**Markel Online Technical overview**

Contents

[Introduction 1](#_Toc469502532)

[Technology Stack 2](#_Toc469502533)

[Architectural Overview 3](#_Toc469502534)

[MOL Extended 3](#_Toc469502535)

[MOL Authorization 4](#_Toc469502536)

[Decision Service 4](#_Toc469502537)

[Service – Authoring and Delivery 6](#_Toc469502538)

[Product Definition Service (PDS) 7](#_Toc469502539)

[Correspondence service 7](#_Toc469502540)

[Submission service 7](#_Toc469502541)

[Configuring Dev Environment 7](#_Toc469502542)

[TSV Importer 9](#_Toc469502543)

[Rule service/ Rule APP 9](#_Toc469502544)

# Introduction

Markel Online is the online commercial insurance system wherein customers (broker/agent/underwriter) can create the policy on the go. We can book the policy with wide range of line of businesses (LOBs).

The full-fledged LOBs that are currently supported in the system are:

1. General Liability
2. Property
3. Liquor
4. Special event
5. OCP
6. Inland Marine
7. Excess Liability

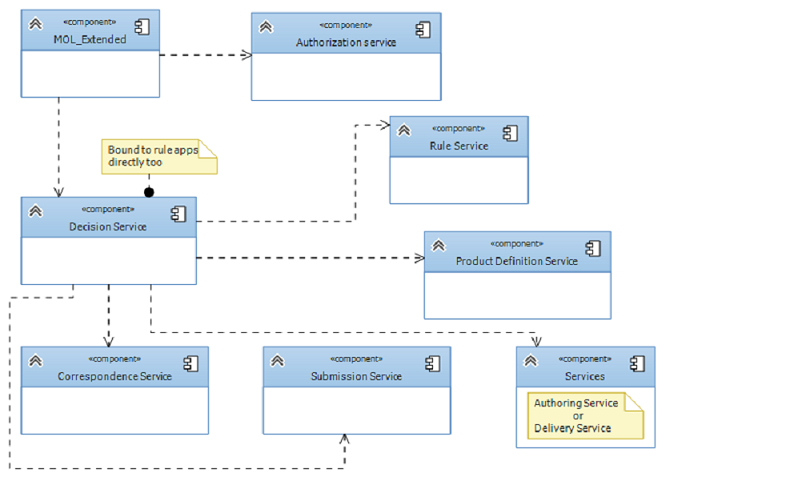
For more details on the business perspective please refer to Domain document.

# Technology Stack

* MVC 5.0
* WEB API
* C#
* In Rule – Rule app
* SQL Server 2008 R2
* Entity Framework 6.0
* Nuget Packages
* TFS server and builds

# Architectural Overview

Authoring/ Deliver DB



Authorization

Submission DB

## MOL Extended

MOl Extended application is the face of the MOL system. This web application is made using MVC 5.0 and is the starting point of all client’s calls. It references 3 core APIs i.e. Authorization, Services and DecisionServices.

Code can be found at: **$/MarkelOnline/(BRANCH)/** **MOL\_Extended**

Ex: $/MarkelOnline/Integration/MOL\_Extended

The architecture of this application is not created the traditional way. Though, we have many forms and partial views, not all the forms that one sees here has a one to one view. Most of the pages here are built runtime and are shown on some generic views. This project contains custom html extensions that are widely used across the views.

Ex: We can have multiple documents inside a policy. And each document can have multiple questions associated to it. These questions are added to the document bucket using Decision services and are all bound at runtime in the document view.

## MOL Authorization

When a broker /Agent/ Underwriter hits the URL for the first time a Login screen is presented to him. The authentication and Authorization part is done by MOL Authorization service. The users of MOL system need to be present in MOL Authorization DB. We are using claim based authorization here. On the development environment we use spoofed users whose information can be found in various XML files in the folder.

Ex: $\MarkelOnline\Integration\MOL\_Extended\MOL\_Extended\testuser-agency1-user1.json

We are unable to debug this service at offshore as there are certain 3rd party components that we do not have access to. We generally don’t need to change anything in this service as this is full proof and rarely requires any change. We might change the way it authorizes in the future.

Code for it can be found at:

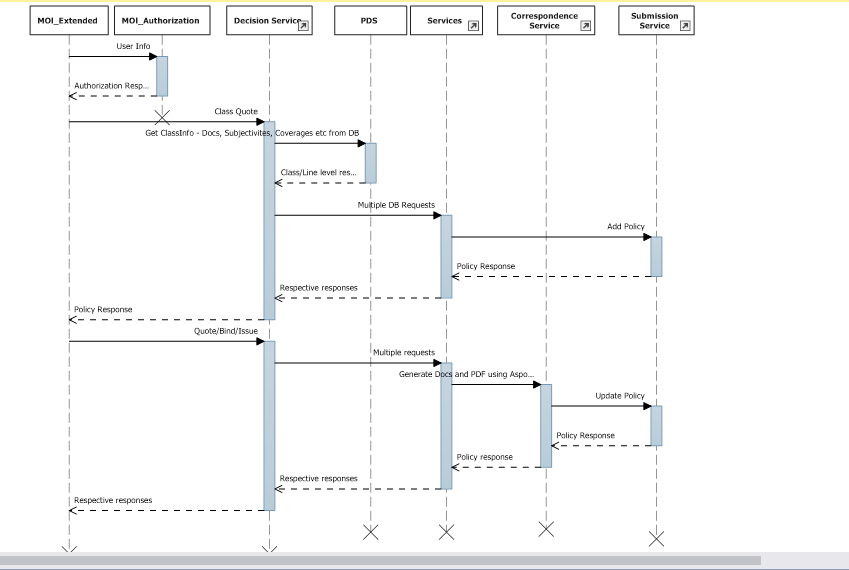
**$/MarkelOnline/(BRANCH)/AuthServices**

Ex: $/MarkelOnline/Integration/AuthServices

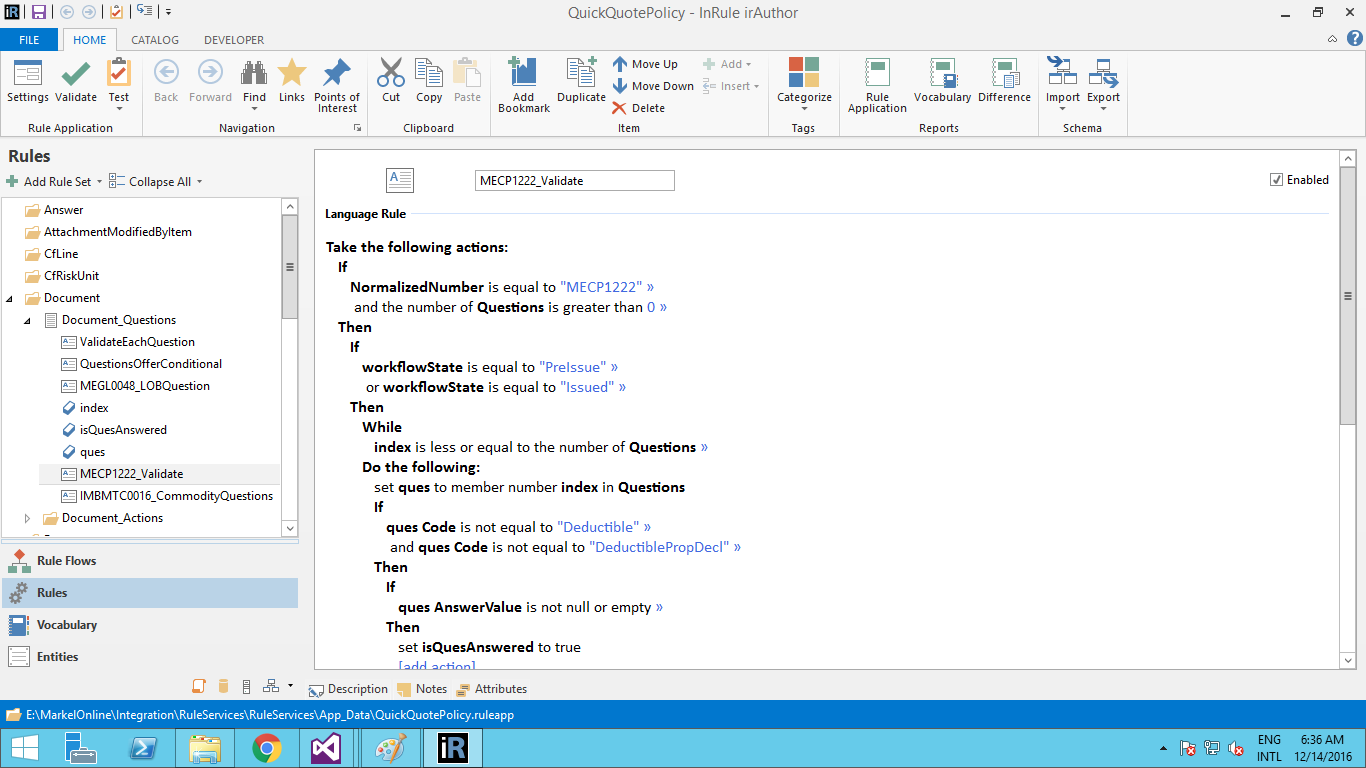
## Decision Service

This is the core business logic component in MOL. All the requests and responses are governed by the calls made using this web Api service. It uses an entity representation model “Decision Model” using which different services and MOL Extended can send requests to Decision services.

The name depicts that it participates in deciding how and where the calls will flow. Below is the diagrammatic depiction of call flow of entire application via Decision service.



This component is coupled with rule engine (An XML based rules application). Thus, the actual business logic to do calculation based on Policy object is done in rule app. The rule app is saved in the form of XML which is written using third party utility irAuthor. Below is the screenshot of inrule application opened using irAuthor editor. Once Decision Service accepts a call to any processing on the Policy object, it invokes the rule app using irAuthor.dll that is referenced in Decision service. The Policy object is serialized in XML format and is available for update in rule engine. Once rule engine does the calculation, it updates the required properties on Policy object and returns the serialized string to Decision service. We deserialize it back to Policy object and store it in the cache or DB.



## Service – Authoring and Delivery

This web api service is again made to counter the generic database related functionalities. This service is wrapped around the configuration DB of MOL i.e. “UWAuthoring” or “UWDelivery”. Both databases are exact replica of each other but any operations requiring TSV update or sql queries update is first done on Authoring DB. Once it’s successful and stable to perform operations using APIs, we publish the Authoring data to Delivery data using Publication web service. At any given point of time only one of the connection will be used in the application as shown in the appSettings.

<connectionStrings>

<add name="UWAuthoring" providerName="System.Data.SqlClient" connectionString="Data Source=STAPP090;Initial Catalog=UWAuthoring;Integrated Security=true;persist security info=false;Trusted\_Connection=true;multipleactiveresultsets=true" />

<add name="UWDelivery" providerName="System.Data.SqlClient" connectionString="Data Source=VA1-DWHLWEB015;Initial Catalog= UWDelivery;Integrated Security=true;persist security info=false;Trusted\_Connection=true;multipleactiveresultsets=true" />

</connectionStrings>

<appSettings><add key="UWContextEnvironment" value="UWAuthoring" /></appSettings>

## Product Definition Service (PDS)

This service is again a DB layer and whatever is said for Services holds true for this web api too. This is built recently so as to improve the system performance. Slowly, we would try to migrate everything from Services to PDS thus reducing the call flow.

Initially we used to fetch data when and where required basis in services but in PDS we are prefetching the data that we already know would be required to complete a policy thus reducing the calls and in turn improving the performance. One more advantage of this service is to support data that is required by Next gen applications. The configuration is similar to that of Services.

From the sequence diagram in Decision service you can see that Decision service calls PDS and services both.

## Correspondence service

This component (web api) takes care of building quote/ bind / issuance documents on the run. Document formats supported are PNG/PDF/DOC/DOCX etc. It reads the policy object and based on the templates provided in the MOL system it applies transformation to the policy object. Using the template’s look n feel and mapping, the policy is transformed into the required output format i.e. PNG, doc or PDF. The third party tool that’s used here is Aspose to generate the document. These documents are generated and saved to disk or DB based on configuration.

## Submission service

Submission service is responsible to manage the operational DB of MOL system. The policies that get generated are stored in Submission Database. We store policy object in the form of json along with additional details like transaction number, agency info, producer info etc in submission database. Every Policy in this system holds a version, this way we can upgrade the older policies if required.

# Configuring Dev Environment

You will be given access to VDI and pre built VM on the VDI. VM is used for all your development activities. Once you receive your credentials try logging on and in cases of any issues talk to your lead.

Once you are able to log in to the system

1. Login to VM
2. Make sure you’ve VS 2015 and SQL server installed on your machine.
3. Configure TFS with <http://stwebdev30:8080/tfs>
4. Take latest code from $/MarkelOnline/$(Branch)
   1. Ask about $(Branch) to your lead
   2. It can be anything ex: $/MarkelOnline/Integration
5. Open all the respective projects as given in Sequence diagram and compile it one by one. Once all projects succeed, try to run all the applications locally.
6. Every project has a Web.Config file. You need to be very particular in choosing the proper configuration.
7. Run services – Keep the Web.config as is
8. Run PDS – Keep the web.config as is
9. Run Submission service – Keep web.config as is and make sure you’ve E:\qq-temp on your system
10. Run Correspondence service
    1. For local keep following setting

<add key="DecisionServices" value="http://localhost:47776/api/" />

<add key="UWManualWebApiUrl" value="http://localhost:59125/api/" />

* 1. For remote keep following setting

<add key="DecisionServicesWebApiUrl" value="http://stapp720:47000/api/" /> <add key="UWManualWebApiUrl" value="<http://stapp720:42000/api/>” />

1. Run Decision services
   1. For local keep following setting

<add key="UWManualWebApiUrl" value="http://localhost:59125/api/" />

<add key="SubmissionServices" value="http://localhost:46547/api/" />

<add key="CorrespondenceServices" value="http://localhost:47992/api/" />

<add key="ProductDefinitionServices" value="http://localhost:16528/api/" />

<add key="RuleServices" value="http://localhost:53102/api/" /> -

Keep RuleServices local only when you’ve irauthor’s license else use remote setting

* 1. For remote keep following setting

<add key="UWManualWebApiUrl" value="http://stapp720:42000/api/" />

<add key="SubmissionServices" value="http://STAPP720:46000/api/" />

<add key="CorrespondenceServices" value="http://STAPP720:48000/api/" />

<add key="ProductDefinitionServices" value="http://STAPP720:52000/api/" />

<add key="RuleServices" value="http://stapp720:49000/api/" />

1. Run MOL\_Extended
   1. For local setting

<add key="UWManualWebApiUrl" value="http://localhost:59125/api/" />

<add key="DecisionServicesWebApiUrl" value="http://localhost:47776/api/" />

* 1. For remote setting

<add key="UWManualWebApiUrl" value="http://stapp720:42000/api/" />

<add key="DecisionServicesWebApiUrl" value="http://stapp720:47000/api/" />

NOTE: for first time, please run all the above services. From next times onwards run the services that require modification. MOl\_Extended and Decision services should always be run locally rest services can remain off and we can use the build server’s services.

# TSV Importer

# Rule service/ Rule APP