel Specification

Release 0.1.0 (Draft 2025-02-8)

Sheikh Mohammad Sajid

Sheikh Mohammad sajid (editor)

Feb 08, 2025

Contents

CHAPTER 1

Introduction

1.1 Introduction

OpenArchitectureModel is a *language for defining and querying software architecture models*. Its main goal is to enable developers to define architecture elements in a standard way.

OpenArchitectureModel is an open standard developed by *Sheikh Mohammad Sajid*.

This document describes version 0.1.0 (Draft 2025-02-8) of the core OpenArchitectureModel standard. It is intended that it will be superseded by new incremental releases with additional features in the future.

1.1.1 Design Goals

The design goals of OpenArchitectureModel are the following:

- Fast, safe, and portable *semantics*:
  - **Well-defined**: fully and precisely defines the syntax.
  - **Language-independent**: does not privilege any particular language, programming model, or object model.
  - **Open**: programs can interoperate with their environment in a simple and universal manner.
- Efficient and portable *representation*:
  - **Modular**: programs can be split up in smaller parts that can be transmitted, cached, and consumed separately.
  - **Portable**: makes no architectural assumptions that are not broadly supported across modern hardware.

1.1.2 Scope

At its core, OpenArchitectureModel is a model for storing and retrieving data about software elements. To encompass their variety and enable maximum reuse, the OpenArchitectureModel specification is split and layered into several documents.

This document is concerned with the core layer of OpenArchitectureModel. It defines the base model, query syntax, validation, and execution semantics. It does not, however, define how OpenArchitectureModel data are stored within a specific environment they execute in, nor how they are invoked from such an environment.

1.2 Overview

1.2.1 Concepts

Open Architecture Model (OAM) introduces a high-level, SQL-like programming language. This language is structured around the following concepts:

Organization An organization operates as a single tenant, maintaining its own set of architectural information that is versioned collectively. Transactions are restricted to the scope of one organization and cannot extend across multiple organizations. User access and permissions are likewise confined to a single organization, ensuring isolated data governance and security at the tenant level.

Space Spaces are functional areas within an organization, designed to group and manage related information specific to teams, departments, groups, or sections. These spaces follow a hierarchical structure, supporting multiple sub-levels to reflect organizational complexity. The hierarchical model ensures that information is organized according to business needs and roles, promoting streamlined workflows and role-based access management.

Application An application is a distinct software component or module that resides within a single Space. Each application is associated exclusively with one Space, allowing it to inherit the permissions, hierarchy, and structural organization of that Space. Applications serve as functional units that provide specific capabilities or services within the organization's system, aligned with the scope and access controls of the Space to which they belong. This ensures that applications operate within defined organizational boundaries, facilitating modular deployment, isolated data handling, and efficient role-based access management.

Framework A framework is an architectural structure that establishes how an application is represented, organized, and managed within the system. It provides a set of conventions, principles, and practices to ensure consistency and clarity in architectural design. A framework can follow standard models such as TOGAF, C4, or Zachman, offering predefined methodologies to address various organizational or technical needs. Alternatively, it may be a custom framework tailored to meet specific requirements or workflows within the system, ensuring flexibility and adaptability for unique use cases.

Archiment An architeture element or *archiment* is the fundamental building block of an architectural structure, representing the smallest unit that cannot be further decomposed while retaining its functionality or purpose. Each element encapsulates a specific aspect of the architecture, such as a component, module, or service, and plays a distinct role within the overall system. Elements are designed to be self-contained and well-defined, enabling precise understanding, efficient communication, and easier management within the architectural framework. Their composition and interactions with other elements form the foundation of the system's design, behavior, and scalability.

Versioning The architecture model is stored as a versioned database, which maintains a complete history of changes to the model over time. This version control mechanism enables the retrieval of any previous version of the architecture, ensuring that historical configurations can be accessed and analyzed when needed. The versioned database supports traceability and auditability, allowing architects and developers to track changes, revert to earlier versions, and maintain consistency across different stages of the architecture's evolution. By managing versions efficiently, the system ensures flexibility, accountability, and continuity in architectural decision-making.

Architecture Query Language The Architecture Query Language (AQL) is a domain-specific language (DSL) designed for querying and manipulating the architecture model. It supports creation, updating, deletion, and retrieval of architectural data, providing a structured approach to model management. AQL is inspired by SQL, offering a familiar syntax that minimizes the learning curve for developers and ensures seamless adoption within existing workflows.

Extensions Extensions are modular add-ons to the system designed to enhance its functionality and adaptability. They achieve this by either extending the capabilities of existing models or introducing entirely new models to the architecture. Extensions allow for system customization without altering the core framework, enabling seamless integration of additional features or workflows. They are often developed to address specific requirements, provide scalability, or support future enhancements, ensuring that the system remains flexible and capable of evolving alongside organizational needs. Additionally, extensions can interact with the core components and other extensions through well-defined interfaces, maintaining consistency and reliability within the overall system.

CHAPTER 2

Architecture Query Language

2.1 Overview

Data Definition Create the structures that store your models.

Data Types Available logical types for declaring the type of various models.

AQL Keywords and Operators All of the language keywords and operators you can use in your AQL expressions.

Syntax and Lexical Structure Overview of the AQL language syntax.

Time Units and Formats Keywords that express time in your AQL expressions.

2.2 Data Definition

This section covers how you create the structures that store your architecture model.

2.3 Data Types

2.4 AQL Keywords and Operators

2.5 Lexical Structure

Index