RFP 16-01

EXHIBIT K

Corporations and Charities System

Logical Architecture Document

April, 2015

TABLE OF CONTENTS

LIST OF FIGURES

Document Revision History

Version	Date	Description
	4/02/2014	Drafted by Sanjeev Batta presented at meeting

1. INTRODUCTION

1.1 Purpose

This document describes the Logical design for the Corporations and Charities System. The logical design document describes the Logical data model, Logical Service and Layers of the architecture and Technology Architecture in terms of technology components.

The Logical Design is described in terms of many static and dynamic views of the architecture to communicate the core areas of intent of the application. These views are described as architectural and design patterns, and practices, but not as an all-encompassing detailed design of every UI element and data element of the application and its processing logic.

The goal for this document is to gain an architectural consensus in logical, technology and data aspects of the application, and get buyoff from the technical and business stakeholders involved in the process. Also this document will provide architectural guidance to the developers on the project and guide the overall design and development effort.

1.2 Scope

The scope of this document addresses the logical architecture for the Corporations and Charities System. Even though system may interface with other systems e.g. Combined Fund Drive and Revenue logical design or changes / impacts to those systems from a logical architecture viewpoint are not covered within this system.

The scope of this architecture includes the high level logical design for both the internal (staff console) and external (web customer facing) aspects of the corporations and Charities system as well as the integration with some of the essential external systems e.g. DOR, Revenue etc.

There is an effort to envision the order entry system for a broader enterprise use and the architecture will be cognizant towards the goal, but the detail mapping of enterprise use of order entry subsystem is not covered in this logical architecture.

1.3 Resources

The information for this document was gathered via design sessions with the EAO and core teams, research on latest Java standards, as well as IBM best practices and RedBooks on IBM Process Server and SOA architectural guidelines.

1.4 Constraint

The design for SOS Corporations and Charities solution is constrained by

- *Use of .Net and Microsoft solution stack as the standard technology platform.*

- *Design for existing interfaces with DOR BLS and UBI systems.*
- *Minimal changes to the existing agency revenue system*
- *Use of Kofax as the document imaging solution.*
- *Use of statewide security standards and other OCIO guidance where possible.*

2. LOGICAL ARCHITECTURE MODEL

2.1 Logical Architecture Overview

The logical architecture for the Corporations and Charities System is based on the following core facets

- Enable a flexible yet balanced approach to a Meta development of the different entity types, transactions and business rules in the system. The system design will support ease of adding new entity types or modification to filing transactions and business rules for existing transactions.
- Keep the workflow and queuing mechanism to be flexible and not overly engineer the business process. The queues are based on a pull mechanism avoiding any complex workflow assignments.
- Drive the user interface using a flexible yet simple UI builder mechanism.
- Deliver most functions of the application as an API so that those APIs can be leveraged by the user interface that is not only developed by SOS but also by other partners as well.
- Enable security at the services layer of the application so that the application services are secured consistently across different usage scenarios. E.g. partner vs. internal.
- The logical architecture is based on a development pattern separating the life cycle of a business process and its corresponding business data. The goal for the underlying architectural approach is, strong decoupling of the respective life cycles and increasing robustness to cope with inevitable changes.
- Develop the order / shopping card and catalog components in a fashion that enables loose coupling with the filing aspects of the system and promotes enterprise use.

This document plans to address some of the basic cross cutting concerns in the overall logical architecture in the following areas

- User Interface Layer
- Services Layer
- Data and Document Storage
- Process and Queuing Mechanisms
- Subscriptions and Notifications
- Security and Token Mechanism
- Document Generation
- Internal Revenue System Integration
- Document Imaging Integration
- Department of Revenue Integration
- Local Print and Scan Management
- Batch Processes

2.2 The User Interface Layer:

The user interface layer of the application will be developed using the AngularJS framework. The basic pattern behind the AngularJS is MVVM (Model View-ViewModel) which allows the front end to be loosely coupled with the services and the data layers of the system.

While the basic discussion and documentation of angular as a framework as well as the MVVM design pattern is outside the scope of this document, the architectural approach calls for several built in AngularJS capabilities that will be further extended to create a flexible yet simple mechanism for a configurable UI for the system.

The user interface will implement core components e.g. shopping card using standard AngularJS architectural paradigm and modular approach. These components will communicate with the back end services using REST and JSON based services.

A major component for the system is the small but numerous variations in different fulfillment form. E.g. Formation data for a for-profit entity may be similar to a large extent with small variations based on entity types. Also certain common components e.g. addresses, parties / people in various roles are similar but slightly variant in different entity types.

The architectural approach calls for these variances to be driven via a Meta data based approach along with the creative and powerful use of Angular partials and hierarchical scope context.

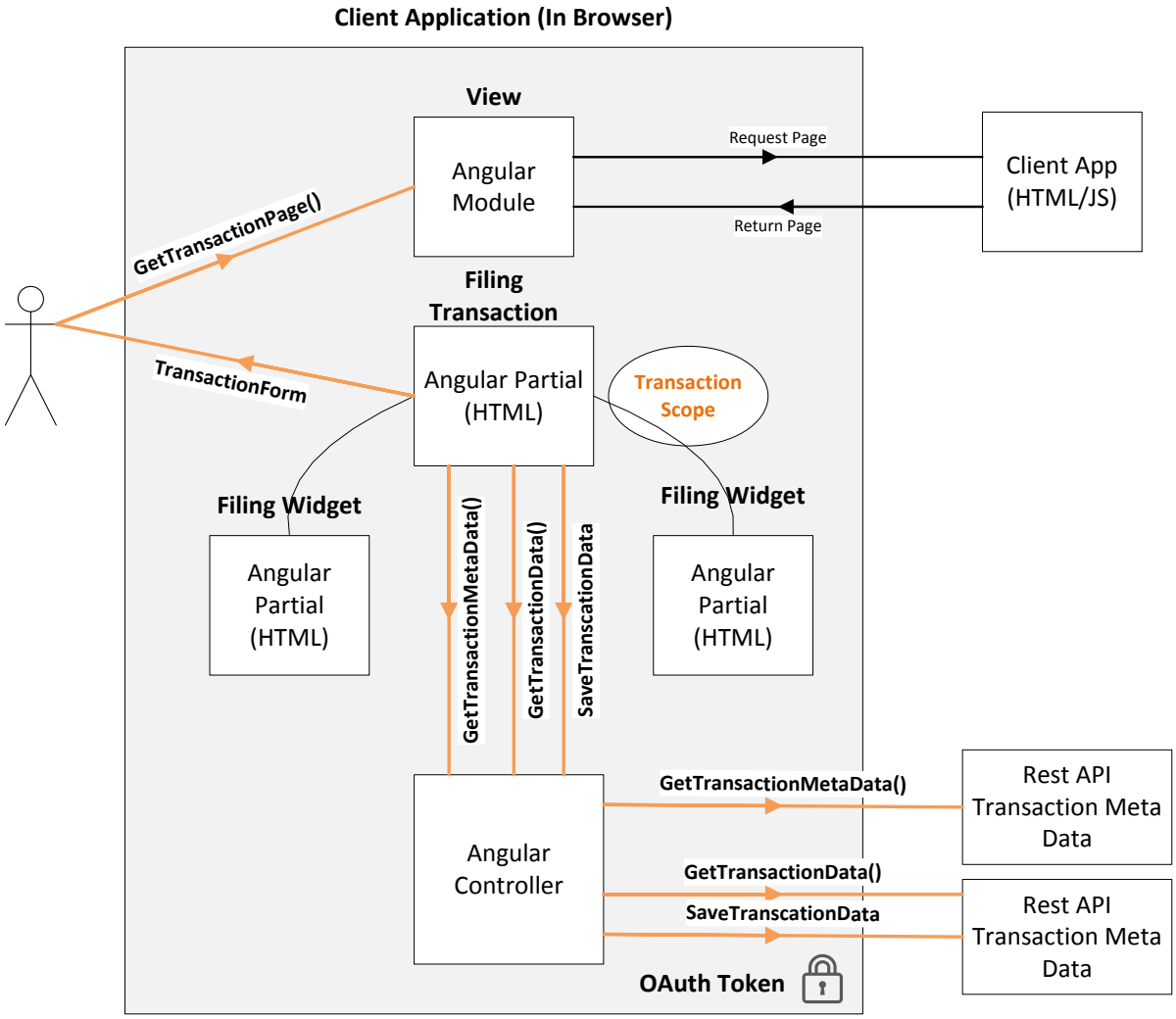

Figure 1: User Interface Layer

.

A baseline list of angular modules / widget will be established during the detailed design phase.

The filing widgets will be self-configuring using the nested meta-data context that will be made available via rest based services. The widgets will be developed in a way that maximum reuse is provided across the internet application and staff console.

2.3 Services Layer

The Service Layer of the application will be developed using .Net WebAPI based RESTful services. These services will provide both the Meta data as

well as data for the saving of different transactions and shopping cart and order management services.

The services layer of the application will be exposed in such a way that the application front ends for internet application, as well as the staff console can both leverage the same set of services. The services will also be available for use by SOS partners and other entities interested in a more API based integration with the services.

The REST services exposed by the solution will be based on resource based services where possible, and follow the REST services standard. The services will be secured using the OAuth token mechanism as described in the security and token section of this document.

The logical services architecture is described below

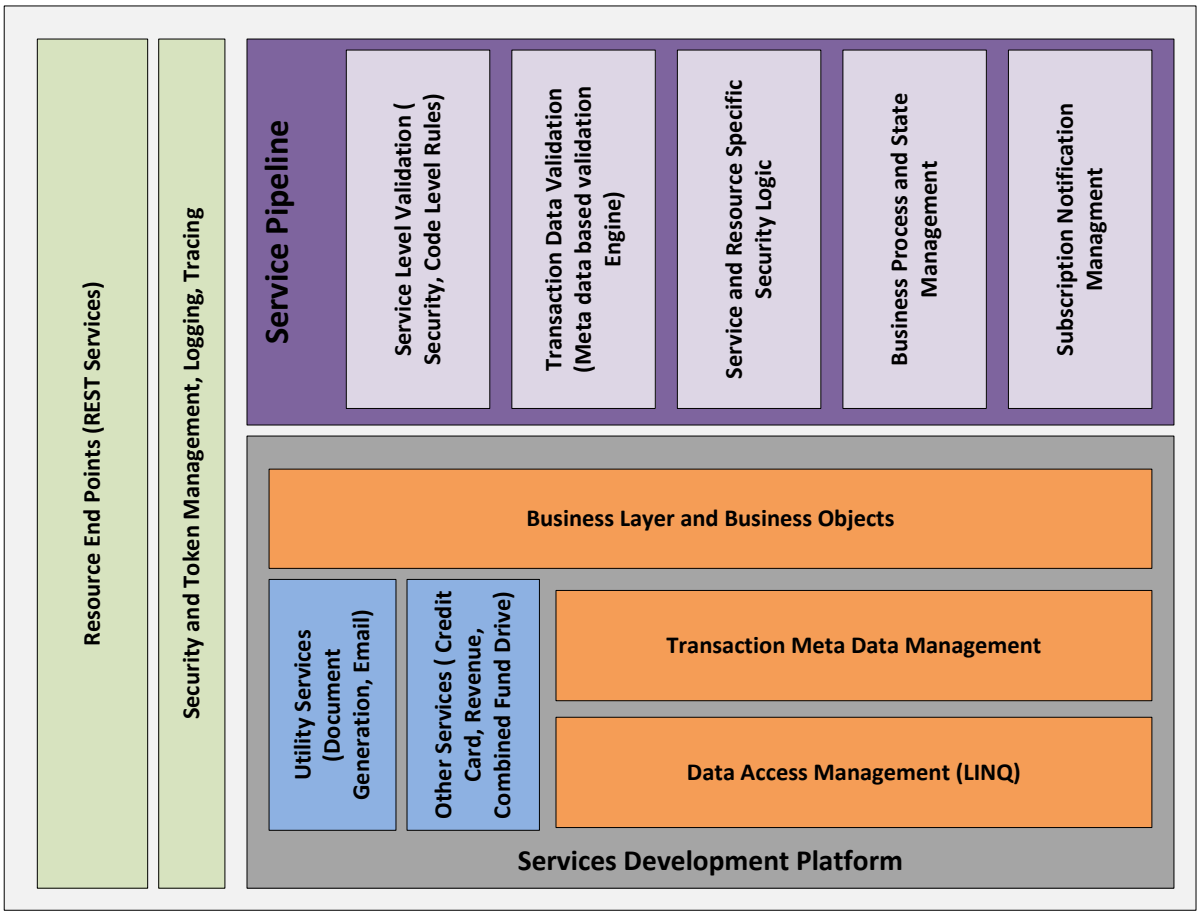

Figure 2: Logical Architecture

The proposed architecture will deliver a service development framework that is based on WebAPI and leverages LINQ and .Net for developing business objects that expose various aspects of the application. These business

objects leverage the Meta data and utility services to deliver an ability to deliver service specific functions exposed by the rest services endpoints.

The service REST endpoints will be secured with a Lightweight OAuth Token / JWT which will be validated and enforced at the REST entry points to ensure the services are secure. The service endpoints will also enforce logging, metering and other functions as required.

Recommendation: Our architectural recommendation is to evaluate a third party tool for service endpoint management. WABOS and OCIO are considering tools that might be available as part of the statewide service initiatives or a product specific to SOS requirements that may also enable other aspects of integration. MuleSoft / APIGEE are some of the front runners in the endpoint management domain.

The service pipeline is proposed as a custom implementation that will enable service level security, business rules management, process and workflow management, as well as a Meta data based transaction data validator that enables a flexible way to manage transaction level rules for various types of filing transactions.

High Level Resources for REST Services

The following diagram is a high level map of the REST resources exposed by the application services. These resources will provide a high level entry point in various areas of the application. A further detailed set of restful services will be defined in the detailed design phase.

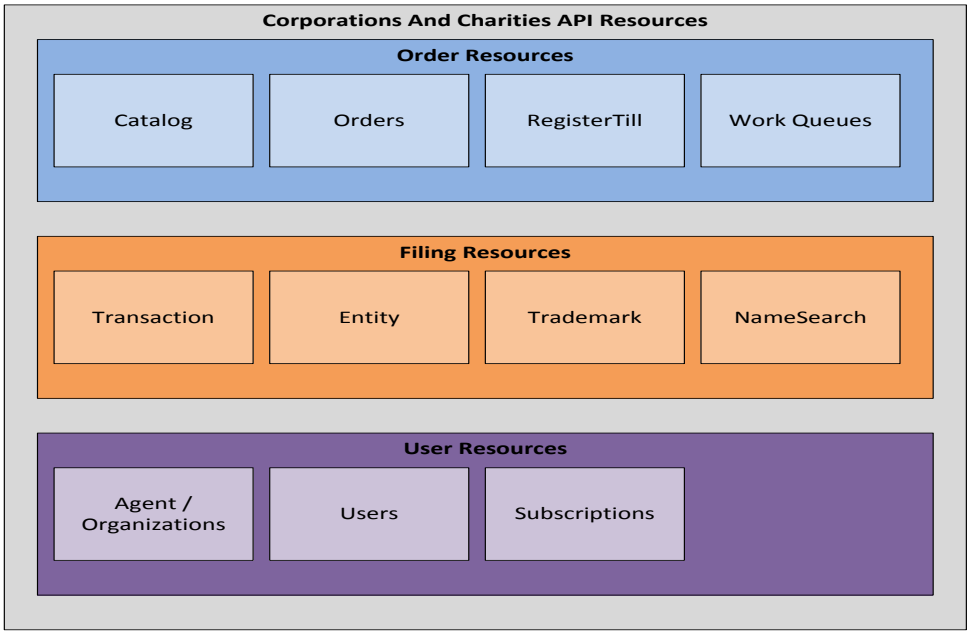

Figure 3: High Level Map Of REST Resources

2.4 Data and Document Storage

The data model for the application is defined in a logical data model and physical data. This section describes the general approach to data, data partitioning and data and document storage that will be used for this project. Some of detailed storage level approaches may be decided during the later stages of design in context of performance, query and other considerations.

The Corporations and Charities data can be categorized into following

- Meta Data: Data about transactions, business rules, required fields etc. Meta data can be stored as combination of relational data and schema and JSON or XML business rules.
- Order Data: Data about orders and order items along with the associated payments and other information for assignment and queuing. This data is best represented as structured relational data and is highly transactional in nature.
- Entity Data: The data in SOS is a more document oriented entity information set, which is made up of entity data along with transaction data. Entity data can be stored as relational data with a combination of XML / JSON data to allow flexibility of transaction data. The entity data could also be stored in a pure document database like Mongo or Raven to provide flexibility and scalability along with performance. The current model for the entity data is modeled as relational.
- Documents: The system has a need to store documents related to transactions. These documents are system of record for transactions

and would grow in size with the system. The document storage for system would either use the SQL Server FileStorage features or an unstructured storage in a document datastore.

As depicted in the services layer the access to data will be managed via the LINQ based data services layer that will encapsulate the storage and persistence of the Meta data, data and documents.

The storage format for the documents will be PDF format. Any annotation of the document will create a new version of the document and older version of the document will be saved for audit and history purpose.

The name search component of the application will be developed using the full text indexing features of SQL Server or the data storage platform for storage of entity information.

2.5 Process and Queuing Mechanisms

The application will manage the order process and queuing via a basic pull mechanism. The order and order item states will be tracked through the system actions and follow receive, augment, fulfill and file process.

Database fields and history tables will be used in order to track the user actions and history of the order item through various stages.

A work queue will be a view on top of the order data e.g. Show me All Orders for New For Profit Entity Filing that are expedited (For Profit expedite Queue). A user will be allowed to check out and pull an item to work on and the checkout information will be logged in the system to develop the work items for that particular user.

System will require security and override mechanisms to allow supervisors or other with appropriate permissions to checkout an order item that is being worked on by the other team member.

Following state model shows the basic flow of the item via different channels. Further state model for exceptions processing and refinements of different state transitions will be done during the detailed design and development.

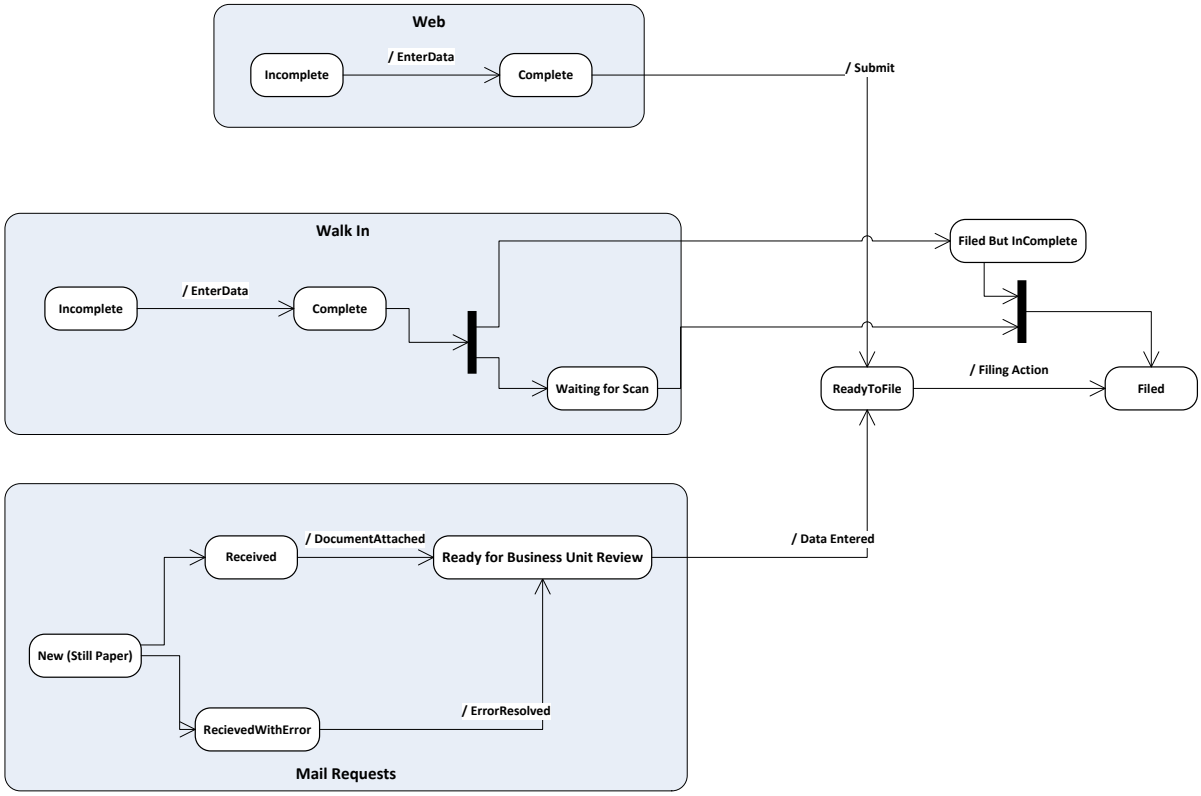

Figure 4: Order Process States

The business process and lifecycle of an entity is managed by the different filings and actions. These actions and state transitions will be managed via a meta data approach towards managing entities, transactions, states and allowed transactions and actions. An entity can have its lifecycle altered by either a filing transaction or via the automated jobs e.g. dissolutions. These entity transactions and actions vary by each type of entity, and will be defined at each entity level and captured in meta data or a mix of meta and .Net business rules.

2.6 Subscriptions and Notifications

Subscriptions and Notifications of the subscriptions will be provided via a simple notifications engine which subscribes to the business events from the business layer. The subscription engine will query the subscription data and find any notifications to be delivered for the business event.

A business event can be a filing event, order event or other events e.g. dissolution or renewal approaching on an entity in the system to which subscriptions can be created.

The subscription engine creates and sends the subscription notifications and records the data in the database for audit and record management.

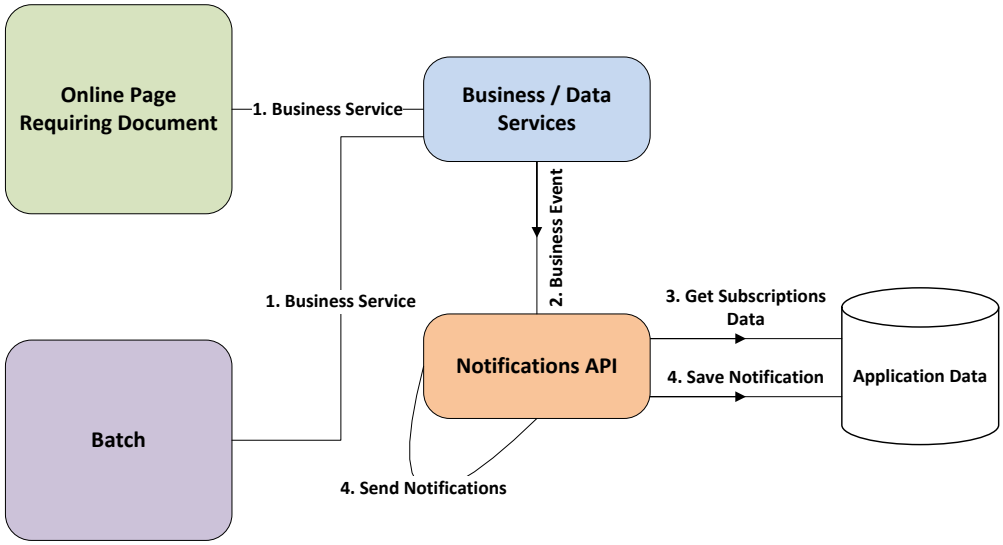

Figure 5: Subscriptions and Notifications Engine

2.7 Security and Token Mechanism

The security and Token design proposed will enable external users to use Secure Access Washington or check out as guest, where appropriate. The internal users will be able to log into the system via the single sign on with active directory. Users working offsite or remotely can also log into system via Secure Access Washington. In both cases system will validate internal user roles and access privileges via active directory groups.

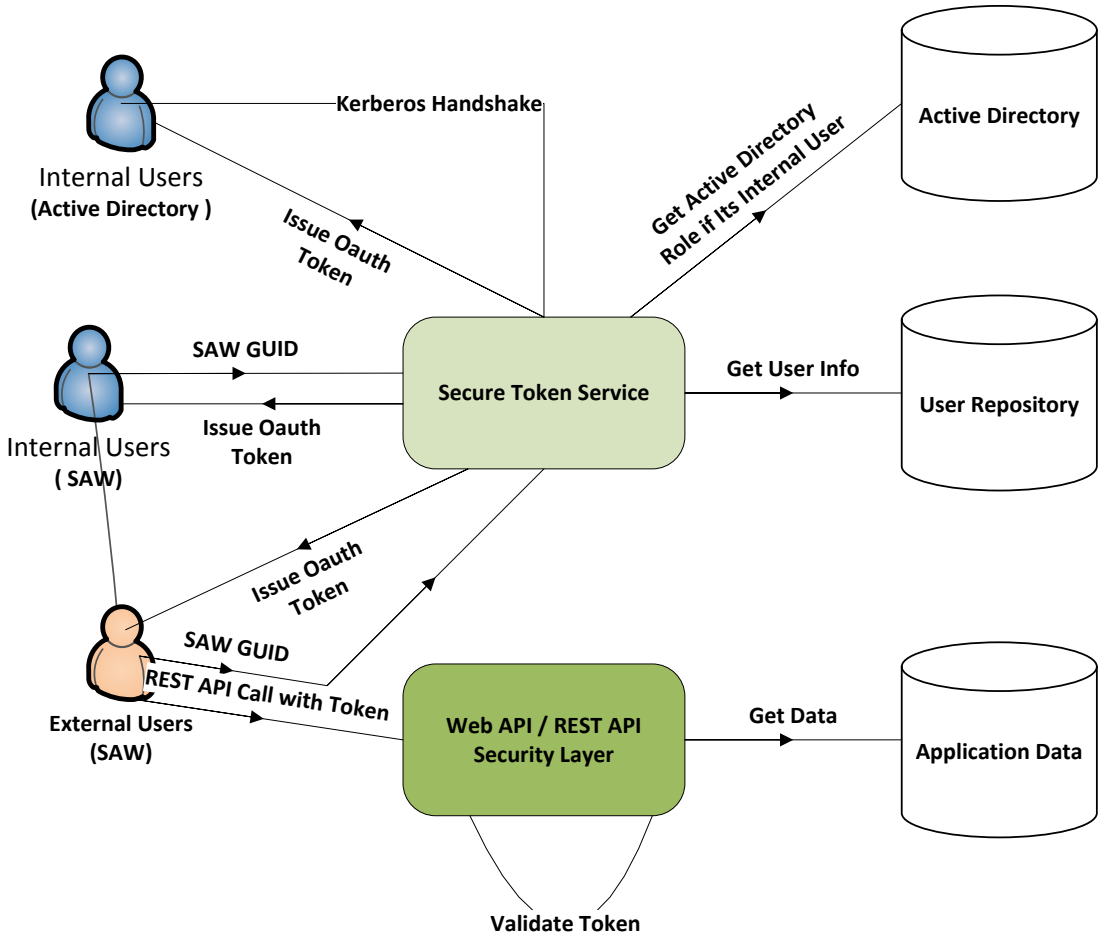

Figure 6: Security and Token Design Mechanism

The architecture calls out for a custom token issuance authority that will issue tokens for both internal and external customers in a unified fashion and allows the application services to be solely depend on the issued token.

The token format used will be an Outh Token with custom OAuth claims tailored to the SOS services and user roles. The Token issuance service also acts as a simple token translator and translates a SAW token to a user identity in the system and issues a valid token to the front end for calling the secured services layer.

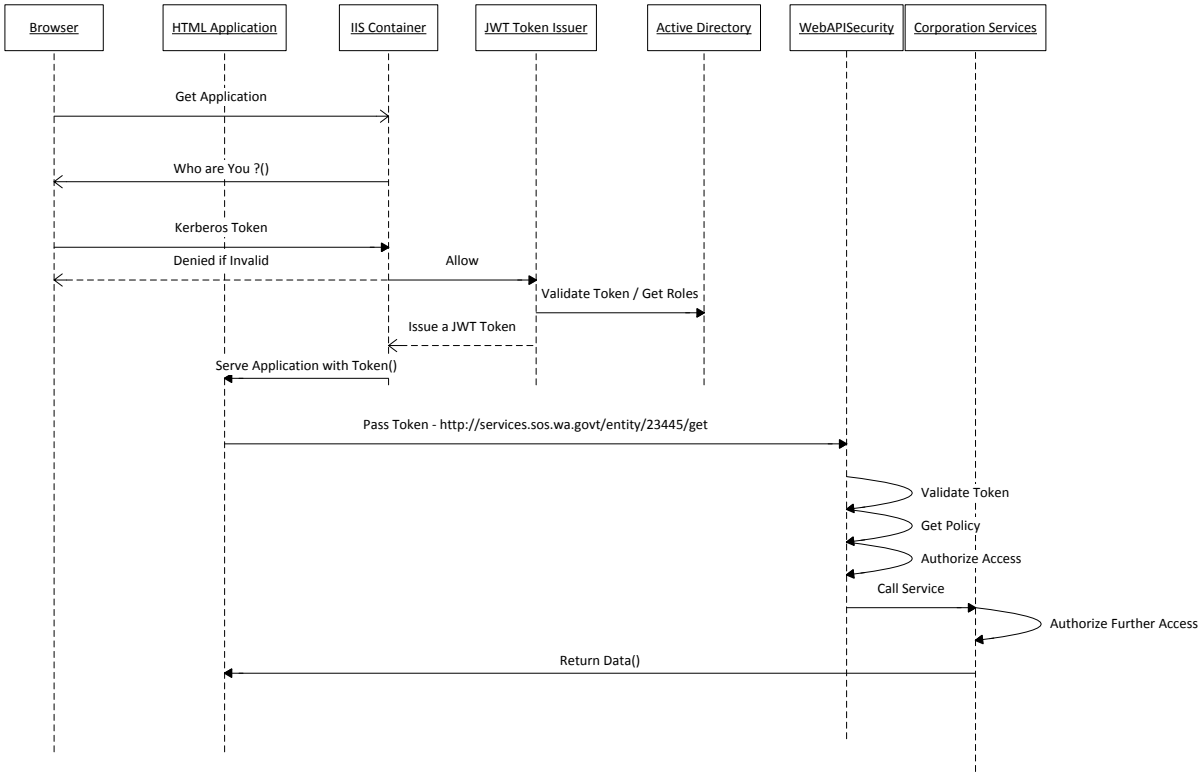

Figure 7: Token Sequence Diagram

2.8 Document Generation

The document generation needs for the project require documents to be generated in two different modes.

- **Online Transactions**: Online transactions or any filing transactions will generate a confirmation / filing document that captures data and a template to create a file copy of the data for records. This data is most easily accessible in the JSON format but can also be converted to other formats if a particular technology was more desirable.
- **Letters and Notices**: Letters are notices are sent by SOS staff at various stages of the workflow. These letter and notices may require custom user input to customize the letter content. Our recommendation is to provide those customizable fields in the application as a text input and merge those with the template, where appropriate. The choice of customization on letters will also lead to some possible technology choices.
- **Batch Transactions**: Various batch transactions will generate printed notices, reminders, dissolution letters and such. These letters need to be printed in a fashion that can be controlled by some batch process and possibly leverage the DES print services where needed.

Document generation logical design falls into two different options based on which during detailed design and implementation an approach will be finalized.

Option 1: Document generation as a service. In this option the document generation is just a template / document merge engine exposed as a service. The responsibility of getting the data for the document is delegated to the caller of the template.

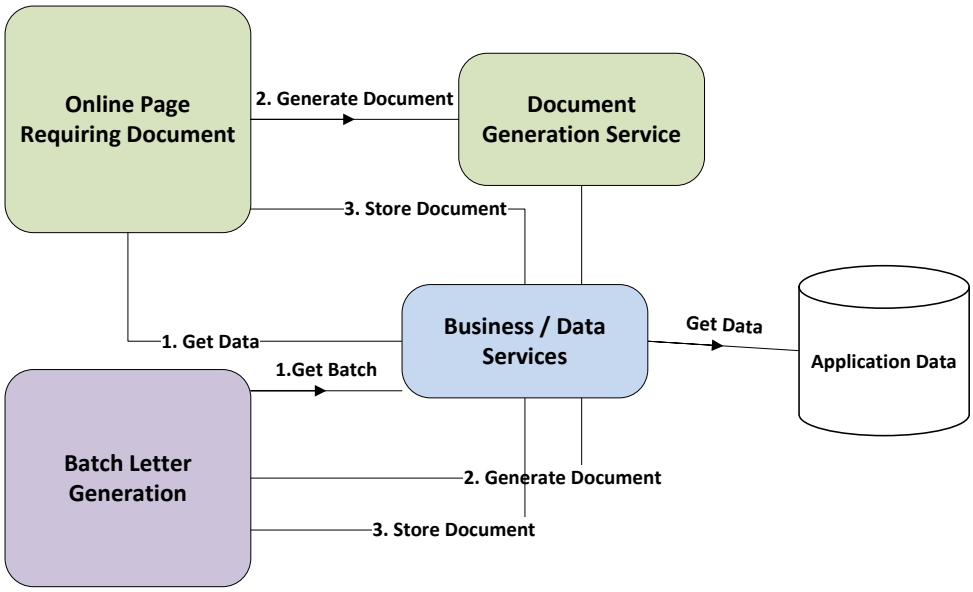

Figure 8: Document Generation Option 1

Option 2: In this case the document generation service also is responsible for knowing the structure of the data inside and querying the document data for producing the document.

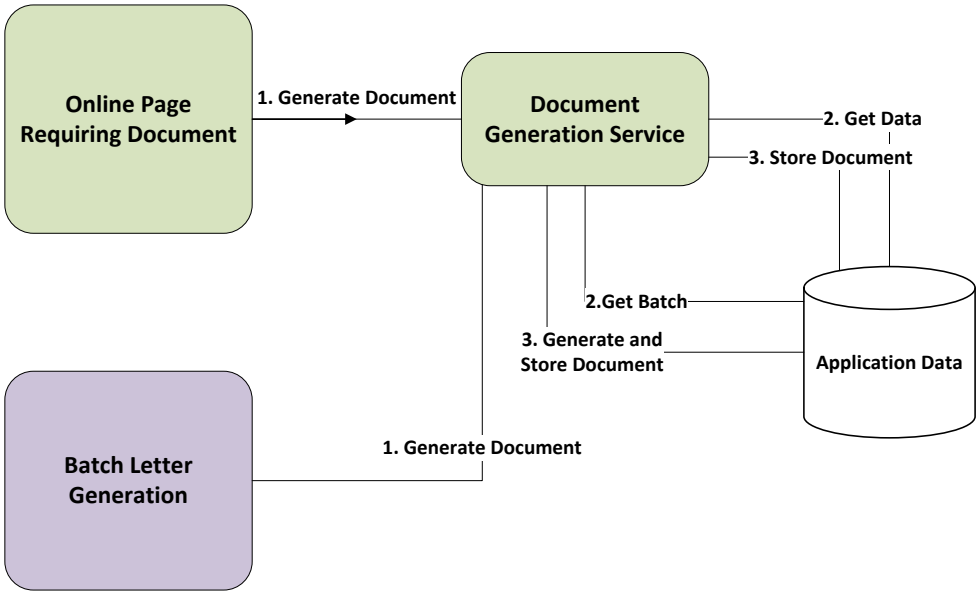

Figure 9: Document Generation Option 2

Based on the two design options above we recommend one of the following products. A final selection of the product can be made based on detailed user experience analysis, ease of use and further proof of concept during detailed design / development phases.

1. ECRION: Ecrion is a commercial document generation engine which can be used in both online document generation as well as for batch document generation options. The main aspect of ECRION is its flexibility in template management, multi lingual capabilities, versioning templates and other mature document management features. The cost of such commercial product and its overall ROI for SOS needs to be evaluated.
2. Microsoft Word: Basic and native Mircrosoft word capabilities can be leveraged within certain constraints to allow generation of letters and merges. The issues in this approach is the general word versioning support and change management, scalability for batch document generation and other template management functionality that will need to be custom developed.
3. SQL Server Reporting Services: SQL reporting services is a good tool for bulk oriented document / report generation and can be leveraged in the option 2 logical design scenario. Licensing requirements around public facing use of SSRS may need to be evaluated and its operability with JSON or non SQL data if that choice is made.
4. ASPOSE: Aspose is a document generation and PDF manipulation tool that provides a rich toolset for document generation and can

provide a cost effective yet a pragmatic alternative to Microsoft Word. Aspose natively allows PDF generation and also allows document merge e.g. If the filing document requires the merge of generated document along with the filed attachments.

2.9 Internal Revenue System Integration

SOS revenue is an aging system and there are plans to replace the system with a commercial off the shelf package or another more modern implementation. The Corporations and Charities system still needs to interface with the revenue system for overall accounting and statewide AFRS interface.

The logical design of the system is developed in a way that it's loosely coupled with the revenue system and can function without hard dependency on the SOS internal revenue system.

The main logical entities of the system are cross referenced and mapped to the revenue system identifiers. The data model for the system allows for these mappings to be optional which allows the process to be either a daily batch release process to revenue or an API based interface.

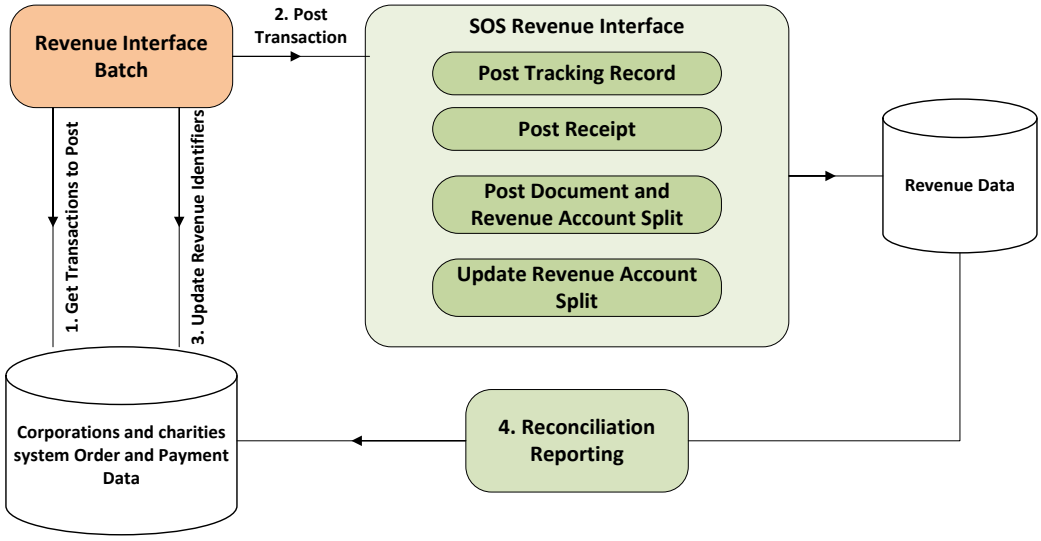

Figure 10: Internal Revenue System Integration

The SOS revenue interface can either be exposed as a set of Web Services or stored procedures that provide the logical abstraction over the revenue tables. The revenue interface batch process will query the Corporations and Charities system for orders, payments and documents that are ready to be posted to the revenue system.

The revenue posting can be done regularly through the day or at the end of the day when batches are reconciled and finalized in the staff console based on the business process and SOS revenue unit's preferences.

The logical design also proposes a reconciliation report that can report on a daily and monthly level on the data between the two systems.

There are two additional processes that still need further discovery and design based on the business process decisions and fine tuning during detailed design and development. These processes relate to the refund for overpayments and NSF processes. The NSF processes may create a new receipt in the system or the debt would be written off and the filed transaction rolled back to the state before the payment. The level of automation and the process via which those transactions are handled will be reviewed in later phases of the project.

2.10 Document Imaging Integration

The scanning and Indexing Component of the System will still leverage Kofax as the technology to perform the scan on the documents.

The Logical overview of the scan index process is depicted in the following diagram.

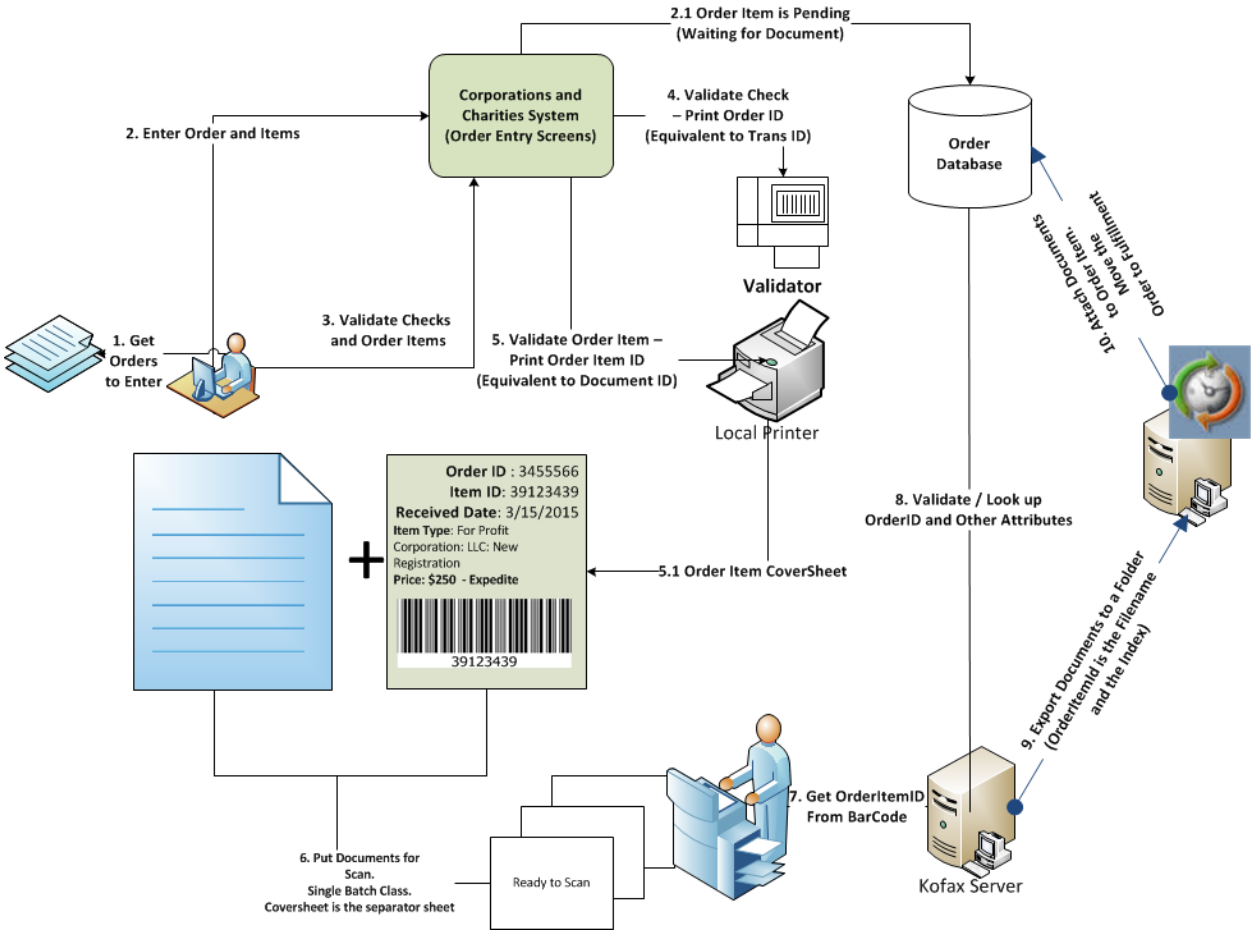

Figure 11: Document and Imaging System

The main design aspect of the Logical model is that the Index information for the order and order item is entered at the order entry state in the new Corporations and Charities System via the staff Console. The Order Item ID cover sheet is then printed to act as the automated index page as well as the separator page for Kofax.

Kofax system will scan the barcode and validate the Order Item ID against the Order database using a database or a service lookup. The Kofax system will only require a single transaction batch since the indexing information e.g. expedite / non-expedite is already entered upfront.

At the end of the validation the Kofax System will release the document batches to a file folder. The filename will contain the Order Item ID to which the document would be associated. A batch job listening to the folder will get the document from the folder and attach the document to the Order Item via an API and also move the Order Item to the next appropriate state e.g. fulfillment or exception if the order item was marked to have exceptions.

There may be other types of exceptions in the scanning indexing module which will need to be addressed and designed based on the learning from the proof of concept effort.

The export of the document from Kofax will be in the PDF format.

2.11 Department of Revenue Integration

The system needs to integrate with department of Revenue system. The current interface with DOR BLS system is via a batch process that runs every 20 minutes. While department of revenue is still in the process of upgrading the system, project approach will be to allow the interface to work as is where possible while also allowing the API level interface for the new BLS system when it gets implemented.

In the current process department of revenue handles a number of transactions on behalf of SOS that include annual renewals, reporting unregistered entities and several batch jobs e.g. dissolutions.

In the new model there are only two expected transactions in the system

- Annual Reports
- Report of New Un Registered Entities

An order batch will be created for transactions from DOR and Annual Reports will be handled via the normal order process by creating an order for each annual report.

Unregistered entities are entities that various agencies (for eg, LNI and others), have found during audit processes and have issued a UBI for those entities in the process. The goal for reporting these entities is that these entities when filing with SOS can be attached to the entity. Currently these entities can be reported through many different mechanisms including email, and SOS interface.

The data for report of unregistered entities will be logged into the system via the system API to create an unregistered entity. No transaction or order will be created for this particular transaction.

The DOR annual report process also sends the images for the transactions. These images will be handled in a similar fashion to the scan index images. These images will be saved to a folder which will then be picked up by the document indexing batch process which will attach the images to the order item transaction and file it.

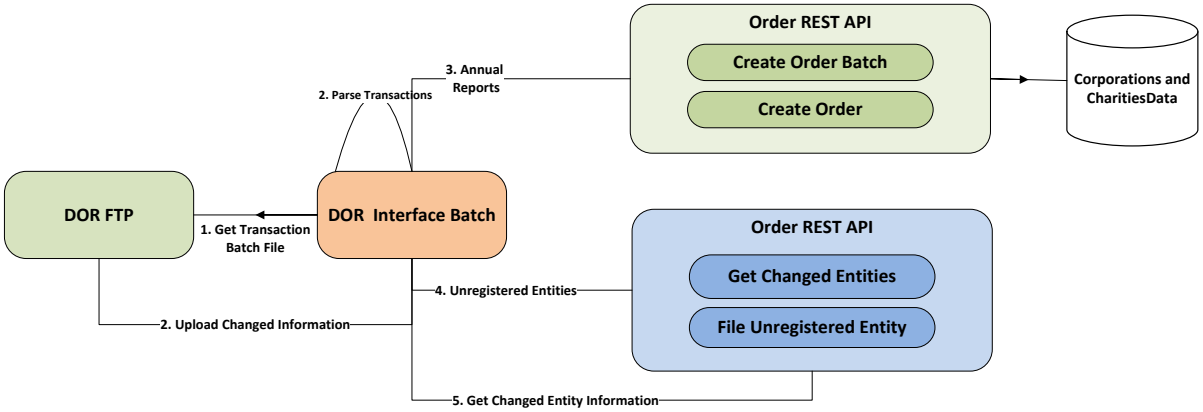

Figure 12: Department of Revenue System Integration

2.12 Local Print and Scan Management

The solution architecture calls for the front end application to be developed using a Javascript and HTML5 based thin client. This application architecture calls for the printing for validation printer, and scanning to be controlled locally via a unique mechanism.

The solution to leverage a browser plugin for enabling desktop scanning and print control is somewhat difficult to maintain and support across browser upgrades.

Proposed logical architecture calls for a locally hosted set of print and scan services that are similar to device drivers that expose RESTful service interface to access for the javascript in the browser to interface with.

One of such service is available via LeadTools and is our recommended solution for solving the local desktop scanning challenge in HTML5 application.

The details for the solution are explained via the LeadTools whitepaper.

https://www.leadtools.com/whitepapers/2015/html5-web-scanning-with-leadtools.pdf

2.13 Batch Processes

A number of batch processes exist in the Corporations and Charities System. These processes range from generating notifications of subscriptions to creating dissolutions and sending notices and marking certain entities delinquent.

These batch processes are most suited for SQL Server based job automation with some additional jobs or processes to produce printed notices. SQL Server Integration Services (SSIS) and SSIS Jobs will be used where possible to create these automated jobs.

Following is the type of batch processes that will be automated

- Annual Renewal Notices
- Delinquent notices where required by the law
- Notifications of subscriptions
- Dissolution of entities
- Triggering any automated holds that are based on wait times
- Sending / Exchanging information with any partner or other systems e.g. Revenue System.

3. LOGICAL DATA MODEL

The logical data model is provided as a separate set of Visio files along with the physical data model.

The logical data model is partitioned into three different schemas

- User and Subscription Schema
    - Contains users, registered agents / professional entities and subscriptions.
- Order Schema
    - Contains order, catalog, payment and other entities for the order and work queue processes.
- Entity Schema
    - Contains entity, transactions and entity related schema.

.

Project Approval Signatures

The signatures below indicate the Logical architecture model was reviewed by all parties and approve of its content.

_____

SOS Technical Steering Committee