# sIntroduction

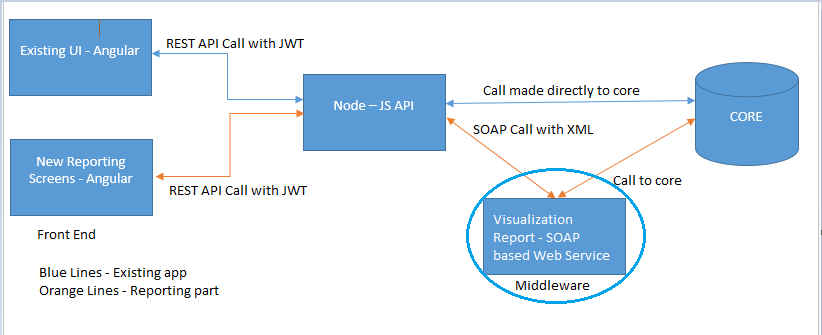
VenAqua Visualization Report application is a middleware application which interacts with CORE DB 2.0 in the existing VenAqua architecture and front-end application. VenAqua Visualization Report fetches required data from CORE DB system and manipulates it as needed to visualize in the front end application. It publishes SOAP based Web Services API, through which an internal components of a VenAqua architecture or a third party applications can communicate to VenAqua Visualization Report application to get a desired data.

VenAqua Visualization Report application collects and manipulates data from the CORE DB system, such as

* Water Consumption from different sources in a site
* Water Consumption for individual house
* Various water types and their demand in a site
* Pump’s ON/OFF states and their yields
* Various water sources and its performance

# Communication Architecture

The below diagram shows how the Visualization Report application can be invoked from the front end application.



In this architecture, the third party applications must be authenticated by the middleware application in order to communicate further. In the subsequent calls, the third party application must send their access token which is received in the previous call.

# Visualization Report Architecture

Visualization Report application has divided into multiple sub modules, each modules have their own roles to prepare desired data to the front end or third party application.

**General Utility**

**SOAP  
Web Services**

**CORE ENGINE**

**DB  
Handler**

**Web Services**

**CORE DB  
2.0**

**Token   
Manager**

**Authentication  
Handler**

**Data Preparer**

**Error Handler**

## General Utility

General utility module contains classes which are related to utility purpose. These classes are used across all the modules in this architecture.

## SOAP Web Services

This module publishes SOAP based web service API with multiple endpoints. Each endpoint is used for different purpose.

* /report – is a base path for the API.
* /report/login – endpoint for login service. Access token will be returned, if the user credentials are valid and client can use this access token for subsequent calls.

This module transfer control to Core Engine module to fetch required data from Core DB2.0 system for all requests.

## Core Engine

Core Engine is a centralized place which communicates with all other modules for different actions.

* Once the request is received, the core engine module validates user credentials or access token with the help of Authentication handler module.
* Uses DB Handler module to interact with the Core DB 2.0 system to fetch requested data and prepares response data by using Data Preparer module.
* Error Handler module will be used to send error responses, in case of any exception while processing client request.

## Authentication Handler

The Authentication handler validates the user credentials during the login call and also checks whether the access token is valid and its expiration with the help of Token manager module. The authentication handler acknowledges back to the core engine module, to process further for a particular request. Authentication failure response will be sent if the user credentials or the access tokens are not valid.

## Token Manager

It manages the access tokens for a user by interacting with the core db2.0 module. This module mainly invoked by the Authentication handler module to complete the authentication process.

## DB Handler

It uses JDBC driver libraries to communicate with the Core DB2.0 system. Major role for this module is manages connection with the database and executes SQL queries to fetch the data. Core Engine module uses this to prepare desired data for a request.

## Data Preparer

## Error Handler