

Administrative Office of the Courts

**Technical Specification**

**External API Administrator’s Portal**

Created from SDLC Template: Technical Specification v2.0

|  |  |
| --- | --- |
| ***Filename:*** | Technical Specification – External API.docx |
| ***Revision:*** | V 1.0.1 |
| ***Last Save Date:*** | Thursday, June 20, 2024 |
| ***Author(s):*** | Gibson, Scott, Amiya Ranjan, Sahayarani Jesudass |

Table of Contents

[1 Introduction 1](#_Toc169607899)

[2 System Architecture 2](#_Toc169607900)

[2.1 Distributed System View 2](#_Toc169607901)

[2.2 Code Management Tool 7](#_Toc169607902)

[3 Front End 9](#_Toc169607903)

[3.1 eAPI Administrator Portal 9](#_Toc169607904)

[4 Middle Tier Objects (Services/Components) 18](#_Toc169607905)

[4.1 Services 18](#_Toc169607906)

[4.2 User Interface Library 22](#_Toc169607907)

[4.3 Data Contract Objects 23](#_Toc169607908)

[5 Database 24](#_Toc169607909)

[5.1 Tables 25](#_Toc169607910)

[5.2 Triggers 25](#_Toc169607911)

[5.3 User Defined Functions 25](#_Toc169607912)

[5.4 Stored Procedures 26](#_Toc169607913)

[5.5 Dependencies 28](#_Toc169607914)

[5.6 Assumptions and Limitations 28](#_Toc169607915)

[5.7 Scripts 29](#_Toc169607916)

[5.8 Domain Data 29](#_Toc169607917)

[Special Considerations 29](#_Toc169607918)

[5.9 Volume Considerations 29](#_Toc169607919)

[5.10 Testing Considerations 29](#_Toc169607920)

[5.11 Production Support Considerations 29](#_Toc169607921)

[5.12 Performance & Response Time 29](#_Toc169607922)

[5.13 Setup and Deployment Strategy 30](#_Toc169607923)

[Appendix A Glossary/ Definitions 31](#_Toc169607924)

[Appendix B Related Documents/References 32](#_Toc169607925)

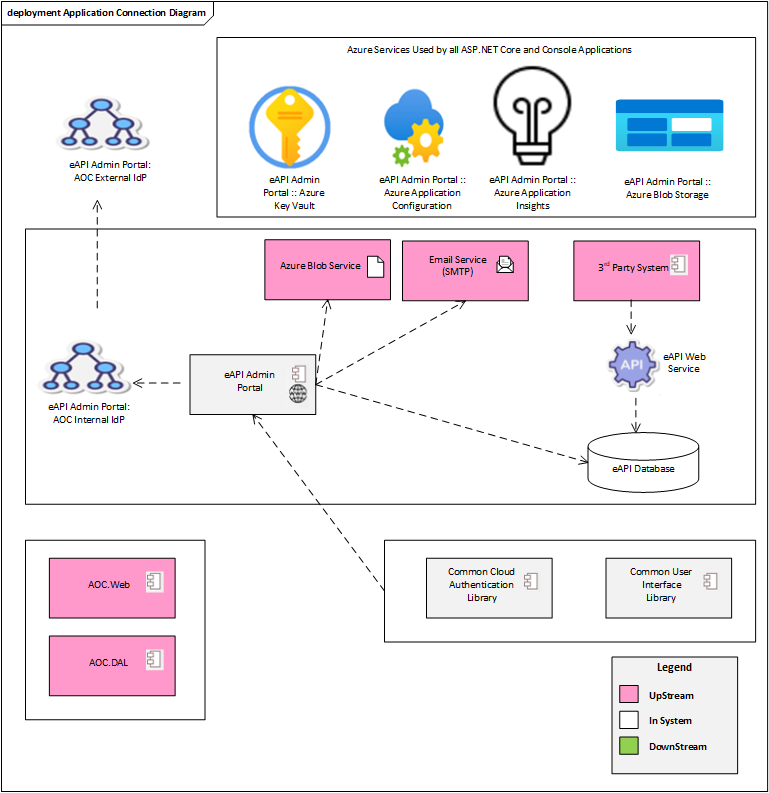
[Appendix C Document Change History 33](#_Toc169607926)

[Appendix D Review and Sign-off 34](#_Toc169607927)

1. Introduction

External API (eAPI) Administration Portal empowers administrators to seamlessly integrate 3rd party systems, manage their access to critical court CMS data, and oversee authentication credentials. Built entirely on .NET 8 solutions, except for robust SQL database, this application ensures efficient data flow between the administration portal service and UI portals via a purpose-built common library. This portal simplifies interactions between UI portals and services, enhancing both security and operational effectiveness

1. System Architecture
   1. Distributed System View
      1. Application Connection Diagram



* + 1. Azure Key Vault

Azure Key Vault is a service provided by Azure to securely manage application secrets. It allows for the centralized control of application configuration within an Azure tenant. Applications consume Key Vault key/value pairs using the appropriate web services. The Azure Key Vault platform is provided and maintained by Microsoft but administered by AOC staff.

* + 1. Azure Application Configuration

Azure Application Configuration is a service provided by Azure to manage application configuration details. It allows for the centralized control of application configuration within an Azure tenant. Applications consume Application Configuration key/value pairs using the appropriate web services. The Azure Application Configuration platform is provided and maintained by Microsoft but administered by AOC staff.

* + 1. Azure Application Insights

Azure Application Insights is a service provided by Azure to allow for observability of applications, particularly around logging, performance timing, and security. It allows for centralized tracking of application health indicators. Applications utilize Application Insights with Microsoft-provided libraries that transmit telemetry to the Azure Cloud. The Application Insights platform is provided and maintained by Microsoft but administered by AOC staff.

* + 1. AOC External ldP (Identity Provider)

The External IdP is only used by the admin portal for its claims extension SQL database which holds the list of administrators used for authorization and data display purposes.

* + 1. AOC Internal ldP (Identity Provider)

The AOC Internal IdP provides the ability to log in with accounts that are centrally managed by AOC, including AOC staff and administrator staff. The AOC Internal IdP platform is provided and maintained by AOC. The internal IdP is backed by Active Directory.

* + 1. Azure Blob Storage

Azure Blob Storage is a cloud-based object storage service provided by Microsoft Azure. It allows you to store enormous amounts of unstructured data, such as text or binary data, in the cloud. Azure Blob Storage provides a reliable and scalable solution for storing and managing enormous amounts of unstructured data in the cloud, with built-in features for security, availability, and cost optimization. This portal will use blob storage to store the uploaded documents in a secured encrypted format.

* + 1. Email Service

The Email Service allows for sending e-mails to the 3rd Party System contacts when System is Removed, Updated and Suspended via the eAPI Administration Web Portal. The Email Service is provided and maintained by AOC and will use SMTP server to execute email operations.

* + 1. eAPI Administration Portal
       1. Application Description

The eAPI Administration Portal allows the administrators to add 3rd party systems to the portal, manage those system’s access to court CMS data, and manage the authentication credentials for the 3rd party system into the eAPI.

* + - 1. Technology Portfolio

This ASP.NET Core MVC web application runs on .NET 8, Windows Server 2019, and IIS 10. It relies on the service client, common user interface library, eAPI Admin Portal service, and eAPI management database. It also relies on jQuery-UI, jQuery-validate, jQuery Data Tables, Font Awesome and Bootstrap to aid in client-side UI rendering and interactions. It may use other jQuery open-source libraries if required during development phase.

* + 1. Common Cloud Authentication Library
       1. Application Description

The common cloud authentication library provides for the reuse of code that is used to integrate with Azure services across all of the web portals in the admin portal.

* + - 1. Technology Portfolio

This is a .NET 6 class library application utilized within the web portals.

* + 1. Common User Interface Library
       1. Application Description

The common user interface library provides for the reuse of user interface elements that are shared across the various web portals within the admin portal.

* + - 1. Technology Portfolio

This is a .NET 6 class library application utilized within the web portals.

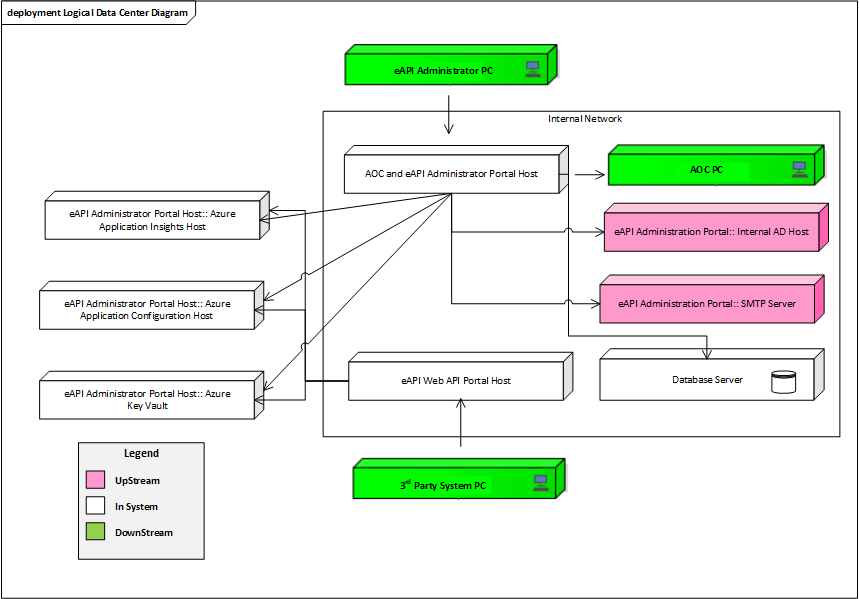
* + 1. Aoc.Web

This NuGet package contains common code utilized by all AOC-maintained web applications. Functions in this library support usage of web service clients, custom exceptions, and App Insights logging, amongst other features.

* + 1. Aoc.DAL

This NuGet package contains common code utilized for data access layers in AOC applications. It contains convenience code for writing database access layer classes.

Logical Host View



* + 1. Azure Key Vault

The Azure Key Vault Host is contained within the Microsoft-provisioned resources supporting the AOC Public Azure Tenant. Configuration of this host is determined by selections made when provisioning the Azure Key Vault environment and is outside of AOC control.

* + 1. Azure App Configuration Host

The Azure App Configuration Host is contained within the Microsoft-provisioned resources supporting the AOC Public Azure Tenant. The configuration of this host is determined by selections made when provisioning the Azure App Configuration environment and is outside of AOC control.

* + 1. Azure Application Insights Host

The Azure Application Insights Host is contained within the Microsoft-provisioned resources supporting the AOC Public Azure Tenant. Configuration of this host is determined by selections made when provisioning the Azure Application Insights environment and is otherwise outside of AOC control.

* + 1. Internal AD Host

This is the internal Active Directory System for AOC.

* + 1. SMTP Server

This is the SMTP server(s) for the AOC SMTP system.

* + 1. AOC PC

The AOC PC is a desktop or laptop issued to AOC staff for their usage in performing their AOC duties. AOC PCs must use the AOC Active Directory to authenticate users for access to AOC resources. While they may not be contained on the AOC internal network, they are expected to utilize the AOC VPN to access resources on the AOC internal network. They must follow established AOC standards and policies around security configuration and access management.

* + 1. AOC and eAPI Administrator Portal Host
       1. Logical Description

The AOC and eAPI Administration Portal host will contain the AOC Administration Portal.

* + - 1. Infrastructure

The Portal Host must be provisioned according to AOC policies and standards governing supported operating systems, hardware, and technology requirements for a web host.

* + - 1. Security Boundaries and policies

The Portal Host is contained within the AOC Internal network following established AOC security standards and policies around security configuration and access management. All sites must be configured with security certificates to support HTTPS communication. Production hosts must also be provided with application identity certificates to be used in authentication with the Azure configuration storage system and identification with each other.

* + - 1. Configuration and Directory Structure

The Portal Host adheres to AOC standards and policies around systems configuration and directory structure for web hosts.

* + 1. eAPI Web API Portal Host
       1. Logical Description

The eAPI Web API portal host will contain the RESTful API service method to get the basic information about the 3rd party system that makes the call. All 3rd party systems can access this method at any time. Access to these methods should be restricted to no more than 100 time per hours per 3rd party system.

* + - 1. Infrastructure

The Portal Host must be provisioned according to AOC policies and standards governing supported operating systems, hardware, and technology requirements for a web host.

* + - 1. Security Boundaries and policies

The Portal Host is contained within the Internal network following established AOC security standards and policies around security configuration and access management. All sites must be configured with security certificates to support HTTPS communication. Production hosts must also be provided with application identity certificates to be used in authentication with the Azure configuration storage system and identification with each other. Production servers must have access to the AOC and administration portal host on the AOC internal network to reach the web service application on those hosts.

* + - 1. Configuration and Directory Structure

The Portal Host adheres to AOC standards and policies around systems configuration and directory structure for web hosts. It should be configured with a single website for the public portal application.

* + 1. Database Server
       1. Logical Description

AOC Database Hosts support databases hosted on the most recent version of SQL Server supported by AOC database administrators. It contains the eAPI database along with the claims extension database for the External IdP used by the eAPI administrator portal.

* + - 1. Infrastructure

The Portal Host must be provisioned according to AOC policies and standards governing supported operating systems, hardware, and technology requirements for a web host.

* + - 1. Security Boundaries and policies

AOC Database Hosts are contained within the AOC internal network following established AOC security standards and policies around security configuration and access management.

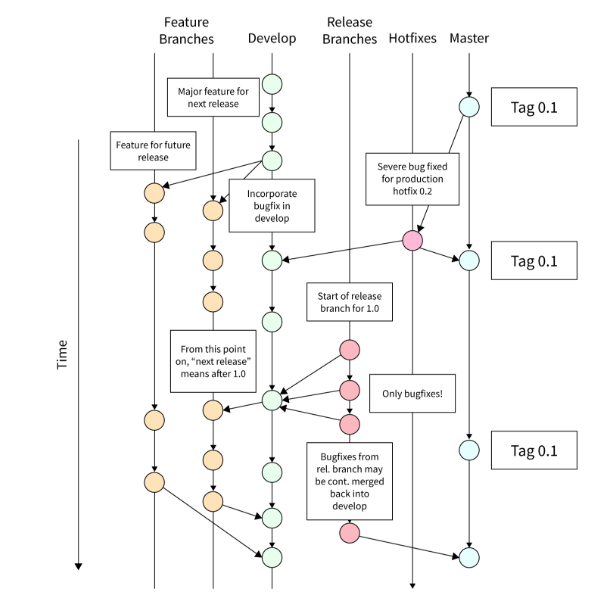
* + - 1. Configuration and Directory Structure

AOC Database Hosts adhere to AOC standards and policies around systems configuration and directory structure for database hosts.

* 1. Code Management Tool

Code management for all components will be performed using Git repositories. Git repositories shall be hosted on the AOC internal instance of Azure DevOps Server.

* + - 1. Branching Strategy



* + - 1. Master Branch

Represents the main branch where stable released are tagged.

* + - 1. Develop Branch

Integration branch where feature branches merge in.

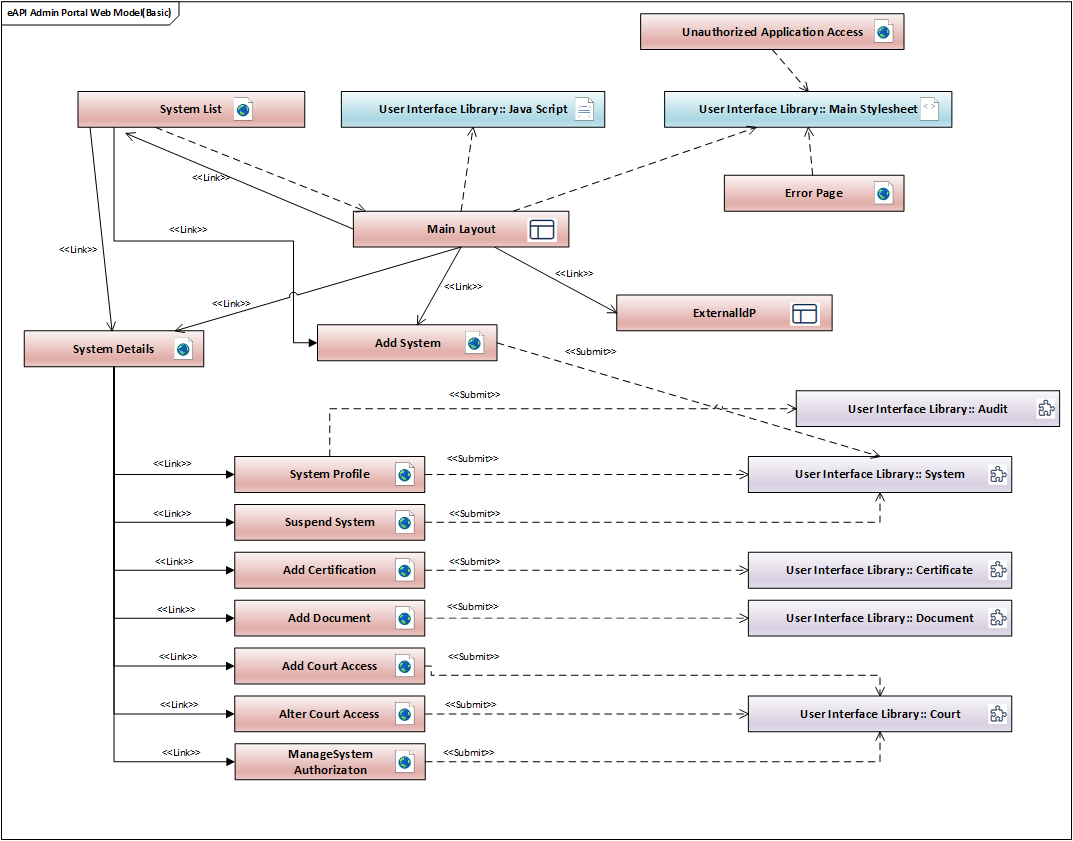
* + - 1. Feature Branches

Branches preparing for a new release.

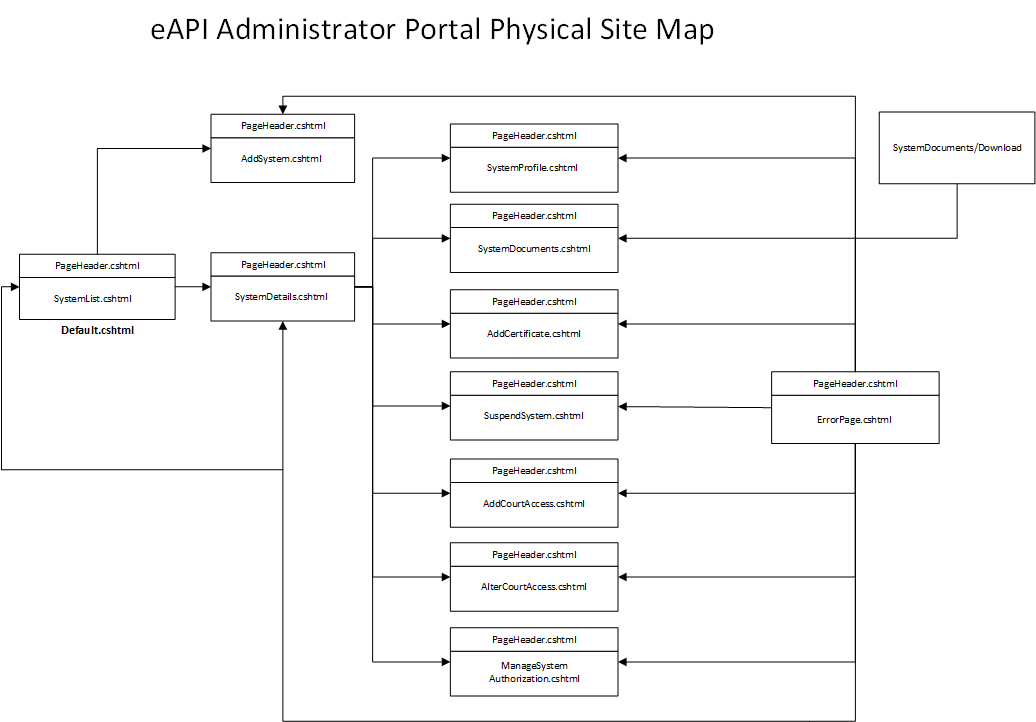
* + - 1. Hotfix

Branches for quickly fixing issues in production.

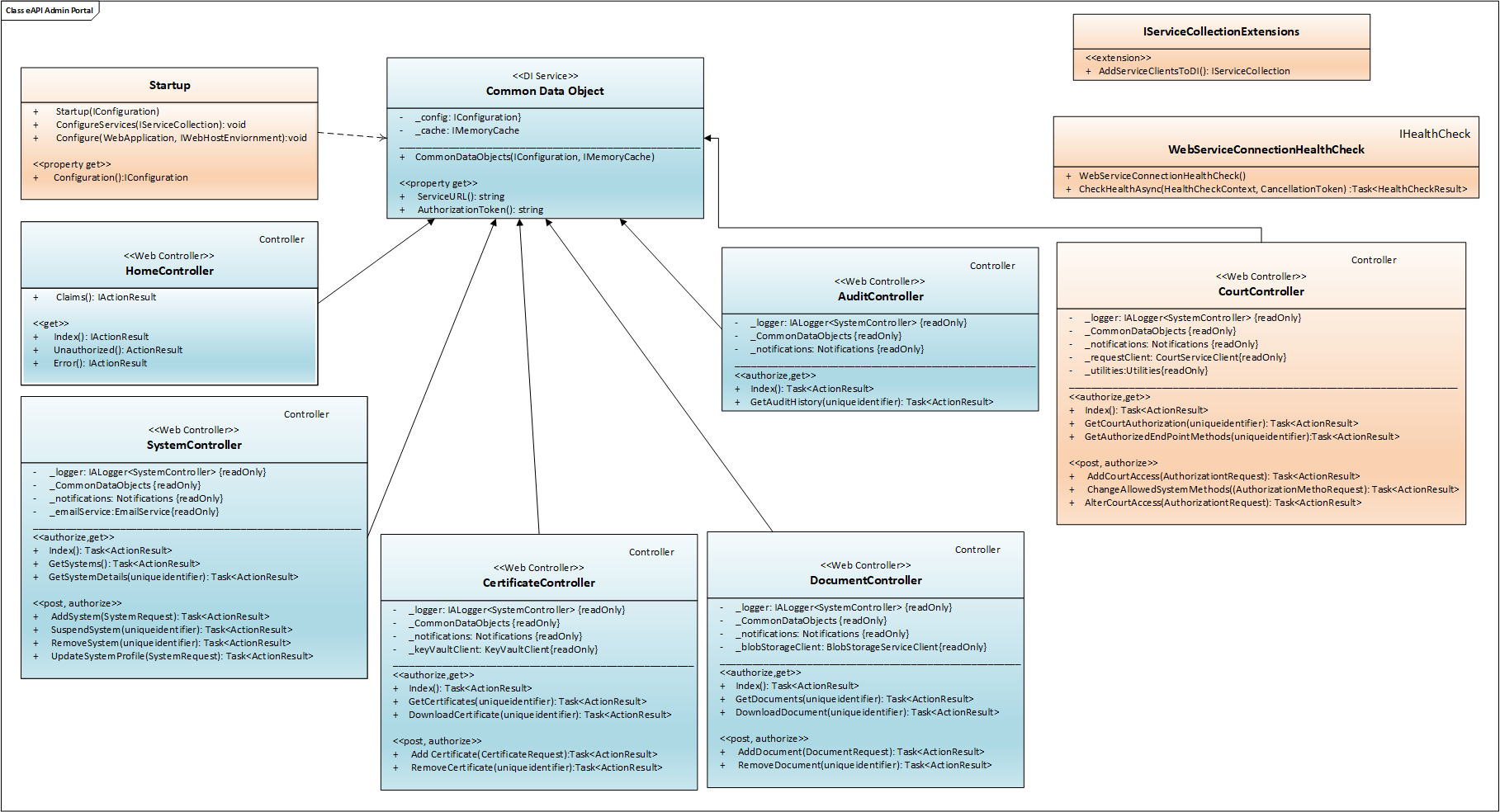
1. Front End
   1. eAPI Administrator Portal
      1. Web Model



* + 1. Physical Site Map



* + 1. Object Model



* + 1. Configuration

A note on configuration keys below: Azure Key Vault does not allow colons in configuration key names[[1]](#footnote-2). As such, this document uses the Key Vault standard double dash (--) to indicate nested configuration values. (Note that, in code, the double dashes are converted back to colons.)

App Configuration colon separators directly; however, for consistency, double dashes are used regardless of the location indicated below.

Keys that may change per-environment are noted with a “Y” in the “Env Label” column below. Environment labels shall be used to control the key value for those keys with a “Y”.

Key Vault secrets and application configuration settings shall be cached in the application for at least eight hours.

| Setting | Location | Env  Label? | Description |
| --- | --- | --- | --- |
| ClientId | Appsettings.json | n/a | The Azure AD id for the application |
| ClientSecret | Appsettings.json | n/a | The passcode for identifying the application (dev & test environments only) |
| ClientThumbprint | Appsettings.json | n/a | The thumbprint for the certificate to identify the application by (production only) |
| TenantId | Appsettings.json | n/a | The tenant the application is registered under |
| CallbackPath | Appsettings.json | n/a | Part of the signature for the application to authenticate itself. |
| VaultPath | Appsettings.json | n/a | The URL for the key vault holding other configuration values. |
| ServiceUrl | Key Vault | Y | The URL for the service location. |
| AppInsightsString | Key Vault | Y | The connection string for application insights. |
| IdPDBConnection | Key Vault | Y | The connection string for connecting the eIdP database. |
| ClientRateLimiting | App Configuration | Y | Configuration section for client-. based rate limiting. |
| ClientRateLimiting--EnableEndpointRateLimiting | App Configuration | Y | Determines whether client-based rate limits apply globally. True for endpoint-specific rules, false for global rules. |
| ClientRateLimiting--StackBlockedRequests | App Configuration | Y | Determines whether rejected requests are added to the throttle counter. |
| ClientRateLimiting--ClientIdHeader | App Configuration | Y | The HTTP header to use to identify a specific client. Used primarily for whitelisting clients from rate limiting restrictions. |
| ClientRateLimiting--HttpStatusCode | App Configuration | Y | The HTTP status code to return when a request is blocked. |
| ClientRateLimiting--GeneralRules | App Configuration | Y | Configuration section for rules configuration. |
| ClientRateLimiting—GeneralRule-—Endpoint | App Configuration | Y | The endpoint that the rule applies for. Asterisk indicates all endpoints. Note that this works in concert with the EnableEndpointRateLimiting configuration flag above. |
| ClientRateLimiting--GeneralRules--Period | App Configuration | Y | The time period to monitor request frequency for. |
| ClientRateLimiting--GeneralRules--Limit | App Configuration | Y | The number of requests that can be made within the request frequency time period before throttling is enabled for that client. |
| IpRateLimiting | App Configuration | Y | Configuration section for IP address-based rate limiting. |
| IpRateLimiting--EnableEndpointRateLimiting | App Configuration | Y | Determines whether IP-based rate limits apply globally. True for endpoint-specific rules, false for global rules. |
| IpRateLimiting--StackBlockedRequests | App Configuration | Y | Determines whether rejected requests are added to the throttle counter. |
| IpRateLimiting--RealIpHeader | App Configuration | Y | Used by the system to determine the HTTP header that contains the actual IP address of the requestor when the user is behind a load balancer or other proxy service. |
| IpRateLimiting--ClientIdHeader | App Configuration | Y | The HTTP header to use to identify a specific client. Used primarily for whitelisting clients from rate limiting restrictions. |
| IpRateLimiting--HttpStatusCode | App Configuration | Y | The HTTP status code to return when a request is blocked. |
| IpRateLimiting--GeneralRules | App Configuration | Y | Configuration section for rules surrounding IP-based rate limiting. |
| IpRateLimiting--GeneralRules--Endpoint | App Configuration | Y | The endpoint that the rule applies for. Asterisk indicates all endpoints. Note that this works in concert with the EnableEndpointRateLimiting configuration flag above. |
| IpRateLimiting--GeneralRules--Period | App Configuration | Y | The time period to monitor request frequency for. |
| IpRateLimiting--GeneralRules--Limit | App Configuration | Y | The number of requests that can be made within the request frequency time period before throttling is enabled for that IP. |
| DefaultAuditSearchPageSize | App Configuration | Y | Default page size that should be used when the web service is queried to search audit data. Specified here in case consumers of the service do not specify values. |
| AD: domain | App Configuration |  | Configuration to connect to EIDP. |
| AD:port | App Configuration |  | Configuration to connect to EIDP. |
| AD:subdomain | App Configuration |  | Configuration to connect to EIDP. |
| AD:unit | App Configuration |  | Configuration to connect to EIDP. |
| AD:zone | App Configuration |  | Configuration to connect to eIdP. |
| MaxFileSizeInMB | App Configuration | n/a | TBD |

NOTE: During the developmental phase, there may be additions of new keys and values to the settings mentioned above.

* + 1. Error Handling

All server-side errors are to be trapped at the controller method level and logged using

Application Insights. The user is to be displayed the error page per the functional specifications

after logging

* + 1. Security
       1. Authentication

The AOC Portal application will be configured to enable Windows authentication. Anonymous authentication will be allowed for static pages such as the error page and the unauthorized page.

* + - 1. Authorization

Users must be in the configured security group. Authorization is checked on each web request to the server for all controller methods except error handling and unauthorized access.

* + - 1. Encryption

All web-based traffic into and out of the application will be encrypted via HTTPS using TLS 1.2 or higher protocols.

* + - 1. App Authentication for Service Requests

The Administration portal uses Azure App Identity to ensure that all requests made to the Service are made by a legitimate requestor.

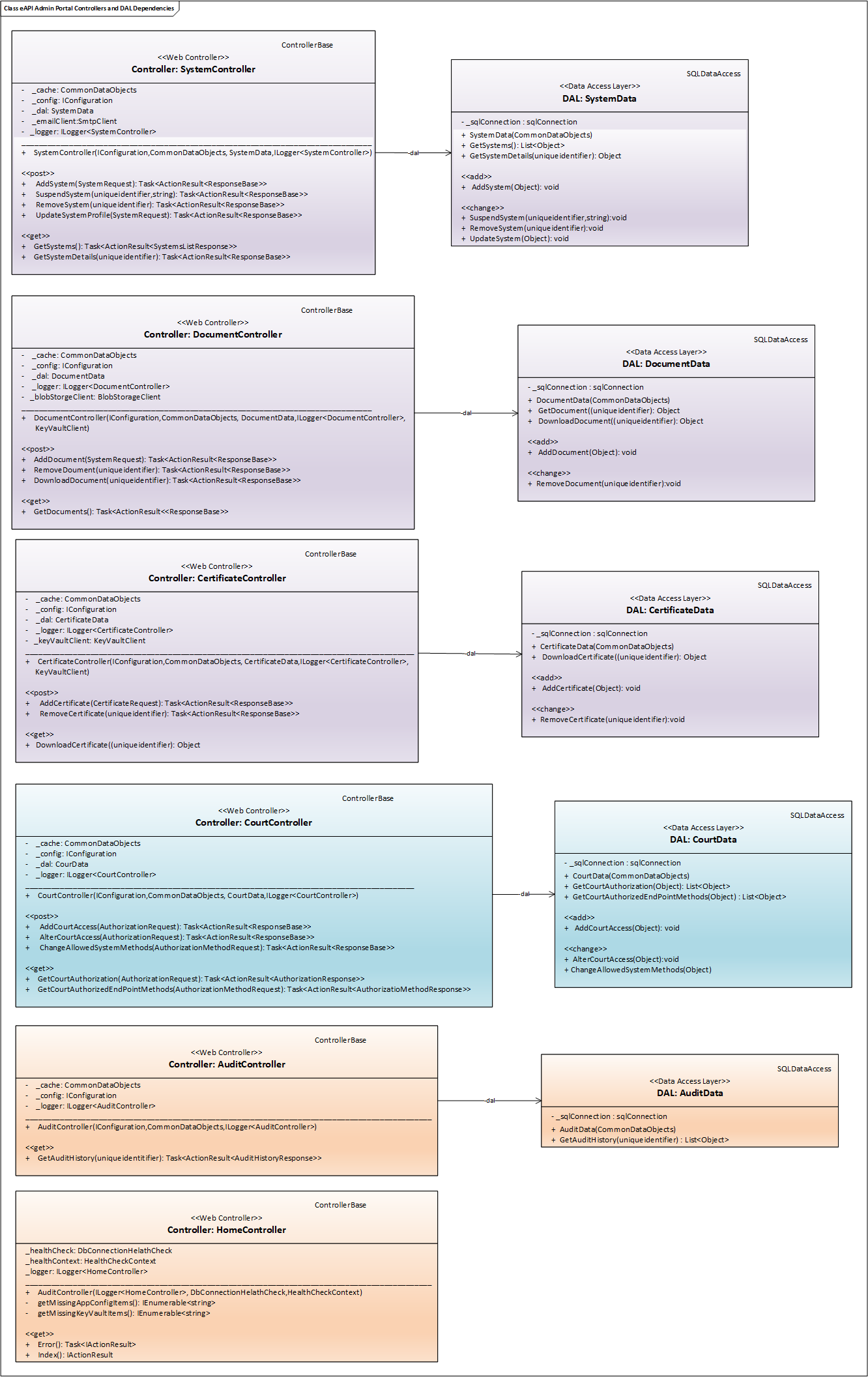
When an authenticated user makes a request on the front-end portal, the Portal uses a local client ID and client Secret to request a JWT from Azure. That JWT is attached to requests made to the Service to verify origin of the request. JWTs are cached up to one hour for performance reasons – if they were not, system performance would degrade significantly because it would mean round-trip request to Azure every request.

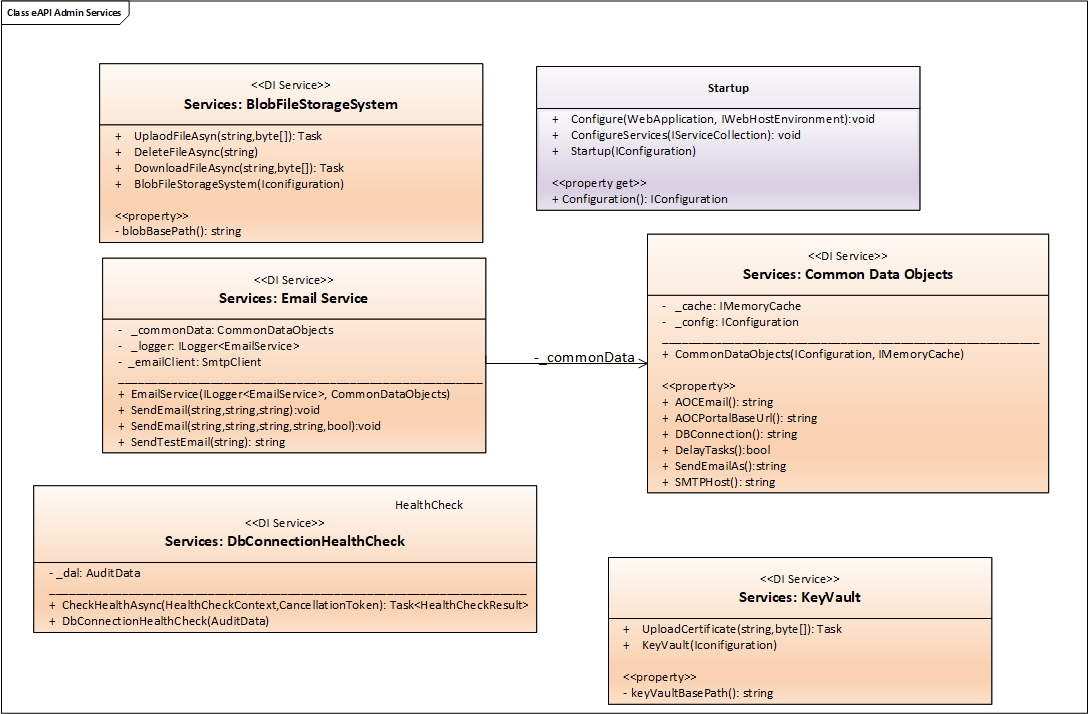
* + - 1. Cross Scripting and Input Security

Securing a .NET web application against Cross-Site Scripting (XSS) and ensuring robust input security involves implementing various measures throughout the development lifecycle. Here are specific steps and best practices tailored for this portal:

Cross-Site Scripting (XSS) Prevention:

1. Input Validation:
   * Purpose**:** Validate and sanitize all user inputs to ensure they meet expected formats and do not contain malicious content.
   * **Implementation:**
     + Use ASP.NET validators (RegularExpressionValidator, RequiredFieldValidator, etc.) for client-side validation.
     + Implement server-side validation in controllers or business logic layers using data annotations, custom validators, or manual validation checks.
     + Use libraries like AntiXSS library (part of Microsoft's Web Protection Library) to sanitize inputs and prevent XSS attacks.
2. **Output Encoding:**
   * **Purpose:** Encode user inputs and other dynamic content before rendering them in HTML to prevent them from being interpreted as HTML or JavaScript.
   * **Implementation:**
     + Use built-in ASP.NET Razor syntax (@Html.Raw, @Html.Encode) to automatically encode content.
     + Explicitly encode output using HttpUtility.HtmlEncode, Server.HtmlEncode, or AntiXss.HtmlEncode methods when rendering user-supplied data.
3. **Content Security Policy (CSP):**
   * **Purpose:** Define and enforce a Content Security Policy to restrict the sources from which certain types of content can be loaded on your web pages.
   * **Implementation:**
     + Configure CSP directives in your ASP.NET application's web.config file or via HTTP headers to specify allowed sources for scripts, stylesheets, images, etc.
     + Use the Content-Security-Policy meta tag in your HTML to set policies directly on individual pages.
4. **Regular Security Audits:**
   * **Purpose:** Conduct regular security audits and vulnerability assessments to identify and mitigate potential XSS vulnerabilities.
   * **Implementation:**
     + Utilize tools like Microsoft's Security Code Analysis extension for Visual Studio or third-party vulnerability scanners to check for XSS vulnerabilities in your codebase.
     + Perform manual code reviews focusing on input validation and output encoding practices.
5. **Secure Development Practices:**
   * **Purpose:** Educate developers on secure coding practices and integrate security into the software development lifecycle (SDLC).
   * **Implementation:**
     + Provide training on ASP.NET security features and best practices for XSS prevention.
     + Establish coding standards that include guidelines for input validation, output encoding, and secure handling of user inputs.
     + Implement security-focused code reviews and incorporate security testing into continuous integration and deployment pipelines.
6. Input Security:
7. **Parameterized Queries:**
   * **Purpose:** Use parameterized queries or stored procedures to prevent SQL injection attacks.
   * **Implementation:**
     + Use ADO.NET parameters to pass user inputs securely to database queries.
     + Avoid dynamic SQL generation with user inputs and validate any dynamic SQL usage thoroughly.
8. **Client-side and Server-side Validation:**
   * **Purpose:** Perform validation on both client-side (for user experience) and server-side (for security).
   * **Implementation:**
     + Use ASP.NET validators and jQuery validation for client-side validation to provide immediate feedback to users.
     + Implement server-side validation in ASP.NET MVC controllers or Web Forms code-behind to enforce data integrity and security.
9. **Input Encoding:**
   * **Purpose:** Encode user inputs appropriately before storing or displaying them to prevent injection attacks.
   * **Implementation:**
     + Use HttpUtility.HtmlEncode or AntiXss.HtmlEncode methods to encode user inputs before rendering them in HTML responses.
     + Encode inputs for other contexts (e.g., SQL parameters) using appropriate encoding methods (SqlParameter in ADO.NET for SQL queries).
10. **Use of Security Libraries and Frameworks:**
    * **Purpose:** Leverage trusted security libraries and frameworks that provide built-in protections against common security vulnerabilities.
    * **Implementation:**
      + Include the Microsoft Web Protection Library (AntiXSS) in your project to enhance XSS prevention capabilities.
      + Use OWASP (Open Web Application Security Project) libraries and guidelines for additional security measures specific to .NET applications.
11. Middle Tier Objects (Services/Components)
    1. Services
       1. Service Description





* + 1. Configuration

|  |  |  |
| --- | --- | --- |
| Setting | Location | Description |
| ClientId | Appsettings.json | The Azure AD id for the application |
| ClientSecret | Appsettings.json | The passcode for identifying the application (dev & test environments only) |
| ClientThumbprint | Appsettings.json | The thumbprint for the certificate to identify the application by (production only) |
| TenantId | Appsettings.json | The tenant the application is registered under |
| CallbackPath | Appsettings.json | Part of the signature for the application to authenticate itself |
| VaultPath | Appsettings.json | The url for the key vault holding other configuration values |
| DBConnection | Key Vault | The database connection string to the eAPI database |
| AOCEmail | Azure Application Configuration | The email address to send AOC staff emails to |
| AppInsightsString | Key Vault | The connection string for application insights |
| SmtpHost | Azure Application Configuration | The host name to use for SMTP relay. |
| SmtpSendAs | Azure Application Configuration | The “send as” e-mail that should be used when sending e-mail via SMTP. |

* + 1. Error Handling

All errors in the data access layer are to be trapped and bubbled up to their calling processing. That process is to log the error and return a generic fault response mapping to the logged error id and a more basic message. Async processes will log the errors and place a message with the logged error id in the result.

* + 1. Security
       1. Authentication

Applications making calls to the service must be registered in Azure AD, registered in and in administrator role.

* + - 1. Authorization

All the resources are only allowed for ADMIN users.

* + - 1. Encryption

All web-based traffic into and out of the application will be encrypted via HTTPS using TLS 1.2 or higher protocols.

* 1. User Interface Library
     1. Library Description



* + 1. Error Handling

Errors encountered while processing are passed directly to the calling application to be handled there.

* 1. Data Contract Objects

N/A

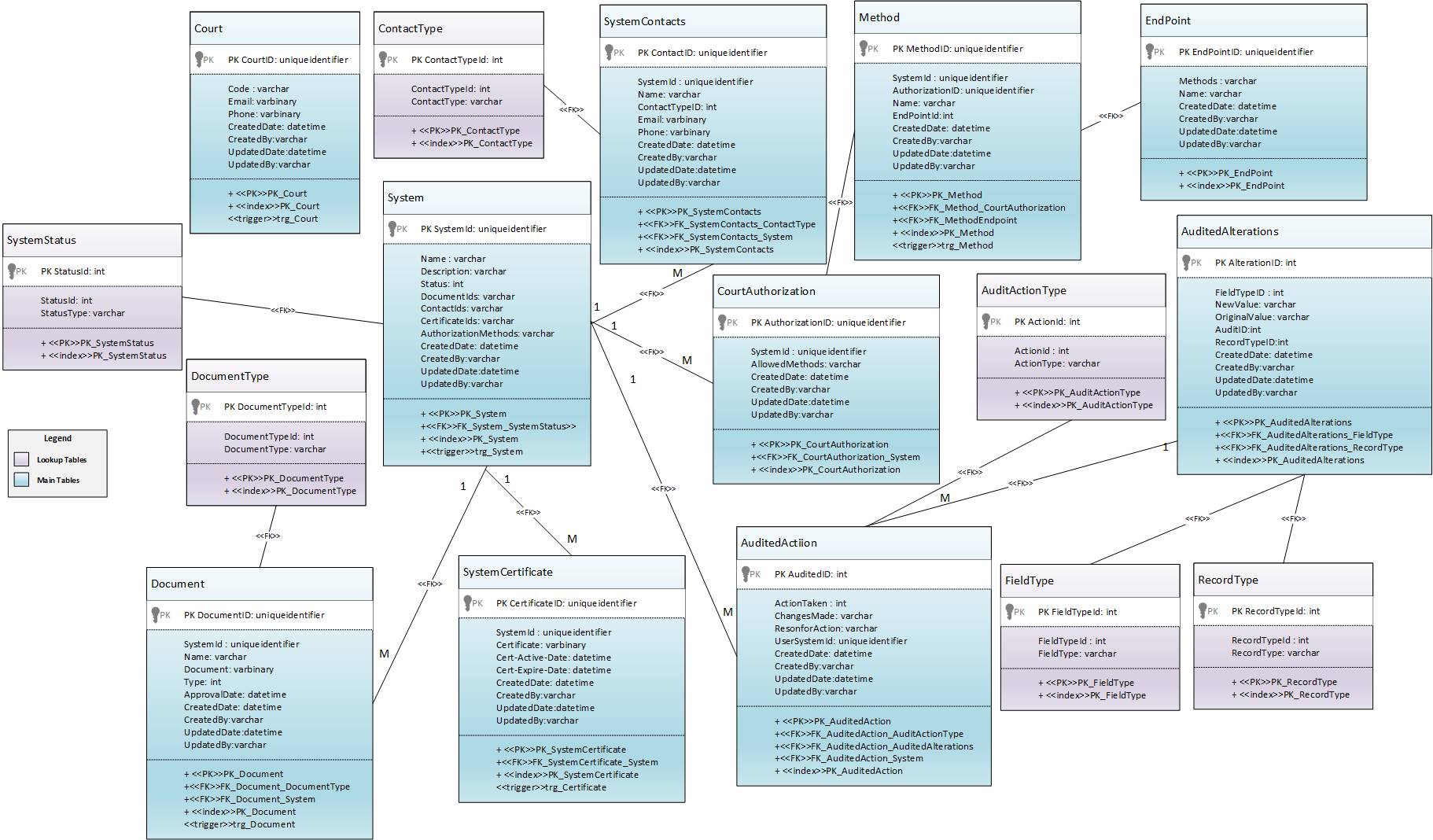
1. Database

This section outlines the database model for eAPI Administration Portal, which is designed to manage System, documents, certificates and contacts. The database module ensures data integrity, performance and scalability.

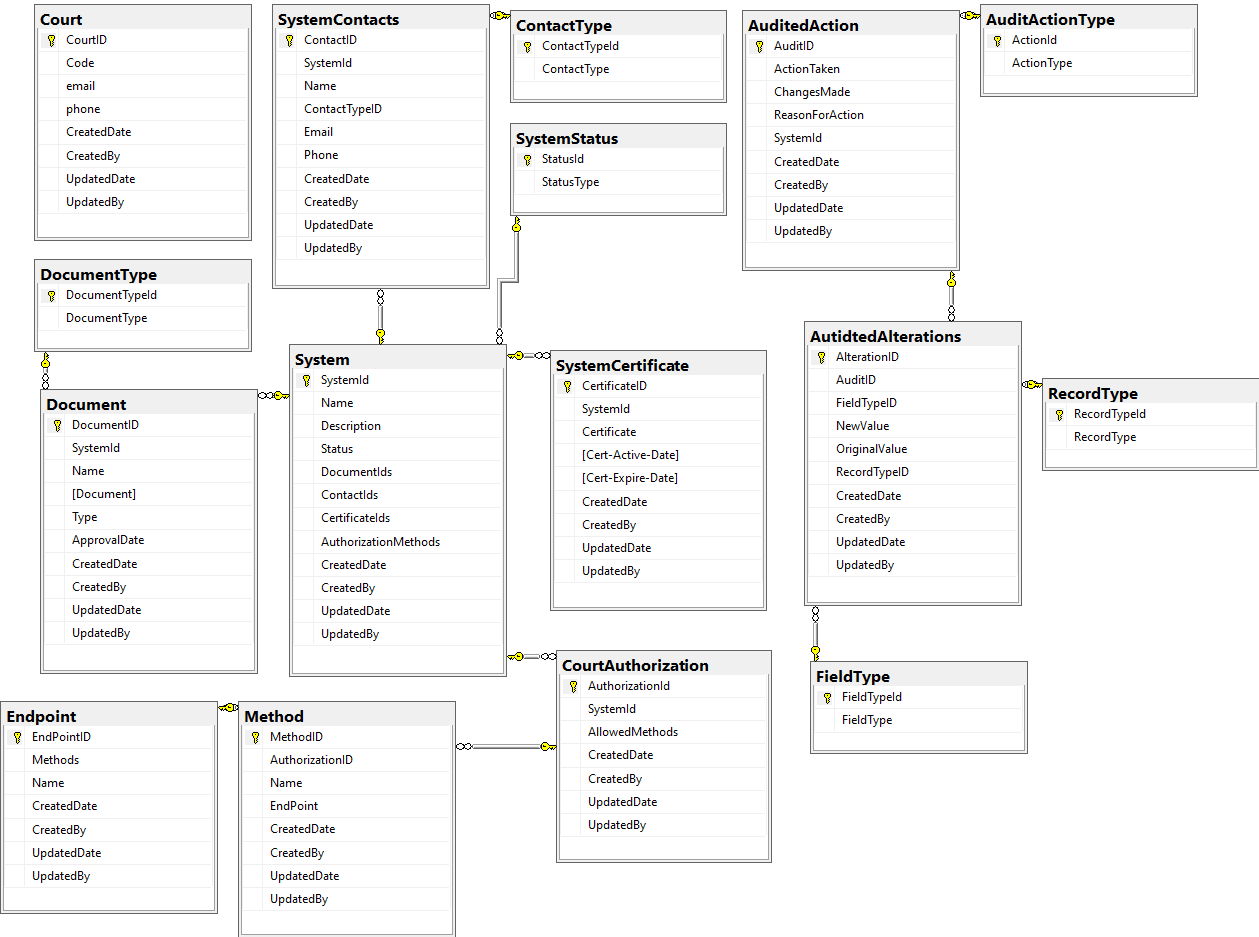
The database must support the following requirements:

* Store information about Systems, Documents, Certificates, and Contacts.
* Ensure data integrity and relationships between tables.
* Support high volume transactions and fast query performance.
* Provide data security and compliance with relevant standards.

The following UML diagram will provide the logical model of the eAPI database that includes entities, attributes and relationships.



* 1. Tables



The above entity relationship diagram depicts the actual eAPI database tables and columns which is required for Administration Portal.

* 1. Triggers

N/A

* 1. User Defined Functions

N/A

* 1. Stored Procedures



* + 1. Sp\_GetSystems

This procedure retrieves all the 3rd party systems details exists in the eAPI database.

* + 1. Sp\_GetSystem

This procedure retrieves the 3rd party system for the given system id.

* + 1. Sp\_System\_Insert

This procedure will insert the new system record to the System table. This system record will be public to the 3rd party systems. The stored procedure records the current date and time as the created time and records the user who created that system record.

* + 1. Sp\_System\_Update

This procedure will update the existing system record to the System table. This system record will be public to the 3rd party systems. The stored procedure records the current date and time as the updated time and records the user who update that system record

* + 1. Sp\_System\_Delete

This procedure will delete the existing system record from the System table.

* + 1. Sp\_System\_Suspend

This procedure will update the existing system record status as “In-Active” in the System table.

* + 1. Sp\_Document\_Insert

This procedure will insert the new document to the Document table. The stored procedure records the current date and time as the created time and records the user who created that document record.

* + 1. Sp\_Document\_Update

This procedure will update the existing document to the Document table. The stored procedure records the current date and time as the updated time and records the user who updated that document record.

* + 1. Sp\_Document\_Delete

This procedure will delete the existing document record from the Document table.

5.4.10 Sp\_Certificate\_Insert

This procedure will insert the new certificate record to the System Certificate table. The stored procedure records the current date and time as the created time and records the user who created that certificate record.

5.4.11 Sp\_Certificate\_Update

This procedure will update the existing certificate record to the System Certificate table. The stored procedure records the current date and time as the updated time and records the user who updated that certificate record.

5.4.12 Sp\_Certificate\_Delete

This procedure will delete the existing certificate record from the System Certificates table.

5.4.13 Sp\_Contact\_Insert

This procedure will insert the new contact record to the System Contact table The stored procedure records the current date and time as the created time and records the user who created that contact record.

5.4.14 Sp\_Contact\_Update

This procedure will update the existing contact record to the System Contact table The stored procedure records the current date and time as the updated time and records the user who updated that contact record.

5.4.15 Sp\_CourtAuthorization\_Insert

This procedure will insert the new authorization record to the CourtAuthorization table The stored procedure records the current date and time as the created time and records the user who created that authorization record.

5.4.16 Sp\_CourtAuthorization \_Update

This procedure will update the existing authorization record to the CourtAuthorization table The stored procedure records the current date and time as the updated time and records the user who updated that authorization record.

5.4.17 Sp\_Method\_Insert

This procedure will insert the new method record to the Method table The stored procedure records the current date and time as the created time and records the user who created that method record.

5.4.18 Sp\_Method\_Update

This procedure will update the existing method record to the Method table The stored procedure records the current date and time as the updated time and records the user who updated that method record.

5.4.19 Sp\_EndPoint\_Insert

This procedure will insert the new endpoint record to the Endpoint table The stored procedure records the current date and time as the created time and records the user who created that endpoint record.

5.4.20 Sp\_EndPoint\_Update

This procedure will update the existing endpoint record to the Endpoint table The stored procedure records the current date and time as the updated time and records the user who updated that endpoint record.

5.4.21 Sp\_Court\_Insert

This procedure will insert the new court record to the Court table The stored procedure records the current date and time as the created time and records the user who created that court record.

5.4.22 Sp\_Court\_Update

This procedure will update the existing court record to the Court table The stored procedure records the current date and time as the updated time and records the user who updated that court record.

5.4.22 Sp\_GetCourts

This procedure will retrieve all the Court records from the Court Table.

5.4.23 Sp\_AuditAction\_Insert

This procedure will take all information to create an audit record including the changed data. The changed

Data is passed to create record in Audit Alteration table. It ensures the admin user with the display name given and then creates the audit action record.

5.4.24 Sp\_SerachAuditAction

This procedure returns the audited actions and their change data in two result sets. If system id is specified, it is to bring back all audited action.

* 1. Dependencies

The database relies on encryption at rest. Database-level encryption must be enabled on the SQL Server configuration of the database when the databases are created by database administrators.

* 1. Assumptions and Limitations

Currently, the database is expected to contain roughly a million records for 3rd Party systems. As each of upload of a system and its associated documents, certificates and contacts is likely to create at least one audit record, the audit tables assumed to be at least double size required for the application.

* 1. Scripts

To create the database, there will be a eAPIAdminPortalDatabase.sql that will provide a script to create the necessary database structure for the system in AOC-hosted database. This script will include population of the domain data required for the application.

A separate eAPIAdminPortalDatabase.sql will provide a script to update the database structure from the baseline database structure to the most current version of the database in an AOC-hosted database. The script will include population of updated domain data required for the application.

* 1. Domain Data

The following tables will hold the domain data and serve as look up tables.

|  |  |  |
| --- | --- | --- |
| ID | Look Up Tables | Description |
| 1 | System Status | List of all System Status |
| 2 | Contact Type | List of all Contact Types associated to System |
| 3 | Document Type | List of all Document Types associated to System |
| 4 | Field Type | List of all Field Types associated to Audited Alterations |
| 5 | Record Type | List of all Record Types associated to Audited Alterations |
| 6 | Audit Action Type | List of all actions that can be taken in the system |

# Special Considerations

* 1. Volume Considerations

The system must be designed to scale for a higher load on uploading cases via the AOC-Admin Portal.

* 1. Testing Considerations

Testing will need to consider both directly testing the appropriate service calls and testing the admin portal to ensure data validity. The integration with the on-premises Active Directory means that testers may need test accounts configured to “act as” ADMIN for proper testing using browsers in private browsing modes. All the related data in configurations entries will need to be configured to match the QA team’s needs. For data integrity testing the testers will need DB access.

* 1. Production Support Considerations

There are no production support considerations.

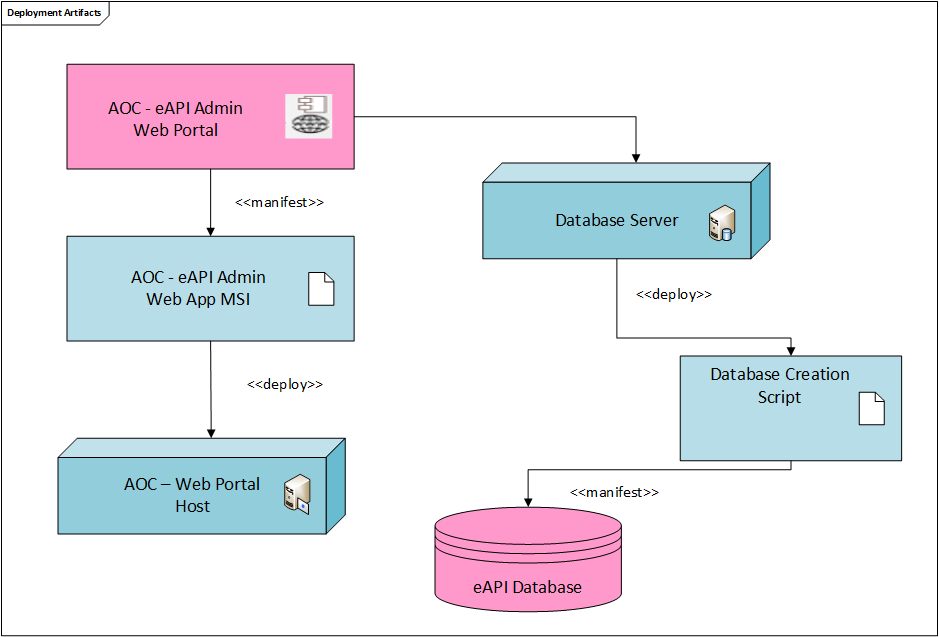
* 1. Performance & Response Time

All Admin portal processes are being designed to meet the 10 second response time standard by showing an intermediate status screen that is immediately shown to the user. That status screen will update the user on the progress of the upload. Once the task is done processing, the user will be met with a final summary screen as summarized in the functional specification.

Additionally, all user requests should present a screen overlay confirming the request is being processed. This overlay should be present on all click or load events.

* 1. Setup and Deployment Strategy

The following deployment diagram depicts the necessary deployment artifacts and the appropriate deployment targets.



1. Glossary/ Definitions

Below is a list of common terms and their definitions that are used throughout this document:

| Term | Definition |
| --- | --- |
| eIdP | External Identity Provider |
| DAL | Data Access Layer |
| SMTP | Simple Mail Transfer Protocol |
| eAPI | External Application Programming Interface |
| AD | Active Directory |

1. Related Documents/References

| Document | Document Location |
| --- | --- |
| Functional Specification 1 | <https://wacourts.sharepoint.com/:w:/r/teams/AOCIntegrationPlatformTeamProject/Shared%20Documents/General/Functional%20Specification%20%E2%80%93%20External%20API.docx?d=wbae1269753c84c69a26ebfd43e8ad20a&csf=1&web=1&e=qc2hYJ> |
| Functional Specification 2 |  |
| COTS Documentation |  |
| Azure Application Insights Documentation | <https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview?tabs=net> |
| Azure Key Vault Documentation | <https://learn.microsoft.com/en-us/azure/key-vault/> |
| Azure App Configuration Documentation | <https://learn.microsoft.com/en-us/azure/azure-app-configuration/> |

1. Document Change History

| Version No. | Date | Name (Alias) | Description of Change |
| --- | --- | --- | --- |
| 1.0.1 | 6/9/2024 | Amiya | Initial drafting. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Review and Sign-off

Below is a list of the project team members and required reviewers

**Revision 1.0.1**

|  |  |  |
| --- | --- | --- |
| Person | Role | Approval Evidence |
| Scott Gibson | Solution Architect |  |
| Winnie Sheeram | QA Lead |  |
|  | Production Support Manager |  |
| Development Lead | Amiya Ranjan Mohnanty |  |

1. <https://learn.microsoft.com/en-us/aspnet/core/security/key-vault-configuration?view=aspnetcore-6.0#secret-storage-in-the-production-environment-with-azure-key-vault> [↑](#footnote-ref-2)