In House CCM Solution Architecture

Title

|  |  |
| --- | --- |
| **Date**: | 2018-03-01 |
| **Author:** | Jim Deane |

# Table of Contents

[Table of Contents 2](#_Toc508137554)

[1 Document Control 4](#_Toc508137555)

[1.1 Document History 4](#_Toc508137556)

[2 Context 5](#_Toc508137557)

[2.1 Background 5](#_Toc508137558)

[2.2 Scope 5](#_Toc508137559)

[2.3 Declaration 5](#_Toc508137560)

[2.4 Cost 5](#_Toc508137561)

[2.5 Note 5](#_Toc508137562)

[3 Solution Context 6](#_Toc508137563)

[3.1 Overview / Vision 6](#_Toc508137564)

[3.2 Future Architecture 6](#_Toc508137565)

[3.3 As-Is Architecture 7](#_Toc508137566)

[3.4 Transition Planning 7](#_Toc508137567)

[4 Architecture Goals and Constraints 8](#_Toc508137568)

[4.1 Enterprise Architecture Alignment 8](#_Toc508137569)

[4.2 Architectural Assumptions and Decisions 8](#_Toc508137570)

[4.3 Solution Architecture Principles and Constraints 9](#_Toc508137571)

[4.3.1 Principles 9](#_Toc508137572)

[4.3.2 Constraints 9](#_Toc508137573)

[4.4 Solution Architecture Attributes 9](#_Toc508137574)

[4.4.1 Technology 9](#_Toc508137575)

[4.4.2 Pattern 10](#_Toc508137576)

[4.4.3 Common Services 10](#_Toc508137577)

[4.4.4 Common Components 10](#_Toc508137578)

[4.4.5 Licensing / Commercials? 10](#_Toc508137579)

[4.4.6 Other Solution Architecture Attributes 10](#_Toc508137580)

[4.5 Non-Functional Requirements 11](#_Toc508137581)

[4.5.1 Portability 11](#_Toc508137582)

[4.5.2 Capacity and Performance 11](#_Toc508137583)

[4.5.3 Availability and Reliability 11](#_Toc508137584)

[4.5.4 Scalability 11](#_Toc508137585)

[4.5.5 System Management, Monitoring, Auditing and Administration 11](#_Toc508137586)

[4.5.6 BC/DR and COOP 11](#_Toc508137587)

[4.5.7 Legal and Compliance 11](#_Toc508137588)

[5 Solution Architecture 12](#_Toc508137589)

[5.1 Component Models 12](#_Toc508137590)

[5.1.1 Components and their Relationships 12](#_Toc508137591)

[5.2 Risks 22](#_Toc508137592)

[6 Data Architecture 23](#_Toc508137593)

[6.1 Data Model 23](#_Toc508137594)

[6.2 Authoritative Data Sources 24](#_Toc508137595)

[6.3 Data Migration Guidance 25](#_Toc508137596)

[7 Costs 26](#_Toc508137597)

[7.1 Overview 26](#_Toc508137598)

[7.2 Basis for Estimation 26](#_Toc508137599)

[8 Security Architecture 27](#_Toc508137600)

[8.1 Security Overview 27](#_Toc508137601)

[9 Infrastructure 28](#_Toc508137602)

[9.1 Deployment Models 28](#_Toc508137603)

[9.2 Failover / resilience 28](#_Toc508137604)

[10 Testing Considerations 29](#_Toc508137605)

[10.1 Unit testing 29](#_Toc508137606)

[10.2 Integration testing 29](#_Toc508137607)

[11 Appendices 30](#_Toc508137608)

[11.1 Appendix A – References 30](#_Toc508137609)

[11.2 Appendix B - Key Terms 31](#_Toc508137610)

[11.3 Appendix D – Security 32](#_Toc508137611)

[11.3.2 Data Security 32](#_Toc508137612)

[11.3.3 Access Control 34](#_Toc508137613)

[11.3.4 Security Information Management 35](#_Toc508137614)

[11.3.5 Threat & Vulnerability Management 36](#_Toc508137615)

[11.4 In House comparison with RFP 37](#_Toc508137616)

# Document Control

## Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Summary of changes |
| 0.1 | 2018-03-01 | Jim Deane | New Document |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Context

## Background

This document is an outline Architecture Design document for a possible in-house development of a Customer Communication System that could, at least partially, provide the functionality of an external vendor’s product such as *Quadient* or *Open Text*; both of which are, at the time of writing, being considered for purchase.

## Scope

The external vendors solutions are extremely comprehensive, and their cost reflects this. In addition, these vendors both have very mature products that are well tested in the market and are clearly robust and reliable.

An in-house development cannot compete with most of these product properties however it can provide ‘good enough’ feature coverage and, of course, would be comprehensively tested.

The scope of the design is intended to provide for the most important elements of the requirements that were presented to vendors in the Request for Proposal made to the vendors mentioned above.

## Declaration

Much of the original design that is the genesis of this proposed design was created by the author for a previous customer for which the Intellectual Property (IP) was retained by the author. To the extent that it is revealed in this document the author makes no claim on that IP and this design may be freely used by Old Mutual Wealth (OMW) provided no claim is made by OMW on that IP.

The re-use of a previous design has made it possible to provide far more detail than a normal Solutions Architecture document would be able to do and enables the provision of an order of magnitude of the potential cost.

## Cost

Section ‎7, Costs(Page 26) provides an outline of the potential costs based on a previous project of similar nature and an estimate of the nature of the user stories involved.

## Note

All diagrams in this document are either embedded (Visio) or are extracted from an ArchiMate Model built using the open source Archi system. This model is embedded here:



# Solution Context

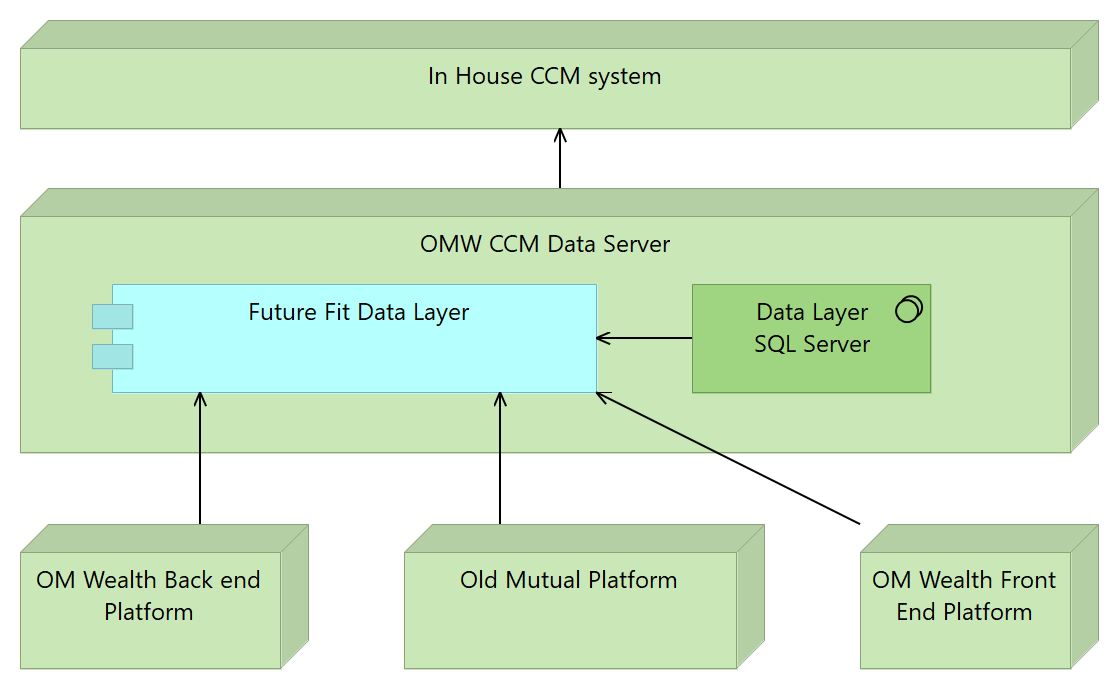
## Overview / Vision

The system described in this document is intended to be a partial implementation of the functionality that is provided by external vendors such as Quadient and Open Text.

## Future Architecture

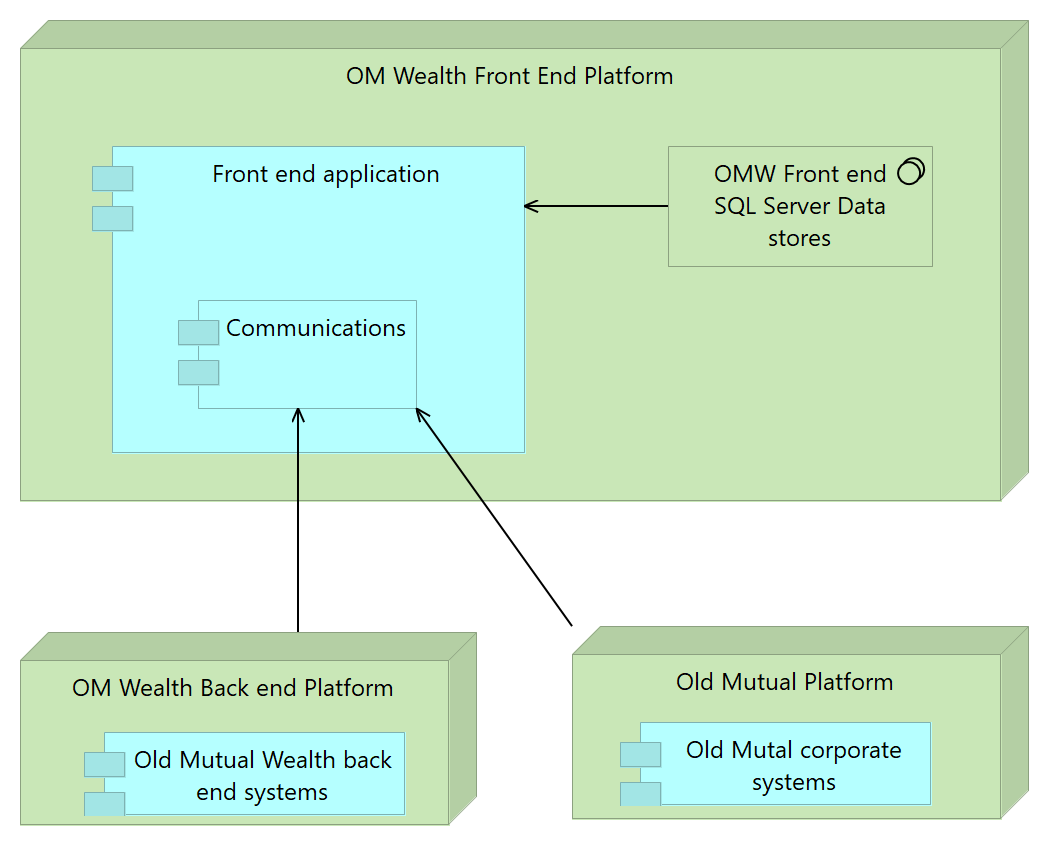
The following diagram illustrates how the proposed system would interact with the current OMW front end systems.

The system as described would be able to be used with any system that can provide a Data Layer that can aggregate data (see Future Fit Data Layer Solutions Architecture Document).



## As-Is Architecture

The current reporting and document production systems for the “Front End”, i.e. customer and advisor facing systems have a simple data architecture that populates data objects as required for specific output templates.



## Transition Planning

The current system depends on the use of standard tools such as Microsoft word and proprietary code. There is no expectation that there will be a migration path from the as is to the to be. Transition will be achieved by completely reworking the current templates and mechanisms within the to be architecture.

This is certain to be performed in phases that are yet to be defined.

# Architecture Goals and Constraints

## Enterprise Architecture Alignment

The architecture and design of this system follows the requirements of Old Mutual (OM) Enterprise Architecture.

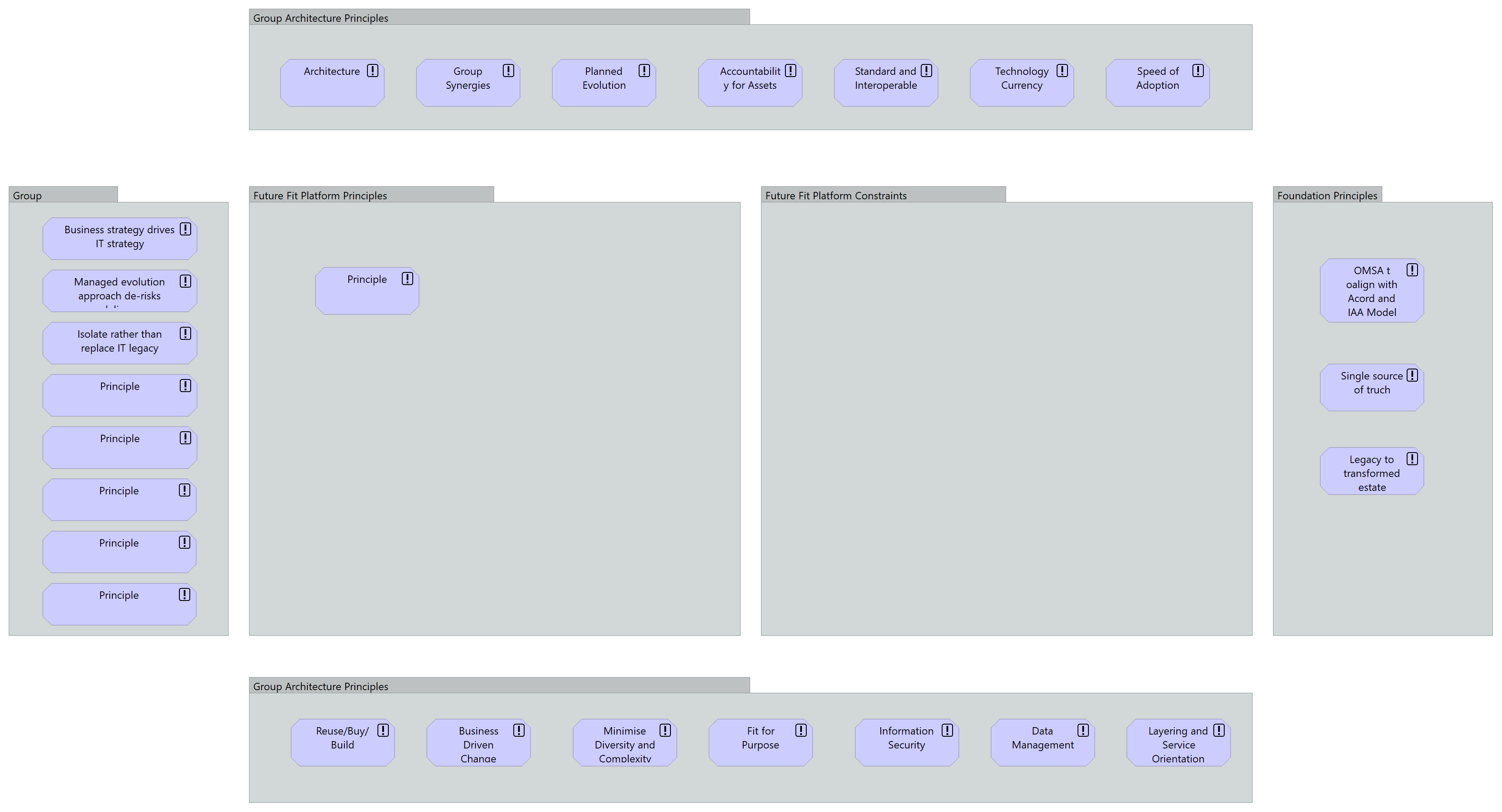
## Architectural Assumptions and Decisions

The architectural design of the Data Layer has, within Enterprise Architecture guidelines, been influenced by the Business goals.

No other assumptions or decisions have yet been made.

## Solution Architecture Principles and Constraints

### Principles



Draft

### Constraints

None so far.

## Solution Architecture Attributes

### Technology

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Vendor** | **Product** | **Version** | **Status** |
| Software | Microsoft | .Net | Core 2.0 |  |
| Software | Microsoft | Entity Framework | Core 2.0 |  |
| Infrastructure | Microsoft | IIS | 7 |  |
| Infrastructure | Microsoft | SQL Server | 2012 or later |  |
| Infrastructure | Microsoft | MSMQ messaging | 6.3 or above |  |
| Libraries | Open Source | GrapesJS <http://grapesjs.com/> | TBA |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table - Hardware and Software Technologies

### Pattern

The construction of the data layer will conform to the following architectural patterns:

* ASP.NET API 2 RESTful services
* ASP.NET MVC for administration, monitoring and dashboards.
* S.O.L.I.D design principles

### Common Services

**IAM Service**

This solution will make use of the IAM services to authenticate users who participate.

Authorisation for administrative access will also be performed through IAM services.

### Common Components

None other than those described in this architecture document.

### Licensing / Commercials?

None, the development of the data layer will be entirely in-house using either standard Microsoft or appropriately licensed open source libraries requiring no additional licensing.

### Other Solution Architecture Attributes

None.

## Non-Functional Requirements

### Portability

The design calls for the use of Microsoft .NET Core 2.0 throughout. This technology may be deployed on Windows and Linux should this be required.

However, some of the infrastructure is currently only supported on Windows (IIS, MSMQ, SQL Server) and alternatives for these infrastructure elements would have to be found.

### Capacity and Performance

The system must be able to provide data services that have the capacity to aggregate data for all the known CCM requirements and be scalable for future capacity.

### Availability and Reliability

As for the OMW “Front End” systems.

### Scalability

The data layer must be able to scale with the requirements of the current OMW “Front End” systems.

### System Management, Monitoring, Auditing and Administration

The system must provide the following:

* Detailed logs of all:
* Errors
* Warnings
* Informational logs

at configurable levels of verbosity.

* Provide a dashboard

### BC/DR and COOP

As for the current “Front End” systems.

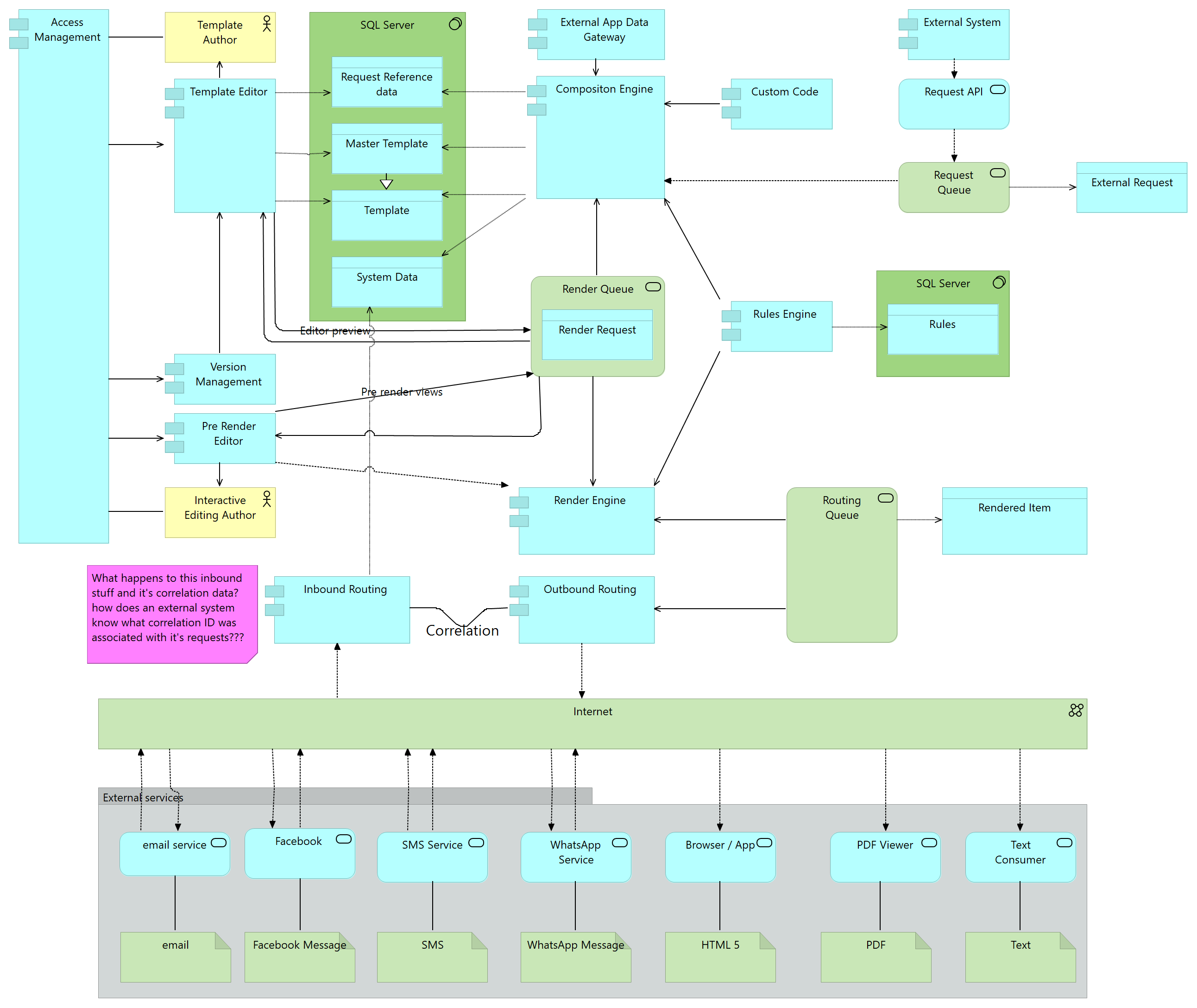
### Legal and Compliance

None.

# Solution Architecture

## Component Models

### Components and their Relationships



Each of the items shown above is described in the following table:

| Item | Description | Item Type |
| --- | --- | --- |
| Access Management | Controls which users and groups have access to specific functions and data within the system.  Access is expected to be federated with the OM IAM system which will provide authentication of a User and triage access to Administrative and Standard Users. | Application Component |
| Browser / App | Either a web browser or an application that is rendering content using HTML / CSS to display that content. | Application Service |
| Data gathering | A component implemented by a calling system to gather data for transmission to this system. | Application Component |
| Composition Engine | This is the core of the system and generates the necessary items for a request.  Section ‎5.1.1.1  Composition Engine *(Page 16)* describes this component in more detail. | Application Component |
| Correlation | Provides the mechanism and data for correlating an outbound message item with a possible return message such as an SMS, email, Facebook message etc. | Data Object |
| Custom Code | This component provides a scaffolding for commonly used or complex functions and rules that can be written by developers and incorporated into the system dynamically.  These functions would be written to a specified Interface in C# (or other CLR language). | Application Component |
| External App Data Gateway | The Data Gateway provides the mechanism for the system to access the data required for merging with templates, evaluating functions and rules. | Application Component |
| External Request | An External Request is the request data object that is passed to the system through an API call to initiate the creation of a set of items defined by the system reference data.  A request consists of:   |  |  | | --- | --- | | Request Type | For example:  POLICY\_WELCOME\_PACK | | External App Data Gateway Keys | A JSON object containing the necessary key values to enable the External App Data Gateway to retrieve the data necessary for the Request Type | | Custom Data | A JSON object containing Name / Value pairs that can be | | Data Object |
| HTML 5 Editor | A drag and drop editor that is used to create the template content.  Section 17HTML Editor *(Page 17)* describes this in more detail. | Application Component |
| Inbound Routing | Receives inbound messages, resolves correlation and stores the content for later querying by the External System. | Application Component |
| Interactive Editing Author | A user who needs to modify output items before rendering to select optional sections and complete free text areas. | Business Actor |
| Master Template | Templates represent the design of the outputs that are to be rendered and distributed. | Data Object |
| Outbound Routing | Takes items from the Routing Queue and finds the External Service that can deliver the item and monitors and reports on the progress and status of deliveries. | Application Component |
| PDF renderer | Mechanism for rendering HTML / CSS as a PDF | Application Component |
| Pre-Render Editor | Interactive editing is the mechanism by which an output items may be presented to a user with various options available to include or exclude nominated blocks in the item as well as optionally manually enter text into nominated free text areas.  Section *‎5.1.1.3 Pre-Render Editor (Page 17)* provides more detail. | Application Component |
| Render Engine | The Render Engine is responsible for rendering items that have been presented on the Render Queue and producing the end format of the item be it PDF, email, SMS etc.  *‎5.1.1.4* Render Engine *(Page 18)* provides more detail. | Application Component |
| Render Queue | Requests processed by the Composition engine are queued for rendering on the Render Queue. The requests on the queues may be served by multiple instances of the Rendering Engine and may have different priorities to influence the order and urgency of the processing. Queueing is provided by MSMQ. | Technology Service |
| Render Request | A Render Request is a request for the rendering Engine to produce a single output which may be a PDF, email, SMS etc. | Data Object |
| Rendered Item | Rendered Items are instances of PDFs, emails, SMSs, HTML documents etc with a specific recipient for each (e.g. copies are rendered into separate items). | Data Object |
| Request API | The Request API provides the sole mechanism for requesting the system to perform its functions. It is secured and ensures that the entire system and its internal functions are opaque to the caller. | Application Service |
| Request Queue | Calls to the Request API are queued for processing on the Request Queue. The requests on the queues may be served by multiple instances of the Composition Engine and may have different priorities to influence the order and urgency of the processing. Queueing is provided by MSMQ. | Technology Service |
| Request Reference data | The systems own tables for directing processing, logs, audit trails and history | Data Object |
| Routing Queue | Items rendered by the Rendering engine are queued for output on the Routing Queue. The requests on the queues may be served by multiple instances of the Routing Engine and may have different priorities to influence the order and urgency of the processing. Queueing is provided by MSMQ. | Technology Service |
| Rules | Rules provide the mechanism for decision making in the system. The rules and the Rules Engine are described in section ‎5.1.1.5  *Rules and* Rules Engine *(Page 20).* | Data Object |
| Rules Engine | Evaluates Rules. The rules and the Rules Engine are described in section *‎0‎5.1.1.5*  *Rules and* Rules Engine *(Page 20).* | Application Component |
| Template | Templates are HTML documents with associated CSS for each of the output formats that are to be supported for the item.  Within Templates data may be inserted or used to evaluate rules and functions.  Section ‎5.1.1.6Templates *(Page 20)* provides more detail. | Data Object |
| Template Author | Template authors can use the template editing tools with variable levels of access according to the user and role permission defined on a template. | Business Actor |
| Template Editor | For creating and editing templates. Described here Template Editor.Templates are designed in a hierarchy so that portions of a template can be locked by an author allowing other editors to modify those portions. Some portions may be modified by Interactive editors. | Application Component |
| Version Management | All templates are stored with a version that is linked to the request at the time it was made to ensure that when reconstructing an output item is associated with the same template(s) with which it was created. | Application Component |

#### Composition Engine

The Composition Engine is responsible for determining which output items are required to be produced for each External Request that is made.

Each external request may have associated with it an arbitrary number of output items for example:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| POLICY\_WELCOME\_PACK | |  |  | | --- | --- | | Email to Policyholder | Welcome email containing a link to the system document repository containing the Policy schedule etc. | | Email to the advisor | Informing the advisor that the policy welcome pack has been created. | | SMS | To the Policyholder confirming the policy created | | PDF document | The policy schedule itself stored in the document repository. | |

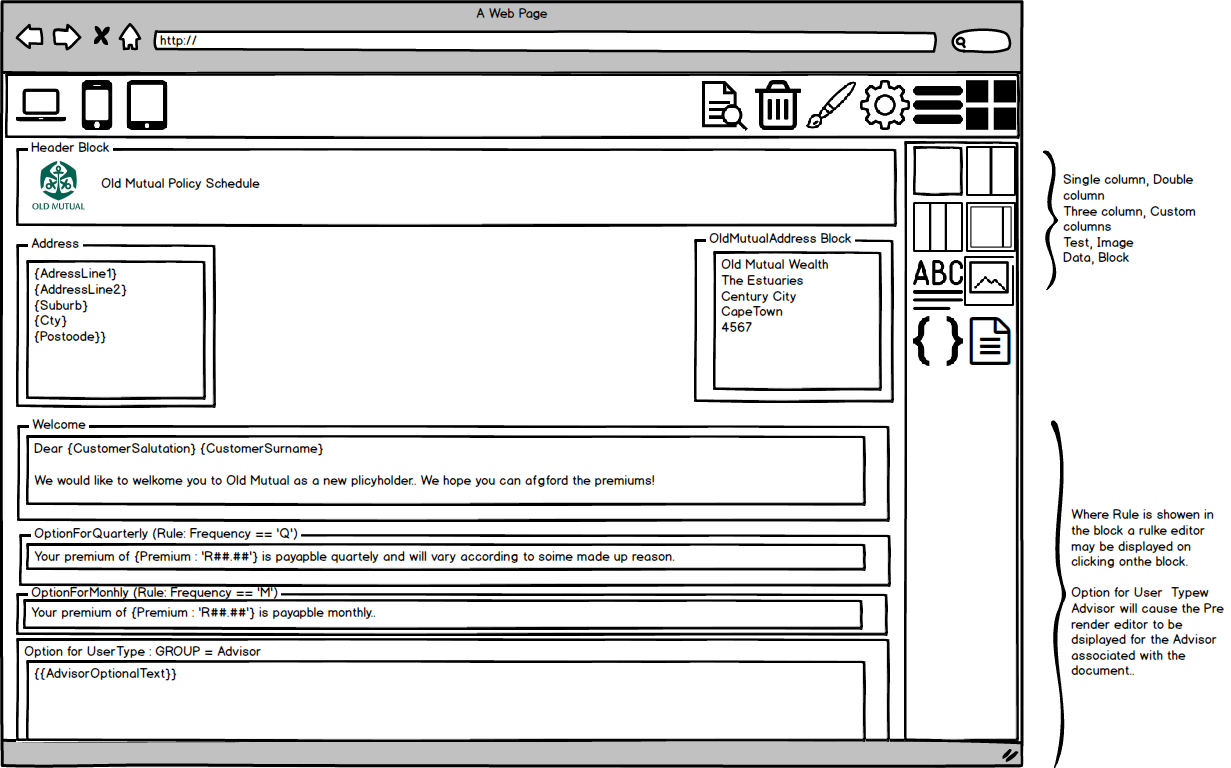
Note, the documents stored in the repository with a link may equally be sent as attachments to the email(s).

#### HTML Editor

This editor is based on an opens source project, [GrapesJS](http://grapesjs.com/) that provides a drag and drop HTML / CSS designer. The designer is extensible and produces pure HTML / CSS that can be used as input to the Composition engine and thence to the Rendering Engine.

Data for insertion is inserted as a dragged element within which the desired data item can be selected.

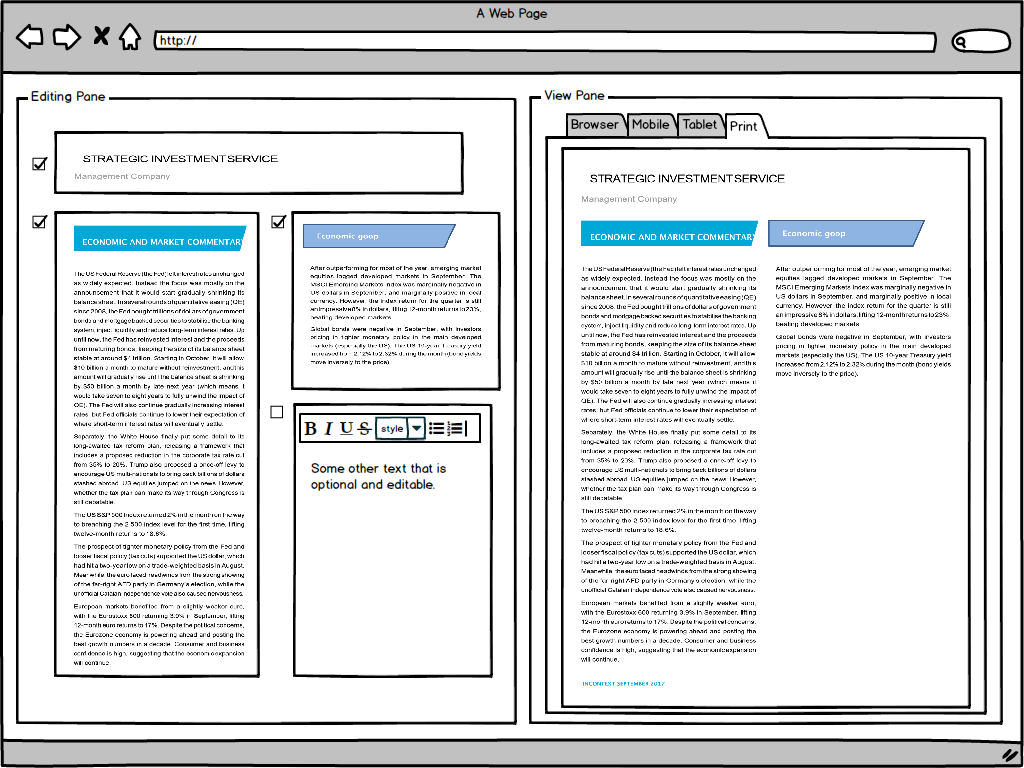
The following is a mock-up of part of the user interface that would be seen by a template editor.



Further UI mock-ups to be constructed.

#### Pre-Render Editor

The following mock-up shows the style of page that will be shown to a user that is



The pre-rendering editor is intended for the use of certain users who are given the ability to

* take items that have been composed by the Composition Engine and elect to show or hide various page elements that have been made selectable in the template editor
* create free text in predefined areas

If these selectable or free text areas are defined, then items are delivered to either a specified user or group and must be viewed before they are returned to the Render queue to be picked up by the rendering engine.

#### Render Engine

The Render Engine takes items from the Render Queue and transforms them from raw HTML / CSS to the destination output format.

The mechanism for this depends on the destination output format itself. The system design allows for new rendering functions to be added dynamically provided that they can take HTML / CSS as the raw input format.

Core rendering functions would be:

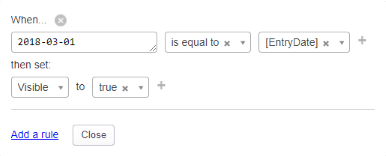
| Output Format | Description |
| --- | --- |
| PDF | PDFs will be rendered using the Print Device CSS Type. (See <https://www.w3.org/TR/css3-page/> for details of the functionality allowed for page-based devices using CSS).  Actual rendering will be achieved using the Headless Chrome application which allows an external application to drive Google Chrome to produce any supported CSS Device output in the background (in a separate process). Chrome has the most complete implementation of CSS devices. |
| HTML | Passed directly through to the output. |
| Text | Passed directly through to the output but with any extraneous HTML tags removed. |
| Messages | As for text. |
| Email | Either as HTML or Text dependant on the output. |

#### Rules and Rules Engine

There are three types of rules that the system is specified to cater for:

* Rules embedded in a template, for example a rule that defines whether a block in a template should be rendered.
* Rules that are common among many templates and are defined separately and outside of any template.

Embedded rules are always defined using a Rule Builder such as:



Common rules are defined as C# expressions and are stored as custom code and may be referred from anywhere that a rule is required.

#### Templates

Templates consist of blocks of HTML and associated CSS that can be rendered into the necessary output format by the rendering engine.

## Risks

There are multiple risks to implementing this design:

* This is a speculative design and therefore the resulting system may not provide sufficient coverage of features to satisfy the real user requirements.
* The scope of the system cannot expect to cover all the features and ongoing road map of a vendor system.
* Any estimates made for the implementation of the system are likely to be unreliable until further detailed design and specification work can be carried out.
* The implementation of the system may not be as scalable and performant as a vendor solution.
* The in-house development and implementation of the system will become the responsibility of OMW and the maintenance of the system will then rely on the availability of personnel that have knowledge of the design and implementation of the system.

# Data Architecture

## Data Model

The following diagram illustrates the important parts of the data model used for this design:



The entities in this model are as follows:

| Entity | Description |
| --- | --- |
| Workflow | This table holds all of the allowable types of External Request that may be made of threw system for the production of output items.  An example would be POLICY\_WELCOME\_PACK which may produce an email, several pages of policy schedule, an alerting SMS as well as an informative email to the Advisor. |
| WorkflowAction | This table provides for multiple steps to be defined for a workflow. For example, where a message sent normally awaits a reply the result of receiving the reply may be defined as a step. |
| EventType | There is one to one relationship between WorkflowAction and EventType. |
| DocumentTypeForEventType | This table lists, for each EventType, the DocumentTypes that are to be produced for that event type. |
| NotificationTypeForEventType | This table lists, for each EventType, the NotificationTypes that are to be produced for that event type. |
| DocumentType | The DocumentType defines the formatting and output type.  This definition holds, for example:   * Master Template name * The output type to be rendered * The data required for the document to be retrieved from the calling system. |
| NotificationType | Notifications describe the type(s) of notifications, for example, SMS or email required to be sent. The content of the notification is defined with its own DocumentType and the mechanism for delivering the notification, for example SMS provider is also defined. |
| AttachementForNotificationTypeAndEventType | Notifications may have attachments and this table defines the DocumentType(s) for each NotificationType that nay be attached. Only allowed for emails. |
| DocumentRequestLog | A log of all requests made. |
| DocumentRequestStatusType | Internal look-up table |
| InteractionStatusType | Internal look-up table |
| InteractionSubType | Internal look-up table |
| InteractionType | Internal look-up table |
| InteractionTypeForInteractionSubType | Internal look-up table |
| NotificationRequestStatusType | Internal look-up table |
| SnapshotType | Internal look-up table |
| TransportType | Internal look-up table |
| WorkflowActionAssignedToType | Internal look-up table |
| WorkflowActionCompletionType | Internal look-up table |
| ClientInteraction | A log of every external output made and its relationship to requests made along with the status of the output success. |
| Document | A record of every document output produced along with a reference to the actual output stored for later retrieval. |
| DocumentRequest | A log of every document processed. |
| Event | A log of every event processed. Events are recorded as a result of WorkflowActions being processed. |
| NotificationRequest | A log of every notification processed. |
| NotificationRequestLog | A log of the progress of delivering notifications. |
| Snapshot | The data retrieved from the external system that was used to process the notification or document. |

## Authoritative Data Sources

The system depends on two sources of data:

* The internal reference data and tables described in the data model above AND
* The data that is supplied through the External App Gateway

These are considered the only authoritative data sources.

## Data Migration Guidance

There is no expectation that current methods of document production will be able to be migrated to this platform.

# Costs

## Overview

TBA after discussion

## Basis for Estimation

TBA

# Security Architecture

## Security Overview

The system will conform to the Old Mutual security architecture.

# Infrastructure

## Deployment Models

To be defined in delivery phase.

## Failover / resilience

As per the “Front End” systems.

# Testing Considerations

## Unit testing

Standard automated unit testing methods will be used.

## Integration testing

As the system is designed to be decoupled from both the data sources and the consuming systems Integration testing can be performed in two phases:

* Each component shown ‎5 Solution Architecture (Page 12) may be tested in isolation using automated methods with predetermined test input and output data on every check-in in a continuous integration environment.
* End to end testing can be performed on a less frequent basis than the above using known data source states and predetermined test input and output data.

# Appendices

## Appendix A – References

Table A.1 below summarizes the documents referenced in this document.

|  |  |
| --- | --- |
| Appendix | Location |
| ‎11.2 Appendix B - Key Terms | ‎11.2, Page 31 |
| ‎11.4 In House comparison with RFP | ‎11.3, Page 32 |

Table A.1: References

## Appendix B - Key Terms

Table B.1 below provides definitions and explanations for terms and acronyms relevant to the content presented within this document.

|  |  |
| --- | --- |
| **Term** | **Definition** |
|  |  |

Table B.1 - Appendix B: Key Terms

## Appendix D – Security

#### Reference Architecture

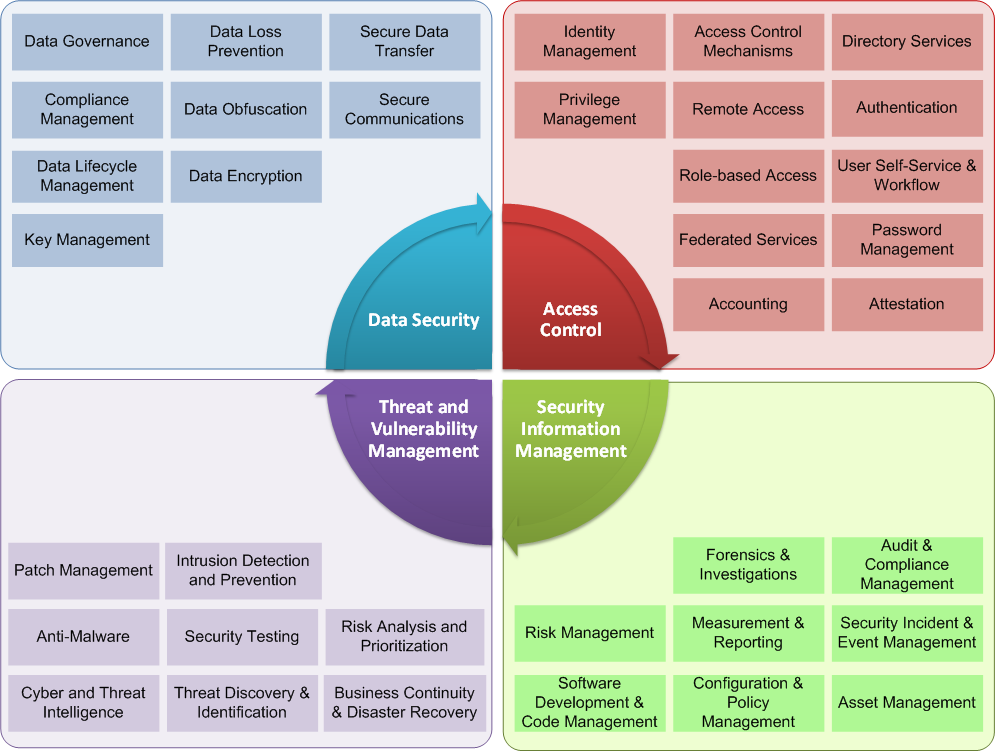


Figure 1: Security Architecture Reference Model

The sections below address each component in the Security Architecture Reference Model or define where the component is addressed in additional documentation.

### Data Security

#### Data governance

According to OM standards.

#### Compliance Management

No further data compliance is necessary for the data layer as the source data is already controlled according to the compliance rules for the source data holding systems.

#### Data Lifecycle Management

There are two possible data lifecycle management implications of the data layer depending upon whether a repository is maintained.

|  |  |
| --- | --- |
| With a repository | If a repository is used to persist the data then the data will be retained until a, yet to be defined, archiving principle is applied. |
| Without a repository | All data is effectively transient within the data layer. |

#### Key Management

Not applicable.

#### Data Loss Prevention

As per “Front End” systems.

#### Data Obfuscation

Not applicable.

#### Data Encryption

Not applicable (other than any encryption provided as standard for SQL server data).

#### Secure Data Transfer

Not applicable.

#### Secure Communications

Not applicable.

### Access Control

#### Identity management

No applicable as this system is only accessed by internal systems only.

#### Privilege management

No applicable as this system is only accessed by internal systems only.

#### Access control mechanisms

No applicable as this system is only accessed by internal systems only.

#### Remote Access

No applicable as this system is only accessed by internal systems only.

#### Role based Access

No applicable as this system is only accessed by internal systems only.

#### Federated Service

No applicable as this system is only accessed by internal systems only.

#### Accounting / Auditing

No applicable as this system is only accessed by internal systems only.

#### Directory Services

No applicable as this system is only accessed by internal systems only.

#### Authentication

No applicable as this system is only accessed by internal systems only.

#### User Self-Service Workflow

No applicable as this system is only accessed by internal systems only.

#### Password Management

No applicable as this system is only accessed by internal systems only.

### Security Information Management

#### Risk Management

#### Software development & Code Management

Standard Old Mutual Wealth development processes apply using Team Foundation Server as the code repository.

#### Forensics & Investigations

See **Error! Reference source not found.** **Error! Reference source not found.**.

#### Measurement Reporting

See **Error! Reference source not found.** **Error! Reference source not found.**.

#### Configuration & Policy Management

Not applicable.

#### Audit & Compliance Management

Not applicable.

#### Security Incident & Event Management

Not applicable.

#### Asset Management

Not applicable.

### Threat & Vulnerability Management

#### Patch Management

Standard OMW mechanisms.

#### Anti-Malware

Standard OMW mechanisms.

#### Cyber & Threat Intelligence

Standard OMW mechanisms.

#### Intrusion Detection & Prevention

Standard OMW mechanisms.

#### Security Testing

Standard OMW mechanisms.

#### Threat Discovery & Identification

Standard OMW mechanisms.

#### Risk Analysis & Prioritization

Standard OMW mechanisms.

#### Business Continuity & Disaster Recovery

###### Business Continuity

As for “Front End” systems.

###### Disaster Recovery documentation

As for “Front End” systems.

## In House comparison with RFP

TBA.