**The Department of Housing and Urban Development**


# Service Layered Architecture Profile (SLAP)

Version 1.0

**<Date>**

**<System Name>**

## Solution Information

| | Information |
|---|---|
| **Solution Name** | |
| **Solution Acronym** | |
| **Project Cost Accounting System (PCAS) Identifier** | |
| **Document Owner** | |
| **Primary Segment Sponsor** | |
| **Version/Release Number** | |

# Document History

| Version No. | Date | Author | Revision Description |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

*Note: The latest version of this document supersedes all previous versions.*

# Table of Contents

# Introduction

The nature of software development and government acquisitions makes it expensive and inefficient to correct projects and their application architectures after they have been developed. Department of Housing and Urban Development (HUD) has adopted a proactive process that allows for prescriptive architecture guidance prior to acquisition activities. This approach ensures that HUD receives full value from their investments. The architecture guidance provided for each selected investment is the result of a detailed review of the investment and its enterprise implications. This document provides the Service Layered Architecture Profile (SLAP) for the Loan Review System (XYZ) and includes architecture requirements and recommendations for XYZ.

The guidance in this document has been derived from a number of sources grouped into the following three categories:

1. Investment Selection Information – Provides the rationale for investing in XYZ.
2. HUD Business Architecture and Loan Review Process Requirements – Specific requirements of the loan review business process and related HUD business processes.
3. Enterprise Architecture Guidance – Architecture standards and best practices HUD has adopted to ensure a flexible and responsive portfolio of systems that can be cost-effectively maintained and operated.

During the HUD investment review process, XYZ went through a Pre-Select phase in which a number of acquisition and technology decisions were made in light of the HUD existing portfolio of systems, the Enterprise Technical Architecture volumes (ETA), and the state of the IT marketplace. The outcome of the Pre-Selection process includes recommendations to create a custom-built solution, developed predominantly with open-source technologies.

The guidance in this document assimilates the XYZ investment rationale, the HUD business architecture and XYZ system requirements and the architecture recommendations from the investment review pre-select phase and introduces technologies that have been demonstrated, implemented, and validated under the Loan Review System Proof of Concept (XYZ PoC). The architecture for XYZ has been elaborated sufficiently to present a series of architectural and design recommendations in this Service Layered Architecture Profile.

## Architecture Guidance Disclaimer


The Enterprise Architecture Team is committed to an architecture approach centered on enablement before enforcement. If at any time the architectural constraints prescribed here-in are not sufficiently enabled to meet the Program schedule, the HUD Architecture Waiver Process may be employed. The HUD Enterprise Technical Architecture was instituted to consolidate architectural and design guidance at HUD. This guidance will evolve, and periodic changes to the guidance will ensue as institutional experience applying these best practices provides feedback to HUD EA on a project-by-project basis. The purpose of the guidance is to simplify HUD's portfolio of applications; ensure interoperability; secure HUD systems; protect citizen's privacy; promote reuse; and improve information assurance.

If you find that this guidance is hindering your solution development, please contact Enterprise Architecture (**EnterpriseArchitecture@Hud.Gov**). Enterprise Architecture has established both a waiver process and an architecture change process to allow project teams to adjust their architecture compliance to an appropriate level for each project. The architecture change process allows program and project teams to influence the enterprise

architecture by making recommendations for technology, product and standards insertion into the EA or for presenting other recommendations to improve the enterprise architecture. Please do not hesitate to contact enterprise architecture with any recommendations, questions or concerns.

## Background

This section provides background information on XYZ for each of the three categories listed above.

### Investment Selection Information

The background presented here is an excerpt from *The Department of Housing and Urban Development, DME Activity Summary: Access to Credit with XYZ*.

"The Access to Credit initiative mitigates some of the deleterious and lingering effects that resulted in the housing market from the subprime mortgage crisis, global economic meltdown, and subsequent economic recession. This initiative directly supports the following HUD Strategic Goals:

- SG1: Strengthen the nation's housing market to bolster the economy and protect consumers
- SG3: Use housing as a platform to improve quality of life
- SG4: Build strong, resilient and inclusive communities

The primary IT objective of this project [XYZ] is to enable the implementation of FHA's new defect taxonomy and methodology for evaluating underwriting defects as the foundational component of XYZ. FHA will refine the way loans are assessed for manufacturing defects, linking defect severity to the outcome of the defect while also providing transparency into the source and cause of the defect. This new basis for rating loans will be used consistently when performing a variety of loan reviews including post-endorsement technical reviews and on-site mortgagee monitoring reviews. FHA's goal is to have a single-issue trigger, a single defect code, and have the severity level and cause of defect indicated in the [defect] code." [1]

### HUD Business Architecture and Loan Review Process Requirements

HUD undertook a loan review business process reengineering effort that competed in October 2015. A subsequent effort to define the requirements in a format suitable for initiating an agile development effort produced approximately 60 user story descriptions in November 2015. These work products and their associated references were the primary sources of information in this category that were considered in the development of the XYZ SLAP. The documents referenced include:

- HUD Loan Review System Business Process Reengineering Results: Loan Review Current Business Process.
- HUD Loan Review System Business Process Reengineering Results: Loan Review Target Business Process.
- HUD Loan Review System Requirements.

In addition, the references identified in the business process and requirements' documents were also analyzed on an as-needed basis. These included a review of the relevant HUD forms and the documents that make-up the loan review binder.

A follow-up effort to refine the XYZ requirements is currently underway. The XYZ SLAP will be revised to reflect the results of this requirements effort once it is complete.

## Enterprise Architecture Guidance

HUD OCIO architecture guidance and standards is a collection of recommendations, processes, models, reviews, and governance best practices. HUD architecture management processes are designed to ensure that all HUD IT

investments are selected, architected, acquired, implemented, deployed and maintained in congruency with HUD OCIO architectural guidance.

HUD investments are reviewed by the HUD OCIO Chief Architect to ensure alignment and interoperability with the HUD Enterprise Architecture. Enterprise Architecture reviews the intended outcomes of each investment to identify opportunities including, but not limited to, the following:

- Advance HUD's transition to a simplified portfolio of systems.
- Provide immediate savings through consolidation of investments in similar capabilities.
- Provide near-term savings by leveraging assets within HUD's current IT portfolio.
- Incorporate modular designs to ensure flexibility.
- Standardize technology to reduce the number of HUD's technology and product dependencies.

Today, HUD spends an inordinate amount of its IT budget on maintenance and operations. Investment in, and commitment to, architecture is necessary to systematically improve the maintainability and manageability of HUD's applications portfolio. HUD architectural guidance integrates industry best practices into a comprehensive resource for all application development efforts. The architectural guidance is continuously improved.

**Exhibit 1 – HUD Architecture Guidance Resources** lists architecture guidance resources relevant to XYZ implementation. The following list is not exhaustive, and other architecture guidance may also apply:

| Name / Title | Description | POC or Universal Resource Locator (URL) Address |
|---|---|---|
| **XYZ Business Process Model and Requirements** | Draft business process model developed by EA (Oct 2015) and supporting User Stories drafted in (Nov 2015). | EnterpriseArchitecture@Hud.Gov |
| **Project Portfolio Management (PPM)** | The Department of Housing and Urban Development's (HUD's) Project Planning and Management (PPM) Life Cycle V2.0 provides practical approaches to optimize innovation, minimize schedule and budget risk, and better plan and execute projects. | http://portal.hud.gov/hudportal/HUD?src=/program_offices/cio/ppm/PPMV20Artifacts |
| **Security** | For information regarding application security requirements please contact the HUD Chief Information Security Officer. | HUD Chief Information Security Officer |
| **Enterprise Technical Architecture Version (ETA) 1.1** | The HUD ETA documents outline layered and tiered architecture principles and standards for developing modern applications. This document provides an overview of the HUD ETA, the layers of that architecture, the security required to safeguard it, and the infrastructure required to operate it. This document is pertinent to application and system stakeholders. | EnterpriseArchitecture@Hud.Gov |
| **Common Application Relational Schema (CARS)** | The Common Application Relational Schema (CARS) logical data model is the Enterprise Data Model for HUD. The purpose of the CARS model | EnterpriseArchitecture@Hud.Gov |

| Name / Title | Description | POC or Universal Resource Locator (URL) Address |
|---|---|---|
| | is to define the data needed by HUD to fulfill the organization's mission using a methodology to allow the model to be used by new development projects to establish the foundation of the data model and data base. | |
| **HUD Cascading Style Sheet** | A Cascading Style Sheet template that may be used for development of websites and web-based applications. | http://portal.hud.gov/hudportal/documents/huddoc?id=hudapril.css |

**Exhibit 1 – HUD Architecture Guidance Resources**

To ensure that the architecture recommended in this XYZ SLAP is appropriately modified to better support XYZ solution delivery without denuding the enterprise value designed into these recommendations, there must be extensive collaboration between the XYZ Program, the XYZ Implementation Team and Enterprise Architectural Design Services (ADS). Consistent, daily or weekly, engagement between the teams is anticipated.

In the spirit of enabling compliance, Enterprise Architecture will supply the XYZ implementation team with:

- A development environment and a test environment in Azure that includes developer workstation virtual machines (VM) and application server VMs that have the software prescribed herein preinstalled.
- An executable architecture framework for the XYZ solution that is consistent with all recommendations presented in this SLAP.
- Pre-integration of available enterprise services.
- Guidance for elaborating and extending the HUD standard architecture in support of XYZ solution delivery.

Collectively, these development aids will speed solution delivery and help facilitate compliance with HUD architecture guidance.

## Scope

The scope of the XYZ SLAP is the XYZ investment and related software development. This document does rely heavily on the loan review business process reengineering effort and subsequent requirements' definition effort. Changes to the target loan review business process and requirements will likely impact this XYZ SLAP requiring commensurate changes to bring this document back into alignment with business needs.

## Audience

The primary audience for this document is the lead architect for any team proposing to implement the XYZ system. Other stakeholders that may find value in the XYZ SLAP include:

- HUD & FHA executives and management
- Application development managers
- Contractors considering submitting proposals to implement the XYZ system
- HUD OCIO Staff that have a role to play in support
- Government oversight organizations

### HUD Reference Architecture Clarification

HUD's architecture strategy warrants some clarification because it consists of two layered architectures intended to dovetail with each other. The first is the **Enterprise Technical Architecture (ETA)** which lays out guidelines for developing three tiered applications. The tiers included are the Presentation Layer, the Business Logic Layer and the Data Access Layer. The guidance documented in the ETA applies to applications, specifically custom-built open-source applications.

The second architecture is The **HUD Layered Service Architecture (see *Appendix A: HUD Layered Service Architecture Guidance*)**. This architecture identifies the layers of an enterprise service architecture designed to enable reuse and the management of enterprise services. This architecture applies to integrating applications. Design patterns for implementing services for each layer are included later in this guidance.

## Purpose

The purpose of this document is to provide architecture guidance for the upcoming XYZ implementation effort. This document prescribes architecture guidance in the form of recommendations. This document provides the architectural blueprint for development of the XYZ application and a framework for future enhancements and maintenance.

## Document Conventions

This document prescribes architecture for XYZ implementation and employs the following conventions to differentiate between mandatory guidance and recommendations.

Mandatory Compliance: These guidelines are identified by the keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT". Exceptions require a waiver and a transition plan.

Recommended use: These guidelines are identified by the keywords "SHOULD", “RECOMMENDED”, "SHOULD NOT", “NOT RECOMMENDED”. These guidelines describe a preferred alternative as judged by HUD EA. Deviation should only be on a limited use and justified by the circumstances.

# Architecture Recommendation Summary

System Summary Statement: HUD has established a standard defect taxonomy to simplify and clarify feedback to lenders. The business domain is the management of loan cases and XYZ will automate parts of the loan case review process. The solution requires integration with other HUD systems to retrieve loan and lender data. Complex business rules are employed to support the loan selection process, loan assignment process and to map results to the defect taxonomy.

## Case Management Capability

The central capability of XYZ is Case Management. The core aspects of Case Management are document management, collaboration, reporting, workflow management and security. However, the workflow and lifecycle are relatively simplistic. The complexity in XYZ lies in other areas, specifically in the following:

- Automating the application of the defect taxonomy to improve the consistency and transparency of loan review results.
- The application of a complex set of business rules to pre-select loans across 9 process areas/categories that meet varying criteria.
- The ability to assess the Home Ownership Center’s (HOC) worker skills and qualifications to review different categories and subtypes of loans requires sophisticated rules’ management.
- The ability to interoperate/interface with other HUD systems requires integration design patterns and data transformations.
- Providing a control panel for the National Coordinator and a dashboard for other executives that provides a holistic view of the current year loan review progress.

Furthermore, HUD has over 70 case management systems within its portfolio. Most of these case management systems are implemented using different COTS products. During the investment review pre-select phase, it was determined that none of these systems should be leveraged to support implementation of the Loan Review System. Instead, the decision was made to implement an open source, custom built solution for the loan review system in order to establish a standard HUD case management platform.

## Case Management Market Analysis

The Case Management arena is filled with hundreds of software products with varying degrees of specialization. This market space has not yet undergone consolidation. As a result, there is a high risk that any off-the-shelf product selected could be jeopardized as the market place consolidates. Most likely, products selected will not survive the anticipated longevity of the investment, which can span more than a decade. Furthermore, the COTS products available do not address the primary areas of complexity for the loan review process. Extensive configuring, hand-coding of business rules and tailoring of COTS software would be required to implement XYZ.

Today, we have already identified approximately 70 systems at HUD that implement Case Management capabilities. XYZ provides an opportunity to implement a simple, open source case management capability at HUD. Once implemented, this low-cost alternative may be extended to support other programs, driving consolidation of some of the case management systems at HUD onto a standardized platform and thereby eliminating many of the unnecessary and costly technology and product dependencies related to these programs.

The following table lists the recommendations from the ADS team for building out and deploying an XYZ application in a HUD environment. As a proof of concept, the XYZ application is being built out by the HUD OCIO/EA/ADS team, based on the architecture recommended in this document, and it is being readied for demonstrations to OCIO in late Jan/early Feb 2016.

## Architectural Recommendations for Implementing the XYZ Application

**Exhibit 2 - Recommendations for XYZ Applications** presents an overview of the architecture guidance for XYZ implementation. Each recommendation is elaborated in a subsequent section.

| | Recommendation | Rationale | Notes |
|---|---|---|---|
| 1 | **XYZ SHALL Be Developed As A Custom-Built Open Source Solution Deployed To The Azure Cloud** | Open Source software provides a low-cost, low-risk alternative to proprietary software. HUD seeks to establish a standardized, managed DevOps environment to speed delivery and reduce the cost of custom-built solutions.<br>This option provides maximum flexibility to HUD in terms of maintenance & operation of the system as business rules and needs change over the years. | HUD OCIO has already made the investment to establish the open-source architecture. |
| 2 | **OCIO will provide Development/Test as a Service (DTaaS)** | The XYZ project will have direct access to and the support of the Enterprise Architectural Design Services team and the HUD CI/CD team to facilitate the use of the provisioned environment. This will expedite delivery of a quality XYZ solution.<br>Expedite delivery by enabling solution team to start work on Day 1. | Improved system quality due to the automated continuous testing capabilities inherent in the HUD CI/CD environment.<br>Reduced total cost of ownership as result of improved system quality. |
| 3 | **XYZ SHALL Be Implemented In Accordance With The HUD Layered Service Architecture** | Establishing interfaces that align with HUD's Enterprise Capability Model and HUD's Layered Service Architecture will provide investment and technology flexibility in the future, simplify HUD's application portfolio, and provide assets that can be extended for enterprise reuse and consolidation. | Positions capabilities developed for XYZ to easily identified and reused in support of subsequent projects. |

| | Recommendation | Rationale | Notes |
|---|---|---|---|
| 4 | **XYZ SHALL Employ Design Best Practices** | Employing consistent design patterns simplifies downstream integration and provides higher degrees of information assurance.<br>The ADS team will provide sample executable code and an application framework that demonstrates the anticipated use of these design patterns for XYZ. | Expected to reduce out-year maintenance and support costs |
| 5 | **XYZ SHALL Employ Utility Services** | As part of custom code development, expose utility services securely as REST based APIs so that other applications do not have to rewrite the business logic. They can simply invoke the RESTful services and perform the desired operation. | Expected to reduce out-year maintenance and support costs |
| 6 | **XYZ SHALL Integrate With The Following known HUD Systems. List will be updated as final scope and business requirements are delivered to OCIO.** | CHUMS, LEAP, AARTS, SFHEDW<br>Please see **Error! Reference source not found.**.<br>The XYZ solution will provide reusable integration patterns for accessing these systems. | These interfaces will be factored in to the cost of any XYZ solution. |
| 7 | **XYZ SHALL Use HUD ESB** | HUD has selected MuleSoft's Enterprise Service Bus (ESB) and established a policy that all applications will utilize the ESB for systems integration. | Will reduce the impact of change between systems integrated using the ESB. |
| 8 | **XYZ SHALL Use HUD EDW For All Data Warehouse Needs** | The data warehouse will provide reliable and consistent data for the enterprise. This will eliminate inconsistencies between isolated views of data that can come from reporting on individual subsystems. | The Enterprise Architecture Enterprise Data Warehouse team will implement any reporting and data warehousing requirements for the XYZ solution. |
| 9 | **XYZ SHALL Align To HUD CARS Data Model** | The Common Application Relational Schema (CARS) logical data model is the enterprise data model for HUD. Alignment to the CARS model and its methodology and design approach will ensure data quality, adhere to consistent business rules, and institutionalize HUD data standards. | The Enterprise Architecture Enterprise Data team will develop CARS data model which will be used by XYZ solution. |
| 10 | **XYZ SHALL use ICAM solution to implement application security** | ICAM solution (TBD) will be used for HUD partners and HUD PIV personnel to authenticate and authorize into the applications. | The Enterprise Architecture team recommends integrating with existing ICAM Solution (infrastructure) but build out ICAM functionality to standards so that there is easy migration to another platform/solution down the road. |

**Exhibit 2 - Recommendations for XYZ Applications**

The following sections will address each recommendation in more detail.

# Recommendation 1: Cloud First - XYZ SHALL Be Developed as a Custom-Built Open-Source Solution Deployed To Azure Cloud

HUD is committed to establishing the capability to reliably deliver custom-built open-source solutions. These solutions provide a lower cost of entry than traditional proprietary software. These solutions also reduce the chance that software products become obsolete due to vendor acquisitions or bankruptcies. Furthermore, employing open-source technologies ensures that HUD will always have open access to the source code that supports their solutions without any of the licensing constraints or risks associated with proprietary software products. As business rules and needs change over the years, this option will provide maximum flexibility to HUD in terms of the maintenance and the operation of the system.

Furthermore, HUD has made an initial investment in Cloud infrastructure-as-a-service. This initial investment in MS Azure SHALL provide the target operating environment for XYZ.

## The HUD Open-Source Application Technology Profile

HUD Enterprise Architecture has identified the application technology profile presented in **Exhibit 3 - Application Technology Profile Overview** for XYZ.

| Technology | Product Name |
|---|---|
| **Java** | Java 8 |
| **Application Build** | Maven |
| **Database** | Oracle |
| **Data Access Layer** | Hibernate |
| **Application Server/Web Server** | Tomcat |
| **Business Service (Integration Layer)** | Mule |
| **BPMN/Workflow Software** | Acitiviti |
| **Message Queue Software (Async)** | TBD |
| **Rules Management** | Drools |
| **User Interface** | HTML/AngularJS/Bootstrap/JQuery |
| **Operating Systems** | Linux |
| **Document Storage Software** | MongoDB |

**Exhibit 3 - Application Technology Profile Overview**

HUD has already invested in establishing a working proof-of-concept of this application technology profile. The proof-of-concept has been tailored to support XYZ. ***The proof-of-concept will be structured as an executable reference implementation and will be provided to the application development contractor as a jump-start.***

## Open-Source Reference Implementation Architecture

**HUD EA is working to provide a tested reference implementation architecture to the XYZ application development team.** The reference implementation architecture will be delivered in the form of code, how-to documentation, javadocs and services as example. All these deliverables will be available in git repository and will be handed off during the onboarding process for XYZ implementation team. The reference implementation is iteratively developed and incrementally delivered during the architecture runway[1].

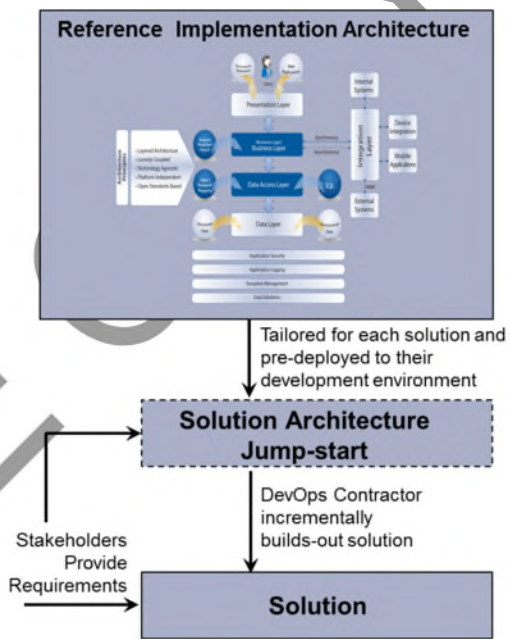



**Exhibit 4 - Value Proposition for HUD Solution Delivery**

**Speeds Delivery** - Will be used to jump-start solutions that fit a prescribed profile by providing a working (executable) framework for the solution as a starting point.

**Reduces Risk** - Reduces architectural complexity for solution architects by standardizing solutions to object relational mapping, referential integrity and other architectural challenges.

**Simplifies Development** - Pre-Integrating standard utility services /components such as authentication, authorization, error handling, audit logging etc.

**Promotes Congruent System Boundaries** – Consistent separation of concerns enables reuse and more effective team organization.

The HUD architecture process includes continuous engagement with the EA team for designated projects. During this period, HUD ADS will define key architectural user stories to be addressed during the architecture runway. The

[1] Architectural Runway provides the technology and architecture enablement necessary for timely, uninterrupted development of new features. This enablement is achieved by ensuring that the affected elements — components, functions, protocols, internal system functions, etc. have the capabilities necessary to support the near-term features on the product roadmap.

To support continuous high delivery velocity, architectural runway needs to be continuously maintained and extended. Enterprise Architecture team builds and extends Architecture Runway to assure continuous investments in enablers. Product/Solution Management together with Architects in collaboration with the business teams, define many of these architectural initiatives, but implementation is the responsibility of the Agile Release Trains.

While providing the enablement for near term delivery success, architectural runway should not over-constrain the development with long-range technical commitments. “Just the right amount” of architectural runway is required. Too much, and the architecture over-constrains the teams and is too disconnected from the current context; too little and the teams will have trouble reliably making and meeting near term commitments.

Continuously extending the architectural runway, exploring and preserving design options, and supporting Adaptive Requirements and Design increases the organization’s ability to respond to technology and business challenges.

ADS team will maintain continued, regular engagement with the project teams through multiple reviews, code validation and through consultancy on architectural issues.

# Recommendation 2: OCIO will provide Development/Test as a Service (DTaaS)

HUD OCIO will provide Development and Test as a Service (DTaaS) in HUD Azure Cloud, this environment includes the ability to automatically provide Developer Workstation Virtual Machines that have been preloaded with the development products identified in **Exhibit 5 - Development Environment Technology Profile**. It also includes the ability to largely automate the provisioning of application servers to support Integration and test environments for project teams.

The following development environment technology profile presents the products that will be preconfigured in the development and test environments for the XYZ application development team.

| Technology | Product Name |
|---|---|
| **Java** | Java 8 |
| **IDE** | Spring Boot with Spring Tool Suite (STS) |
| **Editors** | Notepad++ |
| **UI Development** | HTML/AngularJS/Bootstrap/JQuery |
| **Database Tools** | SQL Developer |
| **Application Build** | Maven |
| **Database** | Oracle |
| **Data Access Layer** | Hibernate |
| **Application Server/Web Server** | Tomcat |
| **Business Service (Integration Layer)** | Mule |
| **BPMN/Workflow Software** | Acitiviti |
| **Messaging Software (Asynchronous)** | TBD |
| **Rules Management** | Drools |
| **Testing Software** | Salinium, Geb, Mockito |
| | Junit |
| **Operating Systems** | Windows 7/8/10 |
| | Linux |
| **Version Control** | Git |
| **Document Storage Software** | MongoDB |

**Exhibit 5 - Development Environment Technology Profile**

HUD EA CI/CD Team will provide support for the Development Test as a Services (DTaaS) environment.

## Recommendation 3: XYZ SHALL Be Implemented in Accordance With HUD Layered Service Architecture

HUD's application portfolio simplification strategy is centered on management of the HUD Layered Service Architecture. By ensuring that systems are congruent with a common capability model that employs consistent rules for integration, OCIO can incrementally deliver a more flexible and responsive application portfolio.

This section presents the results of mapping the XYZ business process model, user stories and requirements to the HUD Enterprise Service Layered Architecture. The results are presented as a service architecture for XYZ that is congruent with the ETA. This congruency is critical to enabling downstream consolidation around targeted capabilities and also for reducing the cost of promoting XYZ solution modules to enterprise assets.

For information on the HUD Enterprise Service Layered Architecture, please refer to

Appendix A: HUD Layered Service Architecture Guidance.

## XYZ Specification Architecture

The XYZ Specification Architecture presents a logical architecture that defines the optimal set of interfaces to be implemented for best alignment with the HUD Layered Service Architecture. Enabling the architecture to flex at these key interfaces will align the XYZ application points of flexibility with the model that the HUD investment review board will be basing their decisions on.

The series of figures in *Appendix D: XYZ Specifications* present the specification view and deployment view for XYZ. The specification figures were derived from the Loan Review System (XYZ) - To-Be Process Flows [5], the XYZ Demo user-stories [6] and the HUD XYZ Demo Functionality [7].

A subsequent Loan Review System requirements effort is currently underway. The specification models presented will require review and modification to reflect changes to the requirements that were the basis of the original design effort.

The XYZ specification view presents a logical service-oriented architecture for XYZ. The specification view indicates a collection of interfaces that can support the XYZ target business process and supporting requirements. The specification model presented here was designed to support the loan review business process reengineering effort that delivered a target architecture in October 2015 and the subsequent requirements elicitation effort that delivered 53 user stories to support the target-state HUD enterprise loan review business process. The specification architecture presented here is in compliance with HUD Layered Service Architecture.

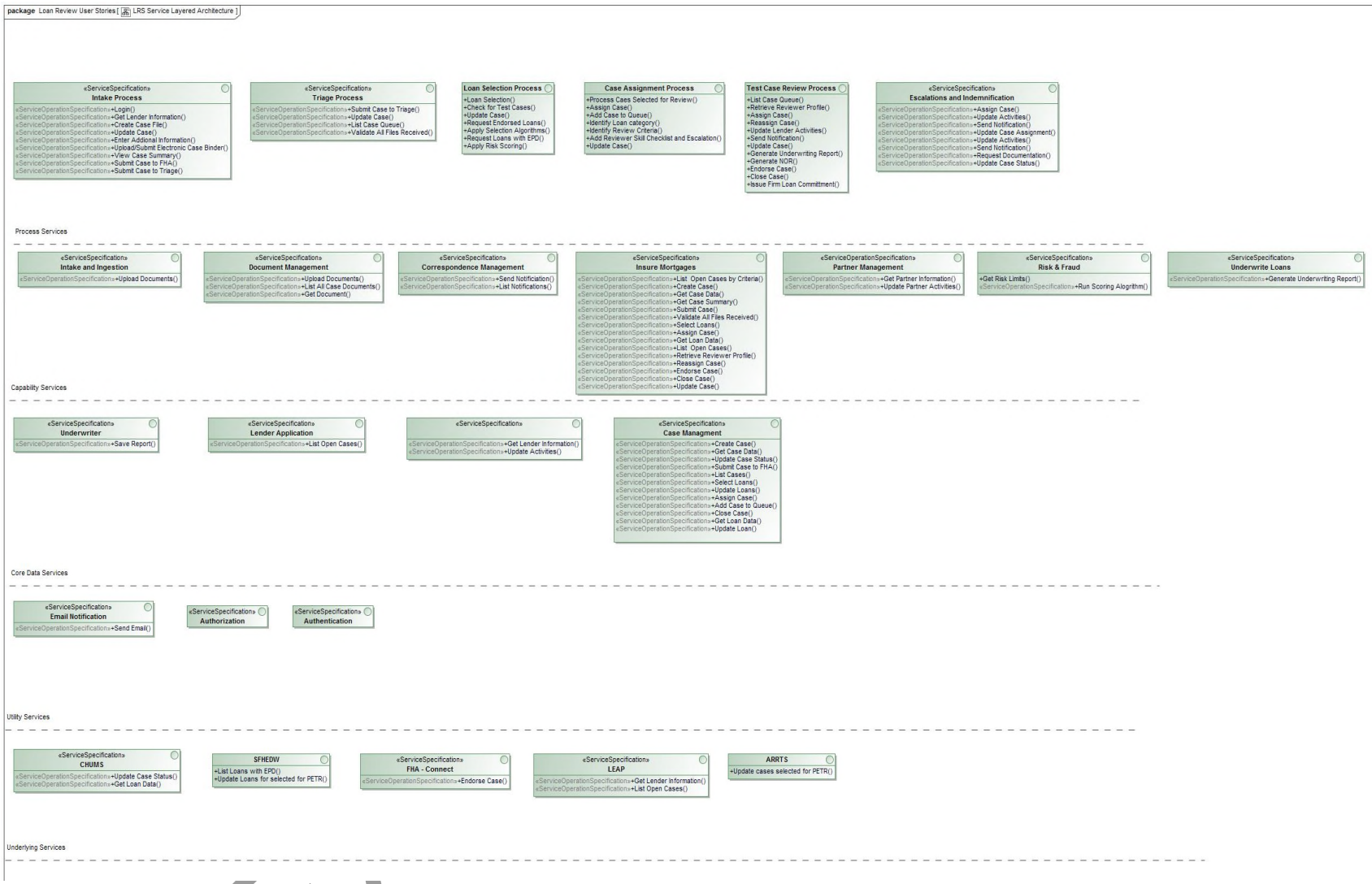



**Exhibit 6 – Overview of Loan Review System Specification Architecture**

*Exhibit 6 – Overview of Loan Review System Specification Architecture* provides a top level view of the service components specified to support the loan review system. *Appendix C: XYZ Specification* provides a dependency diagram to support each of the major loan review process steps.

## Service Layer Examples

The layered services architecture defines a layered framework for organizing enterprise services (see

Appendix A: HUD Layered Service Architecture Guidance). Each layer plays an important role in separating enterprise concerns and project concerns and each layer has different management and operations requirements. For example, the process layer is responsible for interacting with the UI components, and it acts as a controller or a channel that routes requests to other service components within the architecture. Process Services orchestrate the invocation of Capability Services and Core Data Services in order to fulfill requests from the User Interface. The process layer also has access to a BPM Engine (Activiti) and a Rules Engine (Drools) to define the workflow and business rules management that are frequently required by a Process Service. The service layer provides services via HTTPS protocol in accordance with the M-15-13 document (full guidance at https://www.cio.gov).

This section presents example internal implementations for a Process Service, a Core Data Service and an Underlying Service. For the purposes of XYZ, capability services may be implemented as process services.

**Process Service Layer Example**

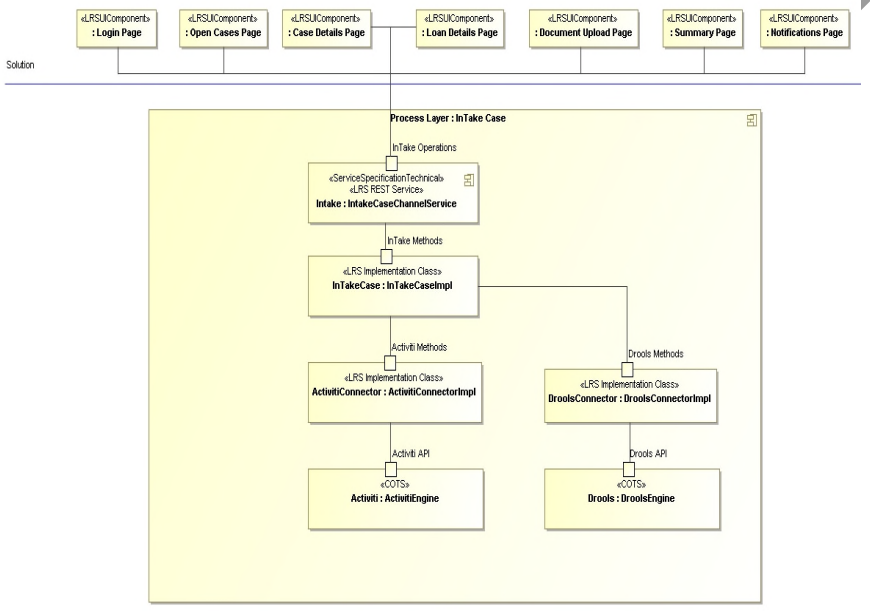



**Exhibit 7 – Example Process Layer Service Internal Design**

**Exhibit 8 – Description of Example Process** Layer provides a description of each component in *Exhibit 7 – Example Process Layer Service Internal Design*. This is a typical example of how a Process Layer component would be packaged but it can be augmented with other components needed to meet a particular requirement. The Process layer component is typically called from the UI or Presentation Layer.

| Component Name | Description |
|---|---|
| **Solution Layer** | User interface components that may call Intake Case service. |
| **Intake Case Channel Service** | An "XYZ REST Service" endpoint with operations exposed for use by the UI. |
| **Intake CaseImpl** | The "XYZ Implementation Class" that implements the necessary business logic that is needed for the Intake Case Channel Service. |
| **ActivitiConnectorImpl** | A common "XYZ Implementation Class" that needs to be called for any interaction with the BPM Activiti Engine. This Implementation class is responsible for any caching and abstraction required by XYZ. |

| | |
|---|---|
| **ActivitiEngine** | The Java API that Activiti exposes for the ActivitiConnectorImpl to consume. |
| **DroolsConnectorImpl** | The common "XYZ Implementation Class" to be called for any interaction with the Drools Rules Engine. This Implementation class is responsible for any caching and abstraction that is needed specific for XYZ. |
| **DroolsEngine** | This is the Java API that Drools exposes for the DroolsConnectorImpl to consume. |

**Exhibit 8 – Description of Example Process Layer Service Internal Design**

**Core Data Service Layer Example**

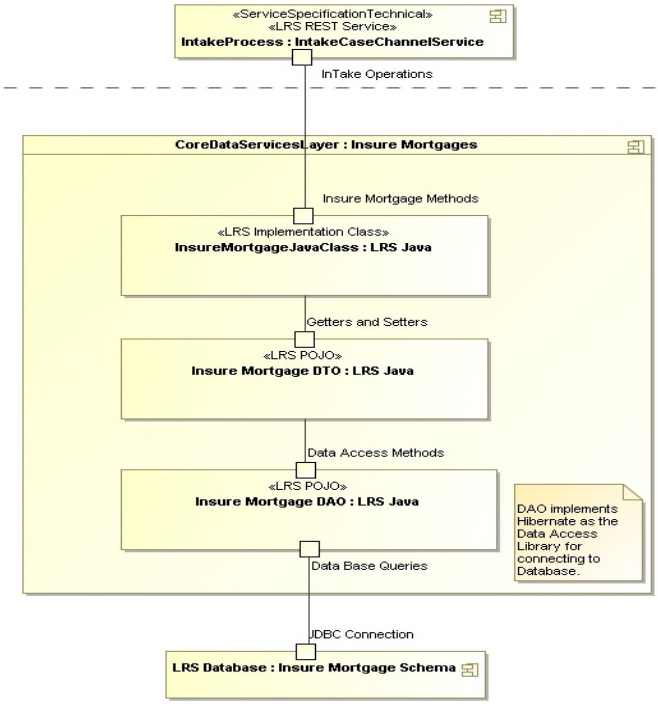



**Exhibit 9 – Example Core Data Service Internal Design**

*Exhibit 10 –Core Data Service Internal Components* provides a description of each component in Exhibit 9. This is a typical example of how a Core Data service might be implemented and packaged. Core Data services components are typically called from the User Interface, Capability or Process Service layers.

| Component Name | Description |
|---|---|
| **InsureMortgage Java Class** | The "XYZ Implementation Class" that implements the business logic for all the service operations that are exposed by the Intake Channel REST service. |
| **InsureMortgage DTO** | A "XYZ Java" or a POJO that holds the data obtained from a persistent object (From the Database) after necessary transformation. |
| **Insure Mortgage DAO** | A "XYZ Java" or a POJO that contains methods to issue CRUD operations against the XYZ database. This needs to be |

| | |
|---|---|
| | implemented using Hibernate and requires integration with Spring Boot. |
| **XYZ Database** | The XYZ oracle database where the loan review data would reside. |

**Exhibit 10 –Core Data Service Internal Components**

**Underlying Service Layer Example**

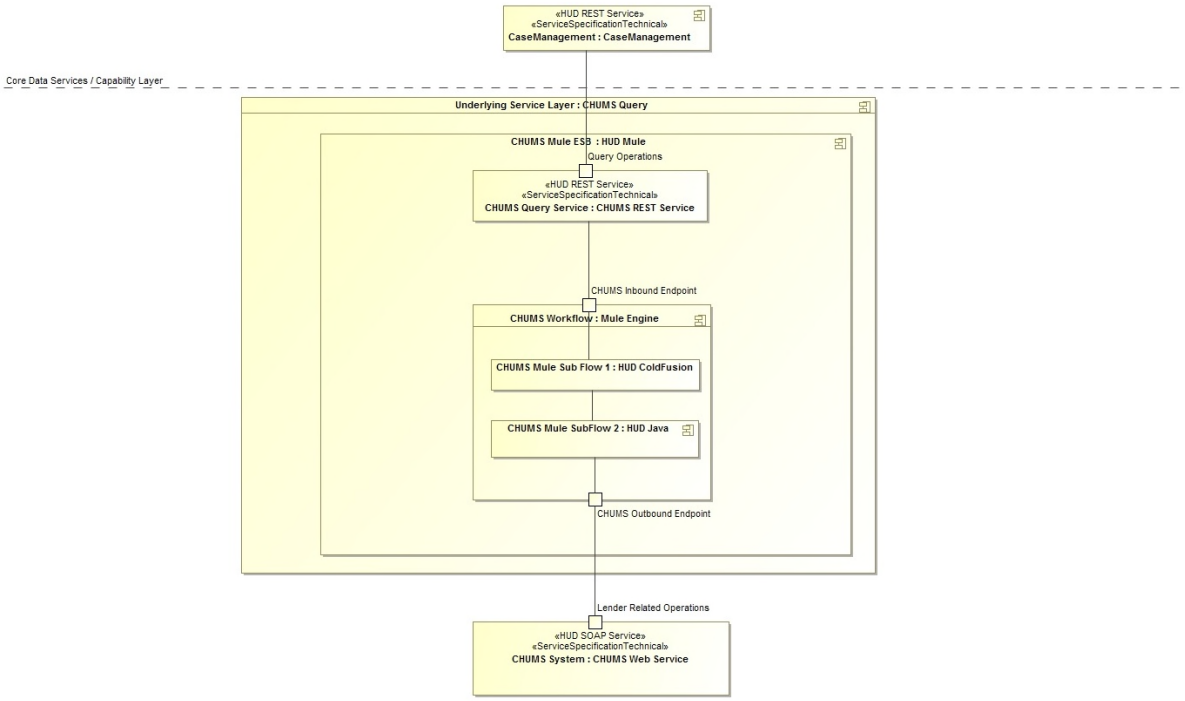



**Exhibit 11 – Example Underlying Service Internal Design**

**Exhibit 12 – Underlying Service Internal Components** provides a description of each component in *Exhibit 9 – Example Core Data Service*. This is a typical example of how the Underlying Services Layer component would be packaged. The underlying services layer components are typically called from the Capability or the Core Data services layer. The underlying services are packaged within a Mule ESB component that encapsulates the internal details of accessing and transforming data from external systems.

| Component Name | Description |
|---|---|
| **CHUMS REST Service** | This is a "HUD REST Service" endpoint with operations that is exposed to the consumers of this service. In this case this would be the Case Management capability layer service component that will access this endpoint to query against external systems. |
| **Mule Engine** | The Mule Engine is the backbone of the underlying services component that provides the EAI capabilities to integrate with other HUD systems as well as external systems. |

| | |
|---|---|
| **HUD ColdFusion** | Mule workflows typically have other sub flows and supports development in via various scripting languages. This is an example of a HUD ColdFusion component that can be part of a Mule workflow. |
| **HUD Java** | Mule workflows typically have other sub flows and supports development in via various scripting languages. This is an example of a HUD Java component that can be part of a Mule workflow. |
| **CHUMS WebService** | This is the legacy web service that the CHUMS system provides to all HUD systems to access Loan data. |

**Exhibit 12 – Underlying Service Internal Components**

# Recommendation 4: XYZ SHALL Employ Design Best Practices

The market place is rife with guidance on industry best practices but it is difficult navigate between best practices that are mutually compatible and can be supported by any given organization. Architecture capability maturity, organizational culture and IT management practices influence the efficacy of best practices that can be employed at HUD. When selecting the design best practices incorporated in this section considerable thought was given to the environmental constraints above. In particular, we are seeking to employ "state-of-the-practice" as opposed to "state-of-the-art". We seek design guidance that is thoroughly proven in a broad array of organizations and technology platforms.

## Model/View/Controller design pattern

One of the most important design patterns that is recommended by ADS team is **Model/View/Controller (MVC)**. The MVC pattern is applied at the application architecture level and promotes effective separation of concerns for the application components and subsystems. MVC consists of three kinds of objects - **Model** is the application object, the **View** is its screen presentation, and the **Controller** defines the way the user interface reacts to user input. There are many benefits of using MVC design pattern including ease of modification of UI as business needs change.

The HUD ETA provides in-depth guidance for the use of MVC in a JAVA environment; however this does not provide holistic guidance for the architecture prescribed for XYZ. As EA seeks to advance the enterprise target architecture, EA will work with the XYZ implementation team to apply this pattern to the new application profile prescribed in *Exhibit 3 - Application Technology Profile Overview*.

## Intended use of JMS and MuleSoft ESB

XYZ will employ both Java Message Service (JMS) and MuleSoft Enterprise Service Bus (ESB). For asynchronous messaging within XYZ application, it is recommended that JMS should be used to send/receive messages. For use cases where XYZ needs to send/receive messages to/from other enterprise applications at HUD, it is recommended that an Underlying Service be implemented as an Adaptor Design Pattern providing access to other system functionality from within the XYZ application. See *Exhibit 11 – Example Underlying Service Internal Design.*
The Java Message Service (JMS) API is a Java Message Oriented Middleware (MOM) is terrible for synchronous request-reply as it will fail slowly (i.e. wait for timeout) when the server is down. When a request is going to fail, you want it to fail fast. A HTTP request to a REST based service will fail immediately (on the TCP connect) if the server is down.

MOM is useful for asynchronous request-reply messaging, but then you'll be left with the problem of where to store the state in-between the request and the reply (Hint: Your options are File or Regular Database, the Message or a NoSQL Database). Often the extra implementation effort is not worth the perceived advantages of asynchronicity. Also REST based services do support asynchronous requests if you really need it.

## Rules Engine

A rules engine solution is overkill for many applications. Choose a rules engine if you can answer "Yes" to all of the following questions:

- Does the algorithm involve significant conditional branching or decision making?
- Are 3 or more conditions present in the rules (i.e. are the rules complex)?

- Are the rules subject to periodic change and/or localization?
- Is the code to be maintained over time?
- Is performance not among the main driving concerns of the system?
- Can the project afford the cost and schedule of a rules solution? e.g., licensing, training, projects must have a duration greater than a year for the ROI to pay off
- Rules are not intended to be used in a procedural manner

## Transaction Management

XYZ will be built on idempotent principle, which means that repeated running of a transaction will not cause any harm to the system. The database design will ensure that duplicate data is not written to the database. During a transaction, if no errors are encountered, transaction data will be committed to the database. Otherwise, the transaction will be rolled back, errors corrected, and the transaction will be tried again. Repeated occurrence of error will be noted and the transaction will be marked as failed. The transaction will be managed in small transaction boundaries.

## System Monitoring

XYZ system will need monitoring by an enterprise class monitoring platform. If there are anomalies found during day-to-day operations of the application, alerts will be sent out via emails, text messaging and phone calls to the operations team and corrective action will be taken. The XYZ system will be designed for resiliency and will employ high availability features of the Microsoft Azure Cloud to ensure a high degree of uptime for XYZ application and proper monitoring of the system is an integral part of the LS resiliency strategy. Today, there is no HUD EA standard open-source monitoring tools (e.g. Nagios) for solutions in the Azure Cloud. HUD EA will work with the XYZ implementation team to determine the XYZ monitoring and instrumentation requirements and select the appropriate monitoring tools.

## Logging

HUD XYZ system SHALL provide Application Logging to log warnings, errors and access from XYZ application. Two important aspects that logging helps with are: (a) code debugging and (b) reviewing application access trail (audit). XYZ SHALL require Secure Auditing feature. Access logs need to be archived in a timely manner for auditing purposes. This would mean tracking data to a transaction level. The logging mechanism SHALL provide ways to "redact" Personally Identifiable Information (PII) from audit logs. Audit logging tools e.g. ELK stack can be used for log viewing purposes – this will help O&M team in viewing the log information without logging to production environment.

## Development/Test as a Service (DTaaS)

XYZ Application will be developed and deployed in an agile manner using industry best practices and within the Dev/Test as a service (DTaaS) environments provided by HUD OCIO; DTaaS are designed and configured to achieve Continuous Integration/Continuous Delivery (CI/CD) using tools for:

- Infrastructure Automation
- Library & Dependency Management (Sona Type Nexus repository)
- Source Control (e.g. git)
- Developers Code Validation (SONAR)

HUD will provide properly configured DTaaS for all its development and integration to achieve CI/CD. Specific tools are provided as examples only; the CI/CD team will refine the CI/CD platform as required.

## Test Automation

The XYZ system SHALL undergo a series of tests, including but not limited to:

- Unit testing
- Functional Testing
- Performance Testing
- Security Testing
- Load Testing

The XYZ system will provided automated test scripts for integration with the CI/CD environment.

The products under consideration as HUD standards include the following:

- JUnit
- JMeter
- JMock
- Selenium
- Spock

## Recommendation 5: XYZ SHALL Reuse Existing Utility Services

HUD EA will establish standard utility components for integration with the HUD Technology Profile. Sourcing activities are underway but is unclear at this time which utility services will be in place and available for use within XYZ. For utility components that have not been sourced in advance of XYZ implementation the XYZ solution provider will be responsible for implementing the utilities. The minimum set of utility services includes:

- Error and Exception Handling
- Audit Logging
- Authentication/Authorization
- Application Logging
- Email and Notification Services

## Recommendation 6: Integrate with the Following HUD Systems

The following systems maintain data that is required by the Target-State Loan Review Business Process.

Exhibit 13 – XYZ Candidate System Interfaces

| | Candidate System Interfaces |
|---|---|
| **1** | Lender Electronic Assessment Portal (LEAP) FHA's approximately 2,900 approved lenders use LEAP for their annual recertification, and to execute post-approval business updates and changes. |
| **2** | Computerized Homes Underwriting Management System (CHUMS) processes single family mortgage insurance applications, from initial receipt through endorsement. Various types of applications are processed in CHUMS, including loans for First Time Homebuyers, Home Equity Conversion Mortgages (HECM) a reverse mortgage available to the elderly to augment income, substantial rehabilitation of existing properties, and VA Certified FHA loans. In addition to tracking and processing assistance, it provides automated assistance in appraisal and mortgage credit evaluation. |
| **3** | Approval/Recertification/Review Tracking System (ARRTS) is a web-based application that provides for tracking, management, and reporting of incoming lender application and recertification packages, as well as tracking of workload related to compliance reviews performed on FHA approved lenders, Office of Inspector General (OIG) audits, and referrals made to the Mortgagee Review Board. |
| **4** | The Single Family Housing Enterprise Data Warehouse (SFHEDW) is an ongoing, fully operational data warehouse that is the key source of data for anyone who needs Single Family data. The integrated data warehouse contains critical Single Family business data from sixteen (16) sources, mostly from FHA Single Family automated systems. |
| **5** | The Active Partners Performance System (APPS) allows HUD's business partners to manage their company and individual participation information and submit their APPS Previous Participation Certification |
| **6** | FHA Connection gives approved FHA lenders real-time access to FHA systems. Applications available through the FHA Connection include Single Family underwriting, default reporting, portfolio management and others FHA programs. |

## Recommendation 7: XYZ Shall Use HUD ESB

The integration layer is a key enabler for a Service Oriented Architecture (SOA) because it provides the capability to mediate, route, and transport service requests from the service requester to the correct service provider. This layer enables the integration of services through the introduction of a reliable set of capabilities; these include the following: modest point-to-point capabilities for tightly coupled endpoint integration as well as more intelligent routing, protocol mediation, and other transformation mechanisms often provided by an Enterprise Service Bus (ESB). Web Services Specifications via API Modeling Language specifies a binding, which defines a physical location of an available service. An ESB, on the other hand, provides a location-independent mechanism for integration.

The integration layer provides a level of indirection between the consumer of functionality and its provider. A service consumer interacts with the service provider by way of the integration layer. As a result, each service specification is only exposed through the integration layer (such as an ESB), never directly. Integration Layer decouples consumers and providers, allowing for integration of disparate systems into new solutions. A key recommendation from ADS team is to utilize HUD ESB for integration to ensure truly SOA for XYZ.

The ESB is a critical component for delivering HUD a manageable enterprise service architecture and is a key component in establishing a more responsive application portfolio. Typical use of MuleSoft ESB is to provide Protocol Transformation and Data Transformation when interfacing with other HUD systems or external systems.

## Recommendation 8: XYZ SHALL Use HUD EDW for Data Warehouse Needs

The enterprise data warehouse will become the unified source that holds all the business information of the agency by making it accessible all across HUD. It will become the "single version of truth" for HUD. The EDW will ensure that all the data is constantly available for analyzing, planning and reporting purposes. It will impose a standard treatment of data and will grow with the business's needs by adding classifications as they emerge in the business model. Additionally, the EDW will provide full access to all the data in the Agency without compromising the security or integrity of that data. Data will be organized such that it is amenable for ad-hoc analysis by enabling business users to work directly with the data with little or no IT support. The EDW will also allow for advanced reporting and analysis of multiple time-periods.

The benefits of the EDW include:

- Integrated Data – Standardized data, Enhanced Data Quality and Consistency
- Detailed/Summarized Data for various analytical needs
- Support flexible access for decision-makers – Allow for Slicing and Dicing the data
- Enhanced Business Intelligence – Convert data into insights and subsequently into data-driven actions
- Historical Intelligence

## Recommendation 9: XYZ Shall Align To HUD CARS Data Model

The Common Application Relational Schema (CARS) logical data model is the Enterprise Data Model for HUD. The purpose of the CARS model is to define the data needed by HUD to fulfill the organization's mission using a methodology to allow the model to be used by new development projects to establish the foundation of the data model and data base.

The CARS' modeling approach is to create a solution data model instead of a high-level logical data model which has been traditionally created for Enterprise Logical data models. The CARS model methodology design approach defines the entities, attributes and relationships that enable the data model and data base to:

- Ensure Data Quality
- Enforce Business Rules
- Enforce HUD data standards

Alignment to the CARS model and its methodology design approach by all the new development will establish a standard data model that will enable easier data integration and enhance data quality and consistency.

## Recommendation 10: XYZ SHALL use ICAM solution to implement application security

XYZ application will fully integrate with the current Identity, Credentials and Access Management (ICAM) platform and build to industry standards to allow the solution to seamlessly migrate to new and enhanced ICAM platform in the future. ICAM cover the following features/capabilities:

- Authentication
- Authorization
- System to System access
- Services Security

ICAM product selection is underway and it is unclear whether these services will be available prior to XYZ release. During XYZ implementation, the development team will work with the selected ICAM vendor to implement above-mentioned features in XYZ application. If HUD ICAM is unavailable, XYZ will integrate with WASS, FHAConnect and DIAMS for authentication of external and internal users respectively.

For end-to-end secured communications, SAML assertions will be used as the standard approach.

**None of the guidance in this document overrides application security guidance from the HUD Chief Information Security Officer**. If there is a conflict then guidance from HUD CISO takes precedence.

This section defines the requirements in several areas where architectural decisions have not been made or product choices are not in place yet. Nonetheless, these are requirements for XYZ implementation team to develop and deploy XYZ application. Product selection and details of implementation for these items will be worked out and this document will be updated.

# References


[1] Department of Housing and Urban Development, DME Activity Summary: Access To Credit With XYZ - *Appendix A – Initiative: Access To Credit*; Project: Loan Review System (XYZ)/Quality Assessment Methodology; Project Evidence Demonstrating Compliance to Statutory Condition 1. (FY2014)

[2] HUD Loan Review System Business Process Reengineering Results: XYZ Target Business Process.

[3] HUD Loan Review Process Requirements

[4] HUD ETA Documents

[5] Loan Review System (XYZ) - To-Be Process Flows 2015-11-06.doc

[6] XYZ -demo-userstories_v1.xls

[7] HUD XYZ Demo Functionality.ppt

# Appendix A: HUD Layered Service Architecture Guidance

In a Service Layered Architecture, we classify services according to the type of capability they provide. By focusing each service on a particular type of capability, it enables greater modularity and separation of concerns, enabling them to be more easily shared or composed into new services. For example, core business services (entity) provide operations focused on managing the information about a key business resource - like customers, orders, or products. These are independent of the processes that use that information. Process services provide operations that enable solution assemblers to interact with the different steps in the process. In turn, the process services then consume the appropriate core business services.

In this way, Solution assemblers can change a process service, add new operations and such, or add new processes, without impacting the core business services (data services). If we mix these capabilities together, we run the risk for example that the entity information is only provided via a specific process. We see this often and it's just too coarse grained and not a very agile approach. Business processes typically change more frequently than the information the business needs to manage. Many different processes require information about the same entity types and entities. To ensure our architecture is responsive to these change patterns, Core business services often have broader scope of use than a single business process.

**HUD-EAS recommends the following Service Layers for XYZ:**

| Architectural Layer | A Service in this layer | Guidance |
|---|---|---|
| **Process Service Layer** | is designed to provide functions that support one specific business process or sub-process. This service SHOULD be independent from any particular user interface design. | MUST be associated with one Business Process or Subprocess, and no other Business Modeling Element. |
| **Capability Service Layer** | is designed to support a particular business capability. This service SHOULD be independent from any particular business process. | MUST be associated with one Business Capability and no other Business Modeling Element. |
| **Core Business Service Layer (data services)** | is responsible for maintaining records about the instances of particular set of business types. This service SHOULD be independent from any particular business process or business capability. | MUST be associated with one or more Business Types, and no other Business Modeling Element. |
| **Utility Service Layer** | provides common or specialized operations that MAY be consumed by any other business service. | MAY be associated with a Process, Business Capability or Business Type or Business Event in cases where appropriate if helpful, but need not be. |
| **Underlying Service Layer** | SHOULD normally be hidden from solution developers, since invocation is difficult or error-prone in some way. For example, the operations are obscure, they are not in the terminology used by your business, some operations are inappropriate, the operations are highly generic and could get used inconsistently. This service | MUST not be associated with any business modelling element, but … may be associated with one or more Applications, which provide some of the implementation of this service, |

| Architectural Layer | A Service in this layer | Guidance |
|---|---|---|
| | can be "hidden" from solution developers by "wrapping" it within higher level services, and making a rule that solution software and process services do not directly request operations of a service in this layer. | through the "provides implementation" association. |

### XYZ Architecture Layer Rules

| Layer: Main Role | Operations are: | Dependencies allowed: | More rules: | Data Storage: |
|---|---|---|---|---|
| Solution Logic (Presentation): provides an effective end-user experience | Non-existent - this layer does not contain services | May call Process, Core Business & Utility services directly | Supplies user interface, validation, user messages, session management | Not normally, except for temporary session data |
| Process Services (Business): Orchestrate other services; apply business process rules. | UI-independent, but designed for a specific business process | May directly call Core Business, Underlying & Utility services | May directly call Core Business, Underlying & Utility services | Only where not stored by Core Business Services. Stored data likely more transient than for Core Services |
| Business Capability Services(Business): Process independent and Channel (UI) independent business services | UI- and business process-independent, so can be used in different contexts | May directly call other Core Business, Underlying and Utility Services | Cyclic dependencies not normally permitted, except for "call-back". May not call Process Services | Enforce Business rules |
| Core Business Services (Business Entity): Apply enterprise-wide business rules | UI- and business process-independent, so can be used in different contexts | May directly call other Core Business, Underlying and Utility Services | Cyclic dependencies not normally permitted, except for "call-back". May not call Process Services | Maintain the principal data and enforce data integrity rules |
| Utility Services Highly reusable services or those employed to standardize technical infrastructure or to hide complex | UI-, business process- and often domain-independent | May call other Utility Services directly. Some may use Underlying Service. | Cyclic dependencies not normally permitted. | Often required — for directories, look up tables and audit trails for example. |

| Layer: Main Role | Operations are: | Dependencies allowed: | More rules: | Data Storage: |
|---|---|---|---|---|
| business calculations | | | | |
| Underlying Services (Integration) difficult for solutions developers to consume correctly | Highly generic or implementation-leaking, so its interface not ideal for exposing to solution developer | May call Utility Services, but normally would not | May not call Core Business or Process Services | Often maintain significant data stores, and core services call underlying services |

## Appendix B: HUD Enterprise Nonfunctional Requirements

See Excel Spreadsheet Attachment named for this appendix.

# Appendix C: XYZ Specification and Deployment Views

**Exhibit 6** is a Unified Modeling Language (UML) class diagram that depicts the services (classes stereotyped as Service Specification) that participate in the XYZ. The services are arranged in different layers and these layers are described in

*Appendix A: HUD Layered Service Architecture* Guidance. The associations between the services are not depicted in this diagram, but are shown in other diagrams presented in this section.

The stereotypes used have two functions: The first function provides additional properties for documentation purposes, and the second function provides the properties needed to generate implementation code from the UML objects. The stereotypes used in the Specification Model are Service Specification and Service Operation Specification. Listed below are the properties of each stereotype:

**Service Specification -** Defines the interface that the service offers.

**Service Operation Specification** – Defines the behavior of an operation offered by the service specification.

The notation for each stereotype attribute denotes the optionality and cardinality of the attribute. [1] means that a value for the attribute must be present. [0..1] denotes an optional attribute. The notation [0..n] reflects an optional attribute that may have up to "n" values. "*" denotes that there is no logical limit to the number of occurrences of this attribute.

**XYZ Intake Process Step – Specification View**

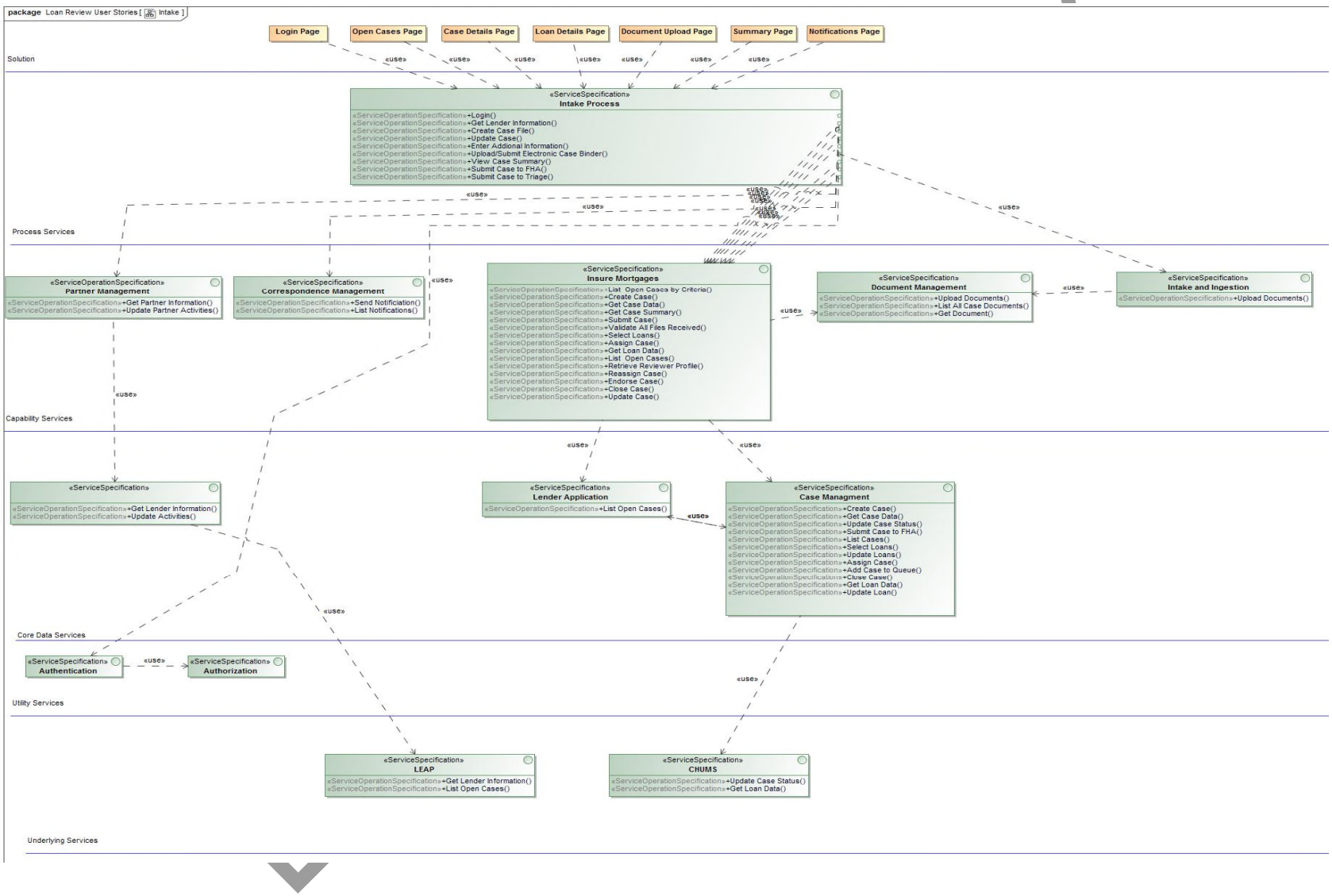


**Exhibit 6 – Overview of Loan Review System Specification Architecture** is a class diagram that depicts the services (classes stereotyped as Service Specifications) that participate in the Intake Process Step. The solution layer classes are identified only to show the interaction of the Solution Layer with the rest of the Service Specifications. The table below provides a description of each service depicted in the diagram and specified in the model.

Services Specified in the Intake Process Step

| Service Specification | Layer | Description |
|---|---|---|
| Intake Process | Process | Controls, executes and tracks the business process. The Process Service should be independent of any user interface or solution design. |
| Partner Management | Capability | Controls and tracks Partner (Lender) information. |
| Correspondence Management | Capability | Tracks and maintains correspondence which can be physical mailings or electronic notices. |
| Insure Mortgages | Capability | Coordinates and tracks the activities associated with cases (mortgage insurance). |
| Document Management | Capability | Tracks and maintains the documents associated with a case. |
| Intake and Ingestion | Capability | Provides the capability to receive documents in both physical and electronic form. |
| Lender Application | Core Data | Stores, retrieves and validates data relating to the Lender's cases. |
| Lender | Core Data | Stores, retrieves and validates data relating to the Lender. |
| Case Management | Core Data | Stores, retrieves and validates the case data. |
| Authentication | Utility | Validates that the user credentials are valid. |
| Authorization | Utility | Determines what functions a validated user is allowed to execute. |
| LEAP | Underlying | Provides an interface to LEAP system to retrieve and update Lender information. The service should be implemented as a façade to the LEAP system so as to not expose any of the technical implementation. |
| CHUMS | Underlying | Provides an interface to CHUMS system to retrieve and update Case and Loan data. The service should be implemented as a façade to the CHUMS system so as to not expose any of the technical implementation. |

**XYZ Loan Selection Process Step – Specification View**

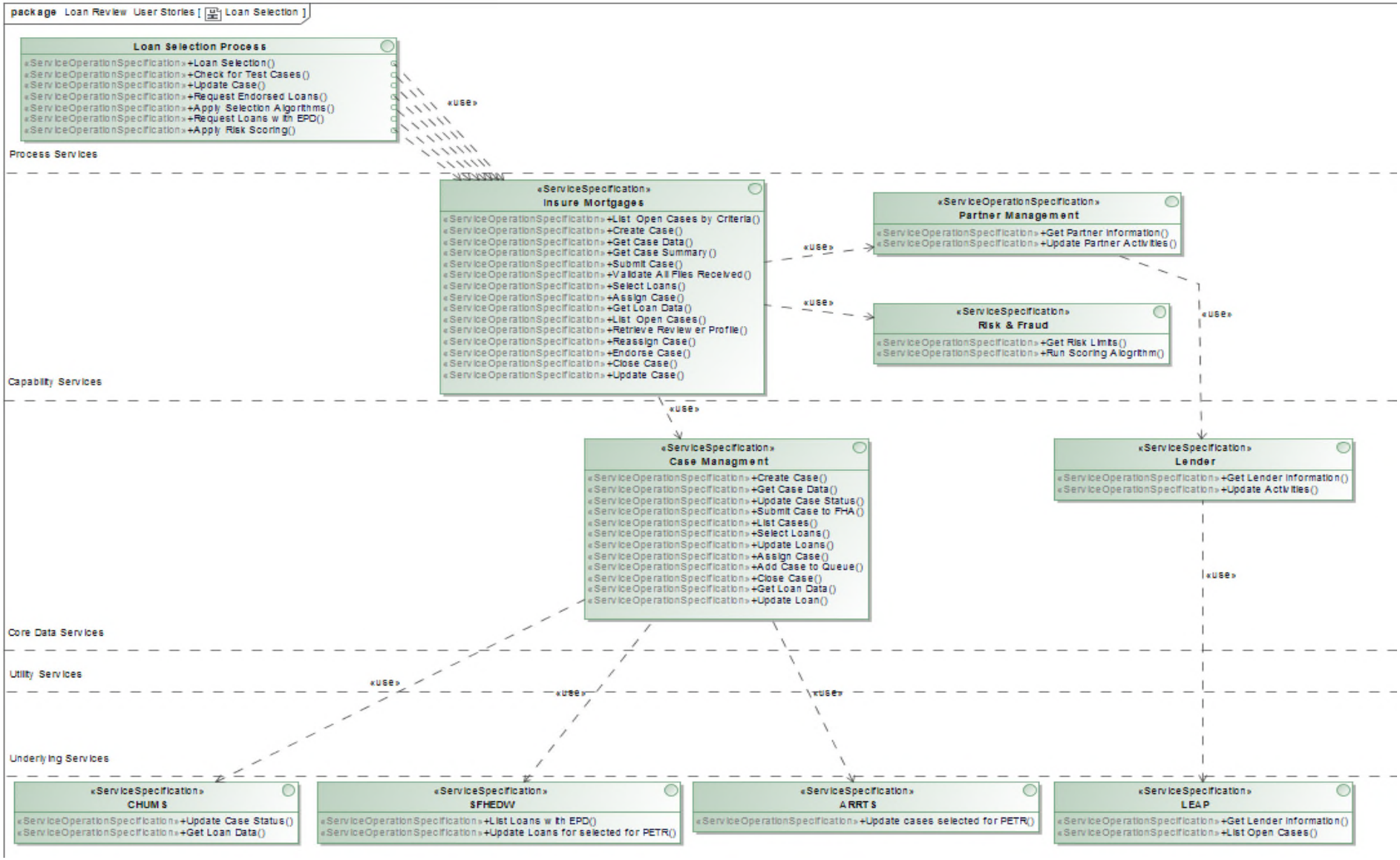


**Exhibit 8 – Description of Example Process** Layer  is a class diagram that depicts the services, classes stereotyped as Service Specifications that participate in the Loan Selection Process. No Solution Layer is shown as this process has no User Interface associated. This process step is initiated nightly. The table below provides a description of each service specification in the diagram.

Services, Classes Stereotyped as Service Specifications in the Loan Selection Process

| Service Specification | Layer | Description |
|---|---|---|
| Loan Selection Process | Process | Controls, executes and tracks the business process.  The Process Service should be independent of any user interface or solution design. |
| Partner Management | Capability | Controls and tracks Partner (Lender) information. |
| Risk and Fraud | Capability | Applies Risk and Fraud algorithms |
| Insure Mortgages | Capability | Coordinates and tracks the activities associated with cases (mortgage insurance). |
| Lender | Core Data | Stores, retrieves and validates data relating to the Lender. |
| Case Management | Core Data | Stores, retrieves and validates the case data. |
| LEAP | Underlying | Provides an interface to LEAP system to retrieve and update Lender information. The service should be implemented as a façade to the LEAP system so as to not expose any of the technical implementation. |
| CHUMS | Underlying | Provides an interface to CHUMS system to retrieve and update Case and Loan data. The service should be implemented as a façade to the CHUMS system so as to not expose any of the technical implementation. |
| ARRTS | Underlying | Provides an interface to ARRTS system to retrieve and update Case and Loan data. The service should be implemented as a façade to the ARRTS system so as to not expose any of the technical implementation. |
| SFHEDW | Underlying | Provides an interface to SFHEDW system to retrieve and update Case and Loan data for the loan selection process. The service should be implemented as a façade to the SFHEDW system so as to not expose any of the technical implementation. |

**XYZ Triage Process Step – Specification Review**

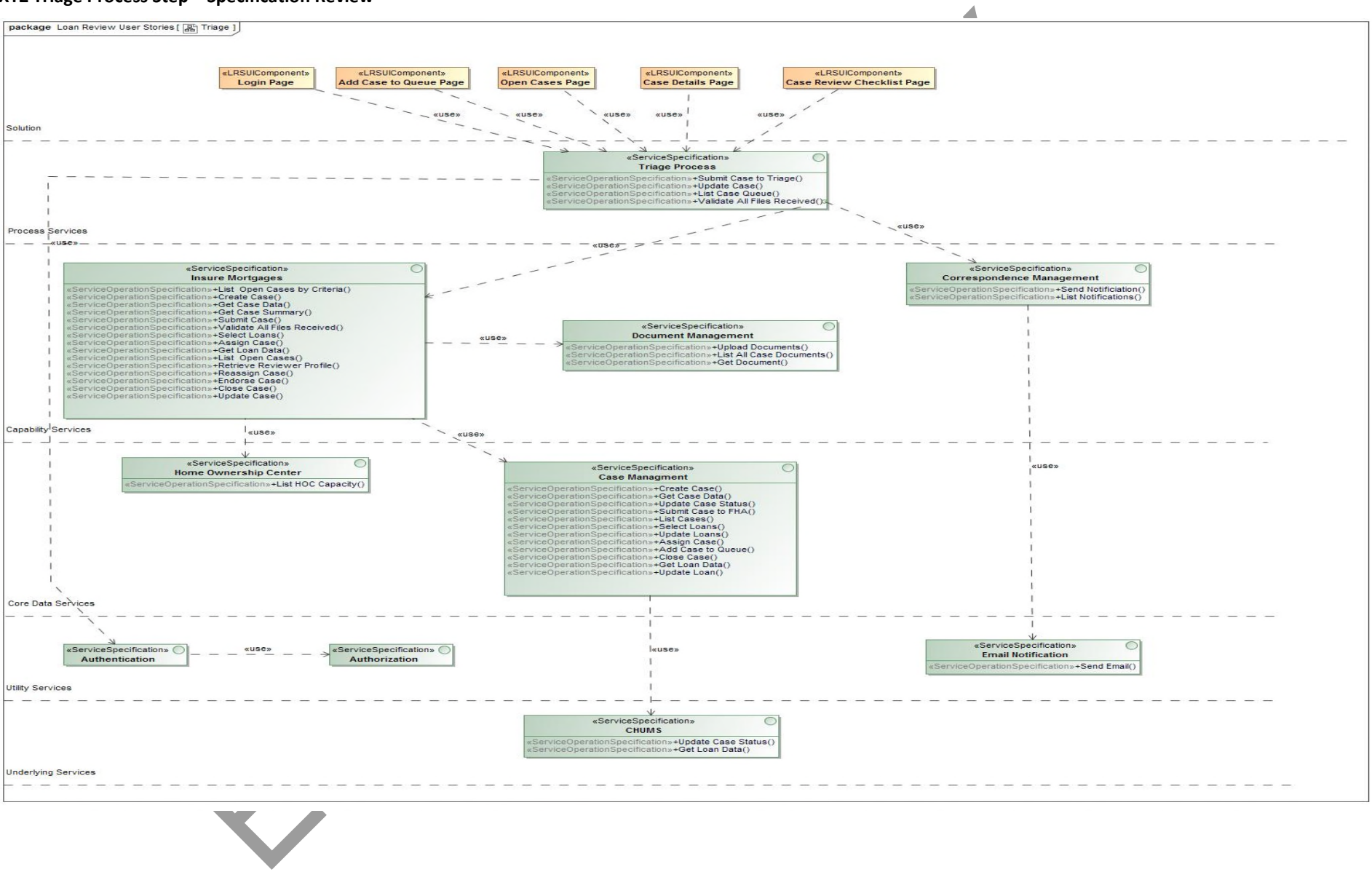


**Exhibit 10 –Core Data Service** is a class diagram that depicts the services, classes stereotyped as Service Specifications that participate in the Triage Process. The table below provides a description of each service specification in the diagram.

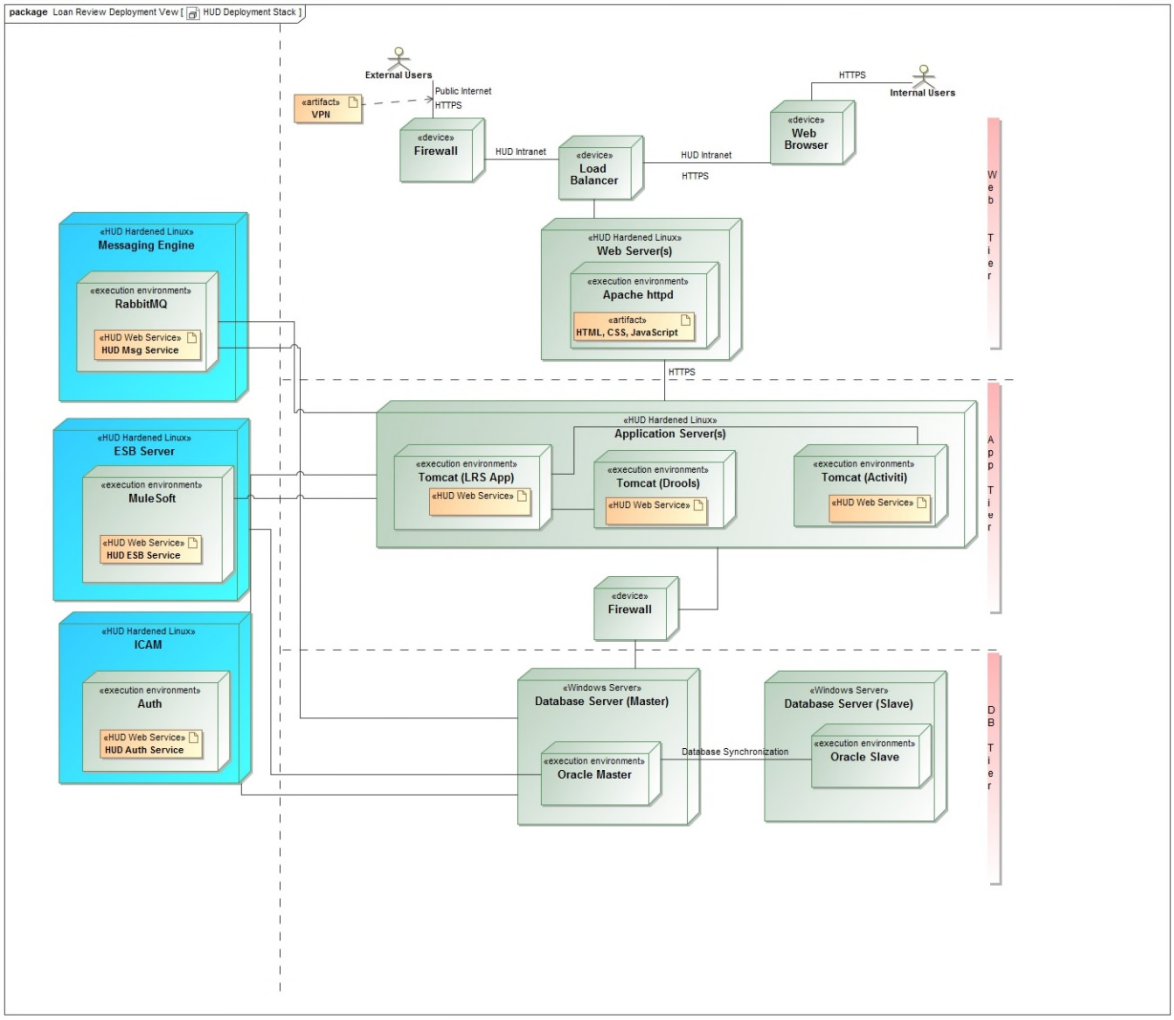
**Services, Classes Stereotyped as Service Specifications in the Triage Process**

| Service Specification | Layer | Description |
|---|---|---|
| Triage Process | Process | Controls, executes and tracks the business process. The Process Service should be independent of any user interface or solution design. |
| Partner Management | Capability | Controls and tracks Partner (Lender) information. |
| Correspondence Management | Capability | Tracks and maintains correspondence which can be physical mailings or electronic notices. |
| Insure Mortgages | Capability | Coordinates and tracks the activities associated with cases (mortgage insurance). |
| Homeownership Center | Core Data | Stores, retrieves and validates the Homeownership Center data. |
| Case Management | Core Data | Stores, retrieves and validates the case data. |
| Authentication | Utility | Validates that the user credentials are valid. |
| Authorization | Utility | Determines what functions a validated user is allowed to execute. |
| CHUMS | Underlying | Provides an interface to CHUMS system to retrieve and update Case and Loan data. The service should be implemented as a façade to the CHUMS system so as to not expose any of the technical implementation. |

**XYZ Deployment View**

The following exhibit presents the deployment view of XYZ application. The left side of the figure (boxes in blue) presents the components deployed at HUD enterprise level – RabbitMQ, MuleSoft ESB server and ICAM for authentication and authorization. The XYZ application interacts with these components, but they are managed and maintained by other teams at HUD. The XYZ components are deployed in 3-tiers (Presentation Tier, Business Logic Tier, and Data Access Tier), which are separated by firewalls for security and to prevent application servers and database servers from external vulnerabilities.

**XYZ Application Deployment View**

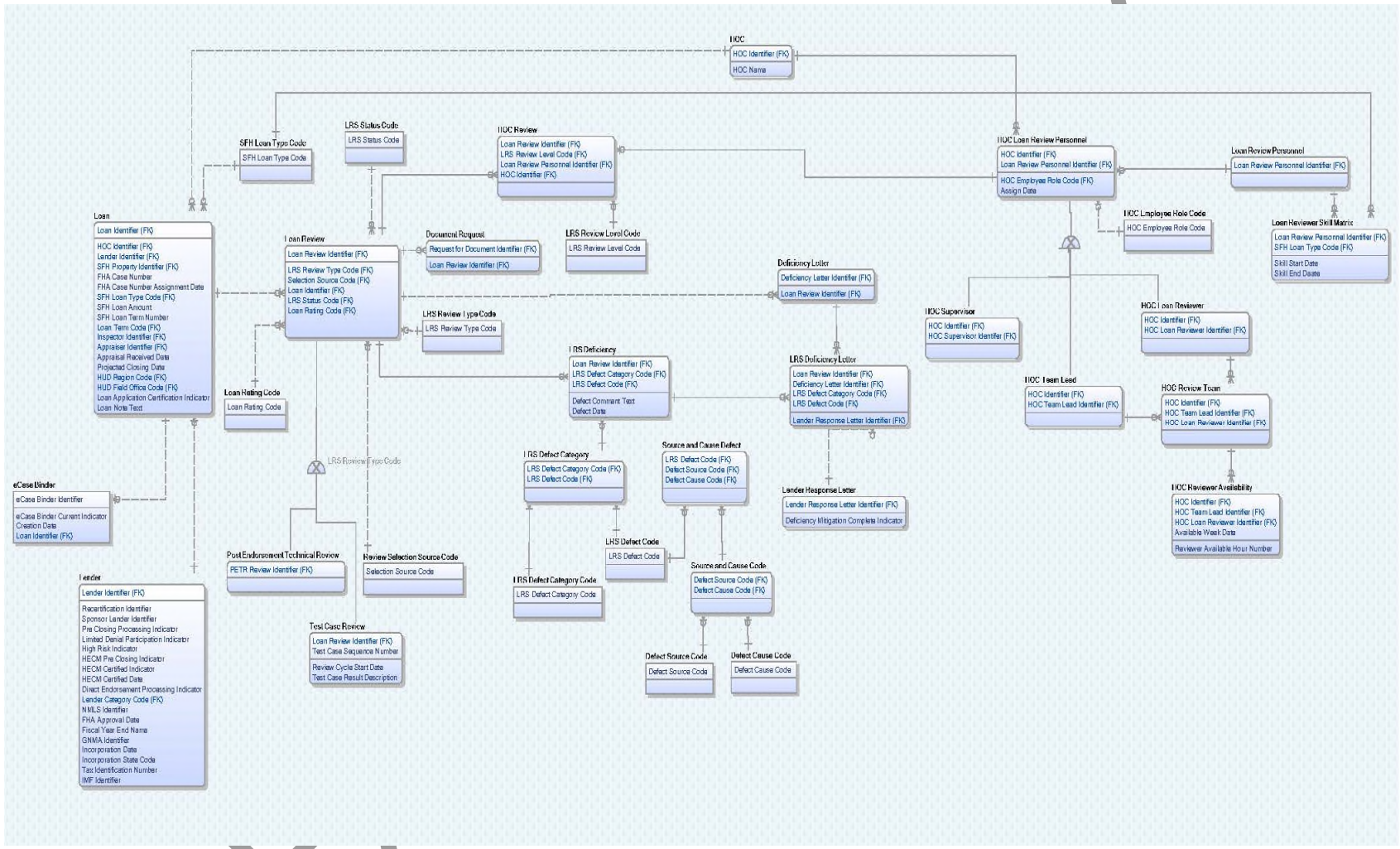


The following table provides a list of all the components represented in the deployment view:

**Exhibit 14 – Deployment View Component Descriptions**

| Component | Tier | Description |
|---|---|---|
| Web Server(s) | Web tier | Apache httpd installed on Linux in web server cluster. The application can scale up and scale out as traffic increases. Additional web servers are provisioned as demand increases. |
| Application Server (XYZ Tomcat) | App tier | Tomcat with XYZ application specific java code and associated libraries installed on Linux. |
| Application Server (Drools) | App tier | Tomcat with Drools rules engine installed on Linux. |
| Application Server (Activiti) | App tier | Tomcat with Activiti BPM engine installed on Linux. |
| Database Server (Oracle Master) | Database tier | Oracle master database. It stores XYZ specific database schema and also databases to support Activiti and Drools. |
| Database Server (Oracle Slave) | Database tier | Oracle slave database (replicates data from Oracle master database in near real-time). |

# Appendix D: XYZ Logical Data Model



The following exhibit provides descriptions for the entities depicted in the entity relationship diagrams (ERD) just presented. The attribution in the ERD is for informational context purposes only. These have not been finalized and no descriptions have been provided.

**Loan Review Entity List and Descriptions**

| Entity Name | Definition |
|---|---|
| **Lender** | Lender is a federally qualified financial institution that lends money to a borrower in return for an obligation to repay. A HUD approved Lender must fully comply with all of the approval and eligibility requirements specified by HUD FHA in order to be approved to participate in the origination, underwriting, closing, endorsement, servicing, purchasing, holding, or selling of FHA-insured Mortgages. |
| **Loan** | Loan (a type of HUD Case) is money lent from a financial institution to a creditworthy borrower(s) over a specified period of time and at a particular interest rate. |
| **HOC** | HOC (Home Ownership Center) is a HUD organization that conducts the mortgage insuring processes. Because the substantive review of a loan occurs after the mortgage is endorsed, Homeownership Centers (HOCs) must continually monitor lender performance and take necessary action as soon as they identify underwriting deficiencies.<br><br>The objective of the HOC action is to:<br>• reduce the risk of defaults, and claims to FHA<br>• improve lender performance, and/or • remove non-complying lenders from the program.<br>HOCs monitor the performance of lenders by:<br>• conducting on-site and remote lender reviews<br>• conducting post endorsement technical reviews (PETRs) of insured loans<br>• analyzing Mortgagee Performance Reports and Underwriting Report System (URS) reports available through FHA Connection (FHAC)<br>• analyzing default and claims data from Neighborhood Watch Early Warning System reports available through FHAC<br>• following up on construction complaints or consumer complaints, and<br>• sharing information among themselves, FHA Headquarters, and the Mortgagee Review Board (MRB). |
| **HOC Loan Review Personnel** | HOC Loan Review Personnel is the staff employed at a Home Ownership Center (HOC) that are qualified to perform loan reviews. |
| **Loan Review** | Loan Review is an FHA loan review process to clearly identify which loans pose too great a risk to FHA and which loans contain errors or other deficiencies. |
| **XYZ Review Type Code** | XYZ Review Type Code is the list of valid values and descriptions that classifies a loan review. |
| **XYZ Defect Code** | XYZ Defect Code is a fundamental characteristic of loan insurability that impact a loan's insurability, credit quality, and compliance. |
| **XYZ Deficiency** | TBD |
| **Deficiency Letter** | Deficiency Letter is a letter the Homeownership Center (HOC) must issue to the lender for each loan receiving a rating of Unacceptable for Mortgage Credit (MC) and/or Valuation/Underwriting (Val/UW). The letter must:<br>• identify the specific deficiencies upon which the unacceptable rating was based, and<br>• provide the lender with 45 days (unless an extension is granted) to submit a response in writing, including any explanations or documentation explaining the decision to approve the mortgage. LI lenders are required to submit their responses electronically within 10 business days. |
| **Document Request** | Document Request is formal request to receive a document in support of a case. |

| Entity Name | Definition |
|---|---|
| **XYZ Deficiency Letter** | XYZ Deficiency Letter identifies a deficiency associated with a Deficiency Letter sent by a HOC (Home Ownership Center) to a Lender. |
| **Test Case Review** | Test Case Review is a type of FHA loan review whereby a Direct Endorsement (DE) lender applying for unconditional approval must submit 15 mortgage loan applications, aka test cases, for review by the Homeownership Center (HOC). The test cases may vary by loan type, and must represent expected underwriting situations. |
| **Post Endorsement Technical Review** | Post Endorsement Technical Review (PETR) is a type of FHA Loan Review that is performed on selected cases to evaluate several factors, including:<br>1. The risk that loans represent to FHA's insurance funds<br>2. The lender's compliance with FHA's underwriting and documentation requirements. |
| **eCase Binder** | eCase Binder is a subset of the lenders "loan file" and is sent to a HUD Home Ownership Center (HOC) where Reviewers and Endorsement Clerks check the paperwork to determine if the mortgage meets the eligibility requirements for insurance and that all required documents and signatures are present. After the lender underwrites and closes the loan, lenders must submit information about the loan organized in a Case Binder (the eCase Binder) in an FHA required stacking order. |
| **Review Selection Source Code** | Review Selection Source Code is the list of valid values and descriptions that describe the method used to select a loan for review.<br>List of Valid Values:<br>1. Manual Selection<br>2. Early Payment Default (EPD)<br>3. New Endorsements Risk Scoring Algorithms - different algorithms for HECM and Forward<br>4. Random Selection - based on sample sizes set by the National Review Coordinator (NRC)<br>5. New Endorsements Sample Selection -based on additional lender or underwriter selection rules<br>6. LDP<br>7. SAMS<br>8. CAIVRS; Credit Alert Interactive Voice Response System (CAIVRS) Authorization function lists information on a party such as Lender or Underwriter past default, claim, judgment, and foreclosure records on government loans. |
| **XYZ Status Code** | TBD |
| **Loan Review Personnel** | Loan Review Personnel is the HUD employee that has been trained in the Loan review process and is qualified to conduct loan reviews. |
| **HOC Supervisor** | HOC (Home Ownership Center) Supervisor is a HUD employee that performs management tasks for the HUD employees assigned to a HOC. |
| **SFH Loan Type Code** | SFH Loan Type Code is the list of valid values and descriptions that classify a Single Family Housing loan. |
| **XYZ Defect Category Code** | TBD |
| **XYZ Defect Category** | TBD |
| **HOC Review** | HOC Review is the assignment of Home Ownership Center (HOC) personnel to loan review. |
| **Loan Rating Code** | Loan Rating Code is represents the risk of a loan's financial stability.<br>For both the Mortgage Credit (MC) and Valuation Underwriting (Val/UW) of a loan, the reviewer must enter a rating. |

| Entity Name | Definition |
|---|---|
| **XYZ Review Level Code** | XYZ Review Level Code is the character that represents<br>List of Valid Values:<br>I - Initial - The initial review level is completed for the first PETR of a loan. The initial review level may be completed by Homeownership Center (HOC) staff or by a contractor.<br>Q - Quality Control - The Quality Control (QC) level is used to enter the results of QC reviews of the initial reviews performed by contractors.<br>S - GTR / Supervisory - The Government Technical Representative (GTR)/Supervisory review level is used to confirm or revise changes in ratings made at the QC level. The GTR/Supervisory review level may be completed by a PUD supervisor or designated senior PUD staff<br>A - Additional Review - The "Additional Review" level of review is used to enter changes in ratings based on lender responses to ratings from earlier level reviews, or otherwise confirm or change ratings from earlier review levels. The Additional Review level is also used to change the Unacceptable rating to Mitigated in either or both "Val/UW" or "MC" rating categories. |
| **HOC Employee Role Code** | HOC (Home Ownership Center) Employee Role Code is the character that represents the job that a HOC employee perform during a loan review process. |
| **Lender Response Letter** | Lender Response Letter is a lender's answer to a deficiency letter that the lender receives as a result of a deficiency discovered during a loan review. |
| **HOC Team Lead** | HOC (Home Ownership Center) Team Lead is a HUD employee who performs leadership tasks for the HUD employees assigned to a team. |
| **HOC Loan Reviewer** | HOC (Home Ownership Center) Loan Reviewer is a HUD employee who is qualified to perform loan reviews. This entails providing objective assessments for credit in order to identify potential loan problems and ensure compliance with established loan policies and federal regulations. |
| **HOC Review Team** | HOC (Home Ownership Center) Review Team is the association of a Loan Reviewer assigned to a specific HOC to a specific team along with the HOC Team Leader of the Team. |
| **Loan Reviewer Skill Matrix** | Loan Reviewer Skill Matrix is a record of a loan reviewer's expertise and qualifications. This enables the case assignment process to assign the case load accordingly. |
| **HOC Reviewer Availability** | HOC (Home Ownership Center) Reviewer Availability is a schedule of hours that a loan reviewer has available to perform loan reviews. This is used to determine the case load for loan reviewers. |
| **Source and Cause Code** | Source and Cause Code is |
| **Source and Cause Defect** | TBD |
| **Defect Source Code** | Defect Source Code is |
| **Defect Cause Code** | TBD |

# Appendix E: Agile Development

HUD OCIO has adopted an agile approach for software development to reduce the risk of project failure, and to assure that the delivered system performs as it is intended. These methods have repeatedly been shown to:

- Improve time-to-mission-value
- Reduce project risk
- Reduce cost
- Improve visibility
- Better adapt to changing needs

Agile approaches use an iterative, incremental process that is characterized by small, frequent releases developed in close collaboration with the customer. These practices provide tight feedback loops and frequent opportunities for course correction. Agile approaches are consistent with "modular development" or "modular IT" as defined in OMB. Modular development of IT capabilities requires that programs deliver functionality in increments, generally of no longer than 6 months each. Agile development provides a best-practices way of conducting such incremental delivery.

The greatest process risk for agile teams is simply following a set of mechanical steps or "ceremonies" rather than adhering to agile values and principles and adopting the agile mindset. Teams that do this are likely to lose the empirical and adaptive benefits of agile, and hence lose the natural controls that are inherent in the agile approach.

Therefore, understanding and practicing the core values and principles of agile are the primary means of obtaining the benefits of agile. The essence of agile methods is the use of an "inspect and adapt" paradigm: agile projects practice continuous improvement of processes, requirements, and design.

Agile Practice (s) – the minimum agile practices required of all agile teams include the following:

1. Frequent Delivery – Projects shall deliver usable product to end users at least quarterly.
2. Time-boxed Iterations – Projects shall adopt a fixed-length iteration (between 2-4 weeks) as their standard cadence for planning, completing, and demonstrating potentially-deployable functionality.
3. User Stories – Project teams shall employ User Stories as the basic unit of planning, executing, and tracking their work, with User Stories being defined as small, independent, and testable units of functionality. Note that User Stories under this definition are not necessarily expressed in classic user story format ("As a <who> I want to <what> in order to <why>"). Indeed, many formats are possible for requirements (for example, samples of reports for BI projects) as long as they are small, independent, and testable.
4. Product Owner – Each project shall have a distinct individual representing the business involved with the team, with the authority to make timely decisions regarding user story development, prioritization, and acceptance.
5. Release Planning – Each project shall conduct a release planning exercise culminating in a release planning review (RPR) and a standard set of deliverables prior to commencement of work on a release.

6. Iteration Reviews – Project teams shall conduct a facilitated review at the end of the each iteration to demonstrate and/or test functionality and to solicit feedback from the project's stakeholders.
7. Retrospectives – Project teams shall hold a facilitated meeting at the end of each iteration to reflect on the team's performance and identify opportunities for improvement.
8. Continuous Testing – Project teams shall test User Stories up to and including in a staging environment during the iteration in which those stories are developed.

The practices discussed in this appendix represent the foundational practices that all projects must adopt to achieve a minimum level of agility within HUD. However, project teams are not limited by this and are encouraged to adopt other practices that advance the goal of continuous deployment.

Principles: The principles are designed to guide how HUD implements agile processes and represents the foundational principles that all projects must adopt to achieve a minimum level of agility within HUD

Lean Thinking

Lean emphasizes seeing the whole (including the end-to-end view of the delivery pipeline), and is not constrained by organizational boundaries and promotes delivering as fast as possible. Lean also emphasizes the "amplification of learning" through techniques, such as retrospection and mentoring, in order to increase the effectiveness of people, and therefore the speed at which work can be done.

Lean supports the concept of amplified learning, through mentoring and focused efforts to have people learn by doing with appropriate supervision. Agile also promotes the concept of a sustainable work pace, which promotes quality and excellence because people have the energy to pay attention to doing things right instead of rushing things through. This means that work must be planned in a way that does not over-promise, and that is based on actual measurement of work capacity instead of aggressive externally imposed deadlines. Rushed work leads to re-work, and does not save time in the long run.

Transparency

Agile requires that people work across business functions. For example, software cannot be developed in an agile manner unless architecture, testing, security, and release management all collaborate to define a way to perform all of those functions in parallel, in a just-in-time, collaborative manner, rather than one group handing things off to the next in sequence. OIT is defining a new processes that crosses business functions, but such processes must be adaptable, and as such, they will never be fully defined. The organization must rely on its staff to evolve processes over time. This requires being highly proactive, with an emphasis on reaching across silos to achieve an objective and not feeling constrained by rigid processes.

For this level of collaboration to be successful, management must provide a safe environment in which people are encouraged to reach out, across business functions, and are not punished for going directly to the staff they need to talk to. Those who reach out should be encouraged, not admonished. This is a significant change from the practices of most established hierarchical organizations, but it is a norm for innovative organizations. Proactiveness and collaboration are strong drivers of grass-roots innovation, and innovation is one of the strategic goals of OCIO.

Along with normalizing change, agile stresses transparency through continual delivery of output. By designing project activities so that every activity has a demonstrable result that the user can understand, and that is easily recognizable as tangible progress, one is able to make sure that a project stays on course. This greatly reduces risk because erroneous approaches become evident very quickly when the output of work is demonstrated.

Continuous Improvement

In an ever-changing world, business processes need to evolve continuously. This calls for regular reflection and process improvement. Continuous delivery advocates that the best way to address a recurring process problem is to address it sooner – right at the outset of the process, if possible – and that will focus attention on it and get it solved. Both lean and agile advocate identifying work that does not add value and eliminating wasteful activity. Measurement is important for continuous improvement because measurement often shows where problems really manifest. Measurement also helps to determine what the organization's true capacity is for handling work. Measurement is therefore a critical enabler for agility.

Mission-Focused and Results Oriented

HUD must be responsive to direction from Congress and other key stakeholders, and be ready to implement policy changes rapidly. The implementation of policy cannot be held up by IT impediments. Continuous delivery emphasizes the creation of a repeatable, reliable pipeline process for building and releasing software, and the automation of almost everything in that process pipeline. Agile methods emphasize the delivery of value frequently. At a tactical level, being mission-focused translates into being results oriented. Agile promotes a focus on tangible outcomes, as opposed to progress against a plan. All business processes and work should be defined and conducted in a manner that emphasizes results, rather than tasks. Completion of a task is not evidence of progress, unless it produces something that can be assessed as tangible progress, and ultimately, that it advances the HUD mission. Continuous delivery advocates the concept that "Done means released", and this means that a task to create an artifact (i.e., software, a business procedure, a document) should never be considered done until it has actually been put into use or operationalized.

Focused on Execution Excellence

Execution excellence means making time to do something right so that it does not cause problems later. Lean promotes the concept of building integrity into a business process, so that the process is self-correcting and does not require manual intervention. This frees the organization to focus its resources on new capabilities. Quality is extremely important in things that will endure: business processes, software, and any artifact that will be used repeatedly. Continuous delivery is enabled by software development processes that promote getting things right continuously, rather than catching errors later. Quality promotes repeatability. Automation also promotes repeatability, and therefore promotes quality.

Welcome Change throughout the Development Process

The strategic goal of responding quickly to mission needs requires allowing for changes to requirements at any time rather than creating rigid plans. Change is disruptive, and it is prudent to think ahead to minimize change; but responding to change is also essential in being responsive to the changing needs of the business, rather than plying ahead with an obsolete plan.

It is not easy to allow for change. It often means that there is work that must be discarded, or that many decisions must be re-visited. But change also means learning. If software is built and the users realize that it needs to change, then the positive view is that the users now have a clearer idea of what is really needed. Agile proposes that expecting users to know exactly what they need before they see it is not realistic. The strategies promoted by the agile and lean communities are to (a) allow for change, and (b) to manage risk by releasing small increments at a time, thereby containing change to those small increments. Lean also promotes the concept of making decisions at the last responsible moment; that is, to postpone a decision until it can be postponed no longer.

Importance of Security and Reliability

HUD is the steward of our nation's Housing data that is both critical and sensitive. It also provides services that are used by the public, and the public's expectations for reliability and security are high. These services affect people's

lives in a dramatic way, and thus the organization cannot afford to have these services be fragile, risky, unreliable or insecure. As mentioned earlier, lean thinking promotes the concept of building integrity into a process or system, rather than creating a substandard process and cleaning up later. For software development, this means building security into an application from day one. It means testing for security throughout the development cycle, not at the end. It means testing for error cases in the same way: continuously, rather than as an afterthought.

Informed by Architecture

Architecture consists of a set of decisions and models about how things should work. As such, architecture informs decision-making at all levels: at a business level (business architecture), and at a technical level (technical architecture). Architecture can be very ineffective and burdensome if it is allowed to operate in a manner that is very disconnected with operations and projects.

What is needed is effective architecture that is solution oriented and that operates in an issue-focused manner, helping the organization to find the right path as needs are considered and implemented. This type of architecture is deeply immersed in solution efforts, and only spends its time on issues that have an actual impact on the business. Most importantly, it operates collaboratively, working with development teams to help them to consider issues and make decisions and choices about technologies and solution approaches. In other words, it shares its knowledge and helps people, rather than focusing its work around documents.

# Appendix F: Acronym and Definitions

This subsection describes all terms and abbreviations used in support of the development of this document and critical to the comprehension of its content that are not contained in the Office of the Chief Information Officer (OCIO) Terms, Acronym and Definitions List.

Below is a list of Acronyms and Definitions contained in this document:

| Acronym | Definition |
|---|---|
| AARTS | Approval/Recertification/Review Tracking System |
| API | Application Program Interface |
| APPS | Active Partners Performance System |
| CARS | Common Application Relational Schema |
| CHUMS | Computerized Homes Underwriting Management System |
| CISO | Chief Information Security Officer |
| COTS | Commercial Off-the-Shelf |
| DTaaS | Development and Test as a Service |
| EA | Enterprise Architecture |
| EAI | Enterprise Application Integration |
| EDW | Enterprise Data Warehouse |
| ESB | Enterprise Service Bus |
| ETA | Enterprise Technical Architecture |
| FHA | Federal Housing Administration |
| HECM | Home Equity Conversion mortgages |
| HOC | Home Ownership Center |
| HUD | The Department of Housing and Urban Development |
| JMS | Java Messaging Service |
| LEAP | Lender Electronic Assessment Portal |
| XYZ | Loan Review System |
| OCIO | Office of the Chief Information Officer |
| OIG | Office of Inspector General |
| PII | Personally Identifiable Information |
| POJO | Plain Old Java Object |
| PPM | Project Portfolio Management |
| SFHEDW | Single Family Housing Enterprise Data Warehouse |
| SLAP | Service Layered Architecture Profile |
| SOA | Service Oriented Architecture |
| UML | Unified Modeling Language |
| VA (Loans) | Veteran Affairs |
| VM | Virtual Machine |