

WINDTWIN

Towards a digital twin for forecasting of power production to wind energy demand

01/06/2024 – 31/05/2027

Call: HORIZON-CL5-2023-D3-02

Project 101147377 — WINDTWIN

D6.1 Specification of Digital Twin Core Components

Lead partner: Open Cascade

Authors: Andrey Smirnov, Open Cascade
Pedro Trindade, Open Cascade
Eduard Babkin, Open Cascade

Submission date: 29/11/2024

Dissemination level		
PU	Public, fully open	X
SEN	Sensitive, limited under the conditions of the Grant Agreement	

Document history

Issue date	Version	Changes made / Reason for this issue
18-11-2024	0.1	Initial version
21-11-2024	0.2.1	Review by FHG-IEE
25-11-2024	0.2.2	Review by BSC
28-11-2024	0.3	New version from Open Cascade based on previous Reviews.
29-11-2024	1.0	Final Version

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

TABLE OF CONTENTS

TABLE OF CONTENTS 2

TABLE OF FIGURES..... 4

TABLE OF TABLES..... 4

LIST OF ACRONYMS 4

EXECUTIVE SUMMARY..... 5

1. OVERVIEW 6

2. OVERALL ARCHITECTURE 6

3. COMPONENTS..... 7

3.1. OPEN CASCADE PLATFORM CORE 7

3.2. DTWIN CORE..... 8

3.3. OPEN CASCADE PLATFORM AUTHENTICATION AND AUTHORIZATION SUBSYSTEM 8

3.3.1. AUTHENTICATION PROXY 8

3.3.2. OPEN CASCADE PLATFORM AUTHENTICATION SERVICE..... 9

- 3.4. OPEN CASCADE PLATFORM FILE MANAGER SERVICE10
- 3.5. GIS SERVICE10
- 3.6. TIME SERIES DATA MANAGEMENT11
- 3.7. SERVICE MESH12
- 3.8. OTHER WINDTWIN COMPONENTS CREATED BY PARTNERS12

- 4. DATA FLOWS13

- 5. MESSAGING13

- 6. DATA STORAGE14

- 7. USER INTERFACE15

- 8. USER MANAGEMENT AND AUTHENTICATION.....16

- 9. DEPLOYMENT.....16

- 10. SECURITY CONSIDERATIONS17
- 10.1. API SERVER SECURITY17
- 10.2. POD SECURITY17
- 10.3. NODE SECURITY17
- 10.4. NETWORK SECURITY.....17
- 10.5. SECRET MANAGEMENT18
- 10.6. MONITORING AND LOGGING18

- 11. PERFORMANCE ASPECTS AND SCALABILITY18

- 12. CONCLUSION.....19

- 13. REFERENCES19

TABLE OF FIGURES

Figure 1 - The overall system architecture of the WinDTwin project 6

TABLE OF TABLES

No table of figures entries found.

List Of Acronyms

AWS Amazon Web Services..... 19

GIS Geographic Information System10, 11, 12

JWT JSON Web Token.....9, 17

KEDA Kubernetes Event-Driven Autoscaling 18

LOD Level Of Details 7

mTLS mutual Transport Layer Security..... 12

NoSQL Not only SQL 18

OS Operating System..... 17

POI Point Of Interest..... 10

RBAC Role-Based Access Control..... 17

REST Representational State Transfer7, 8, 16, 17

SSO Single Sign-On..... 9

TLS Transport Layer Security 17

TSD Time Series Data8, 11

UI User Interface.....15, 16

EXECUTIVE SUMMARY

This document serves as a formal deliverable of the WinDTwin project, detailing the comprehensive project architecture and methodologies employed. The WinDTwin project aims to develop and validate an offshore wind farm digital twin that offers highly accurate predictions of power production and energy demand for end-users. The document begins with an overview of the project's objectives, emphasizing the importance of expanding onshore and offshore wind farms to meet the growing global demand for renewable energy. The overview also addresses the challenges associated with integrating wind farms into larger clusters and virtual power plants, highlighting the project's goal of mitigating long-distance wake effects to maintain efficiency gains.

This deliverable formally documents the overall system architecture, which is designed as a classic web-based client-server setup. The client-side development leverages Angular, a robust framework for building dynamic web applications, while the backend is hosted in a Kubernetes cluster, ensuring scalability and resilience. The architecture is modular, allowing independent development and deployment of each component within Docker containers. Each container exposes a REST API for seamless communication with the client side and inter-cluster interactions, with API specifications provided in OpenAPI format. This approach ensures clear and consistent communication protocols across the system, supporting a "Design first" methodology for API specification drafts.

Furthermore, this document outlines the service provider agnostic design of the system, which can be deployed on any major public cloud service provider or within a private cloud environment. This flexibility ensures that the system can be tailored to meet the specific needs and constraints of different deployment scenarios. Detailed sections are provided on the technical aspects, development teams' responsibilities, and the expected deliverables considering the architecture, thus providing a comprehensive guide for stakeholders involved in the project.

1. OVERVIEW

The WinDTwin project targets to develop and validate an offshore wind farm digital twin for highly accurate prediction of power production and energy demand of the end user. The digital twin will give users tailored access to high-quality information, services, models, scenarios, forecasts, and visualizations, as a central hub for offshore wind decision-makers, seeking to revolutionize the way industry professionals make informed choices.

As the global demand for renewable energy intensifies, expanding both onshore and offshore wind farms becomes crucial. Existing wind farms highlight the importance of strategic micro-siting of turbines and their optimal interconnection. However, as we advance towards integrating wind farms into larger clusters and virtual power plants, the risk of long-distance wake effects becomes evident, potentially compromising the efficiency gains that such clusters are designed to achieve.

To address these challenges, WinDTwin will develop a sophisticated digital twin platform. This platform aims to transform the industry by offering precise predictions of power production and energy demand. It will serve as a central hub for decision-makers, offering access to a suite of high-quality resources, models, scenarios, and visualizations, thus enabling more informed and strategic choices in the management of offshore wind energy.

2. OVERALL ARCHITECTURE

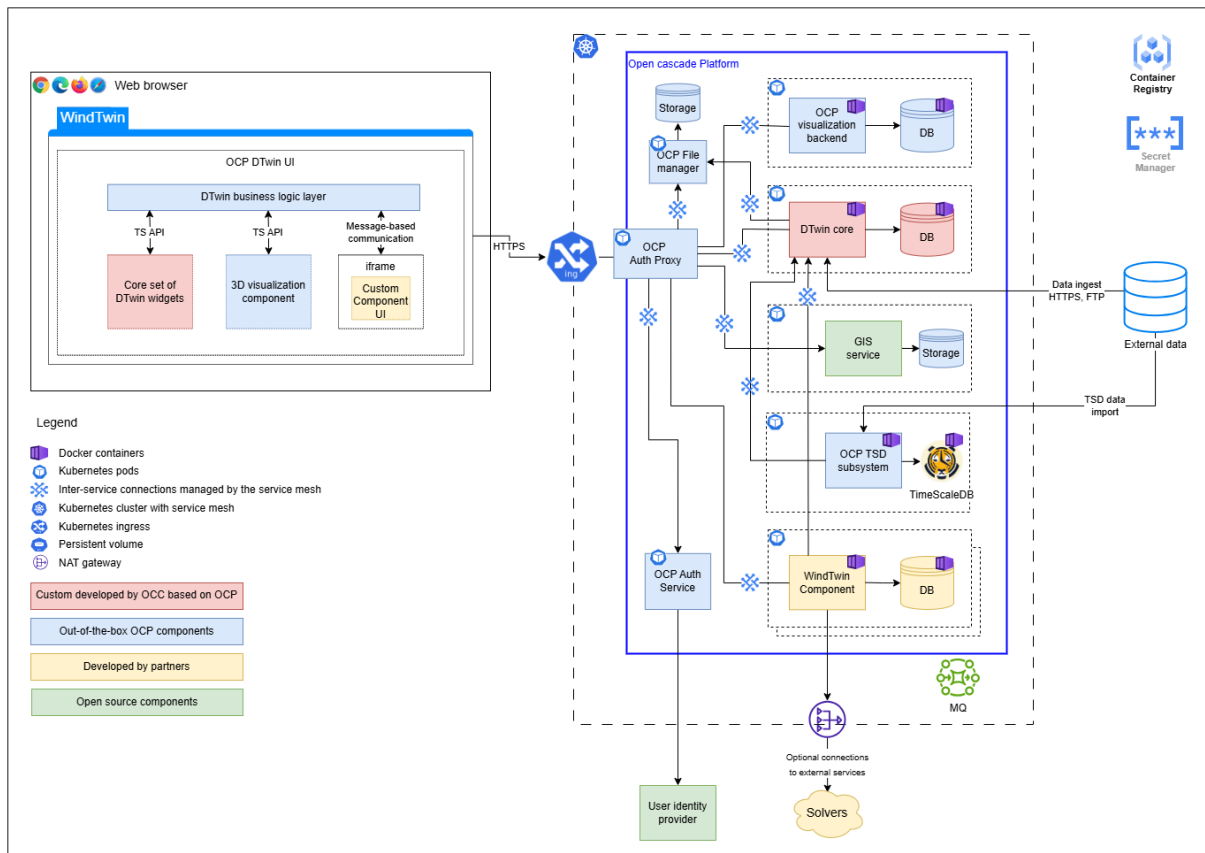


Figure 1 - The overall system architecture of the WinDTwin project

The overall system architecture of the WinDTwin project is designed as a classic web-based client-server setup. The client part of the system runs in a web browser, providing users with an intuitive and accessible interface. The preferred technology for client-side development is Angular, a popular framework known for its robustness and flexibility in building dynamic web applications.

On the backend, the system is hosted in a Kubernetes cluster, which offers a scalable and resilient environment for deploying containerized applications. The business logic running on the backend is divided into multiple modules, each encapsulated within its own Docker container. This modular approach allows for independent development and deployment of each module, enabling dedicated development teams to work on specific components without interfering with each other.

Each development team has the freedom to choose the programming language, tools, and libraries that best suit their needs for developing their respective modules. Once a module is developed, it is delivered to Open Cascade as a Docker container. Each container is expected to expose a REST API for communication with the client side and for inter-cluster interactions. Along with the container, each team provides an API specification in OpenAPI format, ensuring clear and consistent communication protocols across the system. Each development team is expected to follow the “Design first” approach and provide draft of its API specification before final delivery of container image.

The proposed system is designed to be service provider agnostic, meaning it can be deployed on any major public cloud service provider or within a private cloud environment. This flexibility ensures that the system can be tailored to meet the specific needs and constraints of different deployment scenarios.

3. COMPONENTS

3.1. Open Cascade Platform core

The Open Cascade Platform Core is a pivotal component responsible for the ingestion, processing, and visualization of three-dimensional (3D) models. Its architecture is designed to handle 3D models of varying scales, from minuscule to exceedingly large, by leveraging a computational cluster for efficient processing and a web-based 3D viewer for visualization.

The computational cluster within the Open Cascade Platform Core is engineered for horizontal scalability, allowing it to dynamically adjust its capacity based on the current computational load. This scalability is crucial for managing diverse computational tasks, which include both built-in and custom tasks. Each task submitted to the cluster is decomposed into multiple steps and processed in parallel using the map-reduce algorithm. This parallel processing capability ensures optimal performance and efficiency. Custom tasks can be developed in any programming language and encapsulated within Docker containers, providing flexibility and ease of integration.

The 3D viewer component of the Open Cascade Platform Core is built on WebGL technology, enabling it to visualize 3D models of any size, including extremely large ones. This capability is achieved through a preprocessing step that creates a hierarchical structure of levels of detail (LOD). During runtime, the appropriate level of detail is selected for visualization based on the camera's position relative to the

model and the viewing distance. This approach ensures that the visualization remains performant and responsive, even for complex and large-scale models.

3.2. DTwin core

The DTwin Core component is an extension of the Open Cascade Platform, specifically tailored to provide project-specific functionalities that are not available in the standard distribution of the platform. This component is designed to enhance the platform's capabilities by introducing specialized APIs and data management solutions that cater to the unique requirements of the WinDTwin project.

The Open Cascade Platform is built on the principle of extensibility. The platform core offers a foundational set of functionalities, including user and file management, a Time Series Data (TSD) subsystem, and other essential services. The DTwin Core component extends these foundational capabilities to address the specific needs of the WinDTwin project.

The complete set of key features provided by DTwin Core component will be defined in the later stages of the project, but a few of them can be already named. For example, DTwin Core component will provide the ability to manage wind farm entities. This includes the representation of wind turbines, their types and properties, spatial configurations, and operational parameters. DTwin Core component will provide a comprehensive set of APIs for interacting with these entities, enabling users to perform tasks such as adding new turbines, updating their configurations, retrieving operational data, etc.

In addition to managing wind farm entities, the DTwin Core component will be responsible for handling calculation requests. These requests are sent to specialized solvers that perform complex computations related to wind farm operations. The component manages the entire lifecycle of these requests, from submission to execution and retrieval of results.

The DTwin Core component will also play a crucial role in orchestrating user workflows. It will provide a set of tools and interfaces that enable users to define and execute workflows related to wind farm management. These workflows can include tasks such as data ingestion, validation, preprocessing, analysis, and visualization.

3.3. Open Cascade Platform Authentication and Authorization subsystem

3.3.1. Authentication proxy

The Authentication Proxy component of the Open Cascade Platform functions as a sophisticated application gateway, designed to filter, authenticate, and authorize all incoming REST requests directed towards the platform. This component plays a crucial role in ensuring the security and integrity of the platform by acting as the first line of defense against unauthorized access.

At its core, the Authentication Proxy leverages the Authentication Service to retrieve comprehensive information about users, their roles, permissions, and group memberships. This integration allows the proxy to make informed decisions about whether to grant or deny access to specific resources based on the user's credentials and assigned permissions. See Figure 1 for details on how Authentication Proxy and Authentication Service are connected and receive incoming request flow.

The Authentication Proxy operates by intercepting all incoming requests and performing a series of validation checks. Initially, it verifies the authenticity of the request by validating the provided credentials in the form of JWT token against the user database maintained by the Authentication Service.

Once the authentication is successfully validated, the proxy proceeds to the authorization phase. During this phase, it evaluates the user's permissions and roles to determine if they have the necessary access rights to perform the requested action.

The Authentication Proxy is designed to be highly scalable and resilient, capable of handling a large volume of requests without compromising performance. It achieves this through efficient load balancing and fault-tolerant mechanisms that ensure continuous availability and reliability of the authentication and authorization services.

3.3.2. Open Cascade Platform Authentication service

The Open Cascade Platform Authentication Service is a fundamental component designed to ensure secure and efficient user management within the platform. This service is responsible for maintaining a comprehensive user database and managing permission assignments, which are critical for controlling access to various platform resources.

At the core of the Authentication Service is the user database, which stores detailed information about each user, including their credentials, roles, and permissions. Permissions are the basic authorization elements that define what actions a user is allowed to perform within the platform. These permissions can be assigned directly to individual users or grouped into roles, which can then be assigned to users. This hierarchical structure allows for flexible and granular access control, ensuring that users have the appropriate level of access based on their responsibilities and needs.

In addition to managing individual users, the Authentication Service also supports the creation and management of user groups. User groups are collections of users that can be assigned roles and permissions collectively, simplifying the administration of access control. This feature is particularly useful in large organizations where users with similar responsibilities need to have the same access rights.

A unique aspect of the Authentication Service is its ability to maintain a hierarchy of management groups. A management group is a virtual container that can hold one or more projects, forming a tree-like structure. This hierarchical organization allows for efficient management of access rights at different levels, enabling administrators to grant or restrict access to sets of projects based on the user's role within the management group hierarchy.

The Authentication Service also supports various authentication mechanisms, including Single Sign-On (SSO) protocols. This feature allows users to authenticate once and gain access to multiple applications within their IT infrastructure without needing to re-enter their credentials. The service supports widely-used SSO protocols like OpenID Connect, ensuring compatibility with external identity providers and enhancing the overall security and user experience.

See Figure 1 on details how Open cascade Platform Authentication service connects with external identity providers.

Furthermore, the Authentication Service is designed to be highly scalable and resilient. It can handle a large number of authentication and authorization requests without compromising performance. This scalability is achieved through efficient load balancing and fault-tolerant mechanisms, ensuring continuous availability and reliability of the authentication services.

3.4. Open Cascade Platform File manager service

The Open Cascade Platform File Manager Service is an integral component designed to facilitate the efficient management of files within the platform. This service is responsible for the storage, organization, and sharing of files, ensuring that users can seamlessly access and manage their data.

At its core, the File Manager Service maintains an independent storage system for each user, allowing for the organization of files into a hierarchical structure of folders. This structure not only enhances the ease of file management but also supports the logical grouping of files for further processing. The service provides robust interfaces for uploading and downloading files from external sources, such as third-party document management systems and file storage solutions. This capability ensures that users can integrate and manage their files from various sources within a unified platform.

A key feature of the File Manager Service is its ability to facilitate file sharing between users and user groups. This functionality is crucial for collaborative environments where multiple users need to access and work on the same set of files. The service manages the permissions associated with each file, ensuring that only authorized users can access or modify the data. This permission management is critical for maintaining the security and integrity of the files stored within the platform.

The physical storage of files is handled by a clustered file system, which can dynamically scale its capacity based on current needs. This scalability is essential for accommodating varying storage requirements and ensuring optimal performance. The Open Cascade Platform supports multiple choices of file systems, allowing for deployment in both on-premises environments and public clouds. For instance, the platform can utilize open-source file systems like Ceph for on-premises deployments, while leveraging cloud-native file storage services such as AWS EFS or Azure File Service for cloud-based deployments.

In addition to its core functionalities, the File Manager Service also supports advanced features such as version control and metadata management. Version control allows users to track changes to files over time, providing a history of modifications and the ability to revert to previous versions if necessary. Metadata management enables the association of additional information with files, facilitating more efficient search and retrieval of data.

3.5. GIS service

The Open Cascade Platform GIS Service is a component designed to provide comprehensive map visualization capabilities and visualize Points Of Interest (POIs) on these maps. An example of this functionality can be visualization of a map of coastal area and, on top of it, set of icons, representing locations of wind turbines. Turbine icons can be color-coded to represent their power output or

construction stage or other things. This service is integral to the platform's ability to represent spatial data accurately and interactively, thereby enhancing the decision-making processes that rely on geographic information.

At the core of the GIS Service is its sophisticated map visualization engine, which supports various types of geographic data and map layers. This engine is capable of rendering high-resolution maps with multiple layers of information, including topographic details, infrastructure layouts, and environmental data. The visualization engine employs advanced techniques to ensure that maps are rendered efficiently and accurately, even when dealing with large datasets or complex geographic features.

3.6. Time Series Data management

The Open Cascade Platform provides a comprehensive framework for the storage, processing, and visualization of time series data (TSD), which is essential for applications that require the analysis of temporal data patterns. Time Series Data Management service is designed to handle time series data received from external suppliers, either as a continuous stream of samples at runtime or through bulk ingestion.

The architecture of the Time Series Data Management service is built to support high-throughput data ingestion and efficient storage. Time series data is stored in a specialized database optimized for handling large volumes of sequential data. This database supports efficient querying and retrieval of time series data, enabling users to perform complex analyses and derive insights from the temporal patterns present in the data.

Opensource database TimescaleDB is used by Open Cascade Platform to store TSD data. TimescaleDB is a powerful, time-series database built on PostgreSQL, offering scalability, performance, and flexibility for handling time-series data. It excels in managing large datasets through features like automatic data partitioning, compression, and efficient query execution. TimescaleDB supports real-time analytics, continuous aggregates, and policy-based data retention. Its open-source foundation and enterprise-grade features ensure a balance of accessibility and reliability for diverse use cases.

A key feature of the Time Series Data Management service is its ability to visualize time series data in various formats. The service can supply data sources for creation of dashboards that allow users to create and customize charts for visualizing time series data. These charts can display data in real-time, providing users with up-to-date information on the monitored parameters. Additionally, time series data can be integrated into 3D models, allowing for the visualization of temporal data in a spatial context. This integration is particularly useful for applications that require the monitoring of dynamic processes over time.

The service also supports the logical attachment of data sources to elements of 3D models. For example, in an industrial application, time series data from sensors can be linked to the corresponding equipment within a 3D model, enabling users to monitor the performance and condition of the equipment in real-time.

The Time Series Data Management service is designed to be highly scalable, capable of handling large volumes of data from multiple sources simultaneously. This scalability is achieved through the use of distributed computing techniques, which allow the service to distribute the processing load across

multiple nodes in a computational cluster. This approach ensures that the service can handle the demands of large-scale applications without compromising performance.

3.7. Service mesh

The Service Mesh is an optional component of the Open Cascade Platform, designed to enhance the security, observability, reliability, and resilience of inter-component communications. This sophisticated infrastructure layer is responsible for managing the network of microservices that constitute the platform, ensuring seamless and secure interactions between them.

At its core, the Service Mesh provides a robust framework for security management. It employs mutual Transport Layer Security (mTLS) to encrypt communications between services, thereby safeguarding data in transit from potential interception and tampering. This encryption mechanism ensures that all inter-service communications are authenticated and authorized, significantly reducing the risk of unauthorized access and data breaches.

Observability is another critical aspect facilitated by the Service Mesh. It offers comprehensive monitoring and logging capabilities, enabling detailed insights into the behavior and performance of microservices. By collecting and analyzing metrics, logs, and traces, the Service Mesh allows for real-time visibility into the system's operations. This observability framework is essential for diagnosing issues, optimizing performance, and ensuring the overall health of the platform.

The reliability and resilience of the Open Cascade Platform are significantly bolstered by the Service Mesh. It incorporates advanced traffic management features, such as load balancing, circuit breaking, and retry policies, to ensure that services remain available and performant even under adverse conditions. Load balancing distributes incoming requests evenly across multiple service instances, preventing any single instance from becoming a bottleneck. Circuit breaking protects the system from cascading failures by isolating and managing faulty services, while retry policies ensure that transient errors are automatically handled without user intervention.

The Service Mesh also supports advanced routing capabilities, enabling sophisticated traffic routing and segmentation. This feature allows for the implementation of canary deployments, blue-green deployments, and A/B testing, facilitating the safe and controlled rollout of new features and updates. By directing a subset of traffic to new service versions, the platform can validate changes in a production environment without impacting the entire user base.

3.8. Other WinDTwin components created by partners

The Open Cascade Platform is architected with an open and extensible design, allowing for seamless integration of third-party components developed by partners. These components are typically provided as container images, which are capable of running within a Kubernetes cluster. This modular approach ensures that the platform can be continuously enhanced with new functionalities, leveraging the expertise and innovations of external collaborators.

Third-party components can fully utilize the comprehensive suite of services offered by the Open Cascade Platform. These services include user management, authentication and authorization, file management, background task execution, GIS capabilities, and 3D data visualization, among others. By

integrating with these services, third-party components can provide a cohesive and unified user experience, maintaining consistency across the platform.

The communication between third-party components and the native components of the Open Cascade Platform is facilitated through the Service Mesh. This infrastructure layer ensures secure, reliable, and efficient inter-component communication, leveraging mutual Transport Layer Security (mTLS) for encrypted data transmission. The Service Mesh also provides observability features, enabling detailed monitoring and logging of interactions between components, which is crucial for maintaining the platform's overall health and performance.

Partners developing components for the WinDTwin project are encouraged to adhere to the platform's standards and best practices. This includes providing API specifications in the OpenAPI format, which ensures clear and consistent communication protocols. Additionally, components should be designed with scalability and resilience in mind, leveraging the platform's capabilities for horizontal scaling and fault tolerance. Further details on this are to be provided in the scope of deliverable 6.2.

Partners are also required to follow “Design first” approach and develop comprehensive specifications for Application Programming Interfaces (APIs) and data objects of their components prior to the delivery of actual components intended for integration. These specifications must be well documented and disseminated to the entire project team for review and approval. This procedural requirement is pivotal in ensuring the seamless integration and functional cohesion of all components developed by the partners, thereby significantly contributing to success of the project.

4. DATA FLOWS

Data flows and data model specification will be delivered as part of Task 2.1 of Work Package 2. The proposed architecture allows for very flexible definition of data flow patterns in and out of the system.

5. MESSAGING

The Open Cascade Platform employs the Kafka message broker to facilitate asynchronous messaging within its cluster. This architectural choice is driven by Kafka's key features, which include scalability, high throughput, durability, low latency, and flexible parallel consumption. These attributes make Kafka an ideal solution for managing the complex communication needs of the platform's microservices.

Asynchronous messaging, as implemented through Kafka, allows for the decoupling of services. This decoupling is crucial for enhancing the platform's scalability and fault tolerance. By enabling services to communicate without waiting for immediate responses, the platform can handle a higher volume of requests and maintain performance even under heavy load. This approach also allows for more flexible and resilient system architecture, where individual services can be updated or scaled independently without disrupting the overall system.

Kafka's scalability is achieved through its distributed architecture, which allows it to handle large volumes of data by distributing the load across multiple brokers. Each broker is responsible for a subset of the data, ensuring that the system can scale horizontally by adding more brokers as needed. This

scalability is essential for the Open Cascade Platform, which must manage a diverse range of data and communication requirements from various components and services.

High throughput is another critical feature of Kafka, enabling it to process a large number of messages per second. This capability is vital for the platform's performance, ensuring that messages are delivered promptly and efficiently, even during peak usage periods. Kafka achieves high throughput through its efficient storage and retrieval mechanisms, which are optimized for sequential data access.

Durability is ensured by Kafka's ability to persist messages to disk, providing a reliable storage mechanism that guarantees message delivery even in the event of system failures. This durability is crucial for maintaining the integrity and consistency of the platform's data, ensuring that no messages are lost or corrupted during transmission.

Low latency is achieved through Kafka's efficient message handling and delivery mechanisms, which minimize the time it takes for messages to travel from producers to consumers. This low latency is essential for real-time applications and services within the platform, where timely data delivery is critical for performance and user experience.

Flexible parallel consumption is facilitated by Kafka's consumer group mechanism, which allows multiple consumers to read from the same topic in parallel. This parallelism enhances the platform's ability to process large volumes of data quickly and efficiently, ensuring that all services receive the information they need in a timely manner.

6. DATA STORAGE

The Open Cascade Platform offers a versatile and robust data storage framework, designed to accommodate the diverse and dynamic needs of projects built upon it. This framework encompasses multiple storage options, each tailored to specific types of data and usage scenarios, ensuring optimal performance, scalability, and reliability.

File storage within the Open Cascade Platform is implemented through a clustered shared file system, which is accessible to all components hosted under the platform. This storage system is designed to be highly scalable, with the capacity to increase or decrease dynamically at runtime based on the current demands of the project. The reliability of this storage type is ensured through redundancy mechanisms inherent in the file system. The physical implementation of the file system-based storage varies depending on the hosting environment. For on-premises deployments, the open-source Ceph file system is typically utilized, providing a scalable and resilient storage solution. In contrast, deployments on public cloud platforms leverage cloud-native file storage services, such as AWS Elastic File System (EFS) or Azure File Service, to ensure seamless integration and high availability.

In addition to file storage, the Open Cascade Platform incorporates a clustered non-relational database for managing large, schemaless datasets. MongoDB is employed to provide this service, offering a flexible and scalable solution for storing and querying vast amounts of unstructured data. MongoDB's distributed architecture allows it to handle high volumes of read and write operations, making it well-suited for applications that require rapid access to large datasets.

For smaller, more structured data sets, the platform utilizes relational databases integrated with individual components. These databases run within their respective pods, ensuring that each component has access to a dedicated and optimized storage solution. This approach allows for fine-grained control over data management and ensures that the performance of each component is not impacted by the storage needs of others.

The combination of these storage options provides a comprehensive and adaptable data storage framework, capable of meeting the diverse requirements of different projects and applications. By leveraging both file-based and database storage solutions, the Open Cascade Platform ensures that data is stored efficiently and can be accessed quickly and reliably, regardless of the size or complexity of the dataset.

7. USER INTERFACE

The WinDTwin User Interface (UI) is an Angular-based single-page web application designed to provide a seamless and intuitive user experience. This sophisticated interface leverages several advanced features to ensure efficient interaction and data visualization.

One of the core features of the WinDTwin UI is Dynamic Content Loading. This mechanism ensures that only specific parts of the page are updated during interactions, significantly reducing load times and enhancing the overall user experience. By minimizing the amount of data transferred and processed during each interaction, the platform achieves a high level of responsiveness and efficiency.

The Component-Based Architecture of the WinDTwin UI is another critical aspect. This architecture promotes modularity and reusability, allowing developers to create and maintain discrete components that can be independently developed, tested, and integrated. This modular approach not only streamlines the development process but also enhances the scalability and maintainability of the application.

Two-Way Data Binding is employed to synchronize data between the model and the view, ensuring that any changes in the underlying data are immediately reflected in the user interface. This real-time synchronization is crucial for applications that require up-to-date information and interactive data manipulation.

Routing capabilities within the WinDTwin UI enable seamless navigation between different sections of the application without the need for page reloads. This feature enhances the user experience by providing smooth and uninterrupted transitions, allowing users to access various functionalities and data views efficiently.

At the component level, the WinDTwin web application comprises several key elements. The main business logic components, provided by the Open Cascade Platform and customized for the WinDTwin project, form the backbone of the application.

The Core Set of Digital Twin UI Widgets, including tables, lists, and charts, provides users with powerful tools for data visualization and interaction. These widgets are designed to present complex data in an accessible and comprehensible manner, facilitating informed decision-making and analysis.

A WebAssembly-Based 3D Viewer is integrated into the WinDTwin UI, enabling the visualization of three-dimensional models directly within the web application. This viewer leverages the capabilities of WebGL to deliver high-performance 3D rendering, allowing users to interact with detailed and complex models in real-time.

Custom UI Components, provided by partners and running inside iframes, further extend the functionality of the WinDTwin UI. These components can be seamlessly integrated into the application, leveraging the platform's capabilities while maintaining a consistent user experience.

Communication between core components is facilitated by native Angular calls, ensuring efficient and reliable data exchange within the application. For interactions between partner-provided components and the main business logic engine, `postMessage` calls with predefined payloads are used. This approach ensures secure and structured communication, maintaining the integrity and performance of the application.

8. USER MANAGEMENT AND AUTHENTICATION

Open Cascade Platform offers centralized extended user management. User identity database can be internally hosted by Open Cascade Platform or connected externally through Single Sign-On protocol Open ID Connect. User identity and permission management-related features provided by Open Cascade Platform are as follows:

- Individual user accounts, internally managed by the platform or managed by external user identity provider
- User groups
- Atomic permissions attached to REST API endpoints provided by the Platform and assignable to users and user groups
- Default and custom user-created roles that can contain set of atomic permissions and can be assigned to individual users or user groups
- Management groups as containers for users and resources managed by the platform

9. DEPLOYMENT

The general deployment philosophy behind the Open Cascade Platform and applications developed on top of it is that it is cloud-agnostic. This means that the platform can be deployed on-premises as well as in any major public clouds. At the top level, the deployment of the Open Cascade Platform is structured as a Kubernetes cluster, with additional third-party services and data storages connected to it. When the platform is deployed on-premises, it utilizes third-party open-source components that can also be deployed within the same cluster. Conversely, when deployed on public clouds, it often makes sense to use cloud-native services for certain functionalities, such as data storage or database engines. The choice of the exact service type can be made per project.

The deployment process of the Open Cascade Platform is automated using Terraform and Helm. Terraform is employed for infrastructure provisioning, enabling the creation and management of cloud resources in a consistent and repeatable manner. Helm, on the other hand, is used for managing Kubernetes applications, allowing for the deployment, scaling, and management of containerized

applications within the cluster. This combination of Terraform and Helm ensures a streamlined and efficient deployment process, reducing the complexity and potential for errors.

The Kubernetes cluster, which forms the backbone of the platform's deployment, offers a scalable and resilient environment for deploying containerized applications. The cluster can dynamically adjust its capacity based on the current computational load, ensuring optimal performance and resource utilization. This scalability is crucial for managing diverse computational tasks, including both built-in and custom tasks, which can be developed in any programming language and encapsulated within Docker containers.

10. SECURITY CONSIDERATIONS

Given the distributed and dynamic nature of a Kubernetes-based environment such as Open Cascade Platform and applications developed on top of it, securing such deployment involves several layers and practices, addressing network, resource access, data, and infrastructure protection as well as development best practices.

10.1. API Server Security

Authentication and Authorization: Open Cascade Platform contains multiple services providing REST APIs. These services are the primary interface to the cluster and require strong authentication to prevent unauthorized access. Access to all Open Cascade Platform APIs is secured with JWT token-based user authentication. Furthermore, access to particular endpoints and data objects requires the user to be assigned appropriate permissions controlled by Role-Based Access Control (RBAC).

Transport Layer Security (TLS): All communications between external client applications and Open Cascade Platform services are encrypted using TLS to protect data in transit.

10.2. Pod Security

Open Cascade Platform only runs Kubernetes pods with disabled root access and privilege escalations.

As part of the development process, container images are always scanned for vulnerabilities before deployment and image policies are established to ensure only trusted images (from secure registries) are allowed to run on the cluster.

Container images are built on top of minimal base images.

10.3. Node Security

Operating System Hardening: Worker nodes have minimal OS footprints, with only essential software packages and services enabled. Worker nodes receive regular updates at the OS level.

10.4. Network Security

Cluster nodes are deployed in a secure, isolated network environment, with firewalls restricting external access to critical components like the API servers and etc.

Ingress Egress Controllers are setup to manage and restrict inbound and outbound traffic.

Optionally a service mesh is used for encrypted and authenticated inter-service communication.

10.5. Secret Management

Secret management tools are used to manage sensitive data outside the Kubernetes cluster. Hashi Corp Vault is used for on-prem deployments and provider-specific services are used for public cloud deployments.

10.6. Monitoring and Logging

Cluster Monitoring using Prometheus and Grafana is implemented to detect abnormal behavior within the cluster.

Log Aggregation (Grafana/Loki stack) is implemented to centralize and analyze logs across the cluster for potential security incidents.

11.PERFORMANCE ASPECTS AND SCALABILITY

Scalability is an intrinsic feature of the Open Cascade Platform core, designed to ensure optimal performance under varying loads. The platform comprises multiple services, each dedicated to performing specific functions. This modular architecture allows the platform to identify components experiencing high stress and scale them horizontally. Additionally, all third-party components, such as databases, file systems, and messaging systems, are inherently scalable.

The scalability of the cluster is governed by Kubernetes Event-Driven Autoscaling (KEDA), which allows for multiple scaling event sources and metrics, providing granular control over scaling events. This approach enables the selection of the optimal replica count for all pods at runtime, adapting to current usage patterns and minimizing resource costs.

Key components of the Open Cascade Platform and their scaling scenarios include:

1. **Backend Components:** These can be scaled out when under stress from user-issued requests as well as background tasks running on the cluster.
2. **Job Manager:** This component is scaled when there is a need to perform resource-intensive computations, such as preprocessing large 3D models or point clouds.
3. **File System:** When deployed on-premises, the Ceph file system is used. For cloud deployments, the platform typically relies on scalable file systems provided by cloud services. This subsystem is scaled out when there is a need to increase storage or better accommodate a large number of read operations.
4. **Database:** The platform uses MongoDB, a clustered and scalable NoSQL database, to store metadata. MongoDB can be scaled out to adapt to increased storage requirements and/or increased data transfer needs.
5. **Messaging:** The platform employs a clustered message broker, Kafka, to facilitate asynchronous messaging within the cluster.

The platform's architecture ensures that each component can be independently scaled, providing a flexible and resilient environment capable of handling diverse computational tasks. This scalability is crucial for maintaining optimal performance and resource utilization, ensuring that the platform can adapt to varying loads and demands efficiently.

12. CONCLUSION

This document serves as the definitive reference for the architecture of the WinDTwin project, encapsulating the key outcomes and methodologies for ensuring its scalability and performance. It provides a comprehensive overview of how each partner or work package component integrates into and completes the WinDTwin project's development.

For instance, a partner or work package developing a new component can rely on the guidelines and technical specifications outlined here. The document explains how to handle information flow, ensuring that the new component integrates with the existing infrastructure. By following the prescribed architectural paradigms and strategies, partners can contribute to and enhance the project's software platform while maintaining optimal performance and resource utilization.

In conclusion, this document not only outlines the architecture but also serves as a practical guide for partners to develop and integrate their components effectively, ensuring the continued success and scalability of the project's Digital Twin.

13. REFERENCES

1. Angular <https://angular.dev/>
2. Apache Kafka <https://kafka.apache.org/>
3. AWS <https://aws.amazon.com/>
4. Azure <https://azure.microsoft.com/>
5. Ceph <https://ceph.io/>
6. HashiCorp Terraform <https://www.terraform.io/>
7. HashiCorp Vault <https://www.vaultproject.io/>
8. Helm <https://helm.sh/>
9. Kubernetes <https://kubernetes.io/>
10. MongoDB <https://www.mongodb.com/products/self-managed/community-edition>
11. OpenAPI <https://swagger.io/specification/>
12. Open ID Connect <https://openid.net/>
13. PostgreSQL <https://www.postgresql.org/>
14. TimescaleDB <https://github.com/timescale/timescaledb>
15. WebAssembly <https://webassembly.org/>