GAO BLE, RFID & IoT Engine 4.0

2022
Toronto, Ontario
Canada

 http://www.gaorfid.com

Contents

I. Previous system (3.x) review

Over the course of more than 10 years, we have successfully implemented many systems using Version 3.x, and earlier versions 2.x and 1.x, but we have seen lots of room to improve on even the latest version.

The code is branched based on the projects/clients and is not consolidated into a single code base, a solution could be just for a single client and there is so many solutions in the version control system. The situation has limited our ability to improve the system with components/layers, fixed data schema and disconnected solutions with its own layers and components such as the web user interface, middleware, handheld code and data processing:

- Fixed data model and schema for a solution from the bottom data layer which makes the solution could not reuse for other application which has different data models, even minor differences
- Out of date user interface – right away, we can fell the system is old, not for a web 2 application.
- Disconnected repository limits the system grows, if a new feature added to a solution, other solution remains unchanged
- A bug fix can only be for a single solution, have to populate code to all the solutions
- QA for all the solutions is impossible
- No way to figure out which solution to use a new project unless the person has handled the similar project before
- Documentation for all the solutions is way too much compared with a consolidated system

For business reasons, project-based system demonstrated a non-extendable, shallow solution and there is no enterprises system ever did this way. It is hard to manage, to understand, to fix and to scale up affectively.

II. BLE, RFID & IoT Engine Version 4.0

We name the new RFID system 4.0 as GAO BLE, RFID & IoT Engine. If in one sentence, the new system is a consolidated system, a single system to provide all the solutions of the RFID, including BLE technologies.

The goal is to provide a solution that is extendable solution that can easily meet new business requirements and technology changes, and system expansion vertically and horizontally. Basically, this will give us ability to compete in the market with time efficiency, lower cost and better quality.

There are about 30 solutions (more or less) in the version 3, almost every new project to make a copy of the old code and make change on top of it. We should not move on in this the way for version 4. Instead, we should build a system that can be extended with a single solution for the core RFID system with the follow key concepts:

1) The system should be extendable and configurable for business modules and reader devices and various operation models without touching core system code/architecture, except to add new code to meet new business requirements and hardware/device interface (device driver, etc.). The design gaol is to make a system that can communicate with tags without deeply coupled with business logics and able to interpreter the various tag activities based on the configurable business modules as well pluggable devices.
2) We should take advantage of the recent (in the last 10 years) software advancement, such as HTML5, JavaScript, TypeScript/Angular, NoSQL, VPN etc. to address the challenge of dynamic and distributed/cloud nature of the business and technology.
3) The version 3 code has provided a list of tested device drivers and business modules we should move and migrate those tested codes into version 4.

The system will reuse as much as possible the version 3 code by starting with few solutions from version 3, the middleware and the web server components will keep the .Net platform. The web user interface will introduce Google Angular to modernize GUI. The middleware will merge version 3 device drivers and business modules to make a consolidated enterprise system with the following design details:

1 Data Layer and data synchronization

1) NoSQL

Beside using SQL server to manage structured data, such as tag, reader, the new system will use NoSQL database to manage semi structured and dynamic/unstructured data. The use of NoSQL server will provide us the capability to handle dynamic nature of the data schema, simplify and relive our system from direct coupling between data, business logics and to avoid code changes for different data schema which is the key concept for using NoSQL.

In order to provide a stable solution, first we need to have a stable data layer, this is the key for version 4.

2) Data Synchronization

We have seen the requirement to synchronize data for multisite deployment and difficulty for configure data synchronize by using MS SQL Server technology. Also, we have few projects implemented data synchronization by code changes. This issue is rooted in the database servers which are deployed in individual locations.

Today, new technologies provide many valuable options to resolve this challenge, I think we should use VPN technology because of its simple to use and low or no cost with high level or military level security. This approach could be used for future multisite deployment for both version 3 and 4. Without changing code in the middleware, by direct connection to the same database via VPN.

We should identify a VPN technology provider as our preferred vender, there are some free VPN as well, but we need to find one that suitable for our system.

2 BLE, RFID & IoT Server

The widely acceptance of Angular based web user interface has become the industry standard to evaluate web applications and we will use Angular framework to build a true Web2 application. Beside provide modelized user interface, Angular detects errors in compile-time, as TypeScript is a statically typed language, instead of runtime like in JavaScript.

The system will provide the following features:

1) Web browser user interface for managing system user, tag, location, reader and antenna configurations
2) Web browser user interface for CRUD operations for applications such as asset management, people tracking, etc. The interface should provide ability to define application specific metadata, schema that will be used for the target application.
3) Report module provides reports for fixed data tables and reports for dynamic and semi structured data
4) REST service – with fixed CRUD APIs for stable entities, as well as pluggable module for business specific APIs. REST APIs should be published for potential integrators.
5) REST APIs for handheld device
6) Use Google Angular technology for user interface component
7) We should deploy only one web server for multisite environment instead of every site as current version 3 practice
8) Eliminate IIS as the web server, will use native Windows service to handle web user request/response, REST APIs, also provides real time monitor features for publisher and subscriber functions as current SignalR calls.

3 Middleware

The core features for middleware component are the same as the version 3, basically provides functions 1) load business module for different client requirements, 2) load device drivers for different readers and 3) direct driver RFID tag activities to the active business modules.

For version 4, we can use the same version 3, I guess or at least to get most stable version 3 code to start with.

Middleware can be deployed without BLE, RFID & IoT server remotely as long as it can connect to it and database server either locally or by VPN.

4 Handheld Device Components

Handheld device codes are running on different platforms, such as Android. iOS, etc. – we should keep the current code, integrate existing REST APIs into the version 4 REST APIs in the web component.

5 Business Modules

we should have all the functions provided for the version 3.x and consolidate the current code into the new system, the main work for this component will be to interface with the new data layer REST APIs.

We may need to migrate some version 3 business modules, they are:

Name	Subversion URL	Description

6 Fixed Reader Device Drivers

Every RFID fixed reader has a unique driver code to communicate with RFID middleware. Most of the driver code should be inherited from the version 3.x, since it only communicates with middleware components and has no direct coupling with data layer

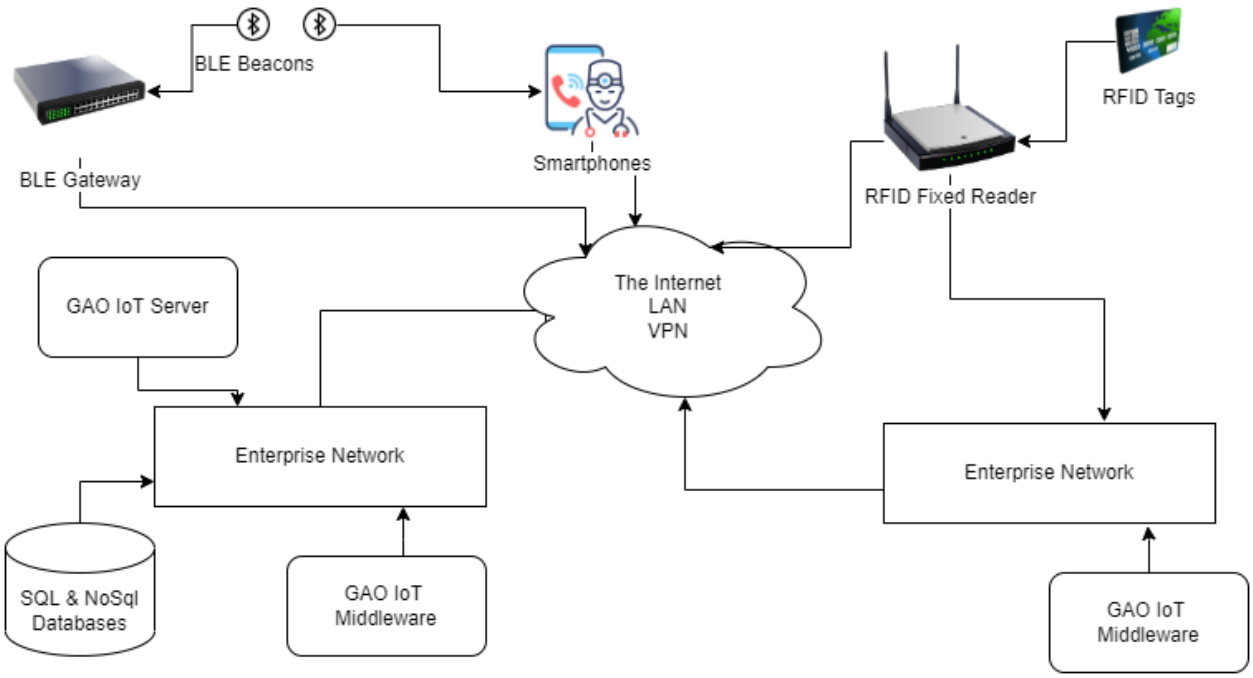
The drivers will be available to version 4:

Name	SKU No.	Subversion URL	Description

III. System Architecture

1. System Deployment Environment

The following system context diagram provides a high-level illustration of the system components and their couplings at runtime. The components connectivity could be on the LAN, Internet and VPN depending on the system performance, data security and complexity. Many enterprises have developed their readers with direct LAN connections with GAO BLE, RFID & IoT middleware which provide high system performance, reliability and security while many other enterprises deployed our solutions on the cloud with fixed readers connect to GAO BLE, RFID & IoT servers on the cloud environment.

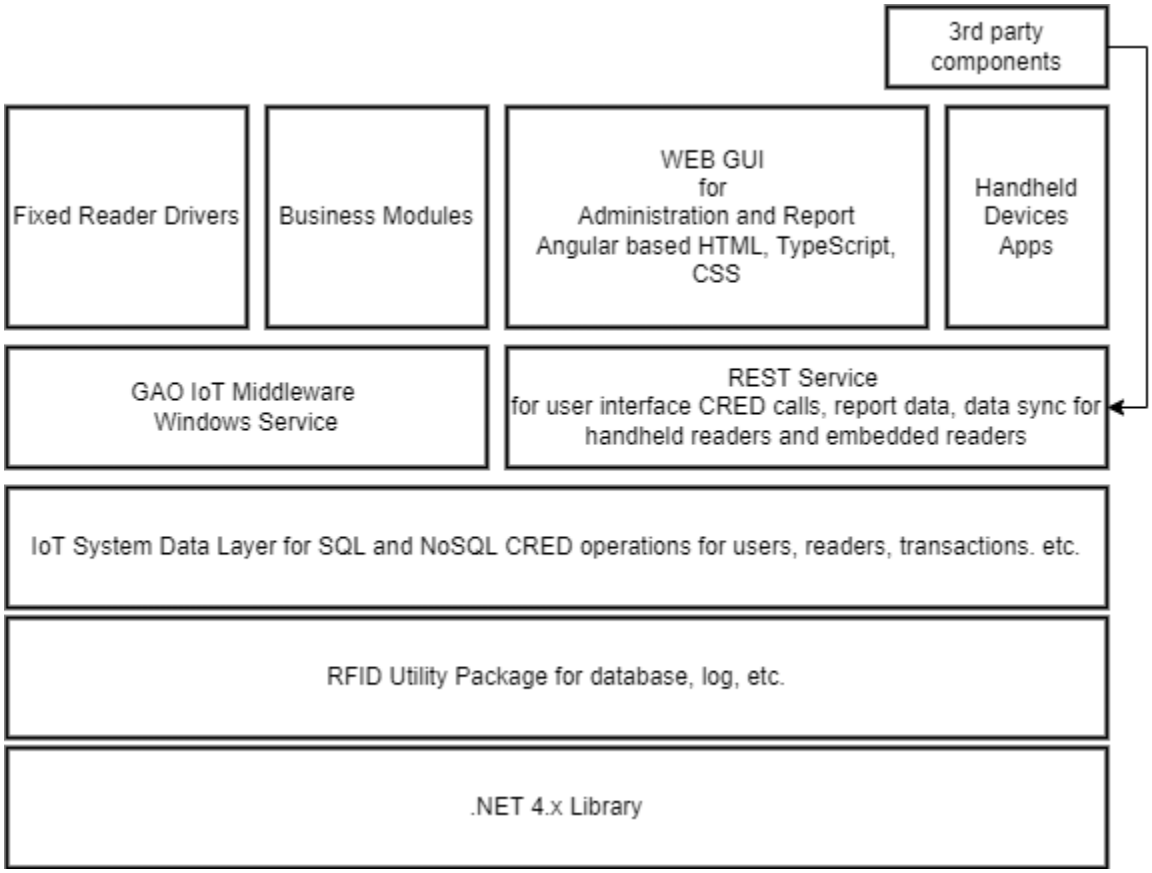


GAO BLE, RFID & IoT system context diagram

2. GAO BLE, RFID & IoT Software Components and Layers

GAO BLE, RFID & IoT system includes middleware, Web server, on reader application and Handheld Reader application.

GAO BLE, RFID & IoT middleware is a Windows based service that provides runtime interfaces for RFID readers, business modules and data synchronization to communicate embedded on reader application and handheld readers. The middleware provides a library of business modules and RFID reader drivers as described below:



GAO BLE, RFID & IoT System Components and Layers

- **GAO BLE, RFID & IoT Middleware** – a Windows based service that provides runtime interfaces for RFID fixed readers, business modules and data synchronization, this module is directly linked with reader drivers and business modules in the runtime. The middleware connects to database server either in the LAN or VPN environment
- **Fixed Reader and Driver Interface** – Every supported GAO Fixed RFID reader has device driver that provides interface to the reader communication specific messages and implements RFID reader device interface that is an abstraction layer with common APIs for active and passive RFID readers that need to connect to the RFID middleware during runtime.

A vender specific reader can implement a reader abstract class to interact with other RFID components. Device driver will pass the received RFID messages to the business module.

- **Business Modules** – this component will be application specific depending on the client requirements. Internally, it will GAO Business Module Interface that provides a business logic abstraction layer that enables various business specific modules to plug into the RFID system by implementing a business module abstract class. This module updates RFID reader activity data to the GAO BLE, RFID & IoT database server.
- **BLE, RFID & IoT REST Services** – REST APIs for system administration related objects, such as users, devices, locations, tags, managed objects (people, assets), report query data and Data Synchronization APIs that provides out-process integration for devices that are not resident in the RFID Middleware process such as handheld reader devices or embedded reader devices. The component provides local data for embedded and handheld reader devices with the latest required information such as location information, reader configuration or user data.
- **Web GUI** – provides end user and administrator interface. This layer will be implemented by Google Angular and deployed in the IIS web server. The layer communicates with REST APIs on the enterprise network via Internet or LAN. The component enables system administrators to manage RFID server, users, configure RFID devices, tags, and access system reports from a Web browser.
- **Handheld Component** – provides user interface based on business specific specification. Each handheld device will have a local database with required information for supported business modules. The local database is updated with a configured server with wireless or USB port. The device stores RFID transactions to the handheld database and uploads transactions to the RFID server whenever the device is connected to the server.
- **Embedded Component** –provides RFID tag reading by using LLRP API, barcode reading, business logic specific processing, and data communication with RFID servers using FTP, UDP and TCP protocols.
- **RFID Utility Library** – this layer provides commonly users classes such as database manager for SQL server, NoSQL server connection and connection pool. Common Log class for the system. Security APIs, communication APIs for external REST request/response calls, etc.
- **Third Party Components** – for any external components that unknow to the BLE, RFID & IoT system. They can integrate with GAO BLE, RFID & IoT system with REST APIs by authenticated sessions. GAO BLE, RFID & IoT REST APIs could be called from any third-party system and it is programming language independent.