**BITS High Level Architecture and Solution Design**

**V0.2**

# Introduction

BITS (Bus Inspection Tracking System) is currently running as Lotus Domino application on IBM xSeries 345 platform (Windows 2008 R2). The platform will be decommissioned by Jun 2018. MTO will re-platform BITS application to keep business operation continuously after Jun 2018.

This document describes high level architecture and solution design for creating a new BITS application.

# Assumptions

* 1. The high level architecture and solution described in this document are mainly based on *BITS Requirement May 2017.pptx* provided by BA
  2. New BITS application will be business-wide like to like basis. However MTO business require a few enhancements to following areas:
* Search and notification
* Bulk Operation
* Integration with MTO systems (Mainframe Vehicle System, Potentially Carrier, AES/RDC etc.)
* Report
* Data Migration and Cleansing
  1. Authorization - reuse enhanced existing IAA (Identity Authentication and Authorization) framework RAMS. (New name: Tree and Role Based Access Control System – TRACS, i.e. RAMS v2.0) for IAA. Existing IAA Framework (RAMS) needs to be enhanced to fit BITS requirements. The changes are fully reusable for other applications as well. The key enhancements are:
* Change RUS IAA framework(RAMS) to become real standalone service architecture
* Remove RUS application and database dependences (with minimized data model changes only. E.g. change table and column names if necessary)
* Remove Application Server (WebLogic) dependencies
* Provide both library and rest API for clients to use
  1. Authentication Service Provider to be determined, potential providers are Go-Secure, Azure Active Directory and ADFS used by MOL.
  2. There is no native mobile client will be created. User still can use BITS application on mobile device provided the device can support browser based UI.
  3. Runtime Environment and Related Tools

Runtime environment and tools are not finalized yet. Based on current discussions (TSO, DEV, DMO, QA, Architect, Tidal Migration teams), assumptions at this time are:

* Azure Cloud
* Docker Image/container runs on Open Shift platform.
* Application Server (Weblogic, JBoss, Tomcat)
* Network connectivity between Azure cloud and Go-Net Data Center is required.
  1. All BITS users are OPS internal users.

# High Level Architecture and Solution

* 1. **High Level Micro Service Based Architecture**



* 1. **Component Design Diagram**



* 1. **BITS Application Data Model**

TBD

* 1. **Integration** 
     1. Integration with MTO Vehicle system

*Just In Time (JIT) Integration (On Demand)*

JIT (On Demand) in this document means whenever vehicle data is used, the system can synchronize it with MTO vehicle system.

Two potential options:

* VSS option (SOAP Web Services)
* IMS Connect option (not preferred)

*Batch Integration (daily)*

Business requires daily batch process to updated BITS system when there are changes on mainframe side (e.g. new vehicle added into vehicle system)

* + 1. Integration with AES/RDC (Out Of Scope)

This is a nice to have functionality to enhance user experiences regarding to inspection result collection operation.

Solution highlights:

* AES/RDC system sends individual raw inspection result into BITS application. BITS application can process AES/RDC data and store required inspection result information into BITS database. This way can make sure there is no BITS business logic inside AES/RDC system.
* BITS still need to provide manual inspection result in case the communication between AES/RDC and BITS are not available.
  + 1. Integration with STS, Go-Secure, Azure Active directory (AAD) or on premise ADFS system.

Refer to section 3.5

* 1. **IAA (Identity Authentication and Authorization) and Security**

Leverage STS, Go-Secure Azure Active Directory (AAD) or On-Premise ADFS for authentication and TRACS for authorization.

* + 1. Business Role (BR) and Authorization Group (AG)

For new BITS application, AG and BR are one to one relationship

* Application Administrator
* District Administrator
* Enforcement Officer
  + 1. Application Role

At least one application role is defined for one request.

* + 1. User Profile Management
* Based on current requirement, there is no person profile management (for administrator and enforcement officer) in BITS. IAA User table defined in TRACS may be sufficient enough (TBD)

(In this case, require TRACS to allow store badge number for enforcement officer)

* + 1. Region and District
* stored as Authorization Node in TRACS
* Region is treated as dictionary entity (means managed through DB script only)
* District may be treated as dictionary entity as well (TBD)
* AGs will linked to region and district
  + 1. Mechanism to secure UI and backend Web API services

Leverages OAuth2 and de-facto standard JSON Web Token (JWT) to perform authentication and authorization.

* Go-Secure – Service API Gateway issues JWT
* STS, AAD / ADFS – STS, AAD/ADFS will issue JWT
* Backend Web API services (REST) may be secured by a system client credential as well.
  + 1. Authentication

All users are OPS internal users; Option 1: use PKI to log into BITS through Go-Secure portal. Option 2: using existing AAD/ADFS service if the application is deployed on Azure Cloud.

* + 1. Authorization

Reuse enhanced RAMS (i.e. TRACS) or built within BITS due to timeline constrains for initial phase.

* 1. **Communication**
* HTTPS/TLS when applicable
* Web Services for on-demand integration
* SFTP for batch integration process (TBD)
* Connectivity between Azure and OPS Data Center is required.
  1. **Performance** 
     1. Design In-Memory Caches and automatic refresh mechanism (configurable)
* Region cache
* District cache
* District – Authorized Enforcement Officer cache
* Caches for dictionary entities
* Status (Id, Code, Name etc.)
* Inspection Category

Note: Region and District entities and relationships are not defined in BITS schema.

The following are sample conceptual cache definitions:



# RAMS (RUS Access Management Service) Enhancement

* 1. Goal
* Fits BITS requirements
* Make it a standalone services (detached from RUS Enterprise Database)
* Separate authorization core and WebLogic related code
* Provide REST API
  1. Strategy
* Minor Entity Relationship changes
* Provide generic names for tables, columns if necessary
* Rename RAMS to a generic service name: Tree and Role based Access Control System (TRACS)
* Separate IAA Core with WebLogic specific code
* Leverages on premise environment

# Report

* Operational Report only. Report provided through BITS UI

# Audit and Logging

* TBD - Reuse RUS Audit services?
* Logging each request and response in log file

# Data Cleansing and Migration (TBD)

* Initial data cleaning and migration will be done through both manual and DB scripts
* On-Going data cleaning may be done by JIT(On demand) and schedule synchronization process initiated by BITS or Mainframe (TBD)