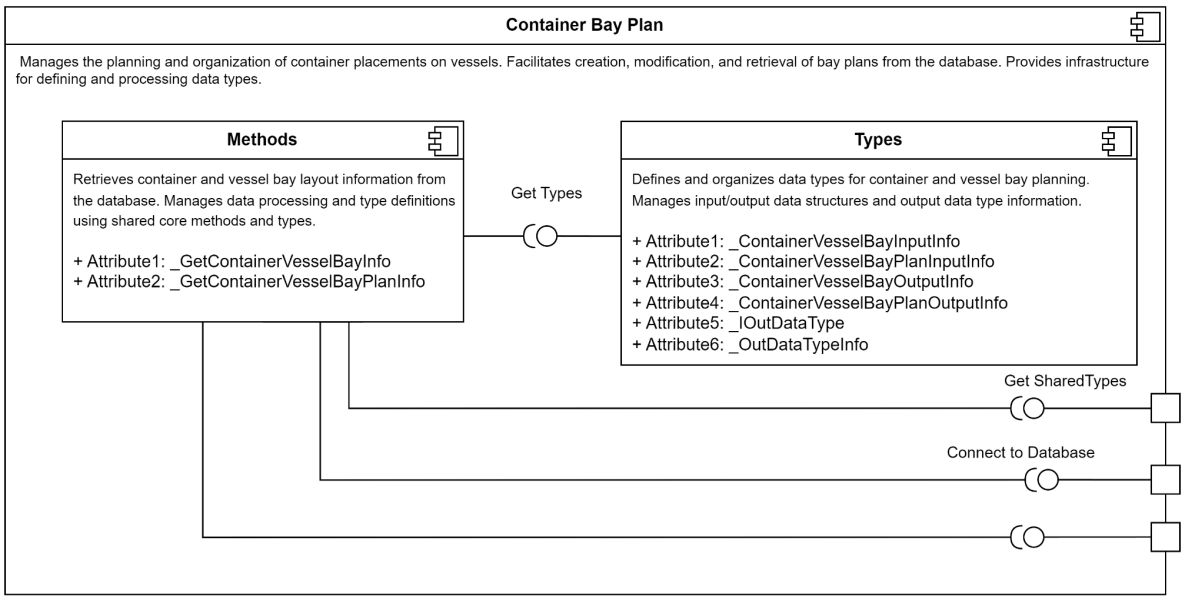
COMPONENT DIAGRAMS

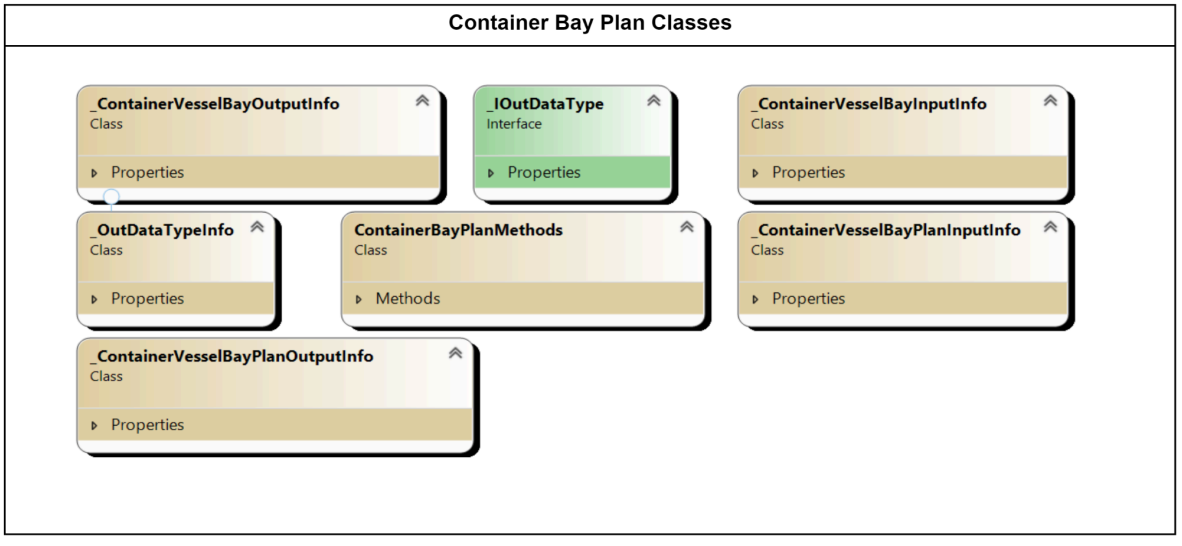
Container Bay Plan



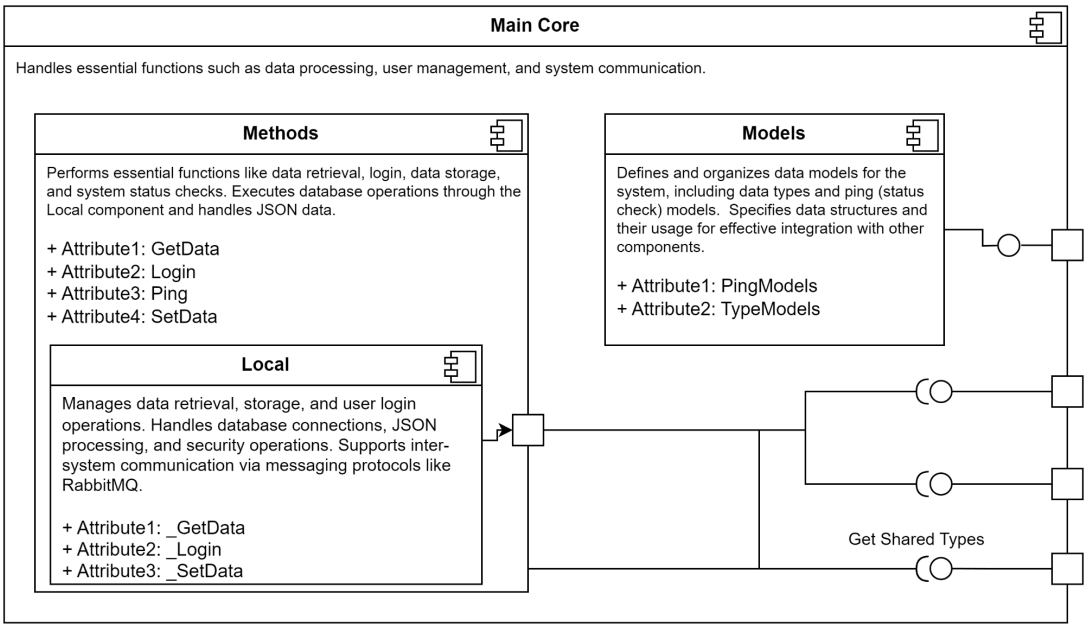
Container Bay Plan: Manages the planning and organization of container placements on vessels. Facilitates creation, modification, and retrieval of bay plans from the database. Provides infrastructure for defining and processing data types.

Methods: Retrieves container and vessel bay layout information from the database. Manages data processing and type definitions using shared core methods and types.

Types: Defines and organizes data types for container and vessel bay planning. Manages input/output data structures and output data type information.



Main Core

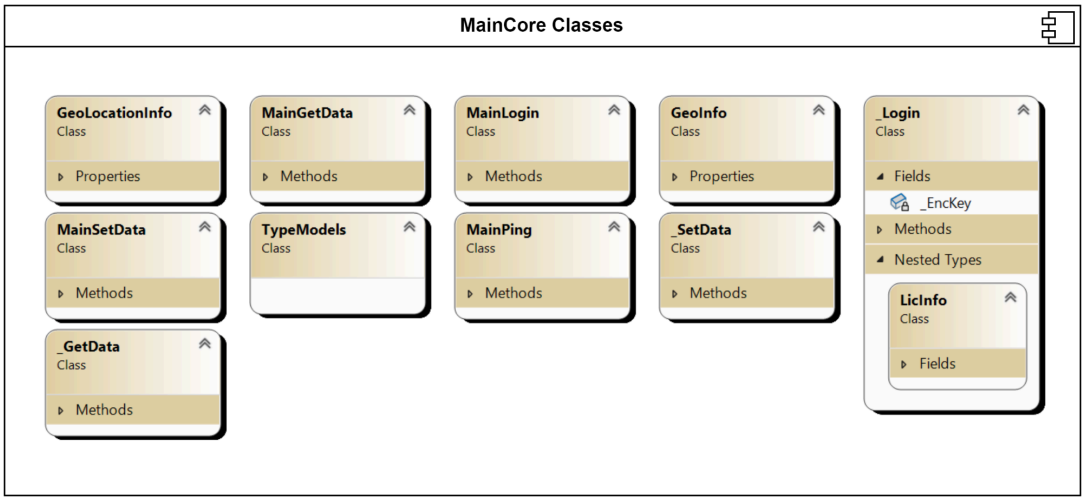


MainCore: Handles essential functions such as data processing, user management, and system communication.

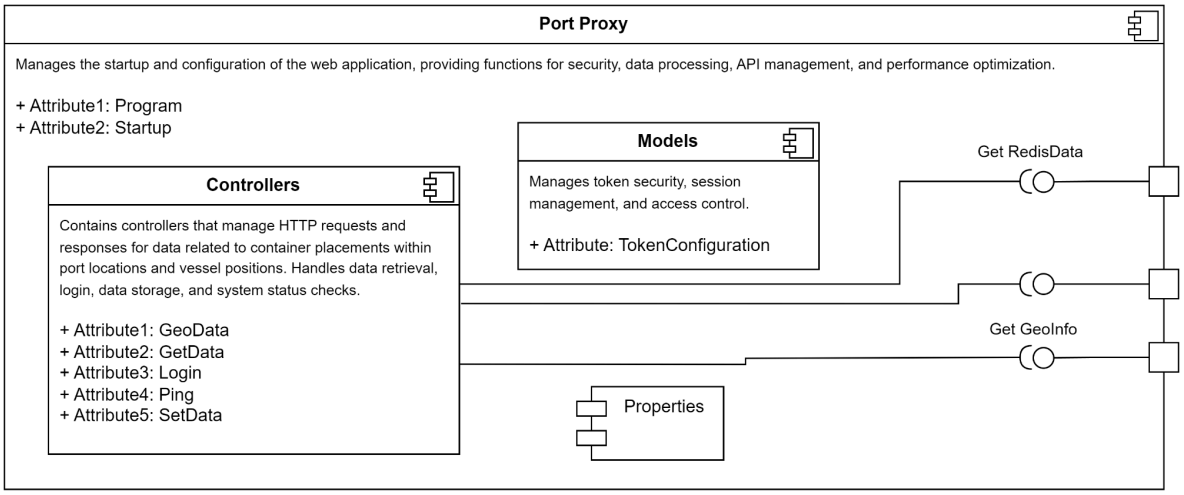
Methods: Performs essential functions like data retrieval, login, data storage, and system status checks. Executes database operations through the Local component and handles JSON data.

Methods.Local: Manages data retrieval, storage, and user login operations. Handles database connections, JSON processing, and security operations. Supports inter-system communication via messaging protocols like RabbitMQ.

Models: Defines and organizes data models for the system, including data types and ping (status check) models. Specifies data structures and their usage for effective integration with other components.



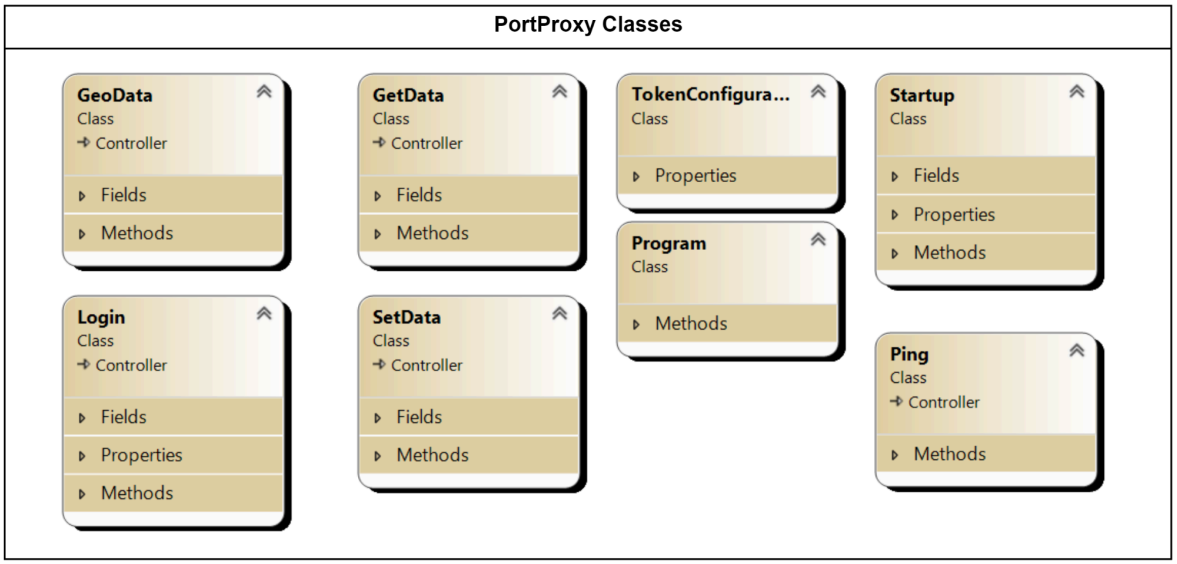
Port Proxy



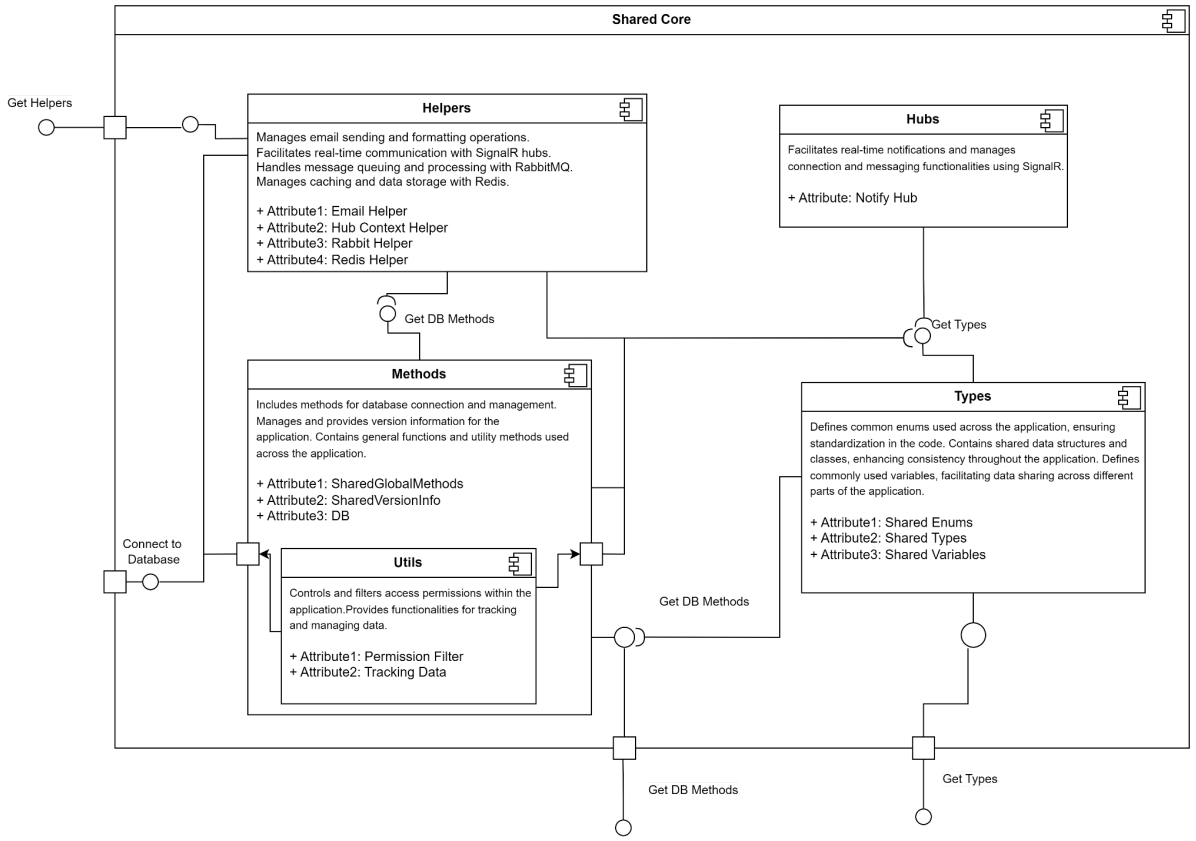
Port Proxy: The PortProxy package is designed to address configuration, service management, and security needs for an API-based application. It facilitates coordination and data management across various components of the application. Technologies such as JWT authentication, Redis integration, and SignalR specifically enhance the application's performance and security.

Controllers: The package hosts various controllers designed to handle and manage API requests. With its modular structure, Redis integration, JWT usage, and security features, it offers robust and secure API management. These controllers enable users to perform various data processing and authentication tasks quickly and securely.

Models: This provides a centralized location for managing the application's configuration settings, facilitating the sharing of configuration data across different components of the application.



Shared Core



Shared Core: The SharedCore module contains methods, types, and helper functions that are commonly used across various components of the application. This module facilitates integration and data sharing between different modules by providing central data processing, utility tools, and communication methods.

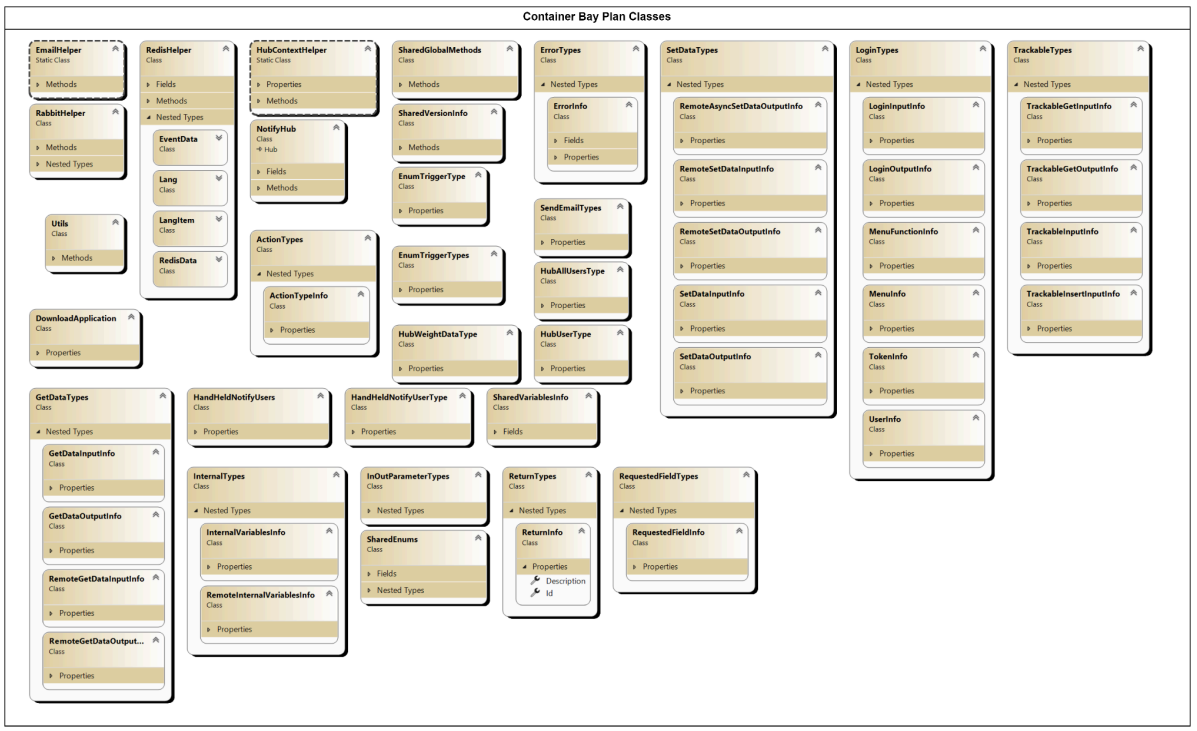
Helpers: Helpers include tools and functions frequently used within the application. These include email sending, Hub Context management, and integration with tools like RabbitMQ and Redis.

Hubs: The Hubs component manages SignalR connections used for real-time communication and signaling. This component handles connections that enable users to interact with the application.

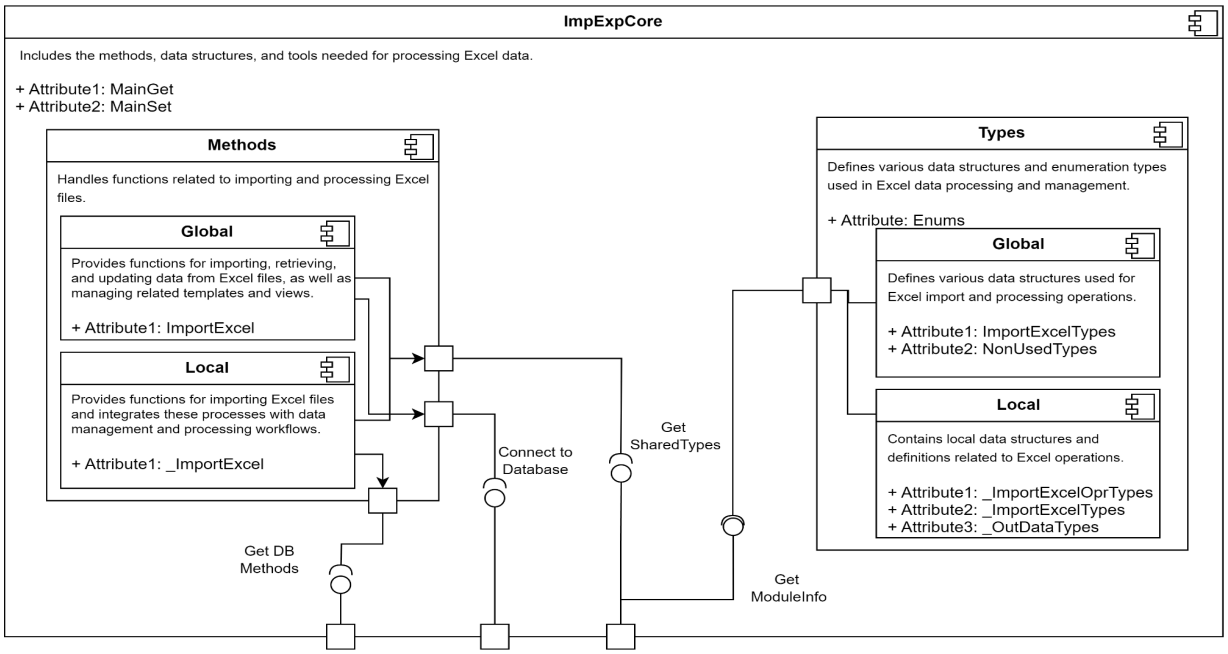
Types: This component contains shared data types that are used throughout the application. Other components extensively utilize these types to define and operate on data structures.

Methods: This component performs fundamental data operations such as SQL transactions, data security, and data manipulation. It also handles tasks like JSON data processing and file management.

Methods.Utils: Utility methods include frequently used operations such as database transactions, JSON processing, and general helper functions. These methods support data processing workflows.



ImpExp Core

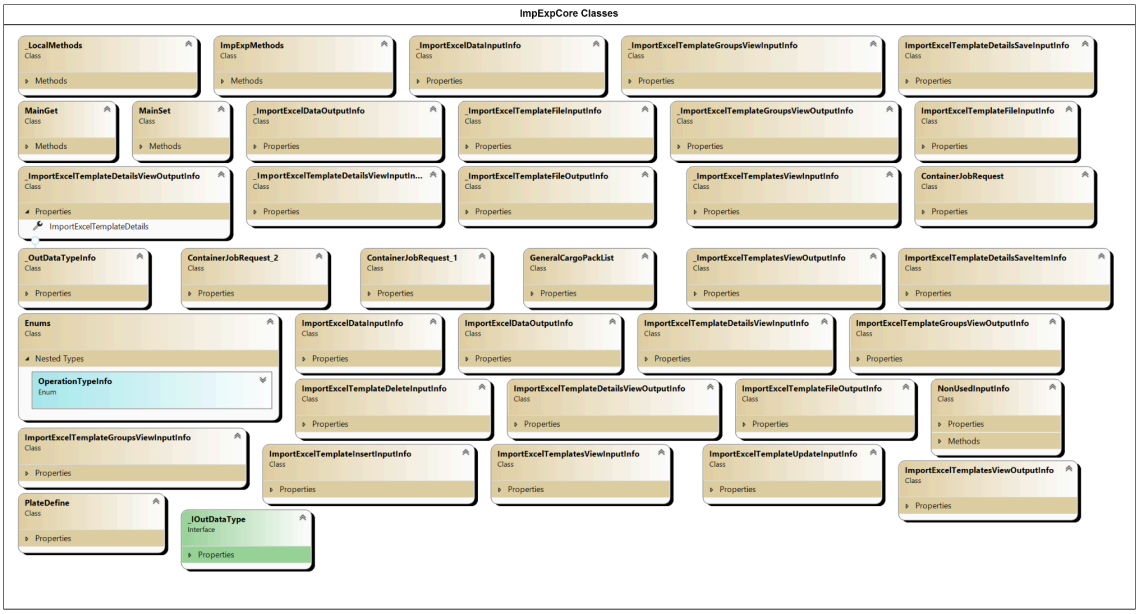


ImpExp Core: The ImpExpCore module manages the import and export of Excel files. This module provides functions for processing, transforming, and managing Excel data through local and global methods. Customized data types and operation types enable efficient handling of Excel operations.

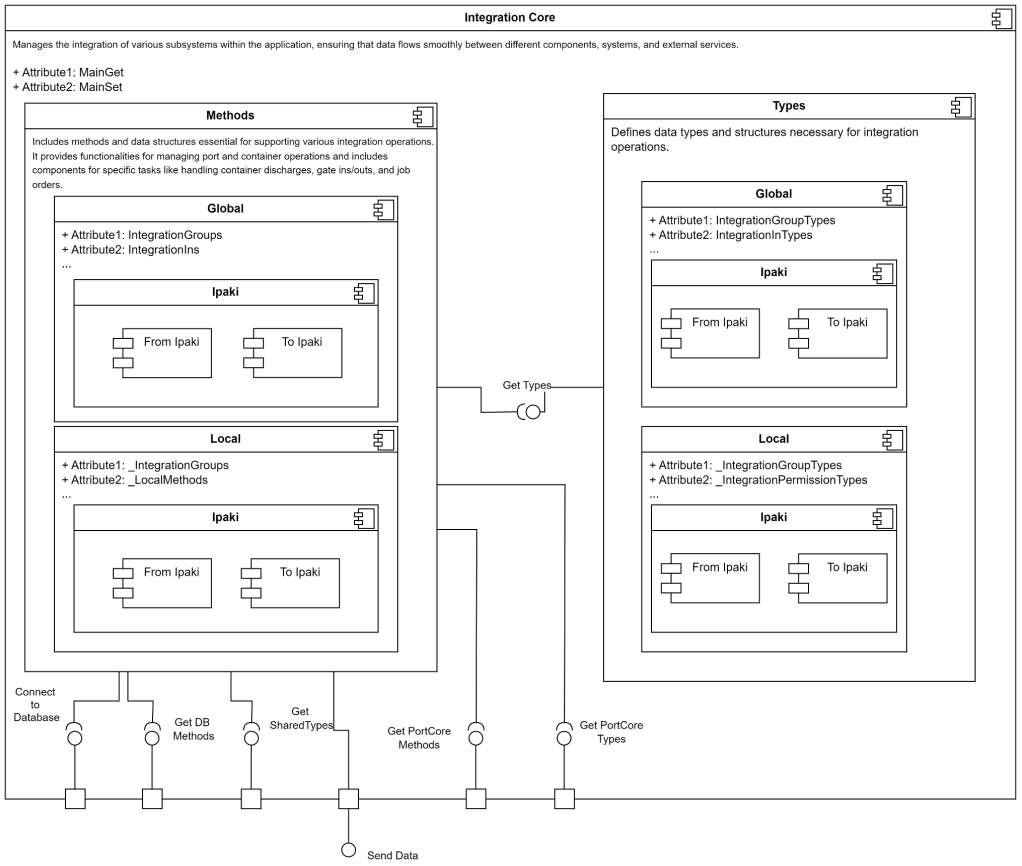
Methods: Local methods perform basic operations related to Excel files. These functions cover tasks such as file reading/writing, data formatting, and transferring Excel data to databases. Global methods are used to manage more extensive Excel data operations and integrations.

These functions handle tasks such as reading data from Excel files, transforming data, and exporting it.

Types: Local types define customized data types used in processing Excel data. These types are used to manage specific data structures within Excel files. Global types include data types used in processing Excel data and integrating it with other application components. These types generally define how Excel files are processed and transferred.



Integration Core



Integration Core: This package provides comprehensive functions for managing data integration and business processes between systems. This package supports various integration tasks related to containers and performs data transmission to specific integration systems using information from the database.

Methods: The package manages integration processes related to containers and uses information from the database to send this information to the appropriate integration systems. It includes functions for checking integration permissions, retrieving data, and processing this data to perform necessary integration operations.

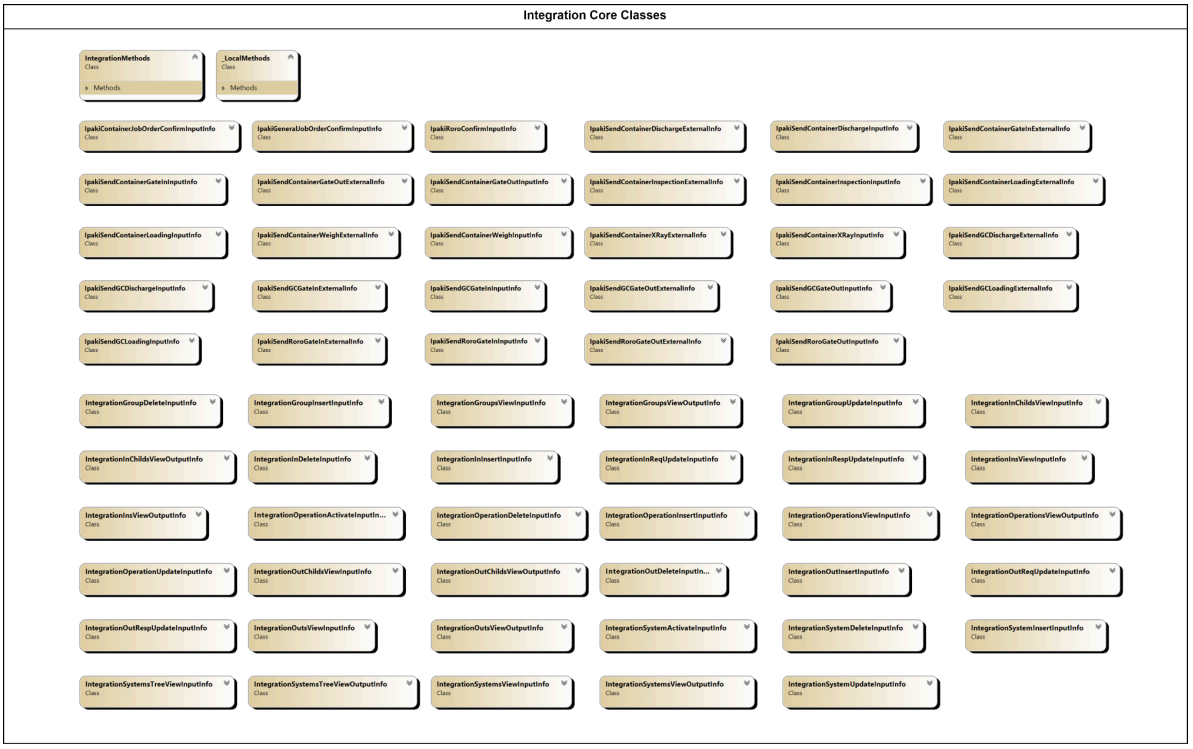
Methods.Global: This package provides classes with core functions to manage integration processes related to containers. These functions involve reading and processing information from the database and sending it to a specific integration system.

Methods.Local: This provides the methods required to retrieve and process information related to system integrations and configurations from the database.

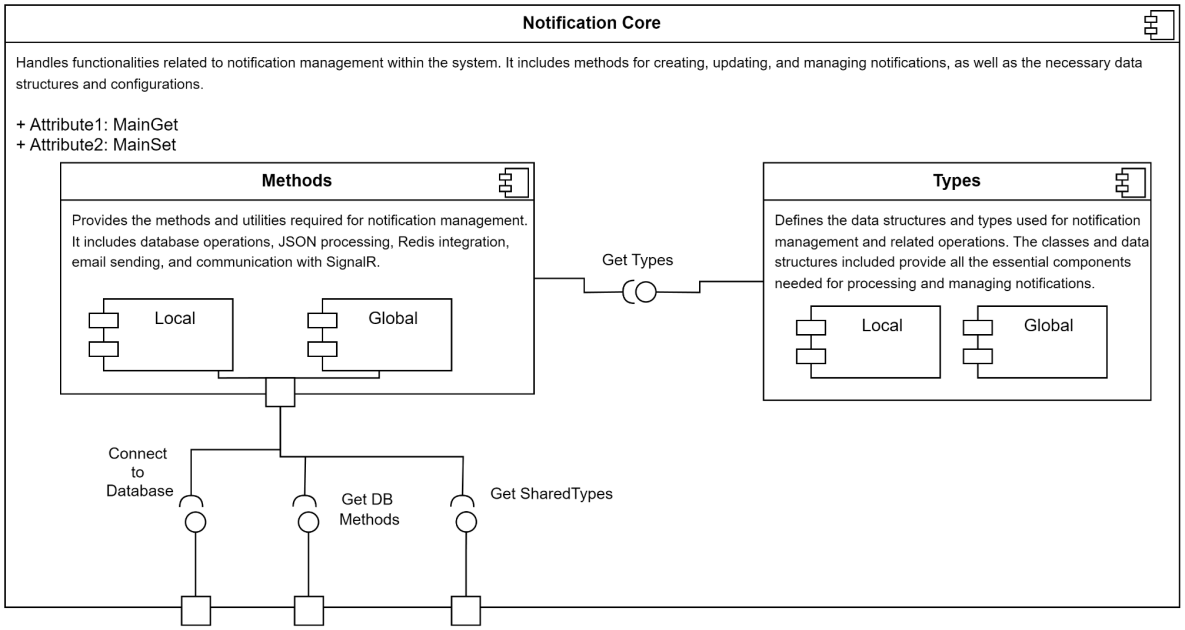
Types: The package contains data structures that support container and cargo transportation processes. It includes various classes related to container and cargo information such as entry, exit, relocation, and general details. These classes provide fundamental data structures for representing, processing, and outputting container and cargo information.

Types.Global: The integration system provides a comprehensive structure for managing its core components. By separating input and output data structures, it offers a flexible and extensible system.

Types.Local: The Local subpackage systematically organizes data flow and management within the integration system. Each class targets a specific data type or operation group, ensuring that data is processed in a structured and orderly manner. This makes the overall operation of the system more efficient and manageable.



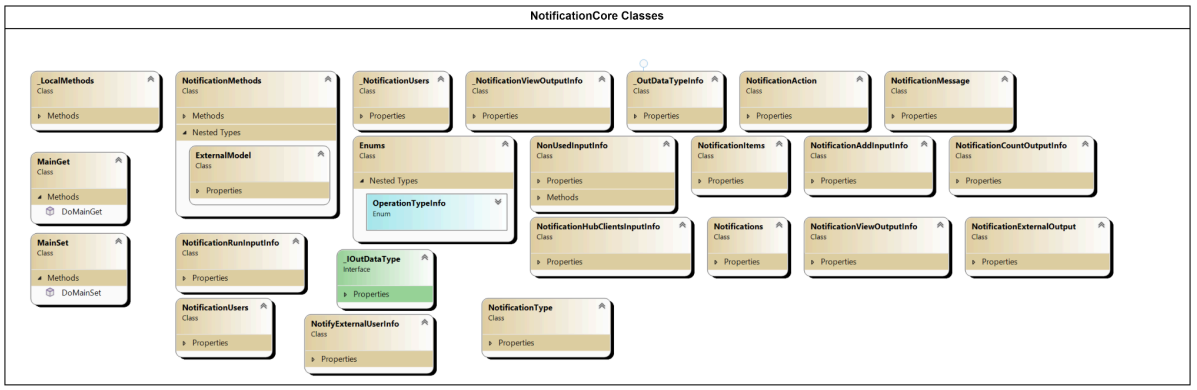
Notification Core



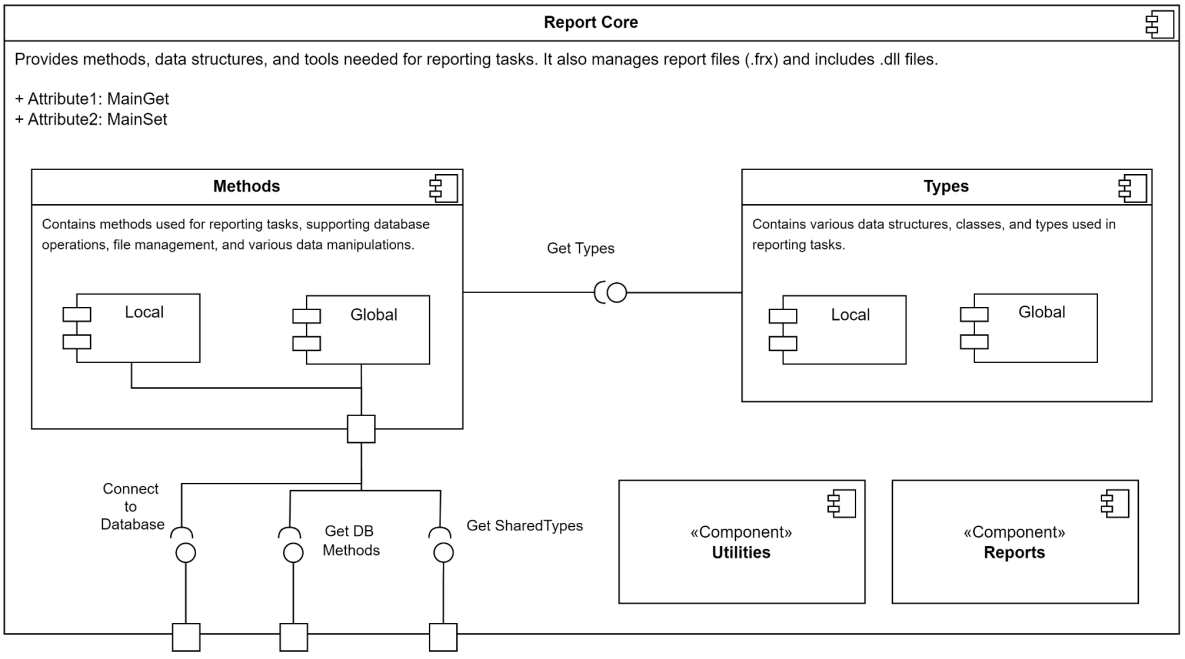
Notification Core: The Notification Core module manages all operations related to notifications. This module provides various notification functions through both local and global methods, along with custom data types.

Methods: Local methods include core functions used in notification processes. These functions typically deal with database operations and notification types. Global methods are used to perform more extensive notification operations. These functions include Redis integration, email notifications, and SignalR features.

Types: Local types define custom data types used in notification operations. These types are designed to be used exclusively within the Notification Core. Global types include data types that can be used across different parts of the application. These types generally define data structures used in notifications.



Report Core



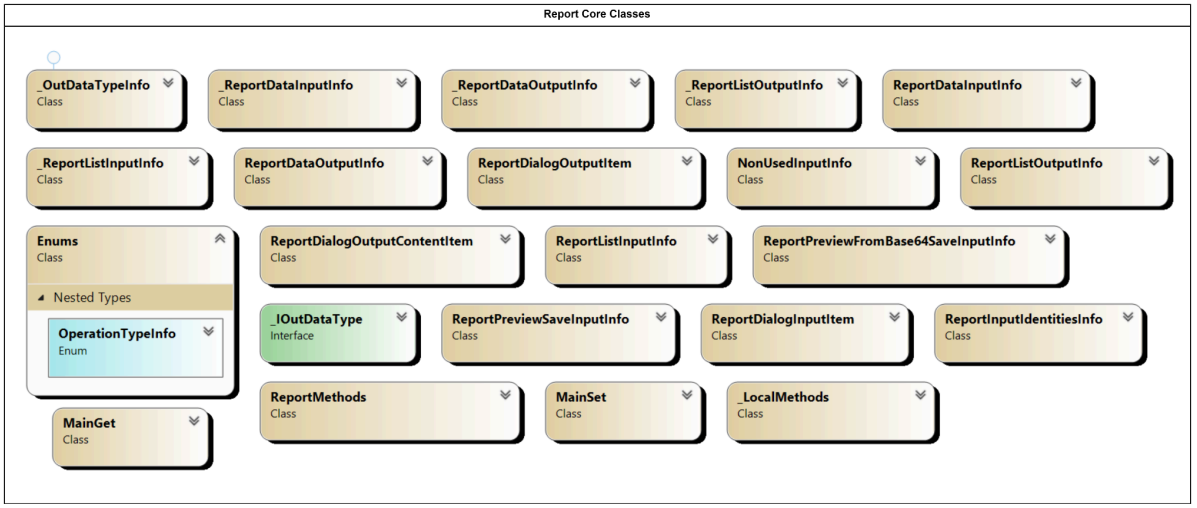
Report Core: This module contains the core components necessary for reporting operations.

Methods: Local methods include functions used in the report creation process. It interacts with FastReport and other libraries. Global methods include functions used for more general and comprehensive operations. These functions work with database operations and shared types.

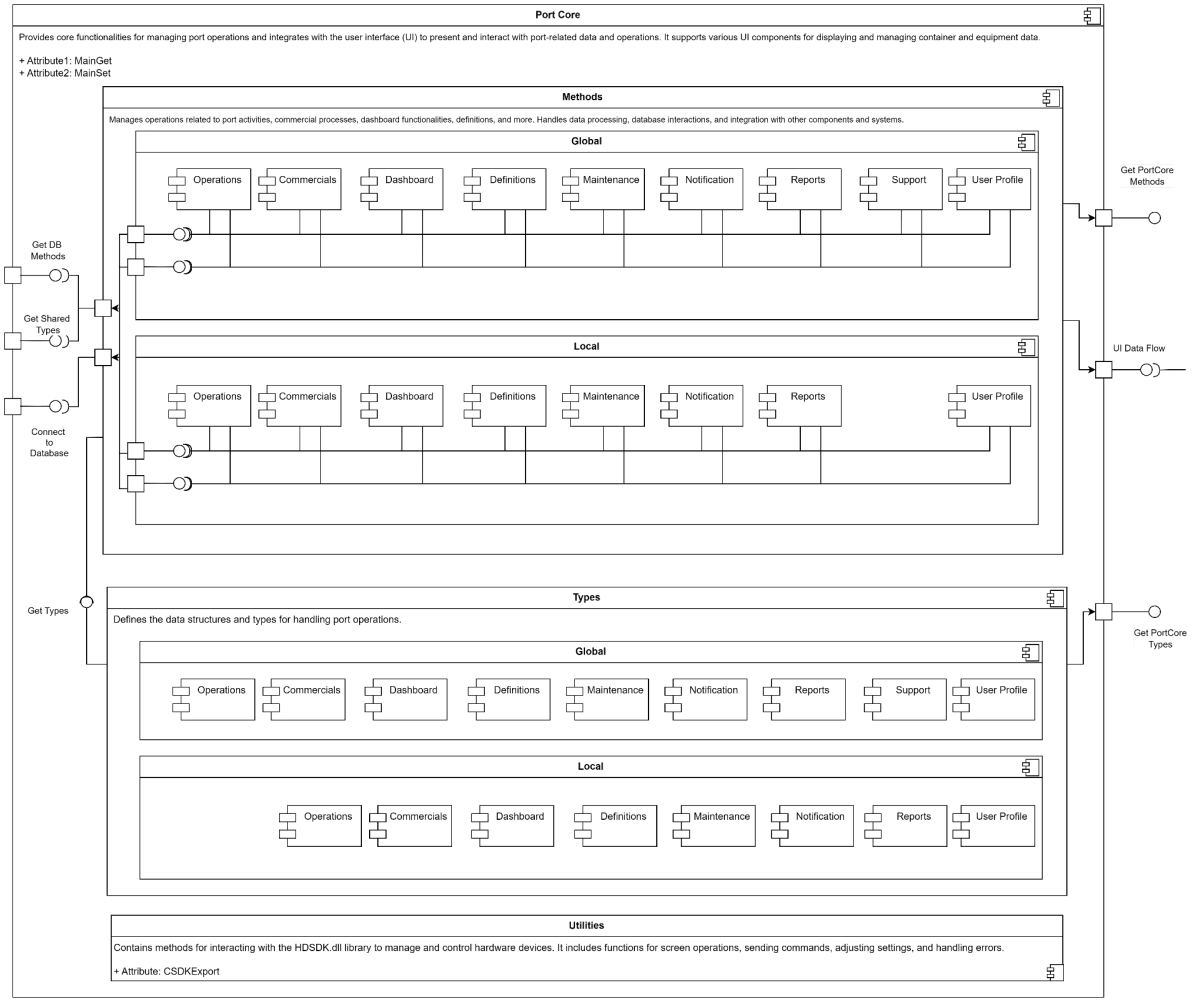
Types: Local types include custom data types used in reporting. These types have a limited interaction scope with other parts of the application. Global types include data types used throughout the application. These types are commonly used in reporting operations.

Reports: Reports contain report templates defined with .frx files. These templates are used in report generation.

Utilities: Utilities contain additional tools and libraries that support the reporting processes. These tools include external libraries like FastReport.



Port Core



Port Core: Provides core functionalities for managing port operations and integrates with the user interface (UI) to present and interact with port-related data and operations. It supports various UI components for displaying and managing container and equipment data.

Methods: This component encompasses all the methods necessary for executing port operations. These methods manage various processes within the port, handle data access and processing. The methods provided range from database operations to reporting and notification management, covering a broad spectrum of functionalities.

Methods.Global: Global subpackage brings together essential methods that cover various operations. It includes subpackages like Support, which handles user request processing and support service management; UserProfile, which manages operations related to user profiles and security processes; Commercials, which oversees financial operations such as invoicing and payments; Dashboard, which monitors the overall status of port operations and manages critical data regarding area information and vessel status; Definitions, which defines the system's core settings and building blocks; Maintenance, which manages equipment and infrastructure maintenance; Notification, which informs users about system events; and Operations, which ensures the efficient execution of port operations, including container operations and vessel job orders.

Methods.Local: This focuses on more specialized and context-specific operations within the port management system. These methods include operations for checking and validating container status,

managing approval processes, and handling gate operations. The class also includes methods for detailed data retrieval and processing, such as fetching various definitions and configurations needed for specific.

Types: Types contains all the type and class definitions needed for the PortCore components. It defines data structures, constants, enums, and other types required by each sub-component (e.g., Commercials, Dashboard, Operations), ensuring consistency and reusability across the components. Local types defines local data types and structures related to port operations. These classes and data structures facilitate data management and processing within the application, providing customised data structures tailored to specific business needs. Global defines general and shared data types and structures related to port operations. These global data structures ensure consistency across the system and facilitate data sharing and interaction between different components.

Utilities: Contains methods for interacting with the HDSDK.dll library to manage and control hardware devices. It includes functions for screen operations, sending commands, adjusting settings, and handling errors.