

ProjectName

Application Architecture

Standard Architecture and Dependencies

Table of Contents

[2 Introduction 3](#_Toc425768024)

[2.1 What is this? 3](#_Toc425768025)

[2.2 Why is Architecture Documentation necessary? 3](#_Toc425768026)

[3 SaaS Software as a Service 3](#_Toc425768027)

[4 Technologies and Dependencies 3](#_Toc425768028)

[5 Standard Architecture Layers 4](#_Toc425768029)

[5.1 Design principles used 4](#_Toc425768030)

[5.2 Common Layers that divide project concerns 4](#_Toc425768031)

[5.2.1 Business Layer 5](#_Toc425768032)

[5.2.2 Common Layer 5](#_Toc425768033)

[5.2.3 Data 5](#_Toc425768034)

[5.2.4 DataAccess 5](#_Toc425768035)

[5.2.5 Resources 5](#_Toc425768036)

[5.2.6 Web 5](#_Toc425768037)

[6 High level diagrams 6](#_Toc425768038)

# Introduction

## What is this?

This document attempts to provide a high level overview of the standard project architecture of AFS Technologies development initiatives in the modern technological landscape.

## Why is Architecture Documentation necessary?

To give stakeholders and vested parties a comprehensive overview of the approach and technologies used to provide rich, purpose built software and services.

* To help identify key areas, communication points, single and cross cutting concerns that affect the development from a macro point of view.
* To provide views of different aspects of the systems which capture and convey the significant architectural elements and how they relate or interact.

# SaaS Software as a Service

The standard distribution model for software at AFS is Software as a Service which frees the user base from complex hardware and software installations and allows AFS to react quickly on a global scale to changing business needs. Simply connect to the software using a modern web browser.

# Technologies and Dependencies

AFS Project Architectures utilize Microsoft technologies, frameworks, coding languages, databases and security models to create structured solutions that facilitate rich functionality that is scalable, secure and maintainable. Projects also leverage modern frameworks and toolkits such as Telerik Kendo MVC, Bootstrap, Font Awesome, jQuery which helps us deliver global solutions with rapid time to value.

* Microsoft .Net Frameworks
* Microsoft SQL Server
* Microsoft IIS
* Microsoft C#
* Microsoft ASP.NET MVC
* Microsoft Entity Framework
* Bootstrap HTML, CSS and JS framework
* Font Awesome font and CSS toolkit
* HTML, HTML5
* jQuery
* Javascript
* Kendo MVC
* NLog
* OpenXML
* ClosedXML

# Standard Architecture Layers

The Standard Architecture layers utilize design principles to modularize areas, to minimize communication points and exposure. This helps to reduce complexity, redundancy, enforce security and make applications resilient to changing business requirements and needs.

## Design principles used

* Separation of concerns.
  + Divide applications into distinct features with minimal overlap.
  + Minimize communication surface areas
  + Decouple as much as possible
* Single responsibility principle.
  + Each module should address a single specific feature or function.
* Principle of least knowledge.
  + An object shouldn’t need to know the internals of another object.
* DRY principle (Don’t Repeat Yourself).
* Least Privilege Principle.
  + Only give the component or user the minimum access it/they require to accomplish the task.

## Common Layers that divide project concerns

* Business
* Common
* Data
* DataAccess
* Resources
* Web
* Security

### Business Layer

The intent of the Business Layer is to cleanly separate the business rules or concerns from other layers of the project. By doing this we have centralized the rules that will ensure proper interaction with data.

### Common Layer

Is used to provide common functionality that spans entities and can be used for crosscutting concerns that affect the entire application. Logging for instance could be a crosscutting concern since we may want to log things in the distinct layers without having code duplicated in some other layer to achieve the same thing i.e. DRY Principle (Don’t Repeat Yourself). Some other examples of crosscutting concerns are but are not limited to Validation, Security, Helper functions, Messaging, Exception Handling and Instrumentation.

### Data

The Data layer is used for all of the Create, Read, Update and Delete methods.

### DataAccess

This layer encapsulates the database communication and serves to cleanly separate the concerns of connecting to various database providers and the communication between those entities.

### Resources

The intent of Resources is to provide a centralized way to specify commonly used text for labels/controls so that it is ubiquitous and not duplicated.

### Web

This layer contains the functionality responsible for User Interaction. This typically consists of the Model, View and Controller elements of the ASP.NET MVC programming model.

# High level diagrams

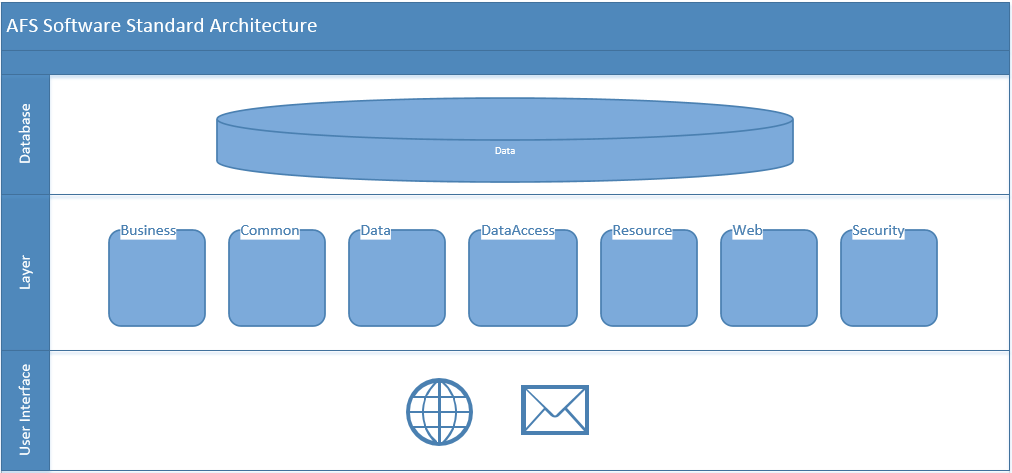


Figure : High level overview

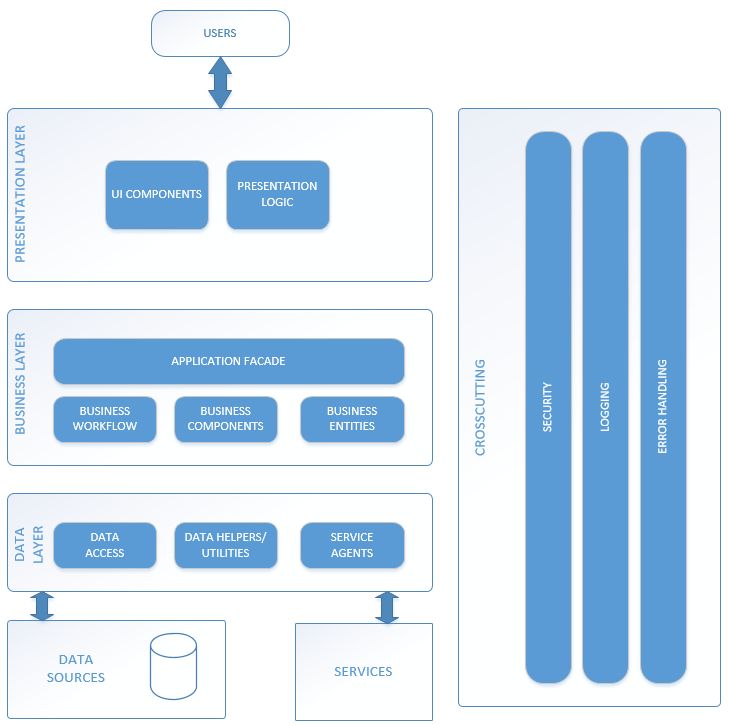


Figure High level architecture with crosscutting concerns