! DISCONTINUED: As of version 4.0, the MyGeotab API Adapter solution has been fully migrated to **Data Model 2 (DM2)**. The original data model has been deprecated and removed from the solution effective January 2, 2026. This document is provided for reference purposes only and is no longer updated or maintained. Moving forward, please refer to the current guide:
MyGeotab API Adapter DM2 — Solution and Implementation Guide [PUBLIC]

MyGeotab API Adapter [DISCONTINUED]

Solution and Implementation Guide

Latest Update: 2025-12-01

Table of Contents

IMPORTANT: Version 3.x - Data Model 2 (DM2)

! IMPORTANT: As of version 3.0, the MyGeotab API Adapter solution has migrated to a **new data model**. The **original data model and the Data Optimizer have been deprecated** and will be removed from the solution after **December 31, 2025**. This provides a reasonable period for integrators to modify any integrations that need to be migrated to the new data model.

Details about these changes are provided in this section. Future updates regarding timing will also be included in this section.

About Data Model 2 (DM2)

The original data model, used exclusively prior to version 3.0.0 of the MyGeotab API Adapter and detailed in this document, is based on the premise of the adapter database serving only as a staging database where "raw" data extracted from the Geotab platform is deposited and there is a need to develop some sort of ETL process to move the data into some downstream "usable" format. Among other limitations, it does not contain indexes or physical relationships required to maintain query performance over time as data volume increases. See the Database Maintenance and Suggested Strategy sections in this document for more information.

The Data Model 2 (DM2) version of the adapter database represents an evolution from the pure staging database paradigm. It is designed with greater performance, scalability and maintainability in mind. Specifically:

- It is normalized with many indexes and relationships between tables.
- Where possible, the string values of Geotab Entity Ids are converted to their underlying numeric values and used as the primary key on the subject table, facilitating fast queries even with many joins and large data volumes. See "GeotabId" and "id" Columns for more information.
- "Feed Data" tables are partitioned by month, week or day. See Database Partitioning for details.
- The MyGeotab API Adapter includes Automated Database Maintenance functionality that helps to keep tables and indexes optimized through regular maintenance while the application is in operation (DM2 only).

With the DM2 version of the adapter database, it is possible to achieve fast throughput speed while also providing the ability to query the data with fast results even with large data volumes.

Important Information

Important information about the evolution of the MyGeotab API Adapter to the new data model is included in the following subsections.

One Application, Two Data Models and Two Guides

Given that the MyGeotab API Adapter solution is Free and Open Source Software (FOSS), subject to MIT license, with a very limited contributor pool of Geotab staff, it was determined that that best way to implement this major change in data model is through an evolutionary process whereby the single application is modified to support both the old and new data models throughout the transition.

The UseDataModel2 Setting

The **UseDataModel2** setting in the DatabaseSettings section of the appsettings.json file determines which data model the API Adapter application will use. However, the adapter database must be set up with the correct data model beforehand, or else it will not work.

If Using Data Model 1 (Original)...

If using the original data model, **follow instructions in this document**.

If Using Data Model 2 (DM2)...

If using the new data model (see the **Data Model 2 (DM2)** section below for more information), **follow instructions in the new guide**: MyGeotab API Adapter DM2 — Solution and Implementation Guide [PUBLIC]

Supported Databases: SQL Server and PostgreSQL

Starting with version 3.0.0, the MyGeotab API Adapter will support SQL Server and PostgreSQL. Oracle database will not be supported moving forward due to very low usage combined with a high cost to develop and maintain.

Data Optimizer Deprecated (Capabilities Moved to Core API Adapter in Version 3.x)

The Data Optimizer is deprecated as of version 3.0. With the greatly enhanced performance and scalability of DM2, the capabilities that were included in the Data Optimizer – primarily attaching location information to StatusData and FaultData records via interpolation using LogRecords – have been replicated and included in the core MyGeotab API Adapter solution. In fact, location interpolation is exponentially faster using DM2 - often able to keep-up with the flow of data even for larger fleets.

In addition to the raw performance improvements, deprecation of the Data Optimizer in favour of including the functionality in the core MyGeotab API Adapter means less complexity and cost resulting from only having to deploy one application and one database as opposed to two of each. Plus, these capabilities are also available for both SQL Server and PostgreSQL (Data Optimizer only supported SQL Server).

In DM2:

- FaultData location information can be found in the FaultDataLocations2 table.
- StatusData location information can be found in the StatusDataLocations2 table.

FAQ

Answers to frequently asked questions are provided below.

Which Geotab Entities are Available with Each of the Data Models?

The following table provides a quick reference showing the Geotab entities that are covered when using the MyGeotab API Adapter with each of the data models. It will be updated as support for additional entities is added to DM2. For more detailed information, refer to the List of Tables in this document for the original data model, or the List of Tables available in DM2.

Key: Available (✔), Not Yet Available (–), Not Available (✘)

Geotab Entity Type	Original Data Model	Data Model 2 (DM2)
BinaryData	✔	✔
ChargeEvent	✔	✔
Condition	✔	✔* (A column in Rules2 table)
DebugData	✔	✘
Device	✔	✔
DeviceStatusInfo	✔	✔
Diagnostic	✔	✔
DriverChange	✔	✔
DutyStatusAvailability	✔	✔
DutyStatusLog	✔	✔
DVIRLog	✔	✔
ExceptionEvent	✔	✔
FaultData	✔	✔
FaultData Locations	✔* (Only with Data Optimizer)	✔
FuelAndEnergyUsed	✘	✔
Group	✔	✔
LogRecord	✔	✔
Rule	✔	✔
StatusData	✔	✔
StatusData Locations	✔* (Only with Data Optimizer)	✔
Trip	✔	✔
User	✔	✔
Zone	✔	✔
ZoneType	✔	✔

VersionInformation	✔	✔

We're new to the API Adapter. Should we use Data Model 2 (DM2) or the Original Data Model?

Once the Geotab entity types currently supported with the original data model are migrated to DM2, the original data model will be deprecated and will eventually disappear from the solution (after a period of time to provide integrators the opportunity to transition).

Check the table in the "Which Geotab Entities are Available with Each of the Data Models?" section above. If the required tables exist in DM2, then it is recommended to use DM2. If there are one or more tables supported in the original data model (see List of Tables in this document) that are not yet available in DM2, there are a few options:

1. **Start your solution with the tables that are currently available in DM2 and wait for the other tables to be added to DM2.** However, since the MyGeotab API Adapter is an open source solution, there are no guarantees on if or when such tables will be available.
2. **Setup two MyGeotab API Adapter instances - one using the original data model and the other using DM2.** Use DM2 where possible and only use the original data model tables for entities that aren't yet supported in DM2. There is more short-term complexity in implementing this approach, but potentially less long-term change required to fully migrate to DM2.
3. **Use the original data model for now and migrate to DM2 in the future.** This involves the least amount of effort in the short term, but the most effort when it comes to migration to DM2 in the future. In some cases, where the API Adapter is used to download data on an ad-hoc basis for analysis, for example, this may be the best option.

We are Already Using the API Adapter. What is the Process to Upgrade to DM2?

Due to the fundamental differences between the original and DM2 data models, combined with the fact that many integrators modify/enhance the adapter database and the fact that this is an open source solution provided at no cost, Geotab cannot provide upgrade scripts to migrate between data models. However, given the many improvements in DM2, it may be beneficial to revisit and alter or replace downstream processes anyway.

Upgrade options are as follows and the best option depends on the individual situation:

1. **Re-extract data using DM2.** Depending on the size of the fleet, the Geotab entities being used and the level of downstream integration, it might be feasible to simply setup the API Adapter with DM2 and re-extract any data that was already collected. For smaller fleets (e.g. less than 200 devices), it may be possible to extract one year's worth of data in only an hour or two.
2. **Setup a second MyGeotab API Adapter instance using DM2 and run it in parallel to the existing instance.** Use DM2 where possible and only use the original data model tables for entities that aren't yet supported in DM2. Running them in parallel may provide additional flexibility to migrate queries and downstream processes and compare efficiency of modified processes using DM2 vs. those using the original data model.
3. **Setup a new MyGeotab API Adapter instance using DM2 and build scripts to migrate existing data from the original adapter database.** This is the most complex option and should be avoided unless absolutely necessary since Geotab is unable to provide support in this regard. If there is a need to build custom data

migration scripts, the data dictionaries and scripts for the two different data models may prove to be useful:

a. **Original Data Model:**
    i. Data Dictionary
    ii. Database Setup (script details in this section)
b. DM2:
    i. Data Dictionary
    ii. Database Setup (script details in this section)

Do we absolutely need to migrate to using DM2?

Strictly speaking, it may not be necessary to migrate an existing instance of the MyGeotab API Adapter to DM2. However, those planning to use the solution for the longer term should plan on migrating. Key points to consider:

- Once all of the Geotab entities supported in the original data model have been added to DM2, the original data model will be deprecated (after a reasonable amount of time – perhaps 6 to 12 months) and DM2 will be the only data model option.
- Starting from MyGeotab API Adapter version 3.0, any new entities supported in the API Adapter will only be added to DM2. The original data model will not change other than to address any bugs or important fixes that might arise.
- Once the original data model is fully deprecated and no longer included in the solution, Geotab will be unable to provide further support around the original data model.
- Since the MyGeotab API Adapter solution is free open source software, if there is some need to continue using the original data model in the longer term, integrators are able to download their own copies of the source code, including database scripts and maintain those themselves or with the help of a third-party.

We really like the Data Optimizer and its ability to attach location information to StatusData and FaultData records. Will that be lost if we migrate to using DM2?

Absolutely not! In fact, with DM2, the capabilities of the Data Optimizer have been rolled into the base API Adapter solution with no need for a separate database and a separate application. See the Data Optimizer Deprecated (Capabilities Moved to Core API Adapter in Version 3.x) section for more information.

What future plans are there for the API Adapter?

While the MyGeotab API Adapter solution is free and open source software and there are, as a result, no guarantees in terms of planned features or timelines, the following points reflect the current outlook. Note that these are subject to change and may be updated in the future.

- The shorter-term focus with the MyGeotab API Adapter will be to add DM2 support for the Geotab entities that are currently supported with the original data model.
- Once all of the currently-supported entity types have been added to DM2, a grace period of 6-12 months will begin during which the original data model will still be supported in the API Adapter alongside DM2. After the grace period, the original data model will be deprecated and removed from the MyGeotab API Adapter solution altogether, leaving DM2 as the only supported data model.
- Support for additional Geotab entity types as well as additional “data enhancement services” may be added to the solution over time.

Revision History

Refer to the Change Log section for information about changes to the MyGeotab API Adapter solution and this document.

Introduction

Streaming of data from the MyGeotab platform into external systems via the Geotab API is accomplished using the data feed - a lightweight and highly-scalable incremental polling mechanism. Building a full-scale integration typically involves utilizing numerous data feeds to pull various types of data from a MyGeotab database. There are many complexities inherent in developing a solid integration.

The MyGeotab API Adapter solution serves as both an example of proper integration via data feeds and the potential foundation for those seeking to develop new integrations with the Geotab platform. Essentially, it uses data feeds to pull the most common data sets from a MyGeotab database and stream the data into tables within a SQL Server, PostgreSQL or Oracle database; this could account for half the work in terms of a unidirectional integration where the data from the database is further processed for integration into an external system.

This document provides detailed information about the MyGeotab API Adapter solution along with instructions related to its deployment.

Data Optimizer - Taking the Adapter to the Next Level!

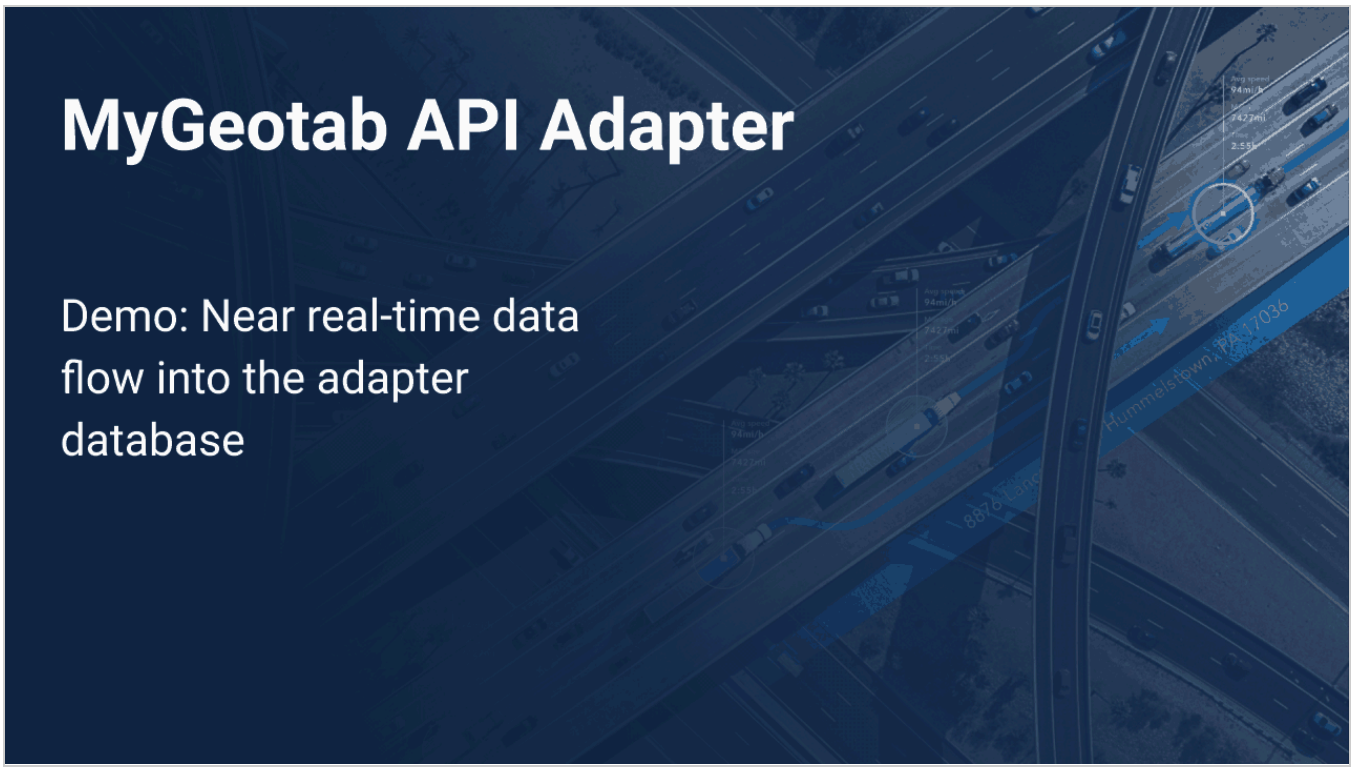
As detailed in the Database Maintenance section of this guide, the adapter database has been designed as a staging database, serving as an intermediary between the Geotab platform and the final repository where the extracted data will ultimately be stored for further processing and consumption by downstream applications.

The **Data Optimizer** is another application that takes the adapter solution to the next level, following the Suggested Strategy outlined in this guide. It migrates data from the adapter database into an "optimizer database" which can then be queried directly by applications. The Data Optimizer includes additional services that enhance the data and overcome certain challenges such as linking data points with different timestamps from different tables. For example, the StatusData and FaultData tables have added Latitude, Longitude, Speed, Bearing and Direction columns that can optionally be populated using LogRecords and interpolation techniques.

Full details pertaining to the Data Optimizer can be found in the MyGeotab API Adapter - Data Optimizer - Solution and Implementation Guide.

Demo - Near Real-Time Flow of Data Into the Adapter Database

The following demonstration shows the near real-time flow of data into the adapter database. In this demo, the adapter output is shown on the left while a query to obtain record counts from certain tables in the adapter database is executed repeatedly on the right. A full-resolution version of this demo can be found here.



MyGeotab API Adapter Highlights

When contemplating a new MyGeotab integration, there are many potential options. This section identifies some key features of the MyGeotab API Adapter solution which may serve both to highlight the benefit of its use as well as to identify likely requirements for a new integration being built from scratch.

In contrast to starting a brand new integration from the ground up, the most obvious benefit of utilizing the adapter solution is that it's already available and open source on GitHub. So, it can be used as-is, or modified as necessary to meet specific requirements. A custom-built solution will most likely need to incorporate many of the features that are built into the MyGeotab API Adapter. Thus, at a bare minimum, the solution can serve to demonstrate ways by which such requirements may be implemented.

Highlights of the MyGeotab API Adapter solution are as follows:

Efficiency

- The number of MyGeotab API calls being made has been minimized via use of data feeds and caching to the extent possible.
- Chattiness with the MyGeotab API Adapter database has also been minimized.
- Asynchronous methods and parallel processing have been incorporated where possible.

Data Integrity

- Feed tokens are tracked and persisted to the MyGeotab API Adapter database.
- Write operations are executed within transactions to ensure all-or-none processing of FeedResult batches and tokens for individual feeds are persisted to the database only upon successful commit.
- Feeds will continue from the last feed versions upon restart of the MyGeotab API Adapter for any reason.

- Safeguards are in-place to prevent missing or duplicating data or inadvertently mixing data from multiple MyGeotab databases.

Database-Agnosticity

- The Dapper ORM (https://github.com/StackExchange/Dapper, https://dapper-tutorial.net/) is used to map .NET objects to rows in corresponding database tables.
- A repository pattern has been used - separating data-access code from application logic.
- A MyGeotab API Adapter database schema has been created for SQL Server (also Azure SQL), PostgreSQL and Oracle and it is possible to switch between them by simply changing the database connection information in appsettings.json.

Resilience

- Mechanisms are in place to allow the MyGeotab API Adapter to continue operation if connectivity is lost either to the MyGeotab API or to the adapter database (e.g. planned maintenance, network issue, etc.) - in such events, the MyGeotab API Adapter will continuously try reconnecting and will resume where it left off once connectivity is re-established. Any partially completed operations are rolled-back and re-executed as noted under the Data Integrity point.

Configurability

- Via appsettings.json, it is possible to configure the MyGeotab API Adapter at a very granular level:
    - Feeds can be configured for individual object types (LogRecord, StatusData, FaultData, Trip, ExceptionEvent, DVIRLog).
    - Feed intervals can be set at a per-feed level.
    - Individual feeds can be enabled or disabled as required.
    - Update and refresh intervals can be set for individual caches of various object types (which also utilize feeds where available), including Controller, Device, Diagnostic, Defect, FailureMode, Group, Rule, UnitOfMeasure, User and Zone.
    - Feed results may be filtered for specific devices and/or diagnostics so that only the data required for a given instance will be persisted to the adapter database.
    - Feeds can be started at the current time, a specific time or based on feed version, thereby enabling consumers to pull as little or as much data as desired into the adapter database.
    - The above configuration options allow the MyGeotab API Adapter to reduce the number of MyGeotab API calls to only those that are required to meet the needs of a specific end-customer implementation, thereby easing the load on the MyGeotab platform as the solution is deployed for many customers.

Deployment Model

- The MyGeotab API Adapter can be published using the self-contained deployment model targeting any supported runtime.
- The solution is intended to be deployed on the basis of having one copy of the MyGeotab API Adapter with a paired adapter database per MyGeotab database.

Logging

- NLog has been incorporated as the logging mechanism.

- Log messages have been added strategically to assist with debugging issues once the solution has been deployed.
- Additional logging tools could be utilized for monitoring purposes.

Code Readability and Reusability

- One of the primary objectives in developing this solution was to ensure maximum reusability.
- In addition to creating this document, effort has been made to ensure extensive code commenting throughout in order to assist integrators.

Quick Start Guide

This section provides a quick summary of the steps required to download and deploy the MyGeotab API Adapter. While it is no substitute for all of the detail provided in this guide, the most important steps are highlighted and this section may serve as a high-level deployment checklist. Additionally, a demo video is provided to illustrate the deployment process and give developers an understanding of the inner workings of the solution.

Quick Start: Download and Deploy the MyGeotab API Adapter

> ✱ **NOTE**: The steps outlined below are for one of several possible ways the solution can be deployed. Information relating to other deployment possibilities can be found throughout this guide. For instance, the MyGeotab API Adapter can
> - be deployed to various operating systems (Windows and Linux-based packages are included with each release published to GitHub);
> - utilize a number of different database types (e.g. SQL Server, PostgreSQL, Oracle, etc.); and
> - be modified to include new or altered capabilities.

The steps to download and deploy the latest release of the MyGeotab API Adapter in a **Windows-based** environment with **SQL Server** as the database are as follows:

1. Download the latest release of the MyGeotab API Adapter from GitHub (i.e. following Steps 1-3 in the Using Published Release from GitHub section). The files to download are:
   - **MyGeotabAPIAdapter_SCD_win-x64.zip**
   - **SQLServer.zip**

   Once downloaded, extract the contents of the zip files.

2. Setup the database (following Steps 1-3 in the SQL Server database setup section):
   1. Create a SQL Server database named **geotabadapterdb**.
   2. Create a login named **geotabadapter_client** along with a user by the same name using the script provided in Step 2 of the SQL Server database setup section.
   3. Execute the database creation script, **geotabadapterdb-DatabaseCreationScript.sql**, which can be found in the *SQLServer* folder extracted from the *SQLServer.zip* file that was downloaded in Step 1.

3 Configure and deploy the MyGeotab API Adapter application (following Steps 1-4 in the Deploy section):

1. Copy the ***MyGeotabAPIAdapter_SCD_win-x64*** folder extracted from the *MyGeotabAPIAdapter_SCD_win-x64.zip* file that was downloaded in Step 1 to the machine on which the MyGeotab API Adapter application is to reside.
2. Modify the **appsettings.json** file, contained in the above application folder, as needed. See the appsettings.json section for more information. Important settings are as follows:
    - The **DatabaseProviderType** and **DatabaseConnectionString** settings in the DatabaseSettings section must be set so that the adapter application can connect to the adapter database using the login created in Step 2 of the database setup, above.
    - The **MyGeotabServer**, **MyGeotabDatabase**, **MyGeotabUser** and **MyGeotabPassword** settings in the LoginSettings section must be set (ideally with the credentials of a service account) so that the adapter application can connect to the target MyGeotab database.
    - In the AppSettings - GeneralFeedSettings section:
        - Adjust the **FeedStartOption** setting, if desired (as well as the **FeedStartSpecificTimeUTC** setting, if *FeedStartOption* is set to SpecificTime).
        - If there is a desire to limit data extracted from the MyGeotab system to a specific set of devices, adjust the **DevicesToTrack** setting. Note that MyGeotab Groups functionality may be used to achieve the same result.
        - If there is a desire to extract StatusData and/or FaultData only for specific diagnostics, adjust the **DiagnosticsToTrack** setting.
    - Enable or disable specific data feeds to pull only the desired data by adjusting the settings for individual feeds in the AppSettings - Feeds section.
3. Review the **nlog.config** file, contained in the above application folder, and make any necessary changes. See the nlog.config section for more information.
    - It should not be necessary to modify the nlog.config file; this step is only included to highlight the fact that modification of logging behaviour is possible.
4. Start the MyGeotab API Adapter by running the **MyGeotabAPIAdapter.exe** file contained in the above application folder.
   ✱ **NOTE**: In a production environment, it is best to setup a process to run the adapter using a system account. On a Windows Server, for example, Windows Task Scheduler can be used to create a task that runs *MyGeotabAPIAdapter.exe* on server startup.

Quick Start: Step-by-Step Demo Video

A step-by-step demo video has been created to assist those looking to get started quickly with the MyGeotab API Adapter. The following is a list of start times of various topics within the recording:

Start Time	Topic
1:23	Intro to the MyGeotab API Adapter
5:41	**How to Download and Deploy a Published Release of the Adapter (with SQL Server)**
22:38	Database Maintenance & Suggested Strategy
25:40	Understanding the Database Through Examples

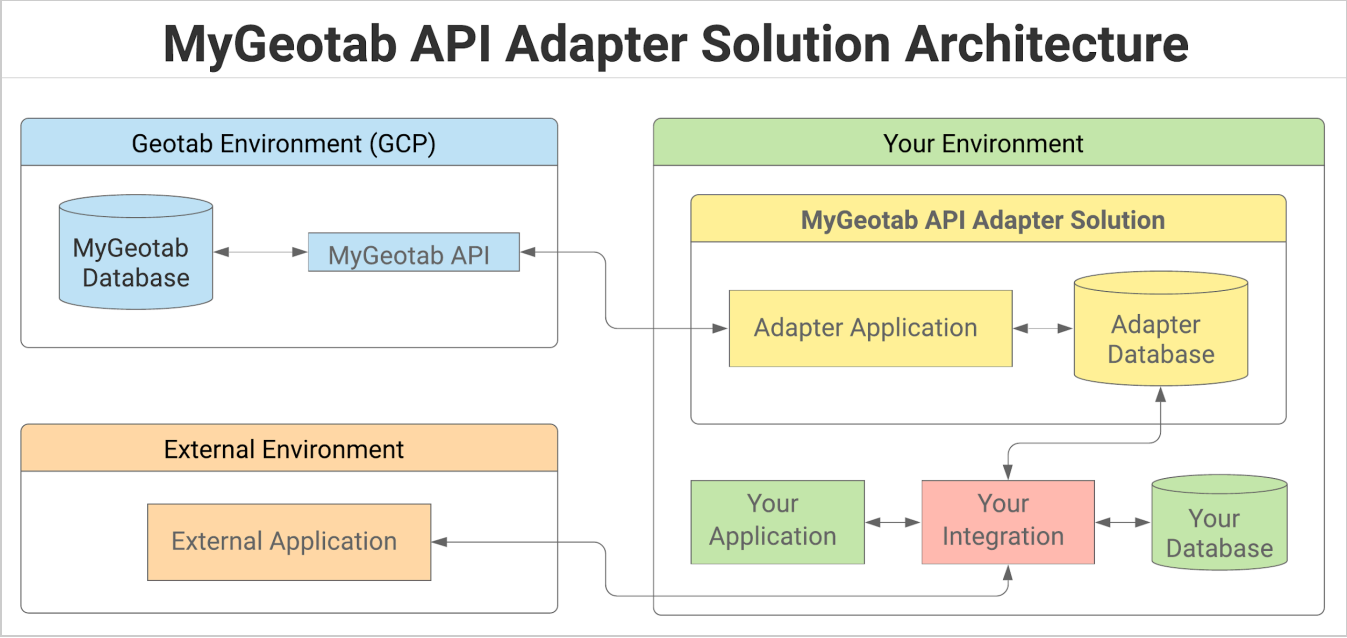
26:35	Logging - Example Showing How to Use Logging to Debug a Configuration Issue
31:15	**Download Source Code and Code Walk-Through (for developers)**
31:49	Download/Clone the Git Repository
33:54	Open the Solution in Visual Studio
35:07	Solution Architecture
37:12	Application Logic
39.06	Code Walk-Through for Developers
1:14:58	Publishing the MyGeotab API Adapter
1:18:18	DVIRLog Manipulator (Bi-Directional Workflow)

Solution Information

This section provides detailed information about the architecture, logic and data models of the MyGeotab API Adapter solution. Usage and deployment-related instructions can be found in the Solution Usage and Implementation section of this guide.

Solution Architecture

The following diagram provides an overview of the MyGeotab API Adapter solution architecture.



The solution consists of two components - the adapter and a database - both of which are deployed in the partner/reseller environment. The adapter is a collection of NET 8.0 Background Services (C#) that interface

between the MyGeotab API and the database, essentially pulling data from the former and writing to tables in the latter. Each customer MyGeotab database must have a dedicated adapter and database pair; it is not possible to mix data from multiple MyGeotab databases. The Geotab partner/reseller is responsible for integrating between the database and partner/reseller databases/applications - potentially with assistance from a third-party solutions integrator.

Database

Out-of-the-box, the MyGeotab API Adapter supports SQL Server, PostgreSQL and Oracle for use as the adapter database into which data retrieved via the MyGeotab API is written. The solution uses a repository pattern and an Object-Relational Mapper - Dapper ORM - making it easy to add support for different types of databases. See the Adding Support for Other Database Types section for specifics.

List of Tables

The following table lists all of the tables contained in the database along with descriptions that include the associated MyGeotab API objects, where applicable. Further detail relating to the structure and fields of individual tables can be found in the Data Dictionary section.

✱ **NOTE**: Each table is assigned to one of the following categories:

- **Feed data:** Records in *feed data* tables generally consist of data points collected using data feeds and are not modified once written to the database. These tables can accumulate vast quantities of records within short periods of time. It is highly recommended to have a data management strategy - especially for tables in this category.
- **Reference data:** These tables generally contain user-added data. Values are referenced by **GeotabId** in the *feed data* tables. Records in *reference data* tables can change over time and only the latest version of each record is maintained. Record counts in *reference data* tables tend to be small and relatively stable over time.
- **Commands:** These tables are utilized for issuing data manipulation commands to the Geotab platform - for example, updating the repair status of DVIRDefects after related work orders have been completed in an external system. Rather than using the MyGeotab API Adapter to extract data from a MyGeotab database and then having to use the MyGeotab SDK to send updates back to the MyGeotab database, the details of these updates can simply be inserted as rows into the relevant *commands* tables and the adapter will take care of the SDK-related work.
- **Command exceptions:** For each *commands* table, there will be an associated *command exceptions* table. If a row is inserted into the *commands* table and it does not pass data validation checks, or if an exception occurs when the adapter attempts to execute the command, a copy of the original row in the *commands* table will be added to the *command exceptions* table along with the related error message. This is to assist in debugging and to provide feedback that would otherwise be provided in the responses to commands issued via the MyGeotab SDK.
- **System info:** These tables are used by the adapter and do not offer any other specific benefit.

WARNING! It is possible for the database to grow very large very quickly, resulting in **potential disk space and performance issues**. In particular, the ***feed data*** tables can accumulate vast quantities of records within short periods of time. See the Database Maintenance section for more information.

Table Name	Category	Description
BinaryData	Feed data	Contains data corresponding to MyGeotab BinaryData objects.
ChargeEvents	Feed data	Contains data corresponding to MyGeotab ChargeEvent objects.
Conditions	Reference data	Contains data corresponding to MyGeotab Condition objects.
DebugData	Feed data	Contains data corresponding to MyGeotab DebugData objects.
Devices	Reference data	Contains data corresponding to MyGeotab Device objects.
DeviceStatusInfo	Reference data	Contains data corresponding to MyGeotab DeviceStatusInfo objects.
Diagnostics	Reference data	Contains data corresponding to MyGeotab Diagnostic objects.
DriverChanges	Feed data	Contains data corresponding to MyGeotab DriverChange objects.
DutyStatusAvailabilities	Reference data	Contains data corresponding to MyGeotab DutyStatusAvailability objects.
DutyStatusLogs	Feed data	Contains data corresponding to MyGeotab DutyStatusLog objects.
DVIRDefectRemarks	Reference data	Contains data corresponding to MyGeotab DefectRemark objects.
DVIRDefects	Reference data	Contains data corresponding to defects associated with DVIRLogs. It includes data derived from MyGeotab DVIRLog, DVIRDefect, Defect and Group objects.
DVIRDefectUpdates	Commands	May be used to send DVIRDefect updates to the MyGeotab database with which the adapter is configured to communicate. See the DVIRLog Manipulator section for more information.
DVIRLogs	Feed data	Contains data corresponding to MyGeotab DVIRLog objects.
ExceptionEvents	Feed data	Contains data corresponding to MyGeotab ExceptionEvent objects.
FailedDVIRDefectUpdates	Command exceptions	Contains copies of any records that were inserted into the *DVIRDefectUpdates* table and did not result in successful

		updates of DVIRDefects in the MyGeotab database. The *FailureMessage* column provides details about the reason why a given command failed. See the DVIRLog Manipulator section for more information.
FaultData	Feed data	Contains data corresponding to MyGeotab FaultData objects.
Groups	Reference data	Contains data corresponding to MyGeotab Group objects.
LogRecords	Feed data	Contains data corresponding to MyGeotab LogRecord objects.
MyGeotabVersionInfo	System info	Contains software version information (obtained from MyGeotab API VersionInformation) about the MyGeotab server and database that the subject adapter database is associated with via the MyGeotab API Adapter.
OServiceTracking	System info	System table used by the MyGeotab API Adapter.
Rules	Reference data	Contains data corresponding to MyGeotab Rule objects.
StatusData	Feed data	Contains data corresponding to MyGeotab StatusData objects.
Trips	Feed data	Contains data corresponding to MyGeotab Trip objects.
Users	Reference data	Contains data corresponding to MyGeotab User objects.
ZoneTypes	Reference data	Contains data corresponding to MyGeotab ZoneType objects.
Zones	Reference data	Contains data corresponding to MyGeotab Zone objects.

Data Dictionary

SQL Server is the main supported database and any database-specific references such as data types will relate to the SQL Server version of the schema. For information such as schema details related to other supported database types, refer to the Database Setup section. The tables and views included in the adapter database schema are detailed in the following subsections.

"GeotabId" and "id" Columns

Most of the tables in the adapter database have "*GeotabId*" and "*id*" columns.

The ***GeotabId*** is the unique identifier for the specific Entity object in the Geotab system; it must be used when relating objects to each other or back to the Geotab system. For example the *DeviceId* in the *LogRecords* table is associated with the *GeotabId* in the *Devices* table.

The ***id*** is the unique identifier for the record in the adapter database table and is entirely unrelated to the Geotab system. It is of *bigint* data type, indexed and defined as the primary key. Its purpose is to allow downstream

processes to delete records (from the *feed data* tables) as part of the suggested strategy while maintaining optimal performance and avoiding database concurrency/contention issues that could arise (if the *GeotabId* were to be used) when trying to delete records while the adapter is actively inserting/updating records.

BinaryData

The *BinaryData* table contains data corresponding to MyGeotab BinaryData objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
BinaryType	nvarchar(50)	Yes	The BinaryDataType.
ControllerId	nvarchar(50)	No	The Id of the Controller associated with the subject BinaryData.
Data	nvarchar(1024)	No	The binary data for the subject BinaryData object.
DateTime	datetime2(7)	Yes	The date and time of the logging of the data.
DeviceId	nvarchar(50)	Yes	The Id of the Device (in the Devices table) associated with the subject BinaryData.
Version	nvarchar(50)	Yes	The version of the entity.
RecordCreationTimeUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating when the subject record was added to the adapter database.

ChargeEvents

The *ChargeEvents* table contains data corresponding to MyGeotab ChargeEvent objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
ChargeIsEstimated	bit	No	Indicates whether the charge values were estimated.

ChargeType	nvarchar(50)	No	The ChargeType provided by the external power source.
StartTime	datetime2(7)	No	The date and time at which the ChargeEvent started.
DeviceId	nvarchar(50)	No	The Id of the Device (in the Devices table) associated with the subject ChargeEvent.
DurationTicks	bigint	No	The length of time the vehicle was charging as measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
EndStateOfCharge	float	Yes	The ending state of charge for this ChargeEvent.
EnergyConsumedKwh	float	Yes	The energy consumed during the ChargeEvent.
EnergyUsedSinceLastChargeKwh	float	Yes	The amount of energy drawn from the battery since the last ChargeEvent.
Latitude	float	Yes	The latitude of the location where the ChargeEvent occurred.
Longitude	float	Yes	The longitude of the location where the ChargeEvent occurred.
MaxACVoltage	float	Yes	The maximum AC Voltage over the ChargeEvent.
MeasuredBatteryEnergyInKwh	float	Yes	The amount of energy *in* measured during charging.
MeasuredBatteryEnergyOutKwh	float	Yes	The amount of energy *out* measured during charging.
MeasuredOnBoardChargerEnergyInKwh	float	Yes	The total amount of energy *in* measured on board during charging.
MeasuredOnBoardChargerEnergyOutKwh	float	Yes	The total amount of energy *out* measured on board during charging.
PeakPowerKw	float	Yes	The peak power used during the ChargeEvent.
StartStateOfCharge	float	Yes	The starting state of charge for this ChargeEvent.
TripStop	datetime2(7)	Yes	The time of the Trip.Stop from the Trip this ChargeEvent occurred in.
Version	bigint	No	The version of the entity.

RecordCreationTimeUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating when the subject record was added to the adapter database.

Conditions

The *Conditions* table contains data corresponding to MyGeotab Condition objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
ParentId	nvarchar(50)	Yes	The Id of the parent Condition, if the subject Condition has a parent.
RuleId	nvarchar(50)	Yes	The Id of the Rule (in the Rules table), with which the subject Condition is associated.
ConditionType	nvarchar(50)	No	The ConditionType of the subject Condition.
DeviceId	nvarchar(50)	Yes	The Id of the Device (in the Devices table) associated with the subject Condition.
DiagnosticId	nvarchar(100)	Yes	The Id of the Diagnostic associated with the subject Condition.
DriverId	nvarchar(50)	Yes	The Id of the Driver (corresponding to the Id in the Users table) associated with the subject Condition.
Value	float	Yes	The specified value that the subject Condition evaluates against.
WorkTimeId	nvarchar(50)	Yes	The Id of the WorkTime that the event must occur inside/outside of for the violation to occur.
ZoneId	nvarchar(50)	Yes	The Id of the Zone associated with the subject Condition.
EntityStatus	int	No	Indicates whether the subject corresponding object is active or deleted in the MyGeotab database. **1 = Active. 0 = Deleted.**

RecordLastChangedUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating the last time that the subject record was updated in the adapter database.

DebugData

The *DebugData* table contains data corresponding to MyGeotab DebugData objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
Data	nvarchar(max)	No	The binary data for the subject DebugData object.
DateTime	datetime2(7)	Yes	The date and time of the logging of the data.
DebugReasonId	bigint	Yes	The numeric value of the enumerator representing the reason for the DebugData record. Used for troubleshooting/debugging purposes only.
DebugReasonName	nvarchar(255)	Yes	The alphanumeric value of the enumerator representing the reason for the DebugData record. Used for troubleshooting/debugging purposes only.
DeviceId	nvarchar(50)	Yes	The Id of the Device (in the Devices table) associated with the subject DebugData.
DriverId	nvarchar(50)	Yes	The Id of the Driver (corresponding to the Id in the Users table) associated with the subject DebugData.
RecordCreationTimeUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating when the subject record was added to the adapter database.

Devices

The *Devices* table contains data corresponding to MyGeotab Device objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.

GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
ActiveFrom	datetime2(7)	Yes	The date the device is active from.
ActiveTo	datetime2(7)	Yes	The date the device is active to.
Comment	nvarchar(1024)	Yes	Free text field where any user information can be stored and referenced for this entity.
DeviceType	nvarchar(50)	No	Specifies the GO or Custom DeviceType.
Groups	nvarchar(max)	Yes	The list of Group(s) the subject entity belongs to. Presented in the form of a JSON array (e.g. [{"id":"GroupVehicleId"},{"id":"b2868"}]). Group Ids in this list correspond with **GeotabId** values in the Groups table. ***SQL Server and PostgreSQL only**.
LicensePlate	nvarchar(50)	Yes	The vehicle license plate details of the vehicle associated with the device.
LicenseState	nvarchar(50)	Yes	The state or province of the vehicle associated with the device.
Name	nvarchar(100)	No	The display name assigned to the device.
ProductId	int	Yes	The product Id. Each device is assigned a unique hardware product Id.
SerialNumber	nvarchar(12)	Yes	The serial number of the device.
VIN	nvarchar(50)	Yes	The Vehicle Identification Number (VIN) of the vehicle associated with the device.
EntityStatus	int	No	Indicates whether the subject corresponding object is active or deleted in the MyGeotab database. **1 = Active. 0 = Deleted.**
RecordLastChangedUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating the last time that the subject record was updated in the adapter database.

DeviceStatusInfo

The *DeviceStatusInfo* table contains data corresponding to MyGeotab DeviceStatusInfo objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description

id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
Bearing	float	No	The bearing (heading) in int degrees. Valued between 0 and 359 inclusive. 0 represents North, 90 represents East, and so on. -1 represents unknown bearing.
CurrentStateDuration	nvarchar(50)	No	The duration between the last Trip state change (i.e. driving or stop), and the most recent date of location information.
DateTime	datetime2(7)	No	The most recent DateTime of the latest piece of status, GPS or fault data.
DeviceId	nvarchar(50)	No	The Id of the Device (in the Devices table) associated with the subject DeviceStatusInfo.
DriverId	nvarchar(50)	No	The Id of the Driver (corresponding to the Id in the Users table) associated with the subject DeviceStatusInfo.
IsDeviceCommunicating	bit	No	A value indicating whether the Device is communicating.
IsDriving	bit	No	A value indicating whether the current Device state. If set true, is driving. Otherwise, it is stopped.
IsHistoricLastDriver	bit	No	Indicates whether the Device has been assigned to "UnknownDriver" and the last Trip Driver is represented in the **DriverId** column.
Latitude	float	No	The last known latitude of the Device.
Longitude	float	No	The last known longitude of the Device.
Speed	real	No	The current vehicle speed (in km/h). NaN represents an invalid speed.
RecordLastChangedUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating the last time that the subject record was updated in the adapter database.

Diagnostics

The *Diagnostics* table contains data corresponding to MyGeotab Diagnostic objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(100)	No	The unique identifier for the specific Entity object in the Geotab system.
GeotabGUID	nvarchar(100)	No	The underlying Globally Unique Identifier (GUID) of the Diagnostic. In the event that the GeotabId changes as a result of the assignment of a KnownId, this GeotabGUID will remain unchanged and can be used for reconciliation of Diagnostic Ids in any downstream integrations.
HasShimId	boolean	No	Indicates whether the Diagnostic is one that has a KnownId on the MyGeotab server side, but is unknown in the MyGeotab .NET API client (Geotab.Checkmate.ObjectModel NuGet package) used at the time of download.
FormerShimGeotabGUID	nvarchar(100)	Yes	If there is an earlier version of the Diagnostic where *HasShimId* is true, this value lists the *GeotabGUID* of that earlier Diagnostic so that the two, along with any associated data, can be logically related.
ControllerId	nvarchar(100)	No	The Id of the Controller related to the diagnostic; if applicable.
DiagnosticCode	int	Yes	The diagnostic parameter code number.
DiagnosticName	nvarchar(max)	No	The name of this entity that uniquely identifies it and is used when displaying this entity.
DiagnosticSourceId	nvarchar(50)	No	The Id of the Source of the Diagnostic.
DiagnosticSourceName	nvarchar(255)	No	The Name of the Source of the Diagnostic.
DiagnosticUnitOfMeasureId	nvarchar(50)	No	The Id of the UnitOfMeasure used by the Diagnostic.
DiagnosticUnitOfMeasureName	nvarchar(255)	No	The Name of the UnitOfMeasure used by the Diagnostic.

OBD2DTC	nvarchar(50)	Yes	The OBD-II Diagnostic Trouble Code (DTC), if the Diagnostic is from an OBD Source.
EntityStatus	int	No	Indicates whether the subject corresponding object is active or deleted in the MyGeotab database. **1 = Active. 0 = Deleted.**
RecordLastChangedUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating the last time that the subject record was updated in the adapter database.

DriverChanges

The *DriverChanges* table contains data corresponding to MyGeotab DriverChange objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
DateTime	datetime2(7)	No	The date and time of the driver change.
DeviceId	nvarchar(50)	No	The Id of the Device (in the Devices table) associated with the subject DriverChange.
DriverId	nvarchar(50)	No	The Id of the Driver (corresponding to the Id in the Users table) associated with the subject DriverChange.
Type	nvarchar(50)	No	The DriverChangeType.
Version	bigint	No	The version of the entity.
RecordCreationTimeUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating when the subject record was added to the adapter database.

DutyStatusAvailabilities

The *DutyStatusAvailabilities* table contains data corresponding to MyGeotab DutyStatusAvailability objects. Return to List of Tables.

> **WARNING!** Duration values in the DutyStatusAvailability table are inaccurate (out-of-date). The amount of inaccuracy can be defined as the duration between the value of the RecordLastChangedUtc field and the current time, in Coordinated Universal Time (UTC). This offset must be applied to the duration values in order

to improve currency of the data upon consumption. See the Maintaining Currency of Values Via Time Offsets section for more information.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
DriverId	nvarchar(50)	No	The Id of the Driver (corresponding to the Id in the Users table) associated with the subject DutyStatusAvailability record.
CycleAvailabilities	nvarchar(max)	Yes	Cycle available to the driver in the future. Presented in the form of a JSON array (e.g. [{"DateTime":"2021-01-14T05:00:00Z","Available":"00:00:00"},{"DateTime":"2021-01-15T05:00:00Z","Available":"00:00:00"},{"DateTime":"2021-01-16T05:00:00Z","Available":"00:00:00"},{"DateTime":"2021-01-17T05:00:00Z","Available":"00:00:00"},{"DateTime":"2021-01-18T05:00:00Z","Available":"20:18:55.6430000"},{"DateTime":"2021-01-19T05:00:00Z","Available":"1.20:18:55.6430000"},{"DateTime":"2021-01-20T05:00:00Z","Available":"2.12:00:00"}]).
CycleTicks	bigint	Yes	The duration of cycle hours left as measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
CycleRestTicks	bigint	Yes	The duration left before cycle rest must be taken. Measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
DrivingTicks	bigint	Yes	The duration left for driving. Measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
DutyTicks	bigint	Yes	The duration of total on-duty time left in the day. Measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.

DutySinceCycleRestTicks	bigint	Yes	The duty hours left since Cycle Rest. Measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
Is16HourExemptionAvailable	bit	Yes	Indicates whether 16 hour exemption is available.
IsAdverseDrivingExemptionAvailable	bit	Yes	Indicates whether adverse driving exemption is available.
IsOffDutyDeferralExemptionAvailable	bit	Yes	Indicates whether off-duty deferral exemption is available.
Recap	nvarchar(max)	Yes	Chronological array representing each day's On-duty time since the beginning of cycle. Presented in the form of a JSON array (e.g. [{"DateTime":"2021-01-07T05:00:00Z","Duration":"1.00:00:00"},{"DateTime":"2021-01-08T05:00:00Z","Duration":"1.00:00:00"},{"DateTime":"2021-01-09T05:00:00Z","Duration":"1.00:00:00"},{"DateTime":"2021-01-10T05:00:00Z","Duration":"1.00:00:00"},{"DateTime":"2021-01-11T05:00:00Z","Duration":"1.00:00:00"},{"DateTime":"2021-01-12T05:00:00Z","Duration":"1.00:00:00"},{"DateTime":"2021-01-13T05:00:00Z","Duration":"15:41:04.3570000"}]).
RestTicks	bigint	Yes	The duration left before rest break must be taken. Measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
WorkdayTicks	bigint	Yes	The duration of workday left in a day. Workday is a consecutive window that begins with first on-duty. Measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
RecordLastChangedUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating the last time that the subject record was updated in the adapter database.

DutyStatusLogs

The *DutyStatusLogs* table contains data corresponding to MyGeotab DutyStatusLog objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
Annotations	nvarchar(max)	Yes	The list of AnnotationLog(s) which are associated with this log.
CoDrivers	nvarchar(max)	Yes	The list of the co-driver User(s) (in the Devices table) for this log.
DateTime	datetime2(7)	No	The date and time the log was created.
DeferralMinutes	int	Yes	The deferral minutes.
DeferralStatus	nvarchar(50)	Yes	The DutyStatusDeferralType.
DeviceId	nvarchar(50)	Yes	The Id of the Device (in the Devices table) associated with the subject DutyStatusLog.
DistanceSinceValidCoordinates	real	Yes	The distance since last valid coordinate measurement.
DriverId	nvarchar(50)	Yes	The Id of the User (corresponding to the Id in the Users table) who created this log.
EditDateTime	datetime2(7)	Yes	The date and time the log was edited. If the log has not been edited, this will not be set.
EditRequestedByUserId	nvarchar(50)	Yes	The Id of the User (corresponding to the Id in the Users table) that requested an edit to this log.
EngineHours	float	Yes	The engine hours for the **DeviceId** at the **DateTime** of this log. The unit is seconds (not hours).
EventCheckSum	bigint	Yes	The event checksum of this log.
EventCode	int	Yes	The event code of this log (Table 6; 7.20 of the ELD Final Rule).
EventRecordStatus	int	Yes	The record status number of this log 1 = active 2 = inactive - changed 3 = inactive - change requested 4 = inactive - change rejected.
EventType	int	Yes	The event type number of this log 1 = A change in driver's duty-status 2 = An intermediate log 3 = A change in driver's indication of authorized personal use of CMV or yard moves 4 = A driver's certification/recertification of records 5 = A

			driver's login/logout activity 6 = CMV's engine power up / shut down activity 7 = A malfunction or data diagnostic detection occurrence (Table 6; 7.25 of the ELD Final Rule).
IsHidden	bit	Yes	Indicates whether the log is hidden.
IsIgnored	bit	Yes	If the log is ignored. True means it will not affect the Driver's HOS availability.
IsTransitioning	bit	Yes	A value indicating whether the log is in transitioning state.
Location	nvarchar(max)	Yes	An object with the location information for the log data.
LocationX	float	Yes	The longitude of the **Location**.
LocationY	float	Yes	The latitude of the **Location**.
Malfunction	nvarchar(50)	Yes	The DutyStatusMalfunctionTypes of this DutyStatusLog record. As a flag it can be both a diagnostic and malfunction state which is used to mark status based records (e.g. "D", "SB") as having a diagnostic or malfunction present at time of recording.
Odometer	float	Yes	The odometer in metres for the **DeviceId** at the **DateTime** of this log.
Origin	nvarchar(50)	Yes	The DutyStatusOrigin from where this log originated
ParentId	nvarchar(50)	Yes	The **Id** of the parent **DutyStatusLog**. Used when a DutyStatusLog is edited. When returning history, this field will be populated.
Sequence	bigint	Yes	The sequence number, which is used to generate the sequence ID.
State	nvarchar(50)	Yes	The DutyStatusState of the DutyStatusLog record.
Status	nvarchar(50)	Yes	The DutyStatusLogType representing the driver's duty status.
UserHosRuleSet	nvarchar(max)	Yes	The linked UserHosRuleSet. Only used to link rulesets to log events that affect the driver's operating zone and/or cycle. (Canadian ELD)
VerifyDateTime	datetime2(7)	Yes	The date and time the log was verified. If the log is unverified, this will not be set.
Version	bigint	No	The version of the entity.

RecordCreationTimeUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating when the subject record was added to the adapter database.

DVIRDefectRemarks

The *DVIRDefectRemarks* table contains data corresponding to MyGeotab DefectRemark objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
DVIRDefectId	nvarchar(50)	No	The Id of the DVIRDefect (in the DVIRDefects table) that the remark applies to.
DateTime	datetime2(7)	Yes	The date and time the remark was created.
Remark	nvarchar(max)	No	The remark text.
RemarkUserId	nvarchar(50)	Yes	The Id of the User (in the Users table) who created the remark.
EntityStatus	int	No	Indicates whether the subject corresponding object is active or deleted in the MyGeotab database. **1 = Active. 0 = Deleted.**
RecordLastChangedUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating the last time that the subject record was updated in the adapter database.

DVIRDefects

The *DVIRDefects* table contains data corresponding to defects associated with DVIRLogs. It includes data derived from MyGeotab DVIRLog, DVIRDefect, Defect and Group objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the DVIRDefect object in the Geotab system.

DVIRLogId	nvarchar(50)	No	The Id of the DVIRLog (in the DVIRLogs table) with which the subject DVIRDefect is associated.
DefectListAssetType	nvarchar(50)	Yes	The asset type of the defect list.
DefectListId	nvarchar(50)	Yes	The Id of the defect list (Group) that the defect belongs to.
DefectListName	nvarchar(255)	Yes	The Name of the defect list (Group) that the defect belongs to.
PartId	nvarchar(50)	Yes	The Id of the part (Group) that has the defect.
PartName	nvarchar(255)	Yes	The Name of the part (Group) that has the defect.
DefectId	nvarchar(50)	Yes	The Id of the Defect.
DefectName	nvarchar(255)	Yes	The Name of the Defect.
DefectSeverity	nvarchar(50)	Yes	The DefectSeverity of the Defect.
RepairDateTime	datetime2(7)	Yes	The date and time the DVIRDefect was repaired.
RepairStatus	nvarchar(50)	Yes	The RepairStatusType of this DVIRDefect.
RepairUserId	nvarchar(50)	Yes	The Id of the User (in the Users table) who repaired this DVIRDefect.
EntityStatus	int	No	Indicates whether the subject corresponding object is active or deleted in the MyGeotab database. **1 = Active. 0 = Deleted.**
RecordLastChangedUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating the last time that the subject record was updated in the adapter database.

DVIRDefectUpdates

The *DVIRDefectUpdates* table may be used to send DVIRDefect updates to the MyGeotab database with which the adapter is configured to communicate. If a record is inserted into this table and it fails to result in the subject DVIRDefect update being propagated to the MyGeotab database, a copy of the record will be inserted into the *FailedDVIRDefectUpdates* table and the reason for the failure will be provided in the *FailureMessage* column. Once a record has been processed, it will be deleted from the *DVIRDefectUpdates* table. For more detail, refer to the DVIRLog Manipulator section of this guide. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.

DVIRLogId	nvarchar(50)	No	The Id of the DVIRLog with which the subject DVIRDefect is associated.
DVIRDefectId	nvarchar(50)	No	The Id of the DVIRDefect that is to be updated in the Geotab system.
RepairDateTime	datetime2(7)	Yes	The date and time the DVIRDefect was repaired.
RepairStatus	nvarchar(50)	Yes	The RepairStatusType of this DVIRDefect.
RepairUserId	nvarchar(50)	Yes	The Id of the User who repaired this DVIRDefect.
Remark	nvarchar(max)	Yes	The remark text.
RemarkDateTime	datetime2(7)	Yes	The date and time the remark was created.
RemarkUserId	nvarchar(50)	Yes	The Id of the User who created the remark.
RecordCreationTimeUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating when the subject record was added to the adapter database.

DVIRLogs

The *DVIRLogs* table contains data corresponding to MyGeotab DVIRLog objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
CertifiedByUserId	nvarchar(50)	Yes	The Id of the User (in the Users table) who certified the repairs (or comments, if no repairs were made) to the Device or Trailer.
CertifiedDate	datetime2(7)	Yes	The date and time that the Device or Trailer was certified.
CertifyRemark	nvarchar(max)	Yes	The remark recorded by the User who certified the repairs (or no repairs made) to the Device or Trailer.
DateTime	datetime2(7)	No	The date and time the log was created.
DeviceId	nvarchar(50)	Yes	The Id of the Device (in the Devices table) associated with the subject DVIRLog.

DriverId	nvarchar(50)	Yes	The Id of the User (in the Users table) who created the log.
DriverRemark	nvarchar(max)	Yes	The remark recorded by the driver for this log.
IsSafeToOperate	bit	Yes	Indicates whether the Device or Trailer was certified as safe to operate.
LocationLatitude	float	Yes	The latitude of the location of the log.
LocationLongitude	float	Yes	The longitude of the location of the log.
LogType	nvarchar(50)	Yes	The DVIRLogType of the log.
RepairDate	datetime2(7)	Yes	The date and time the Device or Trailer was repaired.
RepairedByUserId	nvarchar(50)	Yes	The Id of the User (in the Users table) who repaired the Device or Trailer.
TrailerId	nvarchar(50)	Yes	The Id of the Trailer associated with the log.
TrailerName	nvarchar(255)	Yes	The Name of the Trailer associated with the log.
Version	bigint	No	The version of the entity.
RecordCreationTimeUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating when the subject record was added to the adapter database.

ExceptionEvents

The *ExceptionEvents* table contains data corresponding to MyGeotab ExceptionEvent objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
ActiveFrom	datetime2(7)	Yes	The start date and time of the ExceptionEvent; at or after this date and time.
ActiveTo	datetime2(7)	Yes	The end date and time of the ExceptionEvent; at or before this date and time.

DeviceId	nvarchar(50)	Yes	The Id of the Device (in the Devices table) associated with the subject ExceptionEvent.
Distance	real	Yes	The distance (in KMs) travelled since the start of the ExceptionEvent.
DriverId	nvarchar(50)	Yes	The Id of the Driver (corresponding to the Id in the Users table) associated with the subject ExceptionEvent.
DurationTicks	bigint	Yes	The duration of the ExceptionEvent as measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
LastModifiedDateTime	datetime2(7)	No	The last time this ExceptionEvent was updated (in the MyGeotab database).
RuleId	nvarchar(50)	Yes	The Id of the Rule (corresponding to the Id in the Rules table) that was broken to trigger the subject ExceptionEvent.
State	int	No	The ExceptionEventState of the subject ExceptionEvent. **0 = Valid. 1 = Invalid. 2 = Dismissed.**
Version	bigint	Yes	The version of the entity.
RecordLastChangedUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating when the subject record was added to the adapter database.

FailedDVIRDefectUpdates

The *FailedDVIRDefectUpdates* table is associated with the *DVIRDefectUpdates* table. If a row is inserted into the *DVIRDefectUpdates* table and it does not pass data validation checks, or if an exception occurs when the adapter attempts to execute the command, a copy of the original row in the *DVIRDefectUpdates* table will be added to the *FailedDVIRDefectUpdates* table along with the related error message which will appear in the *FailureMessage* column. For more detail, refer to the DVIRLog Manipulator section of this guide. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
DVIRDefectUpdateId	bigint	No	The value of the *id* field for the original row in the *DVIRDefectUpdates* table that resulted in the failure.

DVIRLogId	nvarchar(50)	No	The Id of the DVIRLog with which the subject DVIRDefect is associated.
DVIRDefectId	nvarchar(50)	No	The Id of the DVIRDefect that was to be updated in the Geotab system.
RepairDateTime	datetime2(7)	Yes	The date and time the DVIRDefect was repaired.
RepairStatus	nvarchar(50)	Yes	The RepairStatusType of this DVIRDefect.
RepairUserId	nvarchar(50)	Yes	The Id of the User who repaired this DVIRDefect.
Remark	nvarchar(max)	Yes	The remark text.
RemarkDateTime	datetime2(7)	Yes	The date and time the remark was created.
RemarkUserId	nvarchar(50)	Yes	The Id of the User who created the remark.
FailureMessage	nvarchar(max)	Yes	The reason why this record failed to result in an update of the subject DVIRDefect in the MyGeotab database.
RecordCreationTimeUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating when the subject record was added to the adapter database.

FaultData

The *FaultData* table contains data corresponding to MyGeotab FaultData objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
AmberWarningLamp	bit	Yes	Indicates whether the amber warning lamp is on.
ClassCode	nvarchar(50)	Yes	The DtcClass code of the fault.
ControllerId	nvarchar(100)	No	The Id of the Controller related to the fault code; if applicable.
ControllerName	nvarchar(255)	Yes	The Name of the Controller related to the fault code; if applicable.
Count	int	No	The number of times the fault occurred.

DateTime	datetime2(7)	Yes	The date and time at which the event occurred.
DeviceId	nvarchar(50)	No	The Id of the Device (in the Devices table) associated with the subject FaultData.
DiagnosticId	nvarchar(100)	No	The Id of the Diagnostic (in the Diagnostics table) associated with the subject FaultData.
DismissDateTime	datetime2(7)	Yes	The date and time that the fault was dismissed.
DismissUserId	nvarchar(50)	Yes	The Id of the User (in the Users table) associated with the subject FaultData entity.
FailureModeCode	int	Yes	The Failure Mode Identifier (FMI) associated with the FailureMode.
FailureModeId	nvarchar(50)	No	The Id of the FailureMode associated with the subject FaultData entity.
FailureModeName	nvarchar(255)	Yes	The Name of the FailureMode associated with the subject FaultData entity.
FaultLampState	nvarchar(50)	Yes	The FaultLampState of a J1939 vehicle.
FaultState	nvarchar(50)	Yes	The FaultState code from the engine system of the specific device.
MalfunctionLamp	bit	Yes	Indicates whether the malfunction lamp is on.
ProtectWarningLamp	bit	Yes	Indicates whether the protect warning lamp is on.
RedStopLamp	bit	Yes	Indicates whether the red stop lamp is on.
Severity	nvarchar(50)	Yes	The DtcSeverity of the fault.
SourceAddress	int	Yes	The source address for enhanced faults.
RecordCreationTimeUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating when the subject record was added to the adapter database.

Groups

The *Groups* table contains data corresponding to MyGeotab Group objects. ***SQL Server and PostgreSQL only**. Return to List of Tables.

Field Name	Data Type	Nullable	Description

id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
Children	nvarchar(max)	Yes	The GeotabIds of any direct children of the subject Group. Presented in the form of a JSON array (e.g. [{"id":"b3D88"},{"id":"b5F70"},{"id":"b2718"}]).
Color	nvarchar(50)	Yes	The color used to render assets in the subject Group in the MyGeotab user interface. Presented in JSON format (e.g. {"a":255,"b":0,"g":0,"r":0}).
Comments	nvarchar(1024)	Yes	A free text field where any user information can be stored and referenced for this entity.
Name	nvarchar(255)	Yes	The name of this entity that uniquely identifies it and is used when displaying this entity.
Reference	nvarchar(255)	Yes	The string reference to add to the database entry for this group.
EntityStatus	int	No	Indicates whether the subject corresponding object is active or deleted in the MyGeotab database. **1 = Active. 0 = Deleted.**
RecordLastChangedUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating the last time that the subject record was updated in the adapter database.

LogRecords

The *LogRecords* table contains data corresponding to MyGeotab LogRecord objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
DateTime	datetime2(7)	No	The date and time the log was recorded.
DeviceId	nvarchar(50)	No	The Id of the Device (in the Devices table) associated with the subject LogRecord.
Latitude	float	No	The latitude of the log record.

Longitude	float	No	The longitude of the log record.
Speed	real	No	The logged speed or an invalid speed (in km/h).
RecordCreationTimeUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating when the subject record was added to the adapter database.

MyGeotabVersionInfo

The *MyGeotabVersionInfo* table contains software version information (obtained from MyGeotab API VersionInformation) about the MyGeotab server and database that the subject adapter database is associated with via the MyGeotab API Adapter. Return to List of Tables.

Field Name	Data Type	Nullable	Description
DatabaseName	nvarchar(58)	No	The name of the MyGeotab database.
Server	nvarchar(50)	No	The server on which the MyGeotab database resides.
DatabaseVersion	nvarchar(50)	No	The current version of the MyGeotab databases on the server.
ApplicationBuild	nvarchar(50)	No	The MyGeotab application build.
ApplicationBranch	nvarchar(50)	No	The MyGeotab application branch.
ApplicationCommit	nvarchar(50)	No	The MyGeotab application commit.
GoTalkVersion	nvarchar(50)	No	The Text to Speech firmware version provided by the server.
RecordCreationTimeUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating when the subject record was added to the adapter database.

OServiceTracking

The *OServiceTracking* table is a system table used by the MyGeotab API Adapter. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
ServiceId	nvarchar(50)	No	An identifier for the subject service.

AdapterVersion	nvarchar(50)	Yes	The version of the MyGeotab API Adapter that the subject service is a part of.
AdapterMachineName	nvarchar(100)	Yes	The name of the machine/server that is hosting the MyGeotab API Adapter instance that the subject service is a part of.
EntitiesLastProcessedUtc	datetime2(7)	Yes	A timestamp, in Coordinated Universal Time (UTC), indicating the last time that the subject service processed any entities (data).
LastProcessedFeedVersion	bigint	Yes	If applicable and the subject service uses the GetFeed method, the *ToVersion* of the last batch of records retrieved by the subject service.
RecordLastChangedUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating the last time that the subject record was updated in the adapter database.

Rules

The *Rules* table contains data corresponding to MyGeotab Rule objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
ActiveFrom	datetime2(7)	Yes	Start date and time of the Rule's notification activity period.
ActiveTo	datetime2(7)	Yes	End date and time of the Rule's notification activity period.
BaseType	nvarchar(50)	Yes	The ExceptionRuleBaseType of the Rule.
Comment	nvarchar(max)	Yes	Free text field where any user information can be stored and referenced for this entity.
Groups	nvarchar(max)	Yes	The list of Group(s) the subject entity belongs to. Presented in the form of a JSON array (e.g. [{"id":"b1769"},{"id":"b2858"}]). Group Ids in this list correspond with **GeotabId** values in the Groups table. ***SQL Server and PostgreSQL only**.

Name	nvarchar(255)	Yes	The name of the rule entity that uniquely identifies it and is used when displaying this entity.
Version	bigint	No	The version of the entity.
EntityStatus	int	No	Indicates whether the subject corresponding object is active or deleted in the MyGeotab database. **1 = Active. 0 = Deleted.**
RecordLastChangedUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating the last time that the subject record was updated in the adapter database.

StatusData

The *StatusData* table contains data corresponding to MyGeotab StatusData objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
Data	float	Yes	The recorded value of the diagnostic parameter.
DateTime	datetime2(7)	Yes	The date and time of the logged event.
DeviceId	nvarchar(50)	No	The Id of the Device (in the Devices table) associated with the subject StatusData.
DiagnosticId	nvarchar(100)	No	The Id of the Diagnostic (in the Diagnostics table) associated with the subject StatusData.
RecordCreationTimeUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating when the subject record was added to the adapter database.

Trips

The *Trips* table contains data corresponding to MyGeotab Trip objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.

GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
AfterHoursDistance	real	Yes	The distance the vehicle was driven after work hours (in km).
AfterHoursDrivingDurationTicks	bigint	Yes	The duration the vehicle was driven after work hours as measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
AfterHoursEnd	bit	Yes	Whether the trip ends after hours.
AfterHoursStart	bit	Yes	Whether the trip starts after hours.
AfterHoursStopDurationTicks	bigint	Yes	The duration the vehicle was stopped after work hours as measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
AverageSpeed	real	Yes	Average speed in km/h. This only includes the average speed while driving.
DeviceId	nvarchar(50)	No	The Id of the Device (in the Devices table) associated with the subject Trip.
Distance	real	No	The distance the vehicle was driven during the trip (in km).
DriverId	nvarchar(50)	No	The Id of the Driver (corresponding to the Id in the Users table) associated with the subject Trip.
DrivingDurationTicks	bigint	No	The duration between the start and stop of the trip as measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
IdlingDurationTicks	bigint	Yes	Total end-of-trip idling (idling is defined as speed being 0 with ignition on). It is calculated from the beginning of the current trip to the beginning of the next trip. Measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.

MaximumSpeed	real	Yes	The maximum speed of the vehicle during this trip (in km/h).
NextTripStart	datetime2(7)	No	The start date and time of the next trip.
SpeedRange1	int	Yes	The number of incidents where the vehicle reached the first range of speeding triggers.
SpeedRange1DurationTicks	bigint	Yes	The duration where the vehicle drove in the first range of speeding triggers as measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
SpeedRange2	int	Yes	The number of incidents where the vehicle reached the second range of speeding triggers.
SpeedRange2DurationTicks	bigint	Yes	The duration where the vehicle drove in the second range of speeding triggers as measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
SpeedRange3	int	Yes	The number of incidents where the vehicle reached the third range of speeding triggers.
SpeedRange3DurationTicks	bigint	Yes	The duration where the vehicle drove in the third range of speeding triggers as measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
Start	datetime2(7)	No	The date and time that the trip started.
Stop	datetime2(7)	No	The date and time the trip stopped.
StopDurationTicks	bigint	No	The duration the vehicle was stopped at the end of the trip. This also includes any idling done at the end of a trip. Measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
StopPointX	float	Yes	The longitude of the Coordinate at which the Trip stopped.
StopPointY	float	Yes	The latitude of the Coordinate at which the Trip stopped.

WorkDistance	real	Yes	The distance the vehicle was driven during work hours (in km).
WorkDrivingDurationTicks	bigint	Yes	The duration the vehicle was driven during work hours as measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
WorkStopDurationTicks	bigint	Yes	The duration the vehicle was stopped during work hours as measured in ticks. A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second.
RecordCreationTimeUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating when the subject record was added to the adapter database.

Users

The *Users* table contains data corresponding to MyGeotab User objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
ActiveFrom	datetime2(7)	No	The date the user is active from.
ActiveTo	datetime2(7)	No	The date the user is active to.
CompanyGroups	nvarchar(max)	Yes	The list of Group(s) the subject entity belongs to. Presented in the form of a JSON array (e.g. [{"id":"b834"},{"id":"b9B1"}]). Group Ids in this list correspond with **GeotabId** values in the Groups table. ***SQL Server and PostgreSQL only**.
EmployeeNo	nvarchar(50)	Yes	The employee number or external identifier.
FirstName	nvarchar(255)	Yes	The first name of the user.

HosRuleSet	nvarchar(255)	Yes	The HosRuleSet the user follows. **Default: None.**
IsDriver	bit	No	Indicates whether the user is classified as a driver.
LastAccessDate	datetime2(7)	Yes	A timestamp, in Coordinated Universal Time (UTC), indicating the last time that the subject user accessed the MyGeotab system.
LastName	nvarchar(255)	Yes	The last name of the user.
Name	nvarchar(255)	No	The user's email address / login name.
EntityStatus	int	No	Indicates whether the subject corresponding object is active or deleted in the MyGeotab database. **1 = Active. 0 = Deleted.**
RecordLastChangedUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating the last time that the subject record was updated in the adapter database.

ZoneTypes

The *ZoneTypes* table contains data corresponding to MyGeotab ZoneType objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(100)	No	The unique identifier for the specific Entity object in the Geotab system.
Comment	nvarchar(255)	Yes	A free text field where any user information can be stored and referenced for this entity.
Name	nvarchar(255)	No	The name of this entity that uniquely identifies it and is used when displaying this entity.
EntityStatus	int	No	Indicates whether the subject corresponding object is active or deleted in the MyGeotab database. **1 = Active. 0 = Deleted.**
RecordLastChangedUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating the last time that the subject record was updated in the adapter database.

Zones

The *Zones* table contains data corresponding to MyGeotab Zone objects. Return to List of Tables.

Field Name	Data Type	Nullable	Description
id	bigint	No	The unique identifier for the record in the adapter database table. Entirely unrelated to the Geotab system.
GeotabId	nvarchar(50)	No	The unique identifier for the specific Entity object in the Geotab system.
ActiveFrom	datetime2(7)	Yes	The date the zone is active from.
ActiveTo	datetime2(7)	Yes	The date the zone is active to.
CentroidLatitude	float	Yes	The latitude of the geographic centre of the zone.
CentroidLongitude	float	Yes	The longitude of the geographic centre of the zone.
Comment	nvarchar(500)	Yes	A free text field where any user information can be stored and referenced for this entity.
Displayed	bit	Yes	A value indicating whether this zone must be displayed when viewing a map or it should be hidden.
ExternalReference	nvarchar(255)	Yes	Any type of external reference to be attached to the zone. May be used to link zones with corresponding entities in other systems.
Groups	nvarchar(max)	Yes	The list of Group(s) the subject entity belongs to. Presented in the form of a JSON array (e.g. [{"id":"b1266"},{"id":"b2998"}]). Group Ids in this list correspond with **GeotabId** values in the Groups table. ***SQL Server and PostgreSQL only**.
MustIdentifyStops	bit	Yes	Indicates whether this zone name must be shown when devices stop in this zone. If *true*, a "zone stop rule" (Rule with BaseType: ZoneStop) will automatically be created for this zone. This is to facilitate reporting on zone stops. The rule is not visible via the MyGeotab UI.
Name	nvarchar(255)	No	The name of this entity that uniquely identifies it and is used when displaying this entity.

Points	nvarchar(max)	Yes	The list of points (see Coordinate) that make up the zone. A zone should be closed (i.e. the first point should consist of the same set of coordinates as the last point). Presented in the form of a JSON array (e.g. [{"X":-79.68085479736328,"Y":43.517887115478516},{"X":-79.6830825805664,"Y":43.51841354370117},{"X":-79.6854019165039,"Y":43.5172004699707},{"X":-79.68214416503906,"Y":43.51369857788086},{"X":-79.6794204711914,"Y":43.51435470581055},{"X":-79.68085479736328,"Y":43.517887115478516}]).
ZoneTypeIds	nvarchar(max)	No	The Id(s) of the ZoneType(s) that this zone belongs to. Presented in the form of a JSON array (e.g. [{"Id":"ZoneTypeOfficeId"},{"Id":"ZoneTypeCustomerId"}]). Ids correspond to GeotabId values in the ZoneTypes table.
Version	bigint	Yes	The version of the entity.
EntityStatus	int	No	Indicates whether the subject corresponding object is active or deleted in the MyGeotab database. **1 = Active. 0 = Deleted.**
RecordLastChangedUtc	datetime2(7)	No	A timestamp, in Coordinated Universal Time (UTC), indicating the last time that the subject record was updated in the adapter database.

Configuration Files

Two files, explained below, are used to configure the MyGeotab API Adapter - *appsettings.json* and *nlog.config*.

Service Interdependencies

Individual MyGeotab API Adapter services can be enabled or disabled using the ***Enable<EntityType>Cache*** settings for reference data tables and the ***Enable<EntityType>Feed*** settings for feed data tables. This provides the flexibility to download only the desired Geotab data, thereby affording potential cost savings in terms of CPU, memory and storage consumption. However, there are certain logical dependencies that must be enforced in order to make the Geotab data usable. For example, the LogRecord object has a *Device* property that contains the Id of the Device object with which it is associated. The LogRecord on its own is useless without knowing which Device it came from. Therefore, someone wishing to enable the LogRecordProcessor should also enable the DeviceProcessor.

To ensure that appropriate combinations of services are enabled, certain service interdependencies are enforced as shown in the following table. For each service, the list of direct service dependencies is provided. When a given service starts-up, checks are performed to ensure that any services on which it depends are already running. If not, warning messages will be written to the log file and the target service will keep checking intermittently until the services on which it depends are running. For example, if the StatusDataProcessor has been enabled, but the

DeviceProcessor and DiagnosticProcessor have not been enabled, the log file will contain messages like the following:

```
2022-10-30 22:16:28.9799|INFO|"******** PAUSING SERVICE:
MyGeotabAPIAdapter.StatusDataProcessor (v2.0.0.0) because of the following:
> The prerequisite DeviceProcessor and DiagnosticProcessor have never been run.
> The prerequisite DeviceProcessor and DiagnosticProcessor are not currently
running.
Please ensure that all prerequisite processors are running. The
MyGeotabAPIAdapter.StatusDataProcessor (v2.0.0.0) will check again at 2022-10-31
2:16:38 AM (UTC) and will resume operation if all prerequisite processors are
running at that time.
```

✱ **NOTE**: There is a service orchestration process that occurs on application startup. Because all of the services start at the same time and some take longer than others to initialize, it is normal to see messages such as that in the above example when the MyGeotab API Adapter first starts. If services have been enabled, appropriately based on the table below, then these messages will stop appearing after the first few minutes of operation, once all of the services have come online.

In the above example, if the DeviceProcessor and DiagnosticProcessor have not been enabled, enable them both and restart the MyGeotab API Adapter to resolve the issue.

The following table lists the direct service dependencies for each service. Note that the dependencies might also have their own direct service dependencies.

Service	Direct Service Dependencies
BinaryDataProcessor	DeviceProcessor
ControllerProcessor	
DeviceProcessor	
DeviceStatusInfoProcessor	DeviceProcessor, UserProcessor
DiagnosticProcessor	
DriverChangeProcessor	DeviceProcessor, UserProcessor
DutyStatusAvailabilityProcessor	UserProcessor
DVIRLogManipulator	DVIRLogProcessor
DVIRLogProcessor	DeviceProcessor, UserProcessor
ExceptionEventProcessor	DeviceProcessor, RuleProcessor, UserProcessor

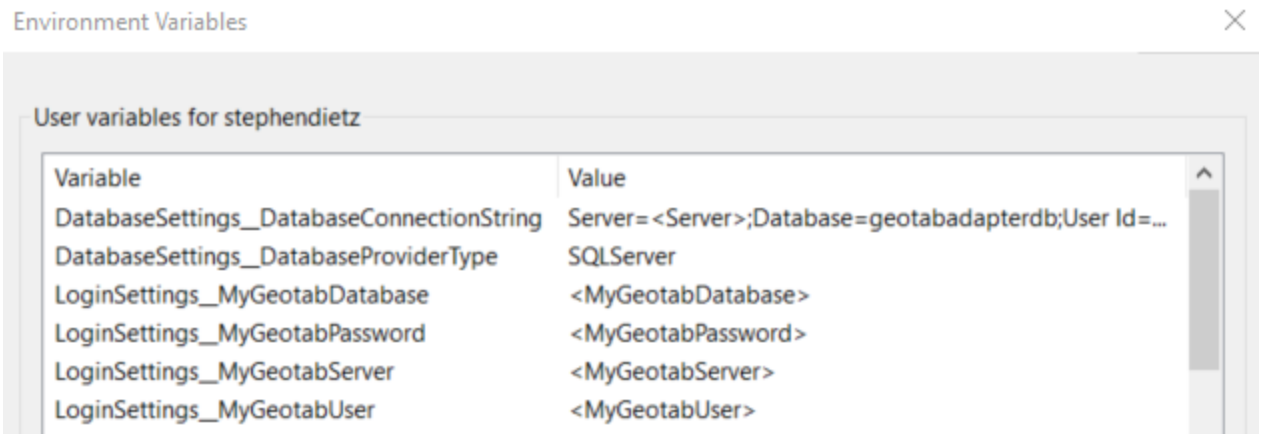
FailureModeProcessor	
FaultDataProcessor	ControllerProcessor, DeviceProcessor, DiagnosticProcessor, FailureModeProcessor
GroupProcessor	
LogRecordProcessor	DeviceProcessor
RuleProcessor	
StatusDataProcessor	DeviceProcessor, DiagnosticProcessor
TripProcessor	DeviceProcessor, UserProcessor
UnitOfMeasureProcessor	
UserProcessor	
ZoneProcessor	
ZoneTypeProcessor	

appsettings.json

Aside from log-related items, all configuration settings governing operation of the MyGeotab API Adapter are found in the *appsettings.json* file, which is located in the same directory as the executable (i.e. *MyGeotabAPIAdapter.exe*). Individual settings are organized into sections for readability. The following tables provide information about the settings contained within each of these sections.

Environment Variables for DatabaseSettings and LoginSettings

In some cases, it may be necessary to store database connection strings and MyGeotab login credentials in environment variables rather than in the appsettings.json file itself. To do so, simply leave the appsettings.json settings as-is (i.e. do not change or delete the subject settings) and create the corresponding environment variables as in the following (Windows) example (note the double underscores "__"):



OverrideSettings

The *OverrideSettings* section contains settings used to override certain aspects of application logic.

Setting	Description
DisableMachineNameValidation	Indicates whether machine name validation should be disabled. In most cases, the value should be false. It can be set to true **only** in cases **where** the application is deployed to a hosted environment in which **machine names are not guaranteed to be static**. **WARNING! Extreme caution must be used when setting this value to true! Improper deployment could lead to application instability and data integrity issues!**

DatabaseSettings

The *DatabaseSettings* section contains settings used to connect to the adapter database that is paired with the MyGeotab API Adapter.

Setting	Description
UseDataModel2	Must be set to **false** when using the original data model covered in this guide. ✱ **NOTE:** There is only a single version of the MyGeotab API Adapter application. It contains all of the original services and database scripts for the original data model. Database scripts and associated application services have also been added to support the second version of the data model. Essentially, the solution is evolving, but will support the original data model until the evolution to the new data model has been completed (and for a TBD period of time thereafter). Indicates whether the adapter database with which the application is paired is using version 2 of the data model. Services will be configured to use the appropriate data model based on the value provided here.

	! WARNING: The MyGeotab API Adapter will not work if the value provided for this setting is incorrect. • Must be set to **false** when using the **original** database scripts (see the Database Setup section of this guide) • Must be set to **true** when using the **Data Model 2** database scripts detailed in the Database Setup section of the new guide (MyGeotab API Adapter DM2 — Solution and Implementation G...).
EnableLevel1DatabaseMaintenance	Only used if **UseDataModel2** is true.
Level1DatabaseMaintenanceIntervalMinutes	Only used if **UseDataModel2** is true.
EnableLevel2DatabaseMaintenance	Only used if **UseDataModel2** is true.
Level2DatabaseMaintenanceIntervalMinutes	Only used if **UseDataModel2** is true.
EnableLevel2DatabaseMaintenanceWindow	Only used if **UseDataModel2** is true.
Level2DatabaseMaintenanceWindowStartTimeUTC	Only used if **UseDataModel2** is true.
Level2DatabaseMaintenanceWindowMaxMinutes	Only used if **UseDataModel2** is true.
DatabaseProviderType	The database provider. Must be one of SQLServer, PostgreSQL or Oracle.
DatabaseConnectionString	The database connection string (e.g. Server=<Server>;Database=geotabadapterdb;User Id=geotabadapter_client;Password=<password>;MultipleActiveResultSets=True;TrustServerCertificate=True).

LoginSettings

The *LoginSettings* section is used to configure the credentials that the internal MyGeotab API object will use to authenticate to the MyGeotab database with which the current MyGeotab API Adapter instance is paired.

Setting	Description
MyGeotabServer	The MyGeotab server (e.g. my.geotab.com).
MyGeotabDatabase	The name of the MyGeotab database to authenticate against. **WARNING!** It is not possible to mix data from multiple MyGeotab databases within a single MyGeotab API Adapter database. Once data has been added

	to the adapter database, it is not possible to change the **MyGeotabDatabase** setting.
MyGeotabUser	The username to be used for authentication.
MyGeotabPassword	The password associated with the MyGeotab user.

AppSettings - GeneralSettings

The *GeneralSettings* section under *AppSettings* includes settings that do not fall into any of the other categories.

Setting	Description
TimeoutSecondsForDatabaseTasks	The maximum number of seconds allowed for an individual adapter database operation (select, insert, update, delete) to complete. If a database operation does not complete within this amount of time, it will be assumed that there is a loss of connectivity, the existing operation will be rolled-back and the MyGeotab API Adapter will resume normal operation after establishing that there is connectivity to the adapter database (e.g. 30). **Minimum: 10**. **Maximum: 3600**.
TimeoutSecondsForMyGeotabTasks	The maximum number of seconds allowed for a MyGeotab API call to wait for a response. If a response is not received within this amount of time, it will be assumed that there is a loss of connectivity, the existing operation will be rolled-back and the MyGeotab API Adapter will resume normal operation after establishing that there is connectivity to the MyGeotab database via API (e.g. 30). **Minimum: 10**. **Maximum: 3600**.

AppSettings - Caches

The *Caches* section under *AppSettings* includes sections that govern the timing and frequency by which the MyGeotab API Adapter updates and refreshes caches of entities obtained from the MyGeotab database configured in the LoginSettings section.

✱ **NOTE**: In order to provide ultimate flexibility, each entity type has its own cache configuration section consisting of the following four settings which include the subject entity type name:

- **Enable<EntityType>Cache:** Indicates whether the cache for the subject entity type should be enabled. Must be set to either true or false.
- **<EntityType>CacheIntervalDailyReferenceStartTimeUTC:** An ISO 8601 date and time string used to specify a time of day to serve as the basis upon which cache update and refresh times are calculated using the associated interval settings. **Only the time portion of the string is used**; the date entered is irrelevant. To avoid time-zone related issues, Coordinated Universal Time (UTC) should be used (e.g. 2020-06-23T06:00:00Z).
- **<EntityType>CacheUpdateIntervalMinutes:** The frequency, in minutes, by which the subject cache should be "updated" - capturing new and changed objects (e.g. 360). **Minimum: 1. Maximum: 10080 (1 week).**
- **<EntityType>CacheRefreshIntervalMinutes:** The frequency, in minutes, by which the subject cache should be "refreshed" - dumped and repopulated to make identification of deleted entities possible,

since deletes do not propagate from MyGeotab through the MyGeotab API data feeds (e.g. 1440). **Minimum: 60 (1 hour). Maximum: 10080 (1 week).**

Cache Configuration Example

As an example of cache configuration, in order to refresh the User cache at 2:00am EDT every day and update it every six hours (i.e. at 8:00am, 2:00pm and 8:00pm), the settings in the User cache section would be configured as follows:

- EnableUserCache: true
- UserCacheIntervalDailyReferenceStartTimeUTC: 2020-06-23T06:00:00Z
- UserCacheUpdateIntervalMinutes: 360
- UserCacheRefreshIntervalMinutes: 1440

! IMPORTANT: Cache refreshes and updates may occur slightly after the configured times due to the amount of time required to process cache and feed data during each iteration.

AppSettings - Caches - Controller

The *Controller* section under *AppSettings > Caches* includes settings that govern the MyGeotab API Adapter cache of Controller objects obtained from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableControllerCache	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
ControllerCacheIntervalDailyReferenceStartTimeUTC	The UTC time of day to serve as the basis upon which cache update and refresh times are calculated using the associated interval settings (e.g. 2020-06-23T06:00:00Z).
ControllerCacheUpdateIntervalMinutes	The frequency, in minutes, by which the subject cache should be "updated" - capturing new and changed objects (e.g. 360). **Minimum: 1. Maximum: 10080 (1 week).**
ControllerCacheRefreshIntervalMinutes	The frequency, in minutes, by which the subject cache should be "refreshed" - dumped and repopulated to make identification of deleted entities possible, since deletes do not propagate from MyGeotab through the MyGeotab API data feeds (e.g. 1440). **Minimum: 60 (1 hour). Maximum: 10080 (1 week).**

AppSettings - Caches - Device

The *Device* section under *AppSettings > Caches* includes settings that govern the MyGeotab API Adapter cache of Device objects obtained from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableDeviceCache	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.

DeviceCacheIntervalDailyReferenceStartTimeUTC	The UTC time of day to serve as the basis upon which cache update and refresh times are calculated using the associated interval settings (e.g. 2020-06-23T06:00:00Z).
DeviceCacheUpdateIntervalMinutes	The frequency, in minutes, by which the subject cache should be "updated" - capturing new and changed objects (e.g. 360). **Minimum: 1. Maximum: 10080 (1 week).**
DeviceCacheRefreshIntervalMinutes	The frequency, in minutes, by which the subject cache should be "refreshed" - dumped and repopulated to make identification of deleted entities possible, since deletes do not propagate from MyGeotab through the MyGeotab API data feeds (e.g. 1440). **Minimum: 60 (1 hour). Maximum: 10080 (1 week).**

AppSettings - Caches - Diagnostic

The *Diagnostic* section under *AppSettings > Caches* includes settings that govern the MyGeotab API Adapter cache of Diagnostic objects obtained from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableDiagnosticCache	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
DiagnosticCacheIntervalDailyReferenceStartTimeUTC	The UTC time of day to serve as the basis upon which cache update and refresh times are calculated using the associated interval settings (e.g. 2020-06-23T06:00:00Z).
DiagnosticCacheUpdateIntervalMinutes	The frequency, in minutes, by which the subject cache should be "updated" - capturing new and changed objects (e.g. 360). **Minimum: 1. Maximum: 10080 (1 week).**
DiagnosticCacheRefreshIntervalMinutes	The frequency, in minutes, by which the subject cache should be "refreshed" - dumped and repopulated to make identification of deleted entities possible, since deletes do not propagate from MyGeotab through the MyGeotab API data feeds (e.g. 1440). **Minimum: 60 (1 hour). Maximum: 10080 (1 week).**

AppSettings - Caches - DVIRDefect

The *DVIRDefect* section under *AppSettings > Caches* includes settings that govern the MyGeotab API Adapter cache of DVIRDefect objects obtained from the MyGeotab database configured in the LoginSettings section.

Setting	Description
DVIRDefectListCacheRefreshIntervalMinutes	The frequency, in minutes, by which the subject cache should be "refreshed" - dumped and repopulated to make identification of deleted entities possible, since deletes do not propagate from MyGeotab through the

	MyGeotab API data feeds (e.g. 1440). **Minimum: 60 (1 hour). Maximum: 10080 (1 week).**

AppSettings - Caches - FailureMode

The *FailureMode* section under *AppSettings > Caches* includes settings that govern the MyGeotab API Adapter cache of FailureMode objects obtained from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableFailureModeCache	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
FailureModeCacheIntervalDailyReferenceStartTimeUTC	The UTC time of day to serve as the basis upon which cache update and refresh times are calculated using the associated interval settings (e.g. 2020-06-23T06:00:00Z).
FailureModeCacheUpdateIntervalMinutes	The frequency, in minutes, by which the subject cache should be "updated" - capturing new and changed objects (e.g. 360). **Minimum: 1. Maximum: 10080 (1 week).**
FailureModeCacheRefreshIntervalMinutes	The frequency, in minutes, by which the subject cache should be "refreshed" - dumped and repopulated to make identification of deleted entities possible, since deletes do not propagate from MyGeotab through the MyGeotab API data feeds (e.g. 1440). **Minimum: 60 (1 hour). Maximum: 10080 (1 week).**

AppSettings - Caches - Group

The *Group* section under *AppSettings > Caches* includes settings that govern the MyGeotab API Adapter cache of Group objects obtained from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableGroupCache	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
GroupCacheIntervalDailyReferenceStartTimeUTC	The UTC time of day to serve as the basis upon which cache update and refresh times are calculated using the associated interval settings (e.g. 2020-06-23T06:00:00Z).
GroupCacheUpdateIntervalMinutes	The frequency, in minutes, by which the subject cache should be "updated" - capturing new and changed objects (e.g. 360). **Minimum: 1. Maximum: 10080 (1 week).**
GroupCacheRefreshIntervalMinutes	The frequency, in minutes, by which the subject cache should be "refreshed" - dumped and repopulated to make identification of deleted entities possible, since deletes do not propagate from MyGeotab through the

	MyGeotab API data feeds (e.g. 1440). **Minimum: 60 (1 hour). Maximum: 10080 (1 week).**

AppSettings - Caches - Rule

The *Rule* section under *AppSettings > Caches* includes settings that govern the MyGeotab API Adapter cache of Rule objects obtained from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableRuleCache	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
RuleCacheIntervalDailyReferenceStartTimeUTC	The UTC time of day to serve as the basis upon which cache update and refresh times are calculated using the associated interval settings (e.g. 2020-06-23T06:00:00Z).
RuleCacheUpdateIntervalMinutes	The frequency, in minutes, by which the subject cache should be "updated" - capturing new and changed objects (e.g. 360). **Minimum: 1. Maximum: 10080 (1 week).**
RuleCacheRefreshIntervalMinutes	The frequency, in minutes, by which the subject cache should be "refreshed" - dumped and repopulated to make identification of deleted entities possible, since deletes do not propagate from MyGeotab through the MyGeotab API data feeds (e.g. 1440). **Minimum: 60 (1 hour). Maximum: 10080 (1 week).**

AppSettings - Caches - UnitOfMeasure

The *UnitOfMeasure* section under *AppSettings > Caches* includes settings that govern the MyGeotab API Adapter cache of UnitOfMeasure objects obtained from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableUnitOfMeasureCache	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
UnitOfMeasureCacheIntervalDailyReferenceStartTimeUTC	The UTC time of day to serve as the basis upon which cache update and refresh times are calculated using the associated interval settings (e.g. 2020-06-23T06:00:00Z).
UnitOfMeasureCacheUpdateIntervalMinutes	The frequency, in minutes, by which the subject cache should be "updated" - capturing new and changed objects (e.g. 360). **Minimum: 1. Maximum: 10080 (1 week).**
UnitOfMeasureCacheRefreshIntervalMinutes	The frequency, in minutes, by which the subject cache should be "refreshed" - dumped and repopulated to make identification of deleted entities possible, since deletes do not propagate from MyGeotab through the

	MyGeotab API data feeds (e.g. 1440). **Minimum: 60 (1 hour). Maximum: 10080 (1 week).**

AppSettings - Caches - User

The *User* section under *AppSettings > Caches* includes settings that govern the MyGeotab API Adapter cache of User objects obtained from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableUserCache	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
UserCacheIntervalDailyReferenceStartTimeUTC	The UTC time of day to serve as the basis upon which cache update and refresh times are calculated using the associated interval settings (e.g. 2020-06-23T06:00:00Z).
UserCacheUpdateIntervalMinutes	The frequency, in minutes, by which the subject cache should be "updated" - capturing new and changed objects (e.g. 360). **Minimum: 1. Maximum: 10080 (1 week).**
UserCacheRefreshIntervalMinutes	The frequency, in minutes, by which the subject cache should be "refreshed" - dumped and repopulated to make identification of deleted entities possible, since deletes do not propagate from MyGeotab through the MyGeotab API data feeds (e.g. 1440). **Minimum: 60 (1 hour). Maximum: 10080 (1 week).**

AppSettings - Caches - Zone

The *Zone* section under *AppSettings > Caches* includes settings that govern the MyGeotab API Adapter cache of Zone objects obtained from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableZoneCache	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
ZoneCacheIntervalDailyReferenceStartTimeUTC	The UTC time of day to serve as the basis upon which cache update and refresh times are calculated using the associated interval settings (e.g. 2020-06-23T06:00:00Z).
ZoneCacheUpdateIntervalMinutes	The frequency, in minutes, by which the subject cache should be "updated" - capturing new and changed objects (e.g. 360). **Minimum: 1. Maximum: 10080 (1 week).**
ZoneCacheRefreshIntervalMinutes	The frequency, in minutes, by which the subject cache should be "refreshed" - dumped and repopulated to make identification of deleted entities possible, since deletes do not propagate from MyGeotab through the

	MyGeotab API data feeds (e.g. 1440). **Minimum: 60 (1 hour). Maximum: 10080 (1 week).**

AppSettings - Caches - ZoneType

The *ZoneType* section under *AppSettings > Caches* includes settings that govern the MyGeotab API Adapter cache of ZoneType objects obtained from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableZoneTypeCache	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
ZoneTypeCacheIntervalDailyReferenceStartTimeUTC	The UTC time of day to serve as the basis upon which cache update and refresh times are calculated using the associated interval settings (e.g. 2020-06-23T06:00:00Z).
ZoneTypeCacheUpdateIntervalMinutes	The frequency, in minutes, by which the subject cache should be "updated" - capturing new and changed objects (e.g. 360). **Minimum: 1. Maximum: 10080 (1 week).**
ZoneTypeCacheRefreshIntervalMinutes	The frequency, in minutes, by which the subject cache should be "refreshed" - dumped and repopulated to make identification of deleted entities possible, since deletes do not propagate from MyGeotab through the MyGeotab API data feeds (e.g. 1440). **Minimum: 60 (1 hour). Maximum: 10080 (1 week).**

AppSettings - GeneralFeedSettings

The *GeneralFeedSettings* section under *AppSettings* contains settings that apply to all data feeds.

Setting	Description
FeedStartOption	Alternate ways that polling via data feeds may be initiated. Must be set to one of the following values: • CurrentTime: Data feeds will be started from the current point in time. • SpecificTime: Data feeds will be started at the specific point in time (in the past) specified by the **FeedStartSpecificTimeUTC** parameter. • FeedVersion: Each data feed will be started at the specific version captured in the **OServiceTracking** table in the adapter database. If the data feed for a specific object type has not yet been run, it will start at version zero, effectively allowing the feed to pull *all* of the respective data from the database. **CAUTION!** If any data has already been captured for a given feed, as determined by the existence of a corresponding entry in the **OServiceTracking** table in the adapter database, the **FeedStartOption** will automatically switch to FeedVersion - overriding any other value that may

	have been set. This mechanism is in place to avoid issues related to data duplication or gaps.
FeedStartSpecificTimeUTC	Only used if the **FeedStartOption** parameter is set to SpecificTime. The UTC time (formatted as yyyy-MM-ddTHH:mm:ssZ) at which to start all data feeds (e.g. 2020-06-23T06:00:00Z).
DevicesToTrack	A comma-separated list of device IDs that correspond to devices to be tracked. The **default** value of * indicates that data will be captured for all devices. Alternatively, if a comma-separated list of device IDs is provided (e.g. b1,b2), any feed data with relations to devices will be filtered such that only the data associated with the specified devices will be persisted to the adapter database. **WARNING!** In order to capture data related to **all devices**, the value of this setting must be *.
DiagnosticsToTrack	A comma-separated list of diagnostic IDs that correspond to diagnostics (StatusData or FaultData) to be tracked. The **default** value of * indicates that data will be captured for all diagnostics. Alternatively, if a comma-separated list of diagnostic IDs is provided (e.g. DiagnosticOdometerAdjustmentId,DiagnosticFuelLevelId,DiagnosticFuelUnitsId,DiagnosticCrankingVoltageId,DiagnosticInvalidGpsMessagesReceivedId), any feed data with relations to diagnostics will be filtered such that only the data associated with the specified diagnostics will be persisted to the adapter database. **WARNING!** In order to capture data related to **all diagnostics**, the value of this setting must be *.
ExcludeDiagnosticsToTrack	Indicates whether the **DiagnosticsToTrack** should be *excluded*, effectively inverting functionality. If *false*, only the data associated with the specified diagnostics will be persisted to the adapter database. If *true*, all data *EXCEPT* for the data associated with the specified diagnostics will be persisted to the adapter database. Must be set to either true or false.
EnableMinimunIntervalSamplingForLogRecords	Indicates whether minimum interval sampling should be applied to the LogRecord feed. Only works if **EnableLogRecordFeed** is true. If set to true, a minimum interval defined by **MinimumIntervalSamplingIntervalSeconds** will be applied between LogRecords that are written to the LogRecords table. Must be set to either true or false. See the Minimum Interval Sampling section for more information.
EnableMinimunIntervalSamplingForStatusData	Indicates whether minimum interval sampling should be applied to the StatusData feed. Only works if **EnableStatusDataFeed** is true. If set to true, a minimum interval defined by **MinimumIntervalSamplingIntervalSeconds** will be applied between StatusData entities with a Diagnostic Id included in the **MinimumIntervalSamplingDiagnostics** list that are written to the StatusData table. *All* StatusData entities with a Diagnostic Id included in the

	DiagnosticsToTrack list but not in the **MinimumIntervalSamplingDiagnostics** list will be persisted to the StatusData table. Must be set to either true or false. See the Minimum Interval Sampling section for more information.
MinimumIntervalSamplingDiagnostics	A comma-separated list of diagnostic IDs for which minimum interval sampling will be applied to the StatusData feed. Only works if **EnableStatusDataFeed** is true. If set to true, a minimum interval defined by **MinimumIntervalSamplingIntervalSeconds** will be applied between StatusData entities with a Diagnostic Id included in this list that are written to the StatusData table. This list must be equal to the list provided in the **DiagnosticsToTrack** setting or a subset thereof. **ExcludeDiagnosticsToTrack** must be set to false. The placeholder wildcard value (*) cannot be used for the **DiagnosticsToTrack** setting nor this setting. See the Minimum Interval Sampling section for more information.
MinimumIntervalSamplingIntervalSeconds	The minimum duration, in seconds, that should be applied between the DateTime values of entities that are written to their respective table(s) in the adapter database for any entity type that has minimum interval sampling enabled. **Minimum: 1. Maximum: 3600 (1 hour).** See the Minimum Interval Sampling section for more information.

AppSettings - Feeds

The *Feeds* section under *AppSettings* includes sections that govern the use of data feeds for the various MyGeotab entity types that the MyGeotab API Adapter supports.

> ✱ **NOTE**: In order to provide ultimate flexibility, each supported entity type has its own feed configuration section consisting of the following settings which include the subject entity type name:
> - **Enable<EntityType>Feed:** Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
> - **<EntityType>FeedIntervalSeconds:** The minimum number of seconds to wait between GetFeed() calls for the subject entity type (e.g. 30). **Minimum: 2. Maximum: 604800 (1 week).**

AppSettings - Feeds - BinaryData

The *BinaryData* section under *AppSettings > Feeds* includes settings that govern the use of the BinaryData data feed pulled from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableBinaryDataFeed	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
BinaryDataFeedIntervalSeconds	The minimum number of seconds to wait between GetFeed() calls for the subject entity type (e.g. 30). **Minimum: 2. Maximum: 604800 (1 week).**

AppSettings - Feeds - ChargeEvent

The *ChargeEvent* section under *AppSettings > Feeds* includes settings that govern the use of the ChargeEvent data feed pulled from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableChargeEventFeed	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
ChargeEventFeedIntervalSeconds	The minimum number of seconds to wait between GetFeed() calls for the subject entity type (e.g. 30). **Minimum: 2. Maximum: 604800 (1 week).**

AppSettings - Feeds - DebugData

The *DebugData* section under *AppSettings > Feeds* includes settings that govern the use of the DebugData data feed pulled from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableDebugDataFeed	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
DebugDataFeedIntervalSeconds	The minimum number of seconds to wait between GetFeed() calls for the subject entity type (e.g. 30). **Minimum: 2. Maximum: 604800 (1 week).**

AppSettings - Feeds - DeviceStatusInfo

The *DeviceStatusInfo* section under *AppSettings > Feeds* includes settings that govern the use of the DeviceStatusInfo data feed pulled from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableDeviceStatusInfoFeed	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
DeviceStatusInfoFeedIntervalSeconds	The minimum number of seconds to wait between GetFeed() calls for the subject entity type (e.g. 30). **Minimum: 2. Maximum: 604800 (1 week).**

AppSettings - Feeds - DriverChange

The *DriverChange* section under *AppSettings > Feeds* includes settings that govern the use of the DriverChange data feed pulled from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableDriverChangeFeed	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.

DriverChangeFeedIntervalSeconds	The minimum number of seconds to wait between GetFeed() calls for the subject entity type (e.g. 30). **Minimum: 2. Maximum: 604800 (1 week).**

AppSettings - Feeds - DutyStatusAvailability

The *DutyStatusAvailability* section under *AppSettings > Feeds* includes settings that govern the use of the DutyStatusAvailability pseudo data feed pulled from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableDutyStatusAvailabilityFeed	Indicates whether the pseudo data feed for the subject entity type should be enabled. Must be set to either true or false.
DutyStatusAvailabilityFeedIntervalSeconds	The minimum number of seconds to wait after retrieving DutyStatusAvailability information for all Drivers before starting the retrieval process again (e.g. 300). **Minimum: 2. Maximum: 604800 (1 week).**
DutyStatusAvailabilityFeedLastAccessDateCutoffDays	Used to reduce the number of unnecessary Get calls when retrieving DutyStatusAvailability information for all Drivers. Data is not queried for Drivers who have not accessed the Geotab system for more than this many days in the past. This value should be set to approximately twice the longest possible cycle for a HOS ruleset (e.g. 30). **Minimum: 14. Maximum: 60.**

AppSettings - Feeds - DutyStatusLog

The *DutyStatusLog* section under *AppSettings > Feeds* includes settings that govern the use of the DutyStatusLog data feed pulled from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableDutyStatusLogFeed	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
DutyStatusLogFeedIntervalSeconds	The minimum number of seconds to wait between GetFeed() calls for the subject entity type (e.g. 30). **Minimum: 2. Maximum: 604800 (1 week).**

AppSettings - Feeds - DVIRLog

The *DVIRLog* section under *AppSettings > Feeds* includes settings that govern the use of the DVIRLog data feed pulled from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableDVIRLogFeed	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.

DVIRLogFeedIntervalSeconds	The minimum number of seconds to wait between GetFeed() calls for the subject entity type (e.g. 30). **Minimum: 2. Maximum: 604800 (1 week).**

AppSettings - Feeds - ExceptionEvent

The *ExceptionEvent* section under *AppSettings > Feeds* includes settings that govern the use of the ExceptionEvent data feed pulled from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableExceptionEventFeed	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
ExceptionEventFeedIntervalSeconds	The minimum number of seconds to wait between GetFeed() calls for the subject entity type (e.g. 30). **Minimum: 2. Maximum: 604800 (1 week).**
TrackZoneStops	Only used if **EnableExceptionEventFeed** is set to true. Indicates whether exceptions with an ExceptionRuleBaseType of *ZoneStop* are to be persisted to the adapter database. Must be set to either true or false.

AppSettings - Feeds - FaultData

The *FaultData* section under *AppSettings > Feeds* includes settings that govern the use of the FaultData data feed pulled from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableFaultDataFeed	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
FaultDataFeedIntervalSeconds	The minimum number of seconds to wait between GetFeed() calls for the subject entity type (e.g. 30). **Minimum: 2. Maximum: 604800 (1 week).**

AppSettings - Feeds - LogRecord

The *LogRecord* section under *AppSettings > Feeds* includes settings that govern the use of the LogRecord data feed pulled from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableLogRecordFeed	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
LogRecordFeedIntervalSeconds	The minimum number of seconds to wait between GetFeed() calls for the subject entity type (e.g. 30). **Minimum: 2. Maximum: 604800 (1 week).**

AppSettings - Feeds - StatusData

The *StatusData* section under *AppSettings > Feeds* includes settings that govern the use of the StatusData data feed pulled from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableStatusDataFeed	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
StatusDataFeedIntervalSeconds	The minimum number of seconds to wait between GetFeed() calls for the subject entity type (e.g. 30). **Minimum: 2. Maximum: 604800 (1 week).**

AppSettings - Feeds - Trip

The *Trip* section under *AppSettings > Feeds* includes settings that govern the use of the Trip data feed pulled from the MyGeotab database configured in the LoginSettings section.

Setting	Description
EnableTripFeed	Indicates whether the data feed for the subject entity type should be enabled. Must be set to either true or false.
TripFeedIntervalSeconds	The minimum number of seconds to wait between GetFeed() calls for the subject entity type (e.g. 30). **Minimum: 2. Maximum: 604800 (1 week).**

AppSettings - DataEnhancementServices

This section and all settings contained within are only used if **UseDataModel2** is true In the DatabaseSettings section.

AppSettings - Manipulators

The *Manipulators* section under *AppSettings* includes sections that govern services responsible for issuing data manipulation commands to the Geotab platform. Manipulators may be used in conjunction with feeds to facilitate bidirectional integration between the Geotab platform and external systems.

AppSettings - Manipulators - DVIRLog

The *DVIRLog* section under *AppSettings > Manipulators* includes settings that govern the use of the DVIRLogManipulator service which provides the capability to update DVIRLogs in the MyGeotab database configured in the LoginSettings section. See the DVIRLog Manipulator section for more information.

Setting	Description
EnableDVIRLogManipulator	Indicates whether the DVIRLogManipulator service should be enabled. Must be set to either true or false.
DVIRLogManipulatorIntervalSeconds	The minimum number of seconds to wait between starts of iterations of the DVIRLogManipulator processing logic (e.g. 30). **Minimum: 2. Maximum: 604800 (1 week).**

AppSettings - AddOns - VSS

The *VSS* section under *AppSettings > AddOns* includes settings related to the Vehicle Signal Specification (VSS) Add-On. Unless this capability is being used, ensure that **EnableVSSAddOn** is set to false. For more information

about the settings in this section, refer to the MyGeotab API Adapter - Vehicle Signal Specification (VSS) Add-On - Solution and Implementation Guide.

nlog.config

The MyGeotab API Adapter utilizes the NLog LoggerProvider for Microsoft.Extensions.Logging to capture information and write it to log files for debugging purposes. NLog configuration settings are found in the *nlog.config* file, which is located in the same directory as the executable (i.e. *MyGeotabAPIAdapter.exe*).

Below is the content of the nlog.config file that is included with the MyGeotab API Adapter. It is followed by instructions that indicate how the **highlighted** settings may be adjusted.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- XSD manual extracted from package NLog.Schema:
https://www.nuget.org/packages/NLog.Schema-->
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
xsi:schemaLocation="NLog NLog.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      autoReload="true"
      internalLogFile="LOG-MyGeotab_API_Adapter-internal.log"
      internalLogLevel="Error" >

  <!-- the targets to write to -->
  <targets>
    <!-- write logs to file -->
    <target xsi:type="File" name="target1" fileName="LOG-MyGeotab_API_Adapter.log"
maxArchiveFiles="100" archiveAboveSize="5120000" archiveEvery="Day">
      <layout xsi:type="CsvLayout" delimiter="Pipe" withHeader="true">
        <column name="Time" layout="${longdate}" />
        <column name="Level" layout="${level:upperCase=true}"/>
        <column name="Message" layout="${message}" />
        <column name="Exception" layout="${exception}"/>
        <column name="Logger" layout="${logger}" />
        <column name="All Event Properties" layout="${all-event-properties}" />
      </layout>
    </target>
    <target xsi:type="Console" name="target2"
            layout="${date}|${level:uppercase=true}|${message}
${exception}|${logger}|${all-event-properties}" />
  </targets>

  <!-- rules to map from logger name to target -->
```

```
  <rules>
    <logger name="*" minlevel="Info" writeTo="target1,target2" />
  </rules>
</nlog>
```

WARNING! Only the settings that are **highlighted** in the above content should be modified as described in the following table. Changing anything else could lead to unpredictable consequences

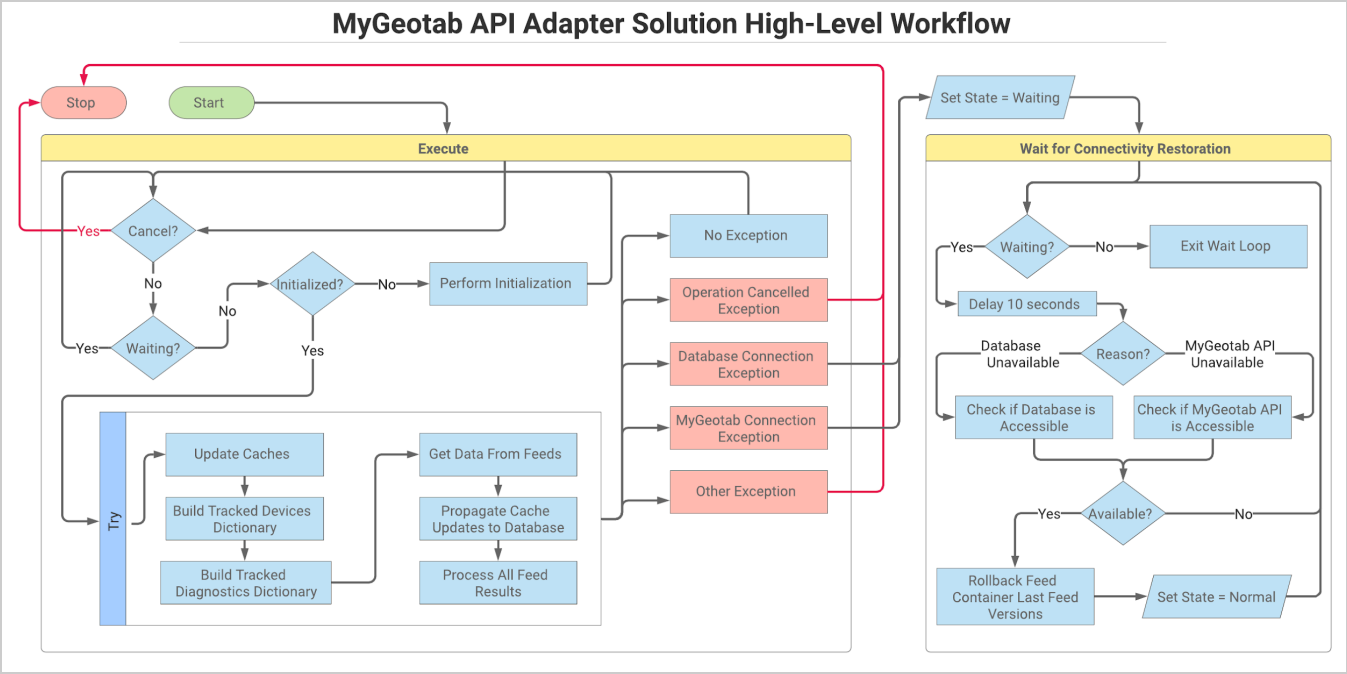
The following table lists the NLog settings, **highlighted** above, that may be adjusted as required.

Setting	Description
autoReload	Indicates whether the *nlog.config* file should be watched for changes and reloaded automatically when changed. Must be set to either true or false.
maxArchiveFiles	Maximum number of archive files that should be kept. If maxArchiveFiles is less or equal to 0, old files aren't deleted. **Default: 100.**
archiveAboveSize	Size in bytes above which log files will be automatically archived. **Default: 5120000 (5 MB).**
archiveEvery	Indicates whether to automatically archive log files every time the specified time passes. **Default: DAY.** Possible values: • Day: Archive daily. • Hour: Archive every hour. • Minute: Archive every minute. • Month: Archive every month. • None: Don't archive based on time. • Year: Archive every year. • Sunday: Archive every Sunday. • Monday: Archive every Monday. • Tuesday: Archive every Tuesday. • Wednesday: Archive every Wednesday. • Thursday: Archive every Thursday. • Friday: Archive every Friday. • Saturday: Archive every Saturday.
minlevel	Indicates the log level, which is the amount of detail to be written to log files. **Default: Debug.** Possible values: • Trace: Very detailed logs, which may include high-volume. information such as protocol payloads. This log level is typically only enabled during development.

	• Debug: Debugging information, less detailed than trace, typically not enabled in a production environment. • Info: Information messages, which are normally enabled in a production environment. • Warn: Warning messages, typically for non-critical issues, which can be recovered or which are temporary failures. • Error: Error messages - most of the time these are Exceptions. • Fatal: Very serious errors!

Application Logic

The high-level application logic of the MyGeotab API Adapter solution is represented in the following exhibit:



Solution Usage and Implementation

This section provides usage and deployment-related instructions for the MyGeotab API Adapter solution. Information related to solution architecture, logic and data models can be found in the Solution Information section of this guide.

Prerequisites

The *MyGeotab API Adapter* requires:

- .NET 9.0 SDK (https://dotnet.microsoft.com/download) or higher. Note: If simply deploying a published release, the deployment package is self-contained (including all dependencies), so **.NET version(s) of the host machine should not be a concern.**
- The following (included via NuGet packages):
  - Geotab.Checkmate.ObjectModel

    - Dapper
    - FastMember
    - Microsoft.Data.SqlClient
    - NLog.Extensions.Logging
    - Npgsql
    - Oracle.ManagedDataAccess.Core
    - Polly
- MyGeotab credentials with all "*View*" clearances enabled on any MyGeotab database with which the MyGeotab API Adapter is to be used. It is recommended that a Service Account be set-up for this purpose. See the Service Account Guidelines document for more details. Specific clearances required by the service account used by the MyGeotab API Adapter include:
    - List devices
    - List Users/Drivers
    - View Asset Inspection logs
    - View binary data
    - View device status information
    - View engine diagnostics
    - View engine failure modes
    - View engine measurement related features
    - View engine units of measurement
    - View exception rules
    - View exceptions
    - View groups
    - View trailers
    - View zones
- If the DVIRLog Manipulator is going to be used, the following clearances are also required:
    - Administer Asset Inspection logs
    - Mark Asset Inspection logs as certified
    - Mark Asset Inspection logs as repaired
    - Perform Asset Inspections
- If **PostgreSQL** is the chosen database provider, access to a PostgreSQL 16 (or greater) server on which the adapter database is deployed.
    - If the adapter and database will reside on separate servers, it may be necessary to ensure that appropriate security and networking steps are undertaken to ensure the ability of the adapter to interact with the database.
    - Although not a strict requirement, it is recommended to have access to a tool such as pgAdmin to view data that the adapter writes to the database.
- If **SQL Server** is the chosen database provider, access to a MS SQL Server instance on which the adapter database is deployed. While developed using SQL Server 2019 Developer (version 15.0.2000.5), given that the solution uses only simple tables and views, it is likely to work on other SQL Server versions without any issues.

    - If the adapter and database will reside on separate servers, it may be necessary to ensure that appropriate security and networking steps are undertaken to ensure the ability of the adapter to interact with the database.
    - Although not a strict requirement, it is recommended to have access to a tool such as SQL Server Management Studio to view data that the adapter writes to the database.
- If **Oracle** is the chosen database provider, access to an Oracle instance on which the adapter database is deployed. Note that Oracle Database XE (18c) was initially used for MyGeotab API Adapter development.
    - If the adapter and database will reside on separate servers, it may be necessary to ensure that appropriate security and networking steps are undertaken to ensure the ability of the adapter to interact with the database.
    - Although not a strict requirement, it is recommended to have access to a tool such as Oracle SQL Developer to view data that the adapter writes to the database.

Getting Started

The *MyGeotab API Adapter* solution can be obtained by cloning the Git repository. Once downloaded, open the solution — **MyGeotabAPIAdapter.sln** — using Microsoft Visual Studio or Visual Studio Code.

Database Setup

Out-of-the-box, the MyGeotab API Adapter supports SQL Server, PostgreSQL and Oracle for use as the database into which data retrieved via the MyGeotab API is written, as described in the Database section. Database setup procedures differ depending on the type of database and are outlined below.

> **WARNING!** Regardless of database type, it is possible for the database to grow very large very quickly, resulting in **potential disk space and performance issues**. For example, running the adapter against a MyGeotab database with a fleet of ~20,000 devices and pulling data for all supported feeds could result in an empty associated PostgreSQL database growing to ~40 GB in size within 7 days. In such a scenario, the database might include ~225,000,000 StatusData, ~65,000,000 LogRecord and ~10,000,000 Trip records.
>
> See the Database Maintenance section for more information.

PostgreSQL

If PostgreSQL is to be used with the adapter, it is necessary to have access to a PostgreSQL server on which to set-up the adapter database. Steps are as follows:

1. Create the **geotabadapter_owner** user using the following script (first replacing `<Password>` with the desired password):

```
CREATE ROLE geotabadapter_owner WITH
  LOGIN
  NOSUPERUSER
  INHERIT
  NOCREATEDB
  NOCREATEROLE
  NOREPLICATION
```

```
  PASSWORD '<Password>';
```

2. Create the **geotabadapter_client** user using the following script (first replacing `<Password>` with the desired password):

```
CREATE ROLE geotabadapter_client WITH
  LOGIN
  NOSUPERUSER
  INHERIT
  NOCREATEDB
  NOCREATEROLE
  NOREPLICATION
  PASSWORD '<Password>';
```

3. Execute the database creation script, **geotabadapterdb-DatabaseCreationScript.sql**, which can be found along with the source code in the *..\mygeotab-api-adapter\MyGeotabAPIAdapter\PostgreSQL* folder (where *mygeotab-api-adapter* is the folder containing the *MyGeotabAPIAdapter.sln* solution file).

SQL Server

If SQL Server is to be used with the adapter, it is necessary to have access to a SQL Server instance on which to set-up the adapter database. Using SQL Server Management Studio, steps are as follows:

1. Create a database named **geotabadapterdb**.
2. Create a login named **geotabadapter_client** along with a user by the same name using the following script (first replacing `<Password>` with the desired password):

```
USE [master];
CREATE LOGIN [geotabadapter_client] WITH
      PASSWORD=N'<Password>',
      DEFAULT_DATABASE=[geotabadapterdb],
      DEFAULT_LANGUAGE=[us_english],
      CHECK_EXPIRATION=OFF,
      CHECK_POLICY=OFF;

USE [geotabadapterdb];
CREATE USER [geotabadapter_client] FOR LOGIN [geotabadapter_client] WITH
DEFAULT_SCHEMA=[dbo];
ALTER ROLE [db_datareader] ADD MEMBER [geotabadapter_client];
ALTER ROLE [db_datawriter] ADD MEMBER [geotabadapter_client];
```

3. Ensure that the database collation is set to be case-sensitive by executing the following:

```
USE [geotabadapterdb]
GO
ALTER DATABASE [geotabadapterdb] COLLATE SQL_Latin1_General_CP1_CS_AS;
GO
```

4. Execute the database creation script, **geotabadapterdb-DatabaseCreationScript.sql**, which can be found along with the source code in the *..\mygeotab-api-adapter\MyGeotabAPIAdapter\SQLServer* folder (where *mygeotab-api-adapter* is the folder containing the *MyGeotabAPIAdapter.sln* solution file).

Oracle

If Oracle is to be used with the adapter, it is necessary to have access to an Oracle instance on which to set-up the adapter database. Using Oracle SQL Developer, steps are as follows:

1. Identify or create an Oracle database that will be used.
2. If running Oracle Database XE, it *may* be necessary to execute the following statement prior to running any of the subsequent scripts:

```
alter session set "_ORACLE_SCRIPT"=true;
```

3. Create a user named **GeotabAdapter_Client** and set grants using the following script (first replacing `<password>` with the desired password):

```
CREATE USER GeotabAdapter_Client IDENTIFIED BY "<password>"
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP;

GRANT CONNECT, RESOURCE TO GeotabAdapter_Client;
GRANT UNLIMITED TABLESPACE TO GeotabAdapter_Client;
GRANT CREATE SESSION TO GeotabAdapter_Client;
GRANT CREATE TABLE, CREATE VIEW TO GeotabAdapter_Client;
```

4. Execute the database creation script, **geotabadapterdb-DatabaseCreationScript.sql**, which can be found along with the source code in the *..\mygeotab-api-adapter\MyGeotabAPIAdapter\Oracle* folder (where *mygeotab-api-adapter* is the folder containing the *MyGeotabAPIAdapter.sln* solution file).

CAUTION! When using Oracle as the adapter database, any field values greater than 4,000 characters in length will be substituted with the message "*Entity value exceeds 4000 character limit and would be truncated. Please use other means to obtain and update this placeholder with the actual value.*" before being inserted or updated. This is to avoid arbitrary truncation of data and is necessary due to Oracle's usage of the non-standard NCLOB data type to store strings greater than 4,000 characters in length.

Other

If another database type is to be used, steps will vary depending on the database provider, but will be similar to those outlined for PostgreSQL. See the Adding Support for Other Database Types section for more information.

Database Maintenance

WARNING! Regardless of database type, it is possible for the database to grow very large very quickly, resulting in **potential disk space and performance issues**. For example, running the adapter against a MyGeotab database with a fleet of ~20,000 devices and pulling data for all supported feeds could result in an empty associated PostgreSQL database growing to ~40 GB in size within 7 days. In such a scenario, the database might include ~225,000,000 StatusData, ~65,000,000 LogRecord and ~10,000,000 Trip records.

The database has been designed to allow data streamed through the adapter to be written to tables as quickly as possible. As such, no physical relationships have been implemented between tables where logical relationships exist. Additionally, the *GeotabId* columns - which represent object identifiers in the Geotab system - are not indexed. Furthermore, no data clean-up mechanisms have been implemented. Consequently, the database will continually grow in size if left unattended and queries against the tables will take increasing amounts of time to execute.

> **CAUTION!** Regardless of database type, the database has been designed to be transactional in nature to allow for data streamed through the adapter to be written as quickly as possible. As such, the database will continually grow in size if left unattended and queries against the tables will take increasing amounts of time to execute.

One might consider options such as adding indexes to some of the fields in the database tables to support query performance once record counts become significant. However, this would slow-down the writing of data being streamed from the MyGeotab database - potentially making the data less *near real-time* than would otherwise be the case. A better approach would be to implement a strategy wherein this database serves as an intermediary between the Geotab platform and the final repository where the extracted data will ultimately be stored for further processing and consumption by downstream applications.

Suggested Strategy

One possible strategy for working with and managing the database is outlined as follows:

1. Utilize the adapter database for its designed purpose as an intermediary transactional database.
2. Create another database to serve as a data warehouse (DWH).
   - It should have a set of tables similar to those in the adapter database where the “raw data” from the adapter database may be inserted.
   - Additional tables and views can then be created to support indexing, aggregation and merging with data from other systems.
3. Build an ETL or polling mechanism that continually monitors the adapter database to capture new data and add it to the DWH.
   - For tables in the **Reference data** category in the List of Tables, the *RecordLastChangedUtc* field can be used to compare values in the adapter database with those in the DWH to identify changed records.
     - ✱ **NOTE**: The *EntityStatus* field indicates whether the corresponding object is active (1) or deleted (0) in the Geotab system; if objects are deleted on the Geotab side, it may require action to be taken.
   - For tables in the **Feed data** category in the List of Tables, the *RecordCreationTimeUtc* field can be tracked on a per-table basis. A single timestamp is applied to all records in a batch of data being written to a given table. Once the new records have been added to the DWH, the records with the subject timestamp can be deleted from the table in the adapter database without risk of deleting any new records that may be added during processing.
     - **WARNING!** When deleting records from tables in the adapter database, the ***id*** column values (**NOT the *GeotabId***) should be used to identify the records to be deleted. This is both to maintain optimal performance and to avoid database concurrency/contention issues that could otherwise arise when trying to delete records while the adapter is

actively inserting/updating records. See the "GeotabId" and "Id" Columns section for more information.
    - **! IMPORTANT**: By deleting records from the adapter database once they have been added to the DWH, it is possible to avoid the potential disk space and performance issues that may arise when the *feed data* tables grow to contain millions of records.

4. Take advantage of the available adapter configuration options. For example, if only LogRecord, StatusData and FaultData feed information is required, then the other feeds such as those for DVIRLogs, ExceptionEvents and Trips can be disabled, thereby eliminating unnecessary processing and database growth. See the AppSettings - Feeds section for more information.

Data Optimizer to the Rescue!
The Data Optimizer is another application that is part of the overall solution and can be paired with the adapter in order to implement the strategy suggested above. Additionally, the optimizer includes capabilities to enhance the data and overcome certain challenges such as linking data points with different timestamps from different tables. For example, the StatusData and FaultData tables have added Latitude, Longitude, Speed, Bearing and Direction columns that can optionally be populated using LogRecords and interpolation techniques.

Full details pertaining to the Data Optimizer can be found in the MyGeotab API Adapter - Data Optimizer - Solution and Implementation Guide.

> ✱ **NOTE**: For testing and monitoring purposes, the script named **CleanDatabaseScript.sql** provides handy queries to delete data from all adapter database tables or return a list of record counts per table. The *PostgreSQL* version of the script can be found along with the source code in the *..\mygeotab-api-adapter\MyGeotabAPIAdapter\PostgreSQL* folder (where *mygeotab-api-adapter* is the folder containing the *MyGeotabAPIAdapter.sln* solution file). The *SQL Server* version of the script can be found along with the source code in the *..\mygeotab-api-adapter\MyGeotabAPIAdapter\SQLServer* folder (where *mygeotab-api-adapter* is the folder containing the *MyGeotabAPIAdapter.sln* solution file). The *Oracle* version of the script can be found along with the source code in the *..\mygeotab-api-adapter\MyGeotabAPIAdapter\Oracle* folder (where *mygeotab-api-adapter* is the folder containing the *MyGeotabAPIAdapter.sln* solution file).

Minimum Interval Sampling

LogRecord and StatusData entities typically account for a very large proportion of the data volume in a MyGeotab database - hundreds of millions of records per day for fleets of over 100K devices, for example. In some cases, this massive data volume may be deemed excessive and there may be a preference to reduce data volume such that data points are no more frequent than one per device every x number of seconds. To address this need, "minimum interval sampling" capability has been added to the MyGeotab API Adapter via the **EnableMinimunIntervalSamplingForLogRecords**, **EnableMinimunIntervalSamplingForStatusData**, **MinimumIntervalSamplingDiagnostics** and **MinimumIntervalSamplingIntervalSeconds** settings in the AppSettings - GeneralFeedSettings section of the appsettings.json file.

> **CAUTION!** Be careful before deciding to utilize the minimum interval sampling capability of the MyGeotab API Adapter. There is **no way to later back-fill data** if minimum interval sampling has been used other than clearing the adapter database, adjusting the appsettings.json file and re-running the extraction process.

✱ **NOTE**: Minimum interval sampling **does not** ensure that LogRecord or StatusData entities are are captured at a regular "polling interval" of one record every n seconds. Rather, it ensures that a *minimum* of n seconds exists between the DateTime values of LogRecord or StatusData entities that are written to the adapter database. Entities that fall within the interval are discarded instead of being written to the adapter database.

Minimum Interval Sampling for LogRecords

With regard to LogRecords, minimum interval sampling is applied on a **per-Device** basis and can be enabled based on the following rules:

1. **EnableLogRecordFeed** must be set to true.
2. **EnableDeviceCache** must be set to true. This is because the LogRecordProcessor requires the DeviceCache to be operational.
3. **EnableMinimunIntervalSamplingForLogRecords** must be set to true. Otherwise, normal processing of LogRecords will occur.
4. **MinimumIntervalSamplingIntervalSeconds** must be set to a value ranging from 1 through 3600.
5. [Optional] **DevicesToTrack** can be used to limit the collection of LogRecords to a specific set of devices.

Take the following example highlighting a specific combination of values in the appsettings.json file:

```
None
...
"EnableDeviceCache": true,
"EnableDiagnosticCache": true,
...
"FeedStartOption": "SpecificTime",
"FeedStartSpecificTimeUTC": "2024-04-01T08:00:00Z",
"DevicesToTrack": "*",
"DiagnosticsToTrack":
"DiagnosticOilPressureId,DiagnosticIgnitionId,DiagnosticEngineRoadSpeedId",
"ExcludeDiagnosticsToTrack": false,
"EnableMinimunIntervalSamplingForLogRecords": true,
"EnableMinimunIntervalSamplingForStatusData": true,
"MinimumIntervalSamplingDiagnostics":
"DiagnosticOilPressureId,DiagnosticEngineRoadSpeedId",
"MinimumIntervalSamplingIntervalSeconds": 300
...
"EnableLogRecordFeed": true,
...
"EnableStatusDataFeed": true,
```

Based on the above setting configuration:

- The LogRecords table in the adapter database will be populated with data for all devices (that the MyGeotab user configured in the LoginSettings section of appsettings.json has access to).

- The minimum interval between successive LogRecords for a given device will be 300 seconds (5 minutes).

Minimum Interval Sampling for StatusData

With regard to StatusData, minimum interval sampling is applied on a **per-Device + Diagnostic** basis and can be enabled based on the following rules:

1. **EnableStatusDataFeed** must be set to true.
2. **EnableDeviceCache** and **EnableDiagnosticCache** must both be set to true. This is because the StatusDataProcessor requires the DeviceCache and DiagnosticCache to be operational.
3. **EnableMinimunIntervalSamplingForStatusData** must be set to true. Otherwise, normal processing of StatusData will occur.
4. **MinimumIntervalSamplingIntervalSeconds** must be set to a value ranging from 1 through 3600.
5. **DiagnosticsToTrack** must be set to a comma-separated list of Diagnostic Ids. The default wildcard (*) value cannot be used if *EnableMinimunIntervalSamplingForStatusData* is set to true.
6. **ExcludeDiagnosticsToTrack** must be set to false if *EnableMinimunIntervalSamplingForStatusData* is set to true.
7. **MinimumIntervalSamplingDiagnostics** must be set to a comma-separated list of Diagnostic Ids that is either the *same* as the list provided in *DiagnosticsToTrack*, or a subset thereof. Minimum interval sampling will only be applied to StatusData records with Diagnostic Ids that are in this list. Normal processing will occur for StatusData records with Diagnostic Ids that are in the *DiagnosticsToTrack* list but not in the *MinimumIntervalSamplingDiagnostics* list.
8. [Optional] **DevicesToTrack** can be used to limit the collection of LogRecords to a specific set of devices.

Take the following example highlighting a specific combination of values in the appsettings.json file:

None

```
...
"EnableDeviceCache": true,
"EnableDiagnosticCache": true,
...
"FeedStartOption": "SpecificTime",
"FeedStartSpecificTimeUTC": "2024-04-01T08:00:00Z",
"DevicesToTrack": "*",
"DiagnosticsToTrack":
"DiagnosticOilPressureId,DiagnosticIgnitionId,DiagnosticEngineRoadSpeedId",
"ExcludeDiagnosticsToTrack": false,
"EnableMinimunIntervalSamplingForLogRecords": true,
"EnableMinimunIntervalSamplingForStatusData": true,
"MinimumIntervalSamplingDiagnostics":
"DiagnosticOilPressureId,DiagnosticEngineRoadSpeedId",
"MinimumIntervalSamplingIntervalSeconds": 300
...
"EnableLogRecordFeed": true,
...
"EnableStatusDataFeed": true,
```

Based on the above setting configuration:

- The StatusData table in the adapter database will be populated with data for all devices (that the MyGeotab user configured in the LoginSettings section of appsettings.json has access to).
- Only StatusData records with the DiagnosticOilPressureId, DiagnosticIgnitionId and DiagnosticEngineRoadSpeedId Diagnostic Ids will be collected.
- All StatusData records with the DiagnosticIgnitionId Diagnostic Id will be collected.
- For StatusData records with the DiagnosticOilPressureId and DiagnosticEngineRoadSpeedId Diagnostic Ids, the minimum interval between successive StatusData records for a given device will be 300 seconds (5 minutes).

DVIRLog Manipulator

The *DVIRLog Manipulator* is a service that runs in parallel with the main Worker service and provides the ability to update DVIRLogs in the MyGeotab database.

Capabilities

Using the DVIRLog Manipulator, it is possible - without directly using the MyGeotab API - to:

- Add repair remarks to existing DVIRDefects
- Change the repair status of existing DVIRDefects

End-to-End Bidirectional Workflow Scenario

A practical example where the DVIRLog Manipulator could be used is an integration between Geotab and an enterprise asset management (EAM) system. In this scenario:

1. Using the Geotab Drive app, drivers completing vehicle inspection reports (DVIRs) log any defects that they discover.
2. The MyGeotab API Adapter's Worker service captures the DVIRLogs and writes any defect information to the adapter database.
3. A third-party integration service extracts the defect information from the adapter database and generates repair work orders in the EAM system.
4. As repair remarks are added to the work orders in the EAM and when the repair orders are closed upon completion of corrective maintenance activities, the third-party integration service writes these remarks and updates to the *DVIRDefectUpdates* table in the adapter database.
5. The DVIRLog Manipulator service captures the repair remarks and status updates as they are written to the DVIRDefectUpdates table and makes the appropriate updates to the corresponding DVIRLogs in the Geotab system.
6. Drivers using the Geotab Drive app as well as supervisors and fleet managers using the MyGeotab web-based application are able to keep up-to-date on the status of repairs as updates flow from the EAM system into the Geotab system via this workflow.

The above workflow is illustrated with diagrams in the DVIRLog Manipulator section of the MyGeotab API Adapter presentation.

Usage

To use the DVIRLog Manipulator, simply ensure that the EnableDVIRLogFeed and EnableDVIRLogManipulator settings are both set to true before starting the adapter.

Repair remarks can be added to existing DVIRDefects and the repair status of existing DVRDefects can be updated in the Geotab system by simply inserting records into the DVIRDefectUpdates table in the adapter database.

Rules for Insertion Into the DVIRDefectUpdates Table

To ensure that updates can be successfully processed, the following rules must be adhered to when inserting these records into the DVIRDefectUpdates table:

1. Values must always be provided for the **DVIRLogId**, **DVIRDefectId** and **RecordCreationTimeUtc** fields.
2. The value provided for the **DVIRLogId** field must correspond to the Id of a DVIRLog in the MyGeotab database with which the adapter is configured to interact with.
3. The value provided for the **DVIRDefectId** field must correspond to the Id of a DVIRDefect associated with the subject DVIRLog.
4. To add a remark to a DVIRDefect:
   - Values must be provided for the **Remark**, **RemarkDateTime** and **RemarkUserId** fields.
5. To update the repair status of a DVIRDefect:
   - Values must be provided for the **RepairDateTime**, **RepairStatus** and **RepairUserId** fields.
6. To add a remark to a DVIRDefect and update the repair status of the DVIRDefect at the same time:
   - Values must be provided for the **Remark**, **RemarkDateTime, RemarkUserId, RepairDateTime**, **RepairStatus** and **RepairUserId** fields.
7. The repair status of a DVIRDefect cannot be changed once it has been set to “Repaired” or “NotNecessary”.
8. The *only* values that may be supplied for the **RepairStatus** field are “Repaired” or “NotNecessary” (other than *null*, when only a repair remark is to be added).
9. User Ids provided for the **RepairUserId** and **RemarkUserId** fields must be valid User Ids in the MyGeotab database with which the adapter is configured to interact. The same User Id can be provided for both fields; separate fields are provided for added flexibility. Additionally, the subject Users must have appropriate clearances in the MyGeotab database:
   - The User associated with the supplied **RepairUserId** must have “**Mark Asset Inspection logs as repaired**” and “**Administer Asset Inspection logs**” clearances.
   - The User associated with the supplied **RemarkUserId** must have “**Perform Asset Inspections**” and “**Administer Asset Inspection logs**” clearances.

Feedback and Exceptions

DVIRLog updates made by the DVIRLog Manipulator service are captured by the main Worker service and written to the adapter database as updates to the DVIRLogs, DVIRDefects and DVIRDefectRemarks tables. Records that have been processed by the DVIRLog Manipulator service are deleted from the DVIRDefectUpdates table in the adapter database.

Any rows in the DVIRDefectUpdates table that do not pass validation checks or for which exceptions are encountered while attempting the associated DVIRLog updates will be copied to the FailedDVIRDefectUpdates table before being deleted from the DVIRDefectUpdates table. The *FailureMessage* column provides details about the reason why a given command failed. This is to assist in debugging and to provide feedback that would otherwise be provided in the responses to commands issued via the MyGeotab API.

> **CAUTION!** Rows are never deleted from the FailedDVIRDefectUpdates table. It is up to the integrator to delete rows from this table once the error messages have been evaluated and appropriate actions have been taken.

Disabling the DVIRLog Manipulator

If the capabilities offered by the DVIRLog Manipulator are not required, simply disable the service by adjusting the EnableDVIRLogManipulator setting to be false. This will prevent the service from starting and avoid unnecessary resource utilization.

Publishing and Deployment

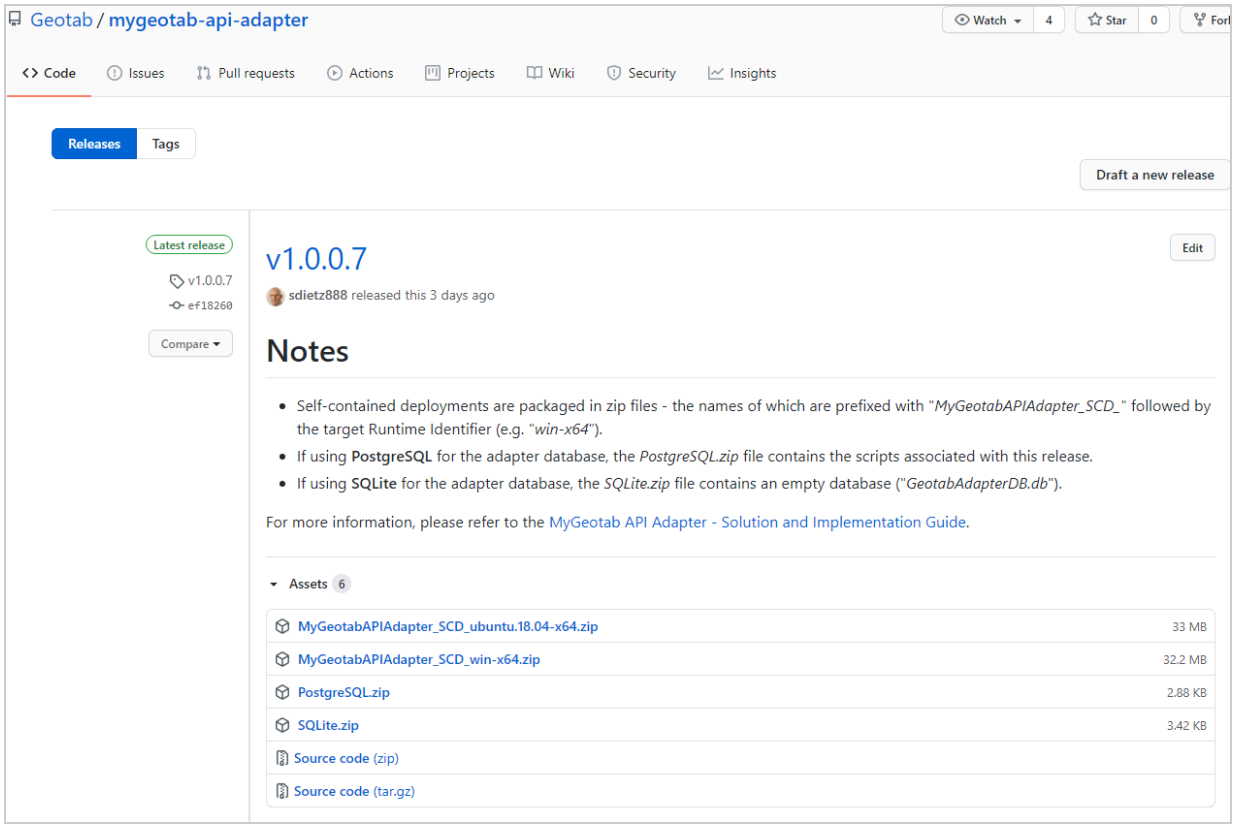
This section provides information related to publishing and deployment of the MyGeotab API Adapter solution.

Using Published Release from GitHub

The latest version of the MyGeotab API Adapter solution is available on GitHub as a pre-published release for one or more target runtimes. If there is a published release version for the desired runtime, it can be downloaded per the instructions in this section and it will then be possible to skip to the Deployment instructions (ensuring first that Deployment Prerequisites are met).

Instructions for obtaining the pre-published release are as follows:

1. Using a web browser, navigate to https://github.com/Geotab/mygeotab-api-adapter/releases.
2. Files associated with the latest release are listed under the **Assets** heading. Self-contained deployments are packaged in zip files - the names of which are prefixed with "MyGeotabAPIAdapter_SCD_" followed by the target Runtime Identifier (e.g. "win-x64"). In the following example, release packages are available for *Ubuntu* and *Windows*. If using **PostgreSQL** for the adapter database, the *PostgreSQL.zip* file contains the scripts associated with this release. If using **SQL Server** for the adapter database, the *SQLServer.zip* file contains the scripts associated with this release. If using **Oracle** for the adapter database, the *Oracle.zip* file contains the scripts associated with this release.



3 Download the appropriate files, extract the contents and proceed with deployment.

Include Database Changes

If any changes are made to the solution that result in database schema changes, such as new tables added to support additional MyGeotab data feeds, it will be necessary to provide an updated database or database creation script, depending on the database type.

> **CAUTION!** The instructions in this section assume that the deployed solution will be **starting with an empty database**. If the solution has already been deployed and changes are being made, but it is necessary to maintain the existing data, steps will need to be modified accordingly.

PostgreSQL

For PostgreSQL, the database creation script, **geotabadapterdb-DatabaseCreationScript.sql**, which can be found along with the source code in the ..*mygeotab-api-adapter**MyGeotabAPIAdapter**PostgreSQL* folder (where *mygeotab-api-adapter* is the folder containing the *MyGeotabAPIAdapter.sln* solution file) should be updated. This can be accomplished using the Backup utility in pgAdmin. Steps used to create the out-of-the-box script are as follows:

1 Ensure that the database is empty. The commented-out lines at the beginning of the **CleanDatabaseScript.sql** script can be used to delete data from the tables and reset the sequences so that the first *id* value for each table will be 1.

2. In pgAdmin, right-click on **geotabadapterdb** in the tree control and select **Backup…** from the context menu.
3. On the *General* tab, set the *Filename* to overwrite the **geotabadapterdb-DatabaseCreationScript.sql** file (or write it to a different location) and set the *Format* to **Plain**.
4. On the *Dump options* tab, set everything to **No** except for the following, which should be set to **Yes**:
   - **Only schema** and **Blobs** (in the *Type of objects* section)
   - **Include CREATE DATABASE statement** (in the *Queries* section)
5. Click the **Backup** button. A dialog box will appear and should indicate that the backup was successfully completed.
6. Using a text editor, open the updated **geotabadapterdb-DatabaseCreationScript.sql** file and add the following lines to the end of the file::

```
-- Manual additions not captured by backup procedure:
GRANT ALL ON SEQUENCE public."DVIRDefectRemarks_id_seq" TO geotabadapter_client;
GRANT ALL ON SEQUENCE public."DVIRDefectRemarks_id_seq" TO geotabadapter_owner;
GRANT ALL ON SEQUENCE public."FailedDVIRDefectUpdates_id_seq" TO geotabadapter_client;
GRANT ALL ON SEQUENCE public."FailedDVIRDefectUpdates_id_seq" TO geotabadapter_owner;
```

> **WARNING!** DO NOT include data in the script. This is to avoid potential issues related to unauthorized access to customer data.

SQL Server

For SQL Server, the database creation script, **geotabadapterdb-DatabaseCreationScript.sql**, which can be found along with the source code in the ..*mygeotab-api-adapter**MyGeotabAPIAdapter**SQLServer* folder (where *mygeotab-api-adapter* is the folder containing the *MyGeotabAPIAdapter.sln* solution file) should be updated. This can be accomplished using SQL Server Management Studio (SSMS). Steps used to create the out-of-the-box script are as follows:

1. In SSMS, right-click on **geotabadapterdb** in the Object Explorer and select **Tasks > Generate Scripts…** from the context menu.
2. Click the ***Next>*** button, if necessary, to get to the **Choose Objects** step of the *Generate Scripts* dialog. Then, choose the **Select specific database objects** option, check the **Tables**, **Views**, and **Stored Procedures** checkboxes, and click the ***Next>*** button.
3. On the **Set Scripting Options** step of the *Generate Scripts* dialog:
   - Choose the **Save as script file** option.
   - Choose the **Single script file** option.
   - Set the **File name** to point to the **geotabadapterdb-DatabaseCreationScript.sql** file.
   - Check the **Overwrite existing file** checkbox, select the **Unicode text** option.
   - Click the **Advanced** button and, on the resulting *Advanced Scripting Options* dialog:
     - Set the **Script for Server Version** value to **SQL Server 2016**.
     - Set the **Script for the database engine edition** value to **Microsoft SQL Server Standard Edition**.
     - Set the **Script Indexes** value to **True**.
   - Click the ***Next>*** button.

4. On the **Summary** step of the *Generate Scripts* dialog, click the ***Next>*** button.

5. On the **Save Scripts** step of the *Generate Scripts* dialog, once processing has completed, click the ***Finish*** button.

WARNING! DO NOT include data in the script. This is to avoid potential issues related to unauthorized access to customer data.

Oracle

For Oracle, the database creation script, **geotabadapterdb-DatabaseCreationScript.sql**, which can be found along with the source code in the *..\mygeotab-api-adapter\MyGeotabAPIAdapter\Oracle* folder (where *mygeotab-api-adapter* is the folder containing the *MyGeotabAPIAdapter.sln* solution file) should be updated. This can be accomplished using Oracle SQL Developer. Steps used to create the out-of-the-box script are as follows:

1. In Oracle SQL Developer, with a connection to the database established, click the **Tools** menu, followed by the **Database Export...** menu item.
2. On the **Source/Destination** screen (*Export Wizard - Step 1 of 5*):
   - Select the appropriate connection.
   - In the *Export DDL* group, check the **Grants**, **Pretty_Print** and **Terminator** checkboxes and select **COMPATIBLE** in the *Version* dropdown list.
   - Uncheck the *Export Data* checkbox.
   - Choose **Single File** for the *Save* option, uncheck the *Compressed* checkbox, choose **Cp1252** for the *Encoding* option. Then, click the **Browse** button and set the *File* to point to the **geotabadapterdb-DatabaseCreationScript.sql** file.
   - Click the ***Next>*** button.
3. On the **Types to Export** screen (*Export Wizard - Step 2 of 5*):
   - In the *Standard Object Types* group, check the **Tables**, **Views**, **Indexes**, **Constraints, Referential Constraints**, **Sequences**, **Procedures** and **Functions** checkboxes.
   - Click the ***Next>*** button.
4. On the **Specify Objects** screen (*Export Wizard - Step 3 of 5*):
   - Click the **More...** button. Then, select **GEOTABADAPTER_CLIENT** in the *Schema* dropdown list and **ALL OBJECTS** in the *Type* dropdown list.
   - Click the **Lookup** button. Then, once the list of items is populated, click the **>>** button to move all of the items into the box on the right side of the dialog.
   - Click the ***Next>*** button.
5. On the **Export Summary** screen (*Export Wizard - Step 4 of 4*):
   - Click the ***Finish*** button. A dialog will appear and show the export progress. When the export has completed, the **geotabadapterdb-DatabaseCreationScript.sql** file will open in a new tab in Oracle SQL Developer.

WARNING! DO NOT include data in the script. This is to avoid potential issues related to unauthorized access to customer data.

Other

If another database type is being supported, steps will vary depending on the database provider, but will be similar to those outlined for PostgreSQL.

Include Configuration File Changes

Ensure that the **appsettings.Publish.json** and **nlog.config** files have appropriate values set for the various items. Both files will be included with the deliverable package as part of the publishing process. They can be found in the *..\mygeotab-api-adapter\MyGeotabAPIAdapter* folder (where *mygeotab-api-adapter* is the folder containing the *MyGeotabAPIAdapter.sln* solution file).

WARNING! For security purposes, take care to avoid including login credentials and database connection information in the *appsettings.Publish.json* file as it will be included in the published package that may be delivered to others. The Database Settings and Login Settings sections should use placeholders instead of actual values and should appear as follows:

```
"DatabaseSettings": {
    "DatabaseProviderType": "PostgreSQL",
    "DatabaseConnectionString":
"Server=<Server>;Port=<Port>;Database=geotabadapterdb;User
Id=geotabadapter_client;Password=<password>"
    //"DatabaseProviderType": "SQLServer",
    //"DatabaseConnectionString": "Server=<Server>;Database=geotabadapterdb;User
Id=geotabadapter_client;Password=<password>"
  },
  "LoginSettings": {
    "MyGeotabServer": "<MyGeotabServer>",
    "MyGeotabDatabase": "<MyGeotabDatabase>",
    "MyGeotabUser": "<MyGeotabUser>",
    "MyGeotabPassword": "<MyGeotabPassword>"
  },
```

Update NuGet Packages

Prior to publishing, it is a good practice to ensure that all NuGet packages used in the solution are updated to the latest stable versions. Use **NuGet Package Manager** to update the packages for each of the projects in the solution:

- MyGeotabAPIAdapter
- MyGeotabAPIAdapter.Database
- MyGeotabAPIAdapter.Tests

Change Assembly Versions

Prior to building and publishing the solution, the assembly version information should be changed. This may be accomplished by editing the **MyGeotabAPIAdapter.csproj**, **MyGeotabAPIAdapter.Database.csproj** and **MyGeotabAPIAdapter.Tests.csproj** files. In each of these files, modify the values of the **Version** and **InformationalVersion** elements. For example, the *Version* element might be set to **1.0.0.6-beta** and the *InformationalVersion* might correspondingly be set to **1.0.0.6-beta: This is a prerelease build.**

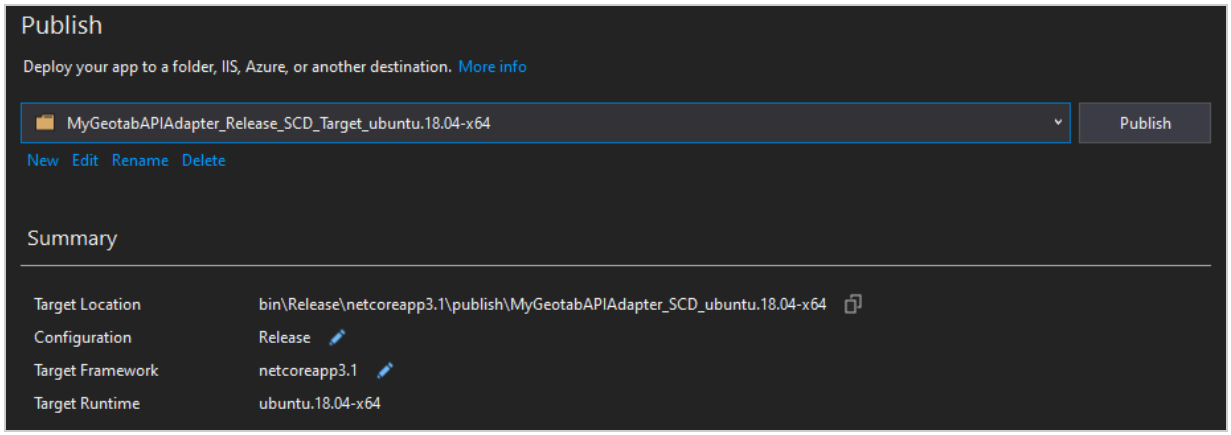
Publish

After completing the preparation steps detailed in the previous sections, it is possible to move-on to the publishing process. Note that the steps outlined here are based on using Microsoft Visual Studio Professional 2019 and that they may differ somewhat if using another IDE or publishing method.

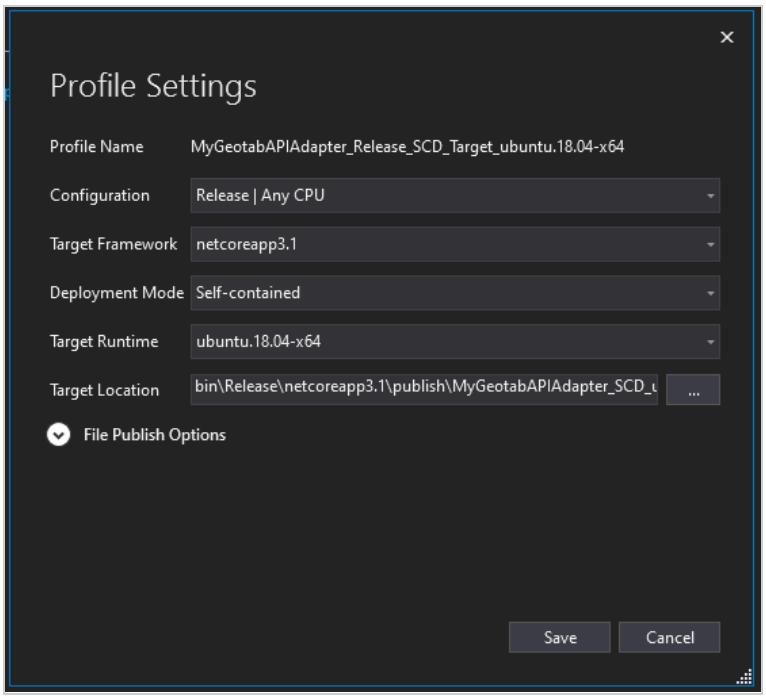
> ✱ **NOTE**: In this example, the target runtime is *ubuntu.18.04-x64*. The chosen target runtime will depend on the server on which the published MyGeotab API Adapter is intended to be deployed. It is possible to create multiple publish profiles to allow the solution to be published for multiple target runtimes. This link from Microsoft contains more information.

Publishing steps are as follows:

1. In the *Solution Explorer*, select the **MyGeotabAPIAdapter** project.
2. Use the main menu to navigate to **Build > Publish MyGeotabAPIAdapter**.
3. Configure the *Target Location*, *Configuration*, *Target Framework*, and *Target Runtime* options similar to the following:

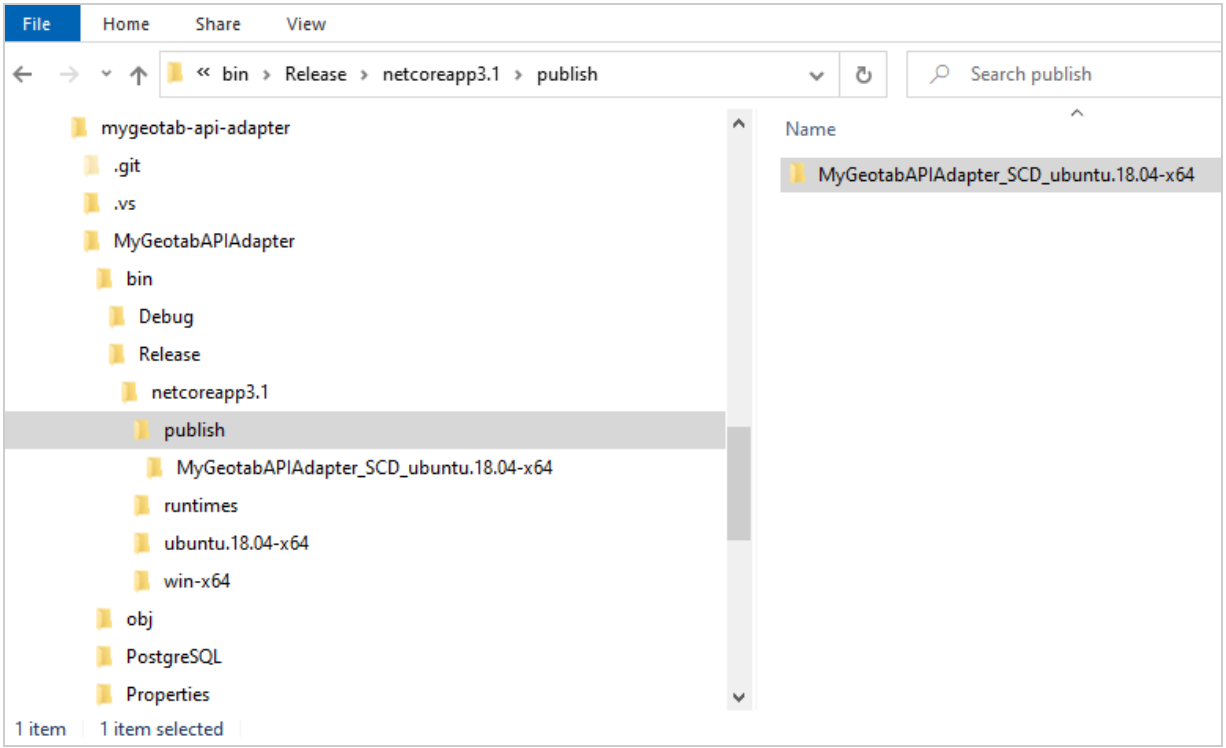


4. Click one of the edit pencils in the previous screenshot to reveal the **Profile Settings** dialog. Then, configure it similar to the following and click the *Save* button:



5 Click the **Publish** button shown in Step 3.

6 The published solution will appear, in this case, as a folder containing all of the files and dependencies needed to run the adapter on a server with the target runtime:



7 The published output in the previous step will include the ***appsettings.json*** file (which is actually the *appsettings.Publish.json* file from the solution) and the ***nlog.config*** file. However, the SQL Server,

PostgreSQL and Oracle database creation scripts are not included and would need to be manually packaged along with the published folder.

Deployment Prerequisites

To provide for a smooth deployment, the following steps should be taken beforehand:

1. Ensure that **permission** has been granted by the owner of the MyGeotab database with which the adapter will be interacting.
2. A **service account** should be established for use by the adapter. See the Service Account Guidelines document for more details.
3. Take appropriate steps (networking, firewall, etc.) to ensure **connectivity** between the server on which the adapter will be deployed and the MyGeotab platform (https://my.geotab.com).
4. Make sure that the adapter **database setup** has been completed per the instructions in the Database Setup section.
5. Make sure that the server hosting the adapter database has enough disk space and that a database maintenance strategy is implemented to prevent unlimited growth.
   WARNING! It is possible for the database to grow very large very quickly, resulting in **potential disk space and performance issues**. In particular, the ***feed data*** tables can accumulate vast quantities of records within short periods of time. See the Database Maintenance section for more information.

Deploy

Because the MyGeotab API Adapter solution has been published using the *Self-contained* deployment mode (see this link from Microsoft for more information) in this example, it is quite simple to deploy:

1. Copy the published folder that was generated in Step 6 of the Publish section to the server on which the adapter is to reside.
2. Modify the **appsettings.json** file as needed. See the appsettings.json section for more information.
   ✱ **NOTE**:Take advantage of the available adapter configuration options. For example, if only LogRecord, StatusData and FaultData feed information is required, then the other feeds such as those for DVIRLogs, ExceptionEvents and Trips can be disabled, thereby eliminating unnecessary processing and database growth. See the AppSettings - Feeds section for more information.
3. Review the **nlog.config** file and make any necessary changes. See the nlog.config section for more information.
4. Run the adapter executable ("*MyGeotabAPIAdapter*" or "*MyGeotabAPIAdapter.exe*", depending on the target runtime) in the published folder that was copied in Step 1.
   ✱ **NOTE**: It is best to setup a process to run the adapter using a system account. On a Windows Server, for example, Windows Task Scheduler can be used to create a task that runs *MyGeotabAPIAdapter.exe* on server startup.

Deploy to Microsoft Azure

With the growing prevalence of cloud computing platforms, questions may arise from those seeking to build integrations that bring Geotab data into such platforms. Although this guide provides instructions that can be used to deploy the MyGeotab API Adapter solution within any number of different cloud computing platforms, integrators may strive to automate such deployments to the greatest extent possible.

In order to demonstrate semi-autonomous deployment of the MyGeotab API Adapter solution into a cloud platform environment, an example has been created using Microsoft Azure as the cloud platform host. For more information, refer to the MyGeotab API Adapter - Guide for Deploying to Microsoft Azure guide.

Adding Support for Other Database Types

Adding support for additional database types should be very straightforward as a result of the fact that the MyGeotab API Adapter solution incorporates a repository pattern to abstract database interaction from business logic (see the Solution Architecture section for more detail). To add support for another type of database:

1. Replicate the PostgreSQL adapter database schema for the subject database type. Refer to the Data Dictionary section for schema information. Note that data types may differ between database providers and it is important to assign field data types based on the closest match.
2. For the *MyGeotabAPIAdapter.Database* project, use NuGet Package Manager to incorporate the .NET data provider for the subject database type.
3. Modify the **ConnectionInfo** class in the *MyGeotabAPIAdapter.Database* project:
   - Modify the **DataAccessProviderType** enum by adding a new constant to represent the type of database for which support is being added.
   - Modify the **RegisterDbProviderFactory** method by adding a case for the subject database type to the *switch* statement.
4. Modify the **DatabaseProviderType** and **DatabaseConnectionString** settings in the DatabaseSettings section of the *appsettings.json* file. Note that the value supplied for the **DatabaseProviderType** setting should be one of the constants defined for the **DataAccessProviderType** enum in the *ConnectionInfo* class (detailed in the previous step).
5. In the **SqlMapperExtensions** class in the *MyGeotabAPIAdapter.Database* project (under *DataAccess* > *Dapper*), modify the **GetAllAsync<T>** method by adding a case for the subject database type to the *switch* statement. Note that case definition will depend on the namespace of the subject data access provider and it may require testing to confirm the correct string to use. The following code extract shows this switch statement at time of publishing this document:

```
...
string connectionProviderType = connection.GetType().Namespace;
switch (connectionProviderType)
{
      case "Npgsql":
            sql = $"select * from \"{name}\" limit {resultsLimit}";
            break;
      case "Microsoft.Data.SqlClient":
```

```
        case "System.Data.SqlClient":
            sql = $"select top ({resultsLimit}) * from \"{name}\"";
            break;
        default:
            throw new NotImplementedException($"Support for the
'{connectionProviderType}' connection provider type has not been implemented in
this method.");
}
...
```

After completing these steps, the adapter should have no issues utilizing an instance of the target database type; there should be no need to modify any code within the solution.

Understanding the Database Through Examples

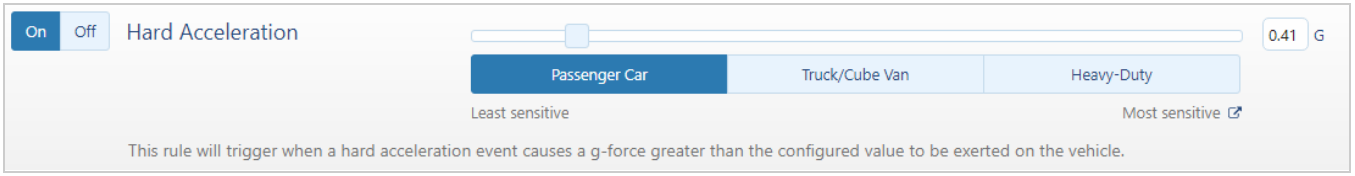
Since the MyGeotab API Adapter solution essentially extracts data from a MyGeotab database and writes it to the adapter database, the MyGeotab SDK and API Reference pages along with the Data Dictionary section of this document should provide the information needed to understand the data. This section includes some examples to augment the aforementioned documentation and provide some insight into ways that data may be queried from the adapter database to address common questions.

Keeping in mind the fact that the adapter database is intended to be an intermediary rather than the actual data warehouse to be queried by applications (as described in the Database Maintenance and Suggested Strategy sections), the queries in these examples have been kept very simple.

Find ExceptionEvents Associated With a Specific Rule

This example demonstrates how to find ExceptionEvents associated with a specific Rule. In this case, the objective is to find all occurrences of **hard acceleration**.

In the MyGeotab database with which the adapter is associated, the target Rule is named *Hard Acceleration* as shown in this screenshot from the MyGeotab user interface:



Identify the Rule Id

Knowing that the target Rule is named “Hard Acceleration”, query the Rules table to find the Id of the subject Rule.

Query:

```
select * from "Rules"
where "Name" = 'Hard Acceleration';
```

Result:

	id	GeotabId	ActiveFrom	ActiveTo	BaseType	Comment	Name	Version	EntityStatus	RecordLastChangedUtc
1	123	RuleJackrabbitStartsId	2017-11-22 21:48:34.713Z	2050-01-01 00:00:00Z	Stock		Hard Acceleration	2028	1	2020-07-13 14:32:06.7764665Z

The result reveals that:

- Only one record is returned, which means that the result shows the correct Rule.
- The Rule Id is "**RuleJackrabbitStartsId**".
- It is a *Stock* Rule.

CAUTION! Customers often create custom Rules - either by modifying stock Rules or by creating entirely new Rules. Custom Rules have unique Ids on a per MyGeotab database basis. As such, when developing an integration, it is important to verify with the customer which Rules are to be used; **do not assume** that the stock Rules are the ones being used by the customer.

Find the Conditions That Form the Rule

Having identified the Rule Id, it is possible to discover exactly which Conditions form the Rule.

Query:

```
select * from "Conditions"
where "RuleId" = 'RuleJackrabbitStartsId';
```

Result:

	id	GeotabId	ParentId	RuleId	ConditionType	DeviceId	DiagnosticId
1	885	aK8QYKCXDxUiOsWeQmDN_2A		RuleJackrabbitStartsId	IsValueMoreThan	NoDeviceId	NoDiagnosticId
2	886	aWtwd5hoqAEulbBSu4J9Dfw	aK8QYKCXDxUiOsWeQmDN_2A	RuleJackrabbitStartsId	FilterStatusDataByDiagnostic	NoDeviceId	DiagnosticAccelerationForwardBrakingId

...

	DriverId	Value	WorkTimeId	ZoneId	EntityStatus	RecordLastChangedUtc
1	NoDriverId	4.059953152			1	2020-07-13 14:32:06.7764665Z
2	NoDriverId				1	2020-07-13 14:32:06.7764665Z

The result reveals that:

- The Rule is based on the Diagnostic named "**DiagnosticAccelerationForwardBrakingId**" having a value greater than 4.05995315167102.
- The Rule is not tied to any specific Devices, Drivers(Users), WorkTimes or Zones.

Get UnitOfMeasure Information From the Condition Diagnostics

In the previous section, the condition is based on a value of 4.05995315167102, but the UnitOfMeasure is still unknown. It can be obtained by retrieving the subject Diagnostic.

Query:

```
select * from "Diagnostics"
```

```
where "GeotabId" = 'DiagnosticAccelerationForwardBrakingId';
```

Result:

	id	GeotabId	ControllerId	DiagnosticCode	DiagnosticName	DiagnosticSourceId
1	24177	DiagnosticAccelerationForwardBrakingId	ControllerNoneId	1	Acceleration forward or braking	SourceSystemId

...

	DiagnosticSourceName	DiagnosticUnitOfMeasureId	DiagnosticUnitOfMeasureName	OBD2DTC	EntityStatus	RecordLastChangedUtc
1	**System	UnitOfMeasureMetersPerSecondSquaredId	m/s^2		1	2020-07-13 14:32:04.7712818Z

The result reveals that:

- The *DiagnosticAccelerationForwardBrakingId* Condition is measured in **metres per second squared**.

Find ExceptionEvents

With the information gathered in the previous steps, it is now possible to find all occurrences of *Hard Acceleration* Rule violations that have been written to the adapter database and have the detail needed for display in a dashboard or consumption by an external application.

Query:

```
select * from "ExceptionEvents"
where "RuleId" = 'RuleJackrabbitStartsId';
```

Result:

	id	GeotabId	ActiveFrom	ActiveTo	DeviceId	Distance	DriverId	DurationTicks	RuleId	Version	RecordCreationTimeUtc
493	7932	aIIMRzZHtr	2017-07-20 21:43:53.543Z	2017-07-20 21:43:53.907Z	b1	0.002512351144	b41B4DC05	36400	RuleJackrabbitStartsId	15461	2020-07-13 14:32:11.5945521Z
494	7933	aG79uR5CD	2017-07-20 21:43:54.273Z	2017-07-20 21:43:54.78Z	b1	0.003511098213	b41B4DC05	50700	RuleJackrabbitStartsId	15462	2020-07-13 14:32:11.5945521Z
495	7934	akt-ay5PDa	2017-07-21 21:34:41.997Z	2017-07-21 21:34:50.187Z	b1	0.0004758710857	b41B4DC05	819000	RuleJackrabbitStartsId	15528	2020-07-13 14:32:11.5945521Z
496	7935	aeIzCj0TCw	2017-07-21 23:15:44.033Z	2017-07-21 23:15:46.12Z	b2	0.01167005859	UnknownDriverId	208700	RuleJackrabbitStartsId	15574	2020-07-13 14:32:11.5945521Z

With the result set, in addition to the times and durations of *Hard Acceleration* Rule violations, it is possible to use the **DeviceId** and **DriverId** values for a given ExceptionEvent to find the associated Device/Vehicle and Driver/User by querying the Devices table and Users table, respectively.

Find Location of ExceptionEvent

ExceptionEvents do not include location information to indicate where they have occurred. This is because location information is available through LogRecord (GPS data) objects which come through a separate data feed. To find the location of an ExceptionEvent (or locations, if the ExceptionEvent occurred over an extended period of time), the LogRecords table can be queried using the *DeviceId*, *ActiveFrom* and *ActiveTo* values of the subject ExceptionEvent.

Query:

```
select * from "LogRecords"
where "DeviceId" = 'b2'
and "DateTime" >= '2017-07-21 23:15:44.033'
and "DateTime" <= '2017-07-21 23:15:46.12';
```

Result:

	id	GeotabId	DateTime	DeviceId	Latitude	Longitude	Speed	RecordCreationTimeUtc
1	264735	b5E2AA	2017-07-21 23:15:46Z	b2	45.0994186	-87.6370163	22	2020-07-13 14:33:13.134752Z

In this example, a single LogRecord is returned showing the location and speed of the vehicle at the time of the ExceptionEvent.

Obtain OBD-II DTC From FaultData

This section demonstrates how to obtain the OBD-II Diagnostic Trouble Code (DTC) associated with a record in the FaultData table.

Start With a FaultData Record

The following example of a record from the FaultData table will be used to demonstrate how to retrieve the OBD-II DTC.

	id	GeotabId	AmberWarningLamp	ClassCode	ControllerId	ControllerName	Count	DateTime
1	3886752	b72D204	0		ControllerObdWwhPowertrainId	Powertrain (P)	1	2020-07-09 17:43:54.807Z

...

	DeviceId	DiagnosticId	DismissDateTime	DismissUserId	FailureModeCode	FailureModeId	FailureModeName	FaultLampState
1	b2DEA	aq0_bgE84UkqEYS3Q3W4bfQ		NoUserId	0	b2761	No sub type information	

...

	FaultState	MalfunctionLamp	ProtectWarningLamp	RedStopLamp	Severity	SourceAddress	RecordCreationTimeUtc
1	Active	0	0	0		61713	2020-08-11 14:12:06.3833709Z

Get the OBD-II DTC From the Associated Diagnostic

Query the Diagnostics table to retrieve the Diagnostic associated with the subject FaultData.

Query:

```
select * from "Diagnostics" where "GeotabId" = 'aq0_bgE84UkqEYS3Q3W4bfQ';
```

Result:

	id	GeotabId	ControllerId	DiagnosticCode	DiagnosticName	DiagnosticSourceId
1	2885	aq0_bgE84UkqEYS3Q3W4bfQ	ControllerObdWwhPowertrainId	309	O2 sensor heater circuit bank 1 sensor 1	SourceObdSaId

...

	DiagnosticSourceName	DiagnosticUnitOfMeasureId	DiagnosticUnitOfMeasureName	OBD2DTC	EntityStatus	RecordLastChangedUtc
1	**Obd Sa	UnitOfMeasureNoneId	None	P0135	1	2020-08-11 14:11:53.9826684Z

If the *DiagnosticSourceId* is "**SourceObdId**" or "**SourceObdSaId**", which it is in this case, the OBD-II DTC can be found in the *OBD2DTC* field: *P0135* in this example.

Obtain J1939 FMI and SPN From FaultData

This section demonstrates how to obtain the J1939 Failure Mode Identifier (FMI) and Suspect Parameter Number (SPN) associated with a record in the FaultData table.

Start With a FaultData Record

The following example of a record from the FaultData table will be used to demonstrate how to retrieve the J1939 FMI and SPN.

	id	GeotabId	AmberWarningLamp	ClassCode	ControllerId	ControllerName	Count	DateTime
1	5356881	b733887	1		ControllerTransmissionNo1Id	Transmission #1	16	2020-06-23 13:55:51.788Z

...

	DeviceId	DiagnosticId	DismissDateTime	DismissUserId	FailureModeCode	FailureModeId	FailureModeName	FaultLampState
1	b3C8D	awwZOakD61kWrgEiPnuKQ2g		NoUserId	5	b2724	Current below normal or open circuit	

...

	FaultState	MalfunctionLamp	ProtectWarningLamp	RedStopLamp	Severity	SourceAddress	RecordCreationTimeUtc
1	Active	0	0	1			2020-08-11 14:12:06.3833709Z

Get the J1939 FMI & SPN Based on the Associated Diagnostic

Query the Diagnostics table to retrieve the Diagnostic associated with the subject FaultData.

Query:

```
select * from "Diagnostics" where "GeotabId" = 'awwZOakD61kWrgEiPnuKQ2g';
```

Result:

	id	GeotabId	ControllerId	DiagnosticCode	DiagnosticName	DiagnosticSourceId
1	3911	awwZOakD61kWrgEiPnuKQ2g	ControllerNoneId	5614	Transmission clutch coarse engagement actuator	SourceJ1939Id

...

	DiagnosticSourceName	DiagnosticUnitOfMeasureId	DiagnosticUnitOfMeasureName	OBD2DTC	EntityStatus	RecordLastChangedUtc
1	**J1939	UnitOfMeasureNoneId	None		1	2020-08-11 14:11:53.9826684Z

If the *DiagnosticSourceId* is "**SourceJ1939Id**", which it is in this case:

- The FMI is found in the *FailureModeCode* field of the FaultData record: *5* in this example.
- The SPN is found in the *DiagnosticCode* field of the Diagnostic record: *5614* in this example.
- The SPN description is found in the *DiagnosticName* field of the Diagnostic record: "*Transmission clutch coarse engagement actuator*" in this example.

Retrieve DVIRLog Information

This example demonstrates how information related to DVIRLogs may be queried from the adapter database.

Find a DVIRLog

The DVIRLogs table can be queried to retrieve DVIRLogs that have been obtained from the MyGeotab database.

Query:

```
select * from "DVIRLogs";
```

Result:

	id	GeotabId	CertifiedByUserId	CertifiedDate	CertifyRemark	DateTime	DeviceId	DriverId	DriverRemark	IsSafeToOperate
1	3861	ataVQAlKkJECxathrw3valg	b3	2020-05-11 19:13:28.54Z		2020-05-11 19:11:50.813Z	b388	b3	Check tires.	1

...

	LocationLatitude	LocationLongitude	LogType	RepairDate	RepairRemark	RepairedByUserId	TrailerId	TrailerName	Version	RecordCreationTimeUtc
1	43.515833	-79.683568	PreTrip	2020-05-11 19:13:13.637Z		b3			10545791	2020-07-13 15:43:05.0772272Z

The result above is for a single DVIRLog record and reveals that:

- The DVIRLog is for a pre-trip inspection where at least one defect was identified and repaired before the DVIRLog was certified (as indicative by the presence of values in the *RepairDate* and *CertifiedDate* fields).
- The inspection, repair(s) and certification were all logged by the same person - likely the driver (as evident in the values of the *DriverId*, *RepairedByUserId* and *CertifiedByUserId* fields).
- The *DeviceId* and *DriverId* values could be used to query the Devices and Users tables, respectively, to obtain more information.

Get Defects Associated With the DVIRLog

A list of all defects associated with the *DVIRLog* examined in the previous section can be retrieved from the DVIRDefects table.

Query:

```
select * from "DVIRDefects"
where "DVIRLogId" = 'ataVQAlKkJECxathrw3valg';
```

Result:

	id	GeotabId	DVIRLogId	DefectListAssetType	DefectListId	DefectListName	PartId	PartName
1	87345	a88gGjG0ye0KFs3DamizpWA	ataVQAlKkJECxathrw3valg	All	b1408	Master Vehicle Inspection List	b135E	Tires

...

	DefectId	DefectName	DefectSeverity	RepairDateTime	RepairStatus	RepairUserId	EntityStatus	RecordLastChangedUtc
1	b276A	Flat or Leaking Air	Normal	2020-05-11 19:13:13.637Z	Repaired	b3	1	2020-07-13 15:43:05.0772272Z

The result reveals that:

- A single defect was identified in this pre-trip inspection: a tire was either flat or leaking air (as determined by examining the *PartName* and *DefectName* values).

Get Remarks Associated With the DVIRDefect

Any comments that are associated with the DVIRDefect identified in the previous section can be retrieved from the DVIRDefectRemarks table.

Query:

```
select * from "DVIRDefectRemarks"
```

```
where "DVIRDefectId" = 'a88gGjG0ye0KFs3DamizpWA'
order by "DateTime";
```

Result:

	id	GeotabId	DVIRDefectId	DateTime	Remark	RemarkUserId	EntityStatus	RecordLastChangedUtc
1	996	aVcVKf2SFeUCJMBsPsdwziA	a88gGjG0ye0KFs3DamizpWA	2020-05-11 19:11:48.247Z	Front passenger tire is flat.	b3	1	2020-07-13 15:43:05.0772272Z
2	997	a1aKUTgX3KECt7PMlaXRGTw	a88gGjG0ye0KFs3DamizpWA	2020-05-11 19:13:13.637Z	Replaced tire.	b3	1	2020-07-13 15:43:05.0772272Z

The result shows that the user provided some additional information about the DVIRDefect:

- The tire in question was on the front passenger side of the vehicle.
- The tire was found to be flat and was subsequently replaced to remedy the DVIRDefect.

Using DutyStatusAvailability Information

This section demonstrates usage of the DutyStatusAvailabilities table in the adapter database.

The Pseudo Data Feed for DutyStatusAvailability

The Geotab API's GetFeed method does not support the DutyStatusAvailability entity type and the results for DutyStatusAvailability Get requests are calculated dynamically, resulting in longer response times than are typical for pre-calculated data. It is also necessary to retrieve DutyStatusAvailability on a per-driver basis using batches of Get requests wrapped in MultiCall requests (in order to support larger fleets where the number of Get<DutyStatusAvailability> requests required to cover all drivers could not be made in a single MultiCall request). The result of the combination of these factors is that it can take some time for DutyStatusAvailability to be retrieved for all drivers in a fleet.

In order to avoid slowing the flow of data for other feeds, DutyStatusAvailability is handled by a separate worker service running in parallel to the main worker service and acting as a pseudo data feed to populate the DutyStatusAvailability table in the adapter database. This service runs in parallel to the main Worker service if the *enableDutyStatusAvailabilityDataFeed* setting in appsettings.json is set to true.

Maintaining Currency of Values Via Time Offsets

> **WARNING!** Duration values in the DutyStatusAvailability table are inaccurate (out-of-date). The amount of inaccuracy can be defined as the duration between the value of the *RecordLastChangedUtc* field and the current time, in Coordinated Universal Time (UTC). This offset must be applied to the duration values in order to improve currency of the data upon consumption. See below for details.

As a result of the pseudo data feed and inherent time required to update DutyStatusAvailability records for all drivers, by the time the last driver's record is being updated, minutes may have elapsed since the first driver's record was updated. The amount of time elapsed since a record was last updated will depend on a number of factors - particularly the number of drivers in the fleet.

Determining How Much Driving and On-Duty Time a Driver Has Left in the Day

The following example of a record from the DutyStatusAvailabilities table will be used to demonstrate how to work with data in this table. For this scenario, the objective is to determine how much driving and on-duty time a particular driver has remaining in the day.

	id	DriverId	CycleAvailabilities	CycleTicks	CycleRestTicks
35	1388	b124	[{"DateTime":"2021-01-16T05:00:00Z","Available":"4.22:30:42.4420000"} ,{"DateTime":"2021-01-17T05:00:00Z","Available":"4.22:30:42.4420000"} ,{"DateTime":"2021-01-18T05:00:00Z","Available":"4.22:30:42.4420000"} ,{"DateTime":"2021-01-19T05:00:00Z","Available":"4.22:30:42.4420000"} ,{"DateTime":"2021-01-20T05:00:00Z","Available":"4.22:30:42.4420000"} ,{"DateTime":"2021-01-21T05:00:00Z","Available":"4.22:30:42.4420000"} ,{"DateTime":"2021-01-22T05:00:00Z","Available":"4.22:30:42.4420000"} ,{"DateTime":"2021-01-23T05:00:00Z","Available":"4.22:30:42.4420000"} ,{"DateTime":"2021-01-24T05:00:00Z","Available":"4.22:30:42.4420000"} ,{"DateTime":"2021-01-25T05:00:00Z","Available":"4.22:30:42.4420000"} ,{"DateTime":"2021-01-26T05:00:00Z","Available":"4.22:30:42.4420000"} ,{"DateTime":"2021-01-27T05:00:00Z","Available":"4.22:30:42.4420000"} ,{"DateTime":"2021-01-28T05:00:00Z","Available":"4.22:38:00.8390000"} ,{"DateTime":"2021-01-29T05:00:00Z","Available":"5.00:00:00"}]	4266424420000	10888798120000

...

	DrivingTicks	DutyTicks	DutySinceCycleRestTicks	Is16HourExemptionAvailable	IsAdverseDrivingExemptionAvailable
35	452166240000	454808390000	2466424420000		1

...

	IsOffDutyDeferralExemptionAvailable	Recap	RestTicks	WorkdayTicks	RecordLastChangedUtc
35	1	[{"DateTime":"2021-01-14T05:00:00Z","Duration":"00:07:18.3970000"}, {"DateTime":"2021-01-15T05:00:00Z","Duration":"01:21:59.1610000"}]		526808390000	2021-01-15 14:32:00

Let's assume that the current UTC date and time is **2021-01-15 14:35:00**. Driving and on-duty time remaining can be determined using the following steps:

1. **Convert the DrivingTicks and DutyTicks values into corresponding durations.** The manner by which this is accomplished depends on the technology being used (e.g. in .NET, the TimeSpan.FromTicks method can be used).

   ✱ **NOTE**: A tick is equal to 100 nanoseconds or one ten-millionth of a second. There are 10,000 ticks in a millisecond and 10 million ticks in a second. For brevity, only the calculated durations will be shown. This .NET Fiddle demonstrates conversion of ticks to TimeSpans (*hours:minutes:seconds.milliseconds* here).
   - DrivingTicks = 452166240000 ticks = **12:33:36.624**
   - DutyTicks = 454808390000 ticks = **12:38:00.839**

2. **Calculate the elapsed time since the subject record was updated.**
   - RecordLastChangedUtc = 2021-01-15 14:32:00
   - Current UTC date and time = 2021-01-15 14:35:00
   - Elapsed time = **00:03.000**

3. **Subtract the calculated offset from the Driving and Duty durations.**
   - Driving Duration = 12:33:36.624 **-** 00:03.000 = **12:30:36.624**
   - Duty Duration = 12:38:00.839 **-** 00:03.000 = **12:35:00.839**

As determined above, in this example, the subject driver has twelve hours and thirty minutes of driving time and twelve hours and thirty-five minutes of on-duty time remaining in the day.

> **WARNING!** Even if time offsets are calculated as noted above, availability values may occasionally still be inaccurate due to driver status changes that may occur in between successive updates.

Using Trip Information

This section demonstrates usage of the Trips table in the adapter database.

Multiple Records for a Single Trip Captured Live

> **CAUTION!** Unlike other entity types, with the MyGeotab Trip entity, the Id field is also used to store the *Version* of the entity. As a result, the Id of a given Trip will change with each update. What this means, from a MyGeotab API Adapter perspective, is that **a single physical trip may be represented by multiple records in the Trips table - each with a different GeotabId**. Read below for further details.

Unlike other entity types, with the MyGeotab Trip entity, the Id field is also used to store the *Version* of the entity. As a result, the Id of a given Trip will change with each update. Since the MyGeotab API Adapter can be configured to start at the current time or at some point in the past, there are some implications:

1. When the adapter retrieves data for Trips that have already been completed (i.e. *historical* Trip data), each physical Trip will be represented by one record in the Trips table in the adapter database.
2. For an ongoing Trip that occurs while the adapter is running (i.e. *live* Trip data), each data update will be captured via the data feed and result in a new record being added to the Trips table in the adapter database. Each of the database records for the subject Trip will have a unique *GeotabId*.

The MyGeotab API Adapter - Trips Table - Comparison of Live vs Historical Data [PUBLIC] spreadsheet will be used to illustrate the above-noted points because the Trips table contains too many columns to be adequately displayed directly in this document. Notes about the content of the spreadsheet:

- The ***Live Trip Data - 3 Trips 1 Device*** sheet contains all of the records that were captured for a single device over the course of three trips that started after the MyGeotab API Adapter was already running.
  - Records are sorted in the order they were received.
  - Background colours are used to distinguish between records associated with each of the three Trips.
  - Red text is used to show changes in field values from one row to the next.
- The ***Historic Trip Data - Same 3 Trips*** sheet contains all of the records that were captured for the same three trips during a second run of the MyGeotab API Adapter (using an empty database) after the Trips had been completed.
  - The background colours distinguishing individual Trips match those of the corresponding Trips in the *Live Trip Data - 3 Trips 1 Device* sheet.
  - Each of these historic Trips is represented by a single record which matches the content of the *last* record for the corresponding Trip in the *Live Trip Data - 3 Trips 1 Device* sheet.

Based on the information above, the following sections provide some insights to assist in using the Trips table.

Uniquely Identify Trips

A unique Trip can be identified by using the combination of **DeviceId** and **Start**. Bear in mind that Trips that occur while the MyGeotab API Adapter is running will have multiple records (as updates are reported during the ongoing Trip) - all with the same combination of *DeviceId* and *Start*. Therefore, if consolidating these into an external

system where each Trip has a single record, it will be necessary to use the combination of *DeviceId* and *Start* to determine whether to insert a new Trip or update an existing one.

Identify Completed Trips

A completed trip will have a **StopDurationTicks** value **greater than zero**. This is because the *StopDurationTicks* is the duration (measured in ticks) the vehicle was stopped at the end of a Trip and can only be calculated once the next Trip starts.

Find Driver Associated with LogRecord, StatusData or FaultData

Certain types of objects, such as LogRecord, StatusData and FaultData, contain Device information that can be used to identify the vehicle (using the *VIN*) to which they apply. However, it is often desirable to also know which Driver was associated with the Device at the time. This example demonstrates how data from the Trips table may be used to accomplish this objective.

In this scenario, the StatusData table contains a record with a *DeviceId* of **b9F** and a *DateTime* of **2020-07-20 19:32:07.039Z**. The following query can be used to find the *DriverId* associated with the StatusData record.

Query:

```
select "DriverId" from "Trips"
where "DeviceId" = 'b9F' and "Start" <= '2020-07-20 19:32:07.039Z'
order by "Start" desc
limit 1;
```

Result:

DriverId
b124

Notes about the result:

- The query simply retrieved the *DriverId* of one of the Trip records associated with the current *Trip* at the subject time for the subject *DeviceId*. Since the *Start* value will be the same for all records associated with a single Trip, it is not necessary to complicate the query with any additional columns in the *order by* clause.
- The *DriverId* that was returned can be used to query the Users table and obtain more information about who was driving at the time the StatusData record was reported.

Vehicle Signal Specification (VSS) Add-On

As a member of the Connected Vehicle Systems Alliance (COVESA), Geotab is engaged in a project which facilitates alignment between automakers and data-oriented vendors. To support this initiative, the MyGeotab API Adapter has been modified to include the capability to translate LogRecord and StatusData records into a universal Vehicle Signal Specification (VSS) format and send them to an Open Vehicle Data Set (OVDS) server. This capability will benefit members of the Connected Vehicle Systems Alliance (COVESA) and related

organizations, but will not be of interest to others. As such, it has been integrated in the form of an “Add-On" that separates the code and functionality to the extent possible from the main solution.

For more information, refer to the MyGeotab API Adapter - Vehicle Signal Specification (VSS) Add-On - Solution and Implementation Guide.

Change Log

This section tracks changes to the MyGeotab API Adapter solution over time by version number in reverse chronological order.

Get Notified About New Releases!

Any time a new release of the MyGeotab API Adapter is published to GitHub, an update will be posted to Geotab's Integrator's Hub. Click the *Join Group* button on the page to join and then choose the desired notification frequency (*Every Post*, *Daily Digest*, *Weekly Digest*, etc.)

Version 3.14.0

- Changes other than those listed below apply only when using Data Model 2 (DM2). For more information, refer to the change log in MyGeotab API Adapter DM2 — Solution and Implementation Guide [PUBLIC]
- NOTE: There are **no database schema or configuration file changes from version 3.13.0 to version 3.14.0**. It is safe to upgrade from version 3.13.0 to version 3.14.0 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 3.13.0.
- Updated NuGet packages to the latest stable release.
- Updated version to 3.14.0.0.

Version 3.13.0

- Changes other than those listed below apply only when using Data Model 2 (DM2). For more information, refer to the change log in MyGeotab API Adapter DM2 — Solution and Implementation Guide [PUBLIC]
- NOTE: There are **no database schema or configuration file changes from version 3.12.0 to version 3.13.0**. It is safe to upgrade from version 3.12.0 to version 3.13.0 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 3.12.0.
- Updated NuGet packages to the latest stable release.
- Updated version to 3.13.0.0.

Version 3.12.0

- Changes other than those listed below apply only when using Data Model 2 (DM2). For more information, refer to the change log in MyGeotab API Adapter DM2 — Solution and Implementation Guide [PUBLIC]
- NOTE: There are **no database schema or configuration file changes from version 3.11.0 to version 3.12.0**. It is safe to upgrade from version 3.11.0 to version 3.12.0 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 3.11.0.
- Updated NuGet packages to the latest stable release.
- Updated version to 3.12.0.0.

Version 3.11.0

- Changes other than those listed below apply only when using Data Model 2 (DM2). For more information, refer to the change log in MyGeotab API Adapter DM2 — Solution and Implementation Guide [PUBLIC]
- NOTE: There are **no database schema or configuration file changes from version 3.10.0 to version 3.11.0**. It is safe to upgrade from version 3.10.0 to version 3.11.0 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 3.10.0.
- Modified GeotabTripDbTripObjectMapper (DM1) to ignore Trips with null Device.
- Updated README.md file.
- Updated NuGet packages to the latest stable release.
- Updated version to 3.11.0.0.

Version 3.10.0

- NOTE: This build **includes changes to the schema of the adapter database and to the appsettings.json files**.
    - To upgrade an existing installation of the MyGeotab API Adapter solution from version 3.9.0 to version 3.10.0, see MyGeotab API Adapter — Upgrade Guide — from v3.9.0 to v3.10.0
- Modified the appsettings.json file:
    - Added a section with settings for the new **FuelAndEnergyUsed** feed.
    - Added a "**PopulateEffectOnComponentAndRecommendation**" setting to the FaultData feed section.
    - Made related changes throughout the application.
- Updated NuGet packages to the latest stable release.
    - Geotab.Checkmate.ObjectModel updated from version 11.68.266 to 11.83.265.
- Updated version to 3.10.0.0.

Version 3.9.0

- NOTE: This build **includes changes to the schema of the adapter database and to the appsettings.json files**.
    - To upgrade an existing installation of the MyGeotab API Adapter solution from version 3.8.0 to version 3.9.0, see MyGeotab API Adapter — Upgrade Guide — from v3.8.0 to v3.9.0
- Added Feedback section to the DM2 guide with a link to the MyGeotab API Adapter - Usage Survey which provides a mechanism by which to provide feedback about the MyGeotab API Adapter solution.
- **FIX:** Modified Diagnostics table:
    - Increased length of **DiagnosticName** column from 255 to max to accommodate new Diagnostics with long names on the MyGeotab side.
- Modified ExceptionEvent feed - setting IncludeInvalidated, IncludeDismissedEvents and IncludeDeleted all to true (previously, only IncludeInvalidated was set to true)
- Modified the appsettings.json files (for both API Adapter and Data Optimizer):
    - Renamed "**OverrideSetings**" to "**OverrideSettings**" (to correct typo).
    - Made related changes throughout the applications.
- Added macOS publish profile (to facilitate deployment to macOS systems)
- Updated NuGet packages to the latest stable release.
- Updated version to 3.9.0.0.

Version 3.8.0

- Changes other than those listed below apply only when using Data Model 2 (DM2). For more information, refer to the change log in MyGeotab API Adapter DM2 — Solution and Implementation Guide [PUBLIC]
- NOTE: There are **no database schema or configuration file changes from version 3.7.0 to version 3.8.0**. It is safe to upgrade from version 3.7.0 to version 3.8.0 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 3.7.0.
- Updated NuGet packages to the latest stable release.
- Updated version to 3.8.0.0.

Version 3.7.0

- Changes other than those listed below apply only when using Data Model 2 (DM2). For more information, refer to the change log in MyGeotab API Adapter DM2 — Solution and Implementation Guide [PUBLIC]
- NOTE: There are **no database schema or configuration file changes from version 3.6.0 to version 3.7.0**. It is safe to upgrade from version 3.6.0 to version 3.7.0 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 3.6.0.
- Updated NuGet packages to the latest stable release.
- Updated version to 3.7.0.0.

Version 3.6.0

- Changes other than those listed below apply only when using Data Model 2 (DM2). For more information, refer to the change log in MyGeotab API Adapter DM2 — Solution and Implementation Guide [PUBLIC]
- NOTE: There are **no database schema or configuration file changes from version 3.5.0 to version 3.6.0**. It is safe to upgrade from version 3.5.0 to version 3.6.0 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 3.5.0.
- Enhanced GenericGeotabObjectFeeder:
    - Added Rollback method to simplify updating of LastFeedVersion and LastProcessedFeedVersion properties along with resetting FeedStartOption to ConfiguredFeedStartOption in cases where rollback occurs on the first GetFeed call for a given entity type.
    - Modified services to use the new GenericGeotabObjectFeeder.Rollback method.
- Updated NuGet packages to the latest stable release.
- Updated version to 3.6.0.0.

Version 3.5.0

- Changes other than those listed below apply only when using Data Model 2 (DM2). For more information, refer to the change log in MyGeotab API Adapter DM2 — Solution and Implementation Guide [PUBLIC]
- NOTE: There are **no database schema or configuration file changes from version 3.4.0 to version 3.5.0**. It is safe to upgrade from version 3.4.0 to version 3.5.0 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 3.4.0.
- Updated NuGet packages to the latest stable release.
- Updated version to 3.5.0.0.

Version 3.4.0

- Changes other than those listed below apply only when using Data Model 2 (DM2). For more information, refer to the change log in MyGeotab API Adapter DM2 — Solution and Implementation Guide [PUBLIC]
- NOTE: There are **no database schema or configuration file changes from version 3.3.0 to version 3.4.0**. It is safe to upgrade from version 3.3.0 to version 3.4.0 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 3.3.0.
- Updated NuGet packages to the latest stable release.
    - Geotab.Checkmate.ObjectModel updated from version 11.62.237 to 11.68.266.
- Updated version to 3.4.0.0.

Version 3.3.0

- Changes other than those listed below apply only when using Data Model 2 (DM2). For more information, refer to the change log in MyGeotab API Adapter DM2 — Solution and Implementation Guide [PUBLIC]
- NOTE: There are **no database schema or configuration file changes from version 3.2.0 to version 3.3.0**. It is safe to upgrade from version 3.2.0 to version 3.3.0 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 3.2.0.
- **Updated the solution from .NET 8.0 to .NET 9.0.** If deploying the solution, there are no issues as the solution is self-contained. If copying/cloning the source code to modify the solution, it will be necessary to install .NET 9.0 SDK on the development machine.
- Updated NuGet packages to the latest stable release.
- Updated version to 3.3.0.0.

Version 3.2.0

- Changes other than those listed below apply only when using Data Model 2 (DM2). For more information, refer to the change log in MyGeotab API Adapter DM2 — Solution and Implementation Guide [PUBLIC]
- NOTE: There are **no database schema or configuration file changes from version 3.1.0 to version 3.2.0**. It is safe to upgrade from version 3.1.0 to version 3.2.0 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 3.1.0.
- Updated NuGet packages to the latest stable release.
- Updated version to 3.2.0.0.

Version 3.1.0

- Changes other than those listed below apply only when using Data Model 2 (DM2). For more information, refer to the change log in MyGeotab API Adapter DM2 — Solution and Implementation Guide [PUBLIC]
- NOTE: There are **no database schema or configuration file changes from version 3.0.0 to version 3.1.0**. It is safe to upgrade from version 3.0.0 to version 3.1.0 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 3.0.0.
- Modified **DutyStatusLog** feed to include a DutyStatusLogSearch with the *IncludeModifications* option set to true to include modification history of the DutyStatusLogs.
- Updated NuGet packages to the latest stable release.
- Updated version to 3.1.0.0.

Version 3.0.0

- NOTE: This build **includes changes to the schema of the adapter database and appsettings.json file.**
    - To upgrade an existing installation of the MyGeotab API Adapter solution from version 2.2.0 or greater to version 3.0.0:
        - If using the **original data model,** see MyGeotab API Adapter — Upgrade Guide — from v2.2.0+ to v3.0.0 [PUBLIC].
        - If using **Data Model 2 (DM2)**, see the "We are Already Using the API Adapter. What is the Process to Upgrade to DM2?" section in this guide.

Data Model 2 (DM2)

Version 3.0.0 represents the next evolution of the MyGeotab API Adapter solution. See the IMPORTANT: Version 3.x - Data Model 2 (DM2) section for more information. Key points are as follows:

- A new data model – Data Model 2 (DM2) – has been added:
    - DM2 is normalized and designed for greater performance and scalability.
    - Database is partitioned (monthly, weekly, or daily).
    - Includes Automated Database Maintenance.
    - Support both SQL Server and PostgreSQL.
        - Starting with version 3.0.0, the MyGeotab API Adapter will support SQL Server and PostgreSQL. Oracle database will not be supported moving forward due to very low usage combined with a high cost to develop and maintain.
    - Data Optimizer deprecated – **location interpolation capabilities** moved directly into the adapter database with exponentially faster performance.
- MyGeotab API Adapter supports both the original data model and DM2:
    - Initial version 3.0.0 release of DM2 includes support for a subset of the Geotab entities currently supported with the original data model.
    - Additional entities will be ported over to DM2 in the coming months.
    - Once DM2 supports all of the original data model entities, the original data model will be deprecated with a grace period of 6-12 months to allow integrators to migrate to DM2 if necessary.
    - Refer to FAQ section.

Other Updates

- Added Groups table (to original data model only for this release) and related columns to Devices, Rules, Users and Zones tables. *SQL Server and PostgreSQL only.
- Updated CleanDatabaseScripts (both SQL Server and Postgres) to more efficiently clear data and use system tables/catalog to obtain approximate counts with no concurrency impact during operation of the API Adapter application.
- Updated **DatabaseResilienceHelper** to use ExceptionHelper to include StackTrace in exception logs. Also enhanced internal exception handling logic to include retry for PK constraint violation exceptions (that are due to TX issues and not actually duplicate PKs).
- Modified **BaseRepository** - added **QueryAsync** method for executing parameterized queries (including MSSQL stored procedures and Postgres functions) that return data.
- Added **DatabaseValidator** to validate adapter database version on application startup (DM2 only).
- Modified BaseRepository and GenericEntityPersister to allow for the optional use of **"standalone" database connections** (with no transactions and outside of units of work).
- Added capability for BackgroundServices to pause for database maintenance (DM2 only).

- Added **BackgroundServiceAwaiter** to consolidate wait logic on behalf of individual BackgroundServices and simplified those services (DM2 versions only) accordingly.
- Removed unnecessary trace method entry/exit logging from all BackgroundServices and various other classes.
- Modified **DatabaseResilienceHelper** - added retry for "current transaction aborted" exceptions.
- Modified **StringHelper.IsValidIdentifierForDatabaseObject** method to allow for dashes (as used in weekly partition names).
- Updated **PrerequisiteServiceChecker** and **ServiceTracker** classes with better logging logic for pauses related to waiting for other services (one pause message and one resumption message per paused service).
- Modified **GenericGeotabObjectCacher** to use AddOrUpdate instead of TryAdd to avoid throwing unnecessary exceptions if entities happen to be updated during the process of cache loading via GetFeed.
- Updated Geotab.Checkmate.ObjectModel from version 11.7.179 to 11.62.237.
- Updated all other NuGet packages to the latest stable release.
- Updated version to 3.0.0.0.

Version 2.2.0.2

- NOTE: There are **no database schema or configuration file changes from version 2.2.0 to version 2.2.0.2**. It is safe to upgrade from version 2.2.0 to version 2.2.0.2 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 2.2.0.
- **Modified the optional minimum interval sampling capability for LogRecord and StatusData feeds to accommodate older data:**
    - Primarily designed for use cases such as mining and other remote operations where vehicles may be outside of cellular reception for days or weeks at a time while occasional data points are received via IOX-SATIRDv2.
    - Added GeotabDateTimeProviderComparer to facilitate sorting of Geotab entities based on their DateTime values.
    - Modified MinimumIntervalSampler class to:
        - sort batches of LogRecords or StatusData entities by DateTime prior to applying minimum interval sampling logic.
        - “reset” the tracked DateTime for the subject DeviceId (for LogRecord entities) or DeviceId + DiagnosticId (for StatusData entities) upon encountering older entities within batches being processed.
- Modified DatabaseResilienceHelper to retry on "Connection is busy" exceptions.
- Updated NuGet packages to the latest stable release.
- Updated version to 2.2.0.2.

Version 2.2.0

- NOTE: This build **includes changes to the API Adapter appsettings.json configuration file.**
    - To upgrade an existing installation of the MyGeotab API Adapter solution from version 2.1.4 to version 2.2.0, see the MyGeotab API Adapter — Upgrade Guide — from v2.1.4 to v2.2.0.
- **Added optional minimum interval sampling capability for LogRecord and StatusData feeds:**
    - See the Minimum Interval Sampling section in this guide for more information.

    - Added “EnableMinimunIntervalSamplingForLogRecords”, “EnableMinimunIntervalSamplingForStatusData”, “MinimumIntervalSamplingDiagnostics” and “MinimumIntervalSamplingIntervalSeconds” settings to the API Adapter appsettings.json file.
    - Added MinimumIntervalSampler class.
    - Updated AdapterConfiguration, LogRecordProcessor, StatusDataProcessor, Program.cs and appsettings.json files.
- Updated NuGet packages to the latest stable release.
- Updated version to 2.2.0.

Version 2.1.4

- NOTE: This build **includes changes to the schema of the adapter database and appsettings.json file.**
    - To upgrade an existing installation of the MyGeotab API Adapter solution from version 2.1.3 to version 2.1.4, see the MyGeotab API Adapter — Upgrade Guide — from v2.1.3 to v2.1.4.
- Added a new DutyStatusLogs table to the adapter database along with an associated configurable DutyStatusLogProcessor to provide the ability to extract DutyStatusLog (HOS) data from the MyGeotab database.
- Updated the DatabaseResilienceHelper to handle PostgreSQL-specific connection exceptions with retry logic.
- Updated all other NuGet packages to the latest stable release.
- Updated version to 2.1.4.

Version 2.1.3

- NOTE: There are **no database schema or configuration file changes from version 2.1.2 to version 2.1.3**. It is safe to upgrade from version 2.1.2 to version 2.1.3 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 2.1.2.
- **Updated the solution from .NET 6.0 to .NET 8.0.**
- Removed *ubuntu2004-x64* publish profiles due to breaking change caused by smaller RID graph in .NET 8 (see https://learn.microsoft.com/en-us/dotnet/core/compatibility/sdk/8.0/rid-graph)
- Updated Geotab.Checkmate.ObjectModel from version 11.6.171 to 11.7.179.
- Updated all other NuGet packages to the latest stable release.
- Updated version to 2.1.3.

Version 2.1.2

- NOTE: This build **includes changes to the schema of the adapter database and appsettings.json file.**
    - To upgrade an existing installation of the MyGeotab API Adapter solution from version 2.1.1 to version 2.1.2, see the MyGeotab API Adapter — Upgrade Guide — from v2.1.1 to v2.1.2.
- Added a new ChargeEvents table to the adapter database along with an associated configurable ChargeEventProcessor to provide the ability to extract ChargeEvent data from the MyGeotab database.
- Updated Geotab.Checkmate.ObjectModel from version 11.0.6226 to 11.6.171.
- Updated all other NuGet packages to the latest stable release.
- Updated version to 2.1.2.

Version 2.1.1

- NOTE: This build **includes changes to the API Adapter appsettings.json configuration file.**

    - To upgrade an existing installation of the MyGeotab API Adapter solution (version 2.0.10 or greater) to version 2.1.1, see the MyGeotab API Adapter — Upgrade Guide — from v2.0.10+ to v2.1.1.
- **Added ExcludeDiagnosticsToTrack setting** to appsettings.json to work as-is if set to true and, if set to false, to exclude entities in the DiagnosticsToTrack list.
- Updated DatabaseResilienceHelper to include handling of transient "command is already in progress" exceptions.
- Updated Geotab.Checkmate.ObjectModel from version 11.0.4143 to 11.0.6226.
- Updated all other NuGet packages to the latest stable release.
- Updated version to 2.1.1.

Version 2.1.0

- NOTE: There are **no database schema or configuration file changes from version 2.0.10 to version 2.1.0**. It is safe to upgrade from version 2.0.10 to version 2.1.0 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 2.0.10.
- **Added Bulk Insert, Update and Delete Capabilities for PostgreSQL:**
    - When pairing the MyGeotab API Adapter application with a PostgreSQL database, records are now automatically inserted, updated and deleted using bulk operations. This dramatically increases throughput capability of the system. For example, whereas it may have previously taken 15-20 seconds to insert a batch of 50,000 records into the StatusData table, that same operation might now be accomplished in 1-2 seconds.
- Updated Geotab.Checkmate.ObjectModel from version 11.0.2 to 11.0.4143.
- Updated all other NuGet packages to the latest stable release.
- Updated version to 2.1.0.

Version 2.0.10.2

- NOTE: There are no code or database changes in this release. The purpose of this update is to remove vulnerabilities in dependencies (Microsoft.NETCore.Platforms:7.0.2) by upgrading NuGet packages.
- Updated all other NuGet packages to the latest stable release.
- Updated version to 2.0.10.2.

Version 2.0.10

- NOTE: This build **includes changes to the API Adapter and Data Optimizer appsettings.json configuration files.**
    - To upgrade an existing installation of the MyGeotab API Adapter solution version 2.0.9 to version 2.0.10, see the MyGeotab API Adapter — Upgrade Guide — from v2.0.9 to v2.0.10.
- Bug fix: Modified GenericDbObjectCache interfaces/classes to use AsyncReaderWriterLock instead of SemaphoreSlim to prevent cache reads while updates are in progress. Resolves an issue whereby duplicate records may on rare occasion be inserted into "Reference Data" tables.
- Added new DisableMachineNameValidation setting for both API Adapter and Data Optimizer - to accommodate hosted deployments in environments with non-static machine names.
- Updated Geotab.Checkmate.ObjectModel from version 10.0.1 to 11.0.2.
- Updated NuGet packages to the latest stable release.
- Updated version to v2.0.10.

Version 2.0.9

- NOTE: This build **includes changes to the schema of the adapter database and appsettings.json file.**
  - To upgrade an existing installation of the MyGeotab API Adapter solution (version 2.0.0 or greater) to version 2.0.9, see the MyGeotab API Adapter — Upgrade Guide — from v2.0.0+ to v2.0.9.
- Added a new DebugData table to the adapter database along with an associated configurable DebugDataProcessor to provide the ability to extract DebugData from the MyGeotab database.
- Bug fix: Improved DB object caching logic to resolve an issue that lead to exceptions indicating duplicate items in in-memory caches.
- Added StringHelper class to assist with string comparisons potentially involving null or empty strings and modified object mapper classes to incorporate the StringHelper. This was to resolve an issue that manifested when using the API Adapter in conjunction with an Oracle database.
- Added publish profiles for the MyGeotab API Adapter and Data Optimizer applications for the ubuntu.20.04-x64 runtime.
- Modified reference data processors (Device, Diagnostic, Rule, User, Zone, ZoneType) - Incorporated Dictionaries to speed-up duplicate item prevention checks and dramatically reduce time to write initial caches to database tables.
- Modified GenericGeotabObjectCacher - added logging to show cache loading progress in 10K record increments.
- Updated Geotab.Checkmate.ObjectModel from version 10.0.0 to 10.0.1.
- Updated NuGet packages to the latest stable release.
- Updated version to v2.0.9 (v2.0.6 through v2.0.8 were not published).

Version 2.0.5

- NOTE: There are **no database schema or configuration file changes from version 2.0.0 to version 2.0.5**. It is safe to upgrade from version 2.0.0 to version 2.0.5 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 2.0.0.
- Bug fix: Removed MyGeotabConnectionException.cs from MyGeotab API Adapter project so that the class of the same name in the MyGeotabApiAdapter.Exceptions project gets used instead. Ambiguity was causing issues with exception handling.
- Resolved issue whereby database-level rounding of sub-millisecond DateTime values resulted in equality comparison issues that further resulted in erroneous “duplicate Id” exceptions.
- Modified DatabaseResilienceHelper.CreateAsyncRetryPolicyForDatabaseTransactions method to include Exception and InnerException checks for timeout strings. Previously, command timeouts would not be caught and transactions would not be retried for timeout exceptions.
- Enhanced DatabaseResilienceHelper and MyGeotabAPIResilienceHelper to add handling for additional known exceptions as well as increasing delay between retries (with jitter) for database commands and transactions.
- Updated NuGet packages to the latest stable release.
- Updated version to v2.0.5.

Version 2.0.3

- NOTE: There are **no database schema or configuration file changes from version 2.0.0 to version 2.0.3**. It is safe to upgrade from version 2.0.0 to version 2.0.3 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 2.0.0.

- Bug fix: Resolved issue whereby introduction of new KnownIds for existing Diagnostics would cause the Data Optimizer application to crash.
- Enhanced DatabaseResilienceHelper and MyGeotabAPIResilienceHelper to add handling for additional known exception types. Also added increasing delay with a jitter of up to 1000 milliseconds between retry attempts for database commands and transactions - to avoid potential spikes in contention issues in scenarios where multiple processes are attempting to operate against the same records.
- Enhanced database object caching logic to refresh caches immediately after persisting changes.
- Updated Geotab.Checkmate.ObjectModel from version 9.0.0 to 10.0.0.
- Updated NuGet packages to the latest stable release.
- Changed version from v2.0.2 to v2.0.3.

Version 2.0.2

- NOTE: There are **no database schema or configuration file changes from version 2.0.0 to version 2.0.2**. It is safe to upgrade from version 2.0.0 to version 2.0.2 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 2.0.0.
- Bug fix: Resolved issue where duplicate values could potentially be inserted into “reference data” tables (associated with caches) causing the API Adapter and/or Data Optimizer applications to crash and become inoperable without first removing the duplicate records from the adapter database.
- Changed version from v2.0.1 to v2.0.2.

Version 2.0.1

- NOTE: There are **no database schema or configuration file changes from version 2.0.0 to version 2.0.1**. It is safe to upgrade from version 2.0.0 to version 2.0.1 by simply downloading the new version and overwriting the respective appsettings.json file(s) with those that were configured for version 2.0.0.
- Incorporated Polly to provide timeout and retry capabilities for MyGeotab API calls. Existing known exceptions that indicate an outage or loss of connectivity to the MyGeotab database continue to trigger the connectivity restoration logic. All other exceptions encountered when making calls to the MyGeotab API are now encapsulated in this timeout and retry logic wherein exception messages are written to the log file, but the API calls are retried indefinitely. The objective is to increase resilience to the extent possible and allow the API Adapter to keep running unattended while external transient or connectivity-related exceptions are resolved.
- Updated Geotab.Checkmate.ObjectModel from version 8.0.1 to 9.0.0.
- Updated NuGet packages to the latest stable release.
- Updated version to 2.0.1.

Version 2.0.0

Version 2.0.0 of the MyGeotab API Adapter solution introduces numerous under-the-hood enhancements aimed at dramatically boosting performance, throughput and resiliency while minimizing changes to database structures and application configuration files. Changes included in this version are listed below.

Major Enhancements

Significant enhancements to the MyGeotab API Adapter solution over the previous version are listed below.

Bulk Insert, Update and Delete Capabilities for SQL Server

When pairing the MyGeotab API Adapter and/or Data Optimizer applications with a SQL Server database, records are now automatically inserted, updated and deleted using bulk operations. This dramatically increases throughput capability of the system. For example, whereas it may have previously taken 15-20 seconds to insert a batch of 50,000 records into the StatusData table, that same operation might now be accomplished in 1-2 seconds. When using any of the other supported database types, the traditional insert, update and delete methods are used.

Parallel Services

Previously, the API Adapter application managed all caches and data feeds through a single physical service. While Geotab API calls were executed in parallel, performance of the overall service was limited to that of the slowest data feed. In this new release, each cache or feed processor is separated into its own physical service, providing a much greater degree of parallelism, better performance and options for splitting the application across multiple servers.

Enhanced Resiliency Through Timeout and Retry Policies

With the massive amounts of data being written to databases by the API Adapter and Data Optimizer, transient exceptions may be encountered. These include things such as occasional timed-out database commands or deadlocks when multiple services attempt to operate on the same records at the same time. Additional resilience has been built into the MyGeotab API Adapter solution to handle these sorts of situations - allowing the API Adapter and Data Optimizer to continue running uninterrupted. This has been accomplished through the incorporation of Polly to provide timeout and retry capabilities for database commands and transactions.

Additional Database Indexing

Indexes have been added to the columns in the adapter database tables that are used to identify record age (i.e. "Date", "RecordLastChangedUtc" or "RecordCreationTimeUtc", depending on the table). This was done to facilitate greater performance of deletion operations when moving data to the optimizer database - fueling greater overall performance and throughput. It also serves to benefit those who may be moving data to their own data lakes or external systems, following the Suggested Strategy outlined in the Database Maintenance section of this document.

Enable/Disable Caches

It is now possible to enable or disable individual “cache” services using the Enable<EntityType>Cache settings in the AppSettings - Caches section of the appsettings.json file. This is useful in cases such as the example where there is only an interest in obtaining LogRecords and the Device cache is the only cache that needs to be enabled.

ExceptionEvent State

A State property was recently added to the ExceptionEvent object to provide the ability to know when ExceptionEvents have become invalidated. New *State* and *LastModifiedDateTime* columns have been added to the ExceptionEvents table in the adapter database to capture this information.

Implementation-Related Changes

This section provides information related to changes that may impact existing implementations. These include database schema changes, configuration file changes and source code changes.

Removed SQLite Support

As a result of increased parallelism introduced in this version, the MyGeotab API Adapter can no longer support SQLite as one of the database options as attempts to use it would result in database locking exceptions. See https://www.sqlite.org/whentouse.html for more details on SQLite suitability. Note that SQL Server, PostgreSQL and Oracle databases all have free options available for development/testing purposes.

Adapter Database Schema Changes

- Added OServiceTracking table.
- Modified ExceptionEvents table:
    - Added **State** and **LastModifiedDateTime** columns.
    - Renamed the **RecordCreationTimeUtc** column to **RecordLastChangedUtc**.
- Modified Rules table:
    - Increased length of the **Comment** column to the maximum (or 4000 characters before truncation for Oracle).
- Removed **ConfigFeedVersions** table.
- Removed **vwRuleObject** view.
- Added indexes to the columns in the adapter database tables that are used to identify record age ("Date" for feed type and "RecordLastChangedUtc" or "RecordCreationTimeUtc" for reference and others). This is to facilitate greater performance of delete operations when moving data to the optimizer database - fueling greater overall performance and throughput. (SOLENG-26904). Affected tables and added indexes:
    - BinaryData - IX_BinaryData_DateTime
    - Conditions - IX_Conditions_RecordLastChangedUtc
    - Devices - IX_Devices_RecordLastChangedUtc
    - DeviceStatusInfo - IX_DeviceStatusInfo_RecordLastChangedUtc
    - Diagnostics - IX_Diagnostics_RecordLastChangedUtc
    - DriverChanges - IX_DriverChanges_RecordCreationTimeUtc
    - DutyStatusAvailabilities - IX_DutyStatusAvailabilities_RecordLastChangedUtc
    - DVIRDefectRemarks - IX_DVIRDefectRemarks_RecordLastChangedUtc
    - DVIRDefects - IX_DVIRDefects_RecordLastChangedUtc
    - DVIRDefectUpdates - IX_DVIRDefectUpdates_RecordCreationTimeUtc
    - DVIRLogs - IX_DVIRLogs_DateTime
    - ExceptionEvents - IX_ExceptionEvents_RecordLastChangedUtc
    - FaultData - IX_FaultData_DateTime
    - LogRecords - IX_LogRecords_DateTime
    - MyGeotabVersionInfo - IX_MyGeotabVersionInfo_RecordCreationTimeUtc
    - OServiceTracking - IX_OServiceTracking_RecordLastChangedUtc
    - Rules - IX_Rules_RecordLastChangedUtc
    - StatusData - IX_StatusData_DateTime
    - Trips - IX_Trips_RecordCreationTimeUtc
    - Users - IX_Users_RecordLastChangedUtc
    - Zones - IX_Zones_RecordLastChangedUtc
    - ZoneTypes - IX_ZoneTypes_RecordLastChangedUtc

Configuration File Changes

The following changes were made to the appsettings.json configuration file associated with the MyGeotab API Adapter application:

- Removed SQLite-related settings from the DatabaseSettings selection.
- Added settings to provide to the enabling or disabling of individual caches. These include:
    - EnableControllerCache
    - EnableDeviceCache
    - EnableDiagnosticCache
    - EnableFailureModeCache
    - EnableGroupCache
    - EnableRuleCache
    - EnableUnitOfMeasureCache
    - EnableUserCache
    - EnableZoneCache
    - EnableZoneTypeCache
- Removed the **EnableDVIRDefectCache** and **DVIRDefectListCacheIntervalDailyReferenceStartTimeUTC** settings as they are no longer used within the application.

Source Code Changes

An extensive refactoring exercise was undertaken in order to achieve the Major Enhancements noted above. This resulted in many changes to the source code. A best effort has been made to categorize these changes below:

Unit of Work Pattern

With the introduction of the Data Optimizer in version 1.5.0.0, overall complexity of the solution increased and instead of largely writing data to one database table at a time, it became necessary to execute multiple inserts, updates and deletes in a single “batch”. To facilitate management of database transactions and ensure data integrity, a Unit of Work (UOW) pattern was introduced into the Data Optimizer project within the solution. This UOW approach has now been extended to include the API Adapter project as well.

Adoption of the UOW approach facilitated related changes that have combined to simplify data access. These include:

- Modified the MyGeotabAPIAdapter.Database project:
    - Removed all service classes.
    - Removed all type-specific repository classes and replaced them with a single generic **BaseRepository** class.

Dependency Injection Pattern

In order to simplify the overall solution and facilitate easier modification and enhancement in the future, a Dependency Injection (DI) pattern was implemented. This resulted in numerous additional supporting projects being added to the solution with functionality split-out into smaller classes with associated interfaces.

Geotab Object Processing Services

Previously, there was a single Worker service that handled retrieval and processing of Geotab data for all caches and feeds configured in the appsettings.json file. While data retrieval and write tasks were executed in parallel tasks, the speed of each execution loop of the Worker service was limited by the slowest object type to complete processing. Each Geotab object type, whether classified as “feed type” or “cache type”, now has its own dedicated service that operates autonomously from the others. This provides for a much greater degree of parallelism and throughput.

While these services are able to run independently, there are certain logical interdependencies related to the Geotab objects being processed. For example, the LogRecord has a Device property which indicates which Device a LogRecord is from. Therefore, if LogRecords are being extracted, it is also necessary to extract Devices. As such, the LogRecordProcessor includes logic that prevents it from running if the DeviceProcessor is not running. In the event that the LogRecordProcessor is enabled but the DeviceProcessor is disabled, warning messages are written to the log file to make it abundantly clear that the DeviceProcessor also needs to be enabled. This type of logic is applied across all of the services.

Generic Classes and Methods

In order to reduce duplication of code while increasing reusability and maintainability of the overall solution, generic classes have been developed and incorporated extensively. These include, but are not limited to:

- MyGeotabAPIAdapter Project:
  - GenericGeotabObjectCacher
  - GenericGeotabObjectFeeder
  - GenericGeotabObjectFiltererBase
  - GenericGeotabObjectHydrator
  -
- MyGeotabAPIAdapter.Database Project:
  - AdapterGenericDbObjectCache
  - GenericGenericDbObjectCache
  - GenericGeotabGUIDCacheableDbObjectCache
  - GenericIdCache
  - OptimizerGenericDbObjectCache
  - BaseRepository
  - GenericDatabaseUnitOfWorkContext
- MyGeotabAPIAdapter.Database.EntityPersisters Project:
  - GenericEntityPersister

Singletons

Given the decoupled design inherent in the implementation of the Dependency Injection (DI) pattern and the splitting-out of the Worker service into multiple Geotab object-focused services, there are many instances where classes need to be used by multiple other classes throughout the application. In order to maximize efficiency and avoid unnecessary duplication (and resource utilization), the Singleton pattern has been implemented where warranted. These can be identified via the ".AddSingleton" calls in the *CreateHostBuilder* method of the Program.cs file (in the *MyGeotabAPIAdapter* or the *MyGeotabAPIAdapter.DataOptimizer* projects).

To illustrate, caches of Geotab objects and corresponding database objects for "reference data" types are set-up as singletons. For example, a single *OptimizerGenericDbObjectCache* class is instantiated to hold an in-memory cache of *DbDeviceT* objects. This single instance is then shared by the BinaryDataProcessor, DeviceProcessor, DriverChangeProcessor, FaultDataProcessor, LogRecordProcessor and StatusDataProcessor classes.

Other Source Code Changes

Other source code changes, many of which are related to the changes outlined in the previous sections, include the following:

- Enhanced MyGeotab connectivity loss detection:

    - Included "*ServiceUnavailableException Service temporarily unavailable.*" and "*HttpRequestException Connection refused*" among the error message strings that trigger connectivity loss detection and restoration logic.
- Increased maximum allowed timeout for MyGeotab tasks:
    - The maximum allowed timeout for Geotab API calls has been increased to 3,600 seconds (1 hour) which is much more than enough to accommodate the largest fleets where initial GetFeed calls starting with a *FromDate* may take significantly longer than subsequent *ToVersion*-based GetFeed calls due to user-scoping combined with additional queries that are sometimes needed.
    - Applied the *TimeoutSecondsForMyGeotabTasks* setting value to MyGeotab API calls to override the API object's default timeout of 5 minutes to support the extended maximum timeout.
- Added the MyGeotabAPIAdapter.GeotabObjectMappers project with separate classes for mapping Geotab API objects to corresponding database objects.
- Created a new RuleProcessor service to handle processing of Geotab Rules, replacing the RuleHelper class associated database view.
- Created GeotabDeviceFilterer and GeotabDiagnosticFilterer classes - implemented as singletons.
- Added the GenericGeotabObjectFeeder class which replaces the previous FeedManager and FeedContainer classes.
- Removed the CacheManager class and replaced it with the GenericGeotabObjectCacher, GenericGenericDbObjectCache (and derived AdapterGenericDbObjectCache and OptimizerGenericDbObjectCache classes) and GenericGeotabObjectHydrator classes.
- In the service classes within the MyGeotabAPIAdapter project, moved database persistence logic outside of conditional logic so that the DbOserviceTracking records get updated on each iteration - even if the subject data feeds have no records to persist. The benefit is that persistence logic only needs to be in one place per class.
- Added logic to allow for the possibility of the MyGeotab API Adapter application to be split across multiple servers in a similar fashion to the Data Optimizer (see the Two Application Servers and Two Databases and Three or more Application Servers and Two Databases sections in the *MyGeotab API Adapter - Data Optimizer - Solution and Implementation Guide* for example) with checks in place to enforce a single application version and one instance of each service across all servers.
- Added MyGeotabAPIAdapter.MyGeotabAPI project with MyGeotabAPIHelper class to replace the existing MyGeotabAPIUtility class in the MyGeotabAPIAdapter project. All Geotab API interaction is handled by this new class and its incorporation into a separate project facilitates reuse for other projects.
- Replaced the MyGeotabAPIAdapter.ConfigurationManager class with a new AdapterConfiguration class in the MyGeotabAPIAdapter.Configuration project.

Other Changes

- Updated all database scripts.
- Upgraded Geotab.Checkmate.ObjectModel from v7.0.0 to v8.0.1.
- Added the Service Interdependencies section to this guide.

Version 1.5.5

- Bug fix: Modified SqlMapperExtensions.cs to resolve a bug which causes exceptions to occur when trying to insert records that would result in database-generated Ids larger than the .NET Int32 maximum of 2147483647.
- Enhanced the solution to accommodate Diagnostic "ShimIds" which occur when new Diagnostics with KnownIds are introduced on the MyGeotab server side, but are not yet known on the client side (as may

occur when a newer version of the MyGeotab .NET API client (Geotab.Checkmate.ObjectModel NuGet package) becomes available but the current client is still using an older version). When the local NuGet package is later updated, new database records are added for the subject Diagnostics and they are related back to their former ShimId Diagnostic counterparts via the FormerShimGeotabGUID property. This allows for data captured and associated with either an old or new Diagnostic Id to be related to the single logical Diagnostic.
    - Modified the Diagnostics table:
        - Increased the length of the GeotabGUID column from 36 to 100 (and changed to nvarchar for SQL Server version of the database).
        - Added new HasShimId and FormerShimGeotabGUID columns.
- Updated NuGet packages to the latest stable release.
- Updated version to 1.5.5.

Version 1.5.4

- No changes to API Adapter - Changes limited to Data Optimizer.
- Updated version to 1.5.4.

Version 1.5.3

- Added a semi-autonomous workflow example for deploying the MyGeotab API Adapter to Microsoft's Azure cloud platform.
    - Added the Deploy to Microsoft Azure section to this guide.
- Updated version to 1.5.3.

Version 1.5.2

- Upgraded Geotab.Checkmate.ObjectModel from v6.0.0 to v7.0.0.
- Updated all other NuGet packages to the latest stable release.
- Updated version to 1.5.2.

Version 1.5.1

- Enhanced the solution to provide handling for changes to Diagnostic Ids:
    - Modified the Diagnostics table:
        - Added a new GeotabGUID column.
- Added "*MultipleActiveResultSets=True*" to SQL Server connection strings in *appsettings.json* to facilitate querying the optimizer database while processor and/or optimizer services are running.
- Migrated solution from .NET 5.0 to .NET 6.0.
- Upgraded Geotab.Checkmate.ObjectModel from v5.7.2103.1 to v6.0.0.
- Updated all other NuGet packages to the latest stable release.
- Added "*TrustServerCertificate=True*" to SQL Server connection strings in *appsettings.json* to remedy a connection issue arising after the migration to .NET 6.0.
- Added an HttpHelper class that utilizes an HttpClient to replace the use of the WebClient which is deprecated in .NET 6.0.
- Updated version to 1.5.1.

Version 1.5.0.0

- Added a new Data Optimizer to the solution. See the MyGeotab API Adapter - Data Optimizer - Solution and Implementation Guide for details.
    - Initiated refactoring of the overall solution to incorporate heavy utilization of Dependency Injection (DI) and a Unit Of Work (UOW) pattern for data access. Developers: note that the *MyGeotabAPIAdapter* project within the solution will eventually be refactored to adopt DI and UOW, which will provide for greater performance, maintainability, testability and deployment flexibility.
- Modified the Devices table:
    - Increased the length of the Comment column to 1024 characters.
- Bug fix: Modified ApplyTrackedDevicesFilterToList and ApplyTrackedDiagnosticsFilterToList methods in Worker.cs to handle null device/diagnostic.
- Various minor bug fixes.
- Removed FedRAMP disclaimer from README.md and solution guide.
- Updated version to 1.5.0.0

Version 1.4.0.9

- Added DeviceStatusInfo table to adapter database along with associated scripts and data feed logic.
- Added BinaryData table to adapter database along with associated scripts and data feed logic.
- Made changes to appsettings.json to accommodate configuration of the new feeds.
- Added a Comment column to the Devices table.
- Added logic in VSSObjectMapper.cs to convert boolean strings to lower-case.
- Upgraded Geotab.Checkmate.ObjectModel from v5.7.2101 to v5.7.2103.1.
- Updated all other NuGet packages to the latest stable release.

Version 1.4.0.8

- Added support for Oracle.
    - Modified the Devices table:
        - Increased the length of the Name column from 50 to 100 characters.
        - Made the SerialNumber column nullable.
    - Modified the Users table:
        - Made the FirstName and LastName columns nullable.
    - Modified the Zones table:
        - Increased the length of the Comment column from 255 to 500 characters.
    - Added information related to Oracle support throughout this document.
- Added the Vehicle Signal Specification (VSS) Add-On. Includes:
    - A supplementary guide which is linked in the Vehicle Signal Specification (VSS) Add-On section of this document.
    - The addition of two new tables to the adapter database: *OVDSServerCommands* and *FailedOVDSServerCommands*.
    - The addition of an *AppSettings > AddOns > VSS* section to the appsettings.json file.
- Improved Task cancellation logic.

Version 1.4.0.7

- Added DriverChange table and data feed.
- Added logic to log warning in case of duplicate Id when updating caches.
- Modified exception logging logic to recursively log InnerExceptions.

Version 1.4.0.6

- Updated all other NuGet packages to the latest stable release.

Version 1.4.0.5

- Altered the Users table, making the HosRuleSet column nullable.
- Resolved an issue of device and diagnostic filtering (per appsettings.json configuration) not working.
- Added logic to handle duplicate values being entered in the DevicesToTrack and DiagnosticsToTrack settings in appsettings.config.
- Added device filtering support for DVIRLog and ExceptionEvent feeds.
- Modified FeedStartOption override on startup to be applied on a per-feed basis only if data has already been obtained for the subject feed.
- Fixed a bug in the RuleHelper class.
- Added assembly version information to log files on startup.
- Enhanced exception handling logic:
    - Treat exceptions with "Geotab.Checkmate.Web.WebServerInvoker" in the stack trace as MyGeotab connectivity issues.
    - Added connectivity restoration logic to DVIRLogManipulator and DutyStatusAvailabilityWorker.
- Migrated solution from .NET Core 3.1 to .NET 5.0.
- Upgraded Geotab.Checkmate.ObjectModel from v5.7.2004 to v5.7.2101.
- Updated all other NuGet packages to the latest stable release.

Version 1.4.0.0

- Introduced bidirectional workflow capability via a new DVIRLogManipulator worker service which makes it possible - without directly using the MyGeotab API - to add repair remarks to existing DVIRDefects and change the repair status of existing DVIRDefects. Related changes:
    - Added DVIRDefectUpdates and FailedDVIRDefectUpdates tables to the database.
    - Added AppSettings > Manipulators section to the appsettings.config file.
    - Added DVIRLog Manipulator section to this guide to fully explain what it does and how to use it.
- Removed RepairRemark column from DVIRLogs table (repair remarks are added to the DVIRDefectRemarks table).

Version 1.3.0.0

- Added DutyStatusAvailabilities table to database.
- Added HosRuleSet and LastAccessDate columns to the Users table.
- Added DutyStatusAvailabilityWorker service, which runs in parallel to the main Worker service (if the DutyStatusAvailability feed is enabled) to keep the DutyStatusAvailabilities table updated through a longer-running pseudo-feed without introducing delay to the processing of the regular feeds.

Version 1.2.0.1

- Added ZonesTypes table to database.
- Added ZoneTypeIds column to the Zones table.
- Added available but previously not included fields to the Trips table.
- Updated the Using Trip Information section in this document.
- Enhanced Rule edit processing logic to delete and replace associated Conditions table records so as to ensure Conditions are always accurately reflected for a given Rule).
- Resolved an issue in *ObjectMapper.cs* whereby methods determining whether a database object requires update would prematurely return false under certain conditions without all properties having been evaluated.
- Resolved an issue whereby records in Reference data tables were not being updated upon detecting changes in Geotab objects due to the existing database *id* values not being applied when converting the Geotab objects to their counterpart database objects.
- Modified SQL Server version of *CleanDatabaseScript.sql* to provide for rapid record counts while the adapter is running and populating the database.
- Updated Checkmate package to latest version (5.7.2004).

Version 1.2.0.0

- Added new *id* columns of the bigint data type to various tables in the adapter database and set these *id* columns to be primary keys with auto-incrementing values. Renamed existing *Id* columns to *GeotabId*. The purpose of this change is to facilitate deletion of records as part of the recommended database maintenance strategy while the adapter is writing new records to the same tables at the same time (i.e. to avoid database concurrency conflicts). The *GeotabId* values must be maintained in order to relate records back to objects in the Geotab platform (i.e. the new *id* values are purely a construct of the MyGeotab API Adapter solution and have no relation to IDs of objects in MyGeotab databases).
- Added a *Points* column to the Zones table to include all points comprising the zone geometry in a JSON array.

Version 1.1.0.1

- Enhanced logic to process feed results immediately upon receipt on a per-feed basis - resulting in the observed ability to process the same data approximately 43% faster.
- Enhanced database connectivity issue handling logic.

Version 1.1.0.0

- Added support for SQL Server.
    - Modified the ExceptionEvents table:
        - Replaced the *Duration* column with the *DurationTicks* column.
    - Modified the Trips table:
        - Replaced the *DrivingDuration* column with the *DrivingDurationTicks* column.
        - Replaced the *StopDuration* column with the *StopDurationTicks* column.
    - Added information related to SQL Server support throughout this document.
- Updates to facilitate retrieval of OBD-II Diagnostic Trouble Codes (DTCs) and J1939 Fault Mode Indicators (FMIs) & Suspect Parameter Numbers (SPNs) associated with FaultData records.
    - Includes the addition of fields to the Diagnostics and FaultData tables.

    - Added the Obtain OBD-II DTC From FaultData and Obtain J1939 FMI and SPN From FaultData sections to this document.

Version 1.0.0.8

- Added Zones table to database along with propagation-related code. Added Zones table information to the Data Dictionary.
- Updated Checkmate package to latest version

Version 1.0.0.7

- Initial post-beta release.