Volpe CVP Safety Evaluation

Technical / Installation document

November 2019

Authored by: Acrotron, Bjorn Kuiper

The latest version of this document is stored on GitHub within the following project:

<https://github.com/VolpeUSDOT/cvp-safety-sql-plugin>

# Document history

|  |  |  |  |
| --- | --- | --- | --- |
| Date (yyyy-mm-dd) | Name | Version | Description |
| 2019-11-08 | Bjorn Kuiper | 1.0 | Initial document. |
| 2019-11-26 | Bjorn Kuiper | 1.1 | Added information about Web prototype. |

# Introduction

As part of the VCSE effort we have developed a SQL CLR DLL that can be loaded into a Microsoft SQL Server to help execute the Kinematic Equations, as defined in the CVP Data Processing Technical Memorandum, against the different datasets within the CVP project.

# Intended Audience

The intended audience are software developers and people responsible for maintaining the VCSE (Volpe CVP Safety Evaluation) SQL CLR DLL. The audience is expected to be familiar with GitHub and how to clone and work on a repository and how to use Visual Studio 2019 and navigate the codebase within Visual Studio.

# Acronyms

|  |  |
| --- | --- |
| VCSE | Volpe CVP Safety Evaluation |
| CVP | Connected Vehicle Pilot |
| SQL | Structured Query Language |
| CLR | Common Language Runtime |
| DLL | Dynamic Link Library |

# Technical overview

This project outcome consists of an SQL CLR DLL that can be loaded into Microsoft SQL Server instance. The SQL CLR DLL exposes SQL functions that can be used within the Microsoft SQL Server instance. The functions are based on the kinematic equations that are defined in the CVP Data Processing Technical Memorandum. The source-code is written in C# .NET Framework and stored in GitHub as part of the Volpe US DOT GitHub account.

# Source-code

The latest version of the source-code can be found on the Volpe US DOT GitHub account at:

<https://github.com/VolpeUSDOT/cvp-safety-sql-plugin>

***NOTE: this is a private repository, thus only visible to project members.***

It consists of a C# .NET Framework solution.

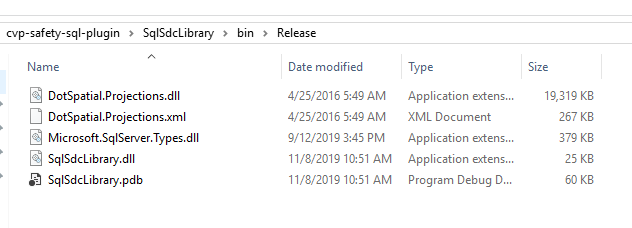
The kinematic equations are implemented in Functions.cs class and then exposed to SQL through the SqlFunctions.cs class, both within the SqlSdcLibrary solution. Testing of the functions is done through the SqlSdcLibrary.Specs and SqlSdcLibrary.Test projects. The SqlSdcLibrary.Specs project uses SpecFlow (<https://specflow.org/>) for binding Business requirements to .NET code.

Example queries build against specific data sets can be found in the sql-queries folder, together with the register.sql query which helps registering the SQL CLR DLL with Microsoft SQL Server instance.

# Building the solution

The solution is opened through the *SqlSdcLibrary.sln* within the SqlSdcLibrary folder. The solution is developed in Visual Studio 2019 against .NET Framework v4.7.2. The solution is built with the Debug or Release configuration against Platform “Any CPU” with the *SqlSdcLibrary* project as the Startup project. The resulting DLLs can be used to register against an SQL Server instance:

* DotSpatial.Projection.dll
* Microsoft.SqlServer.Types.dll
* SqlSdcLibrary.dll



# Deploying the CLR DLL

Deploying the CLR DLLs consists of copying the associated DLLs to a folder on the SQL Server machine. The folder should be accessible to the SQL Server instance. We would recommend creating a *C:\sql-clr-dll\* folder and change the permissions on the folder to allow for full access by the SQL Server Instance. Then open SQL Server Management studio on the SQL Server machine and open the register.sql query that is provided within the GitHub repository. Update the path within the register.sql when opened in SQL Server Management Studio and make sure you are targeting the correct database. You could do this by additionally adding a “USE <databasename>” at the start of the register.sql.

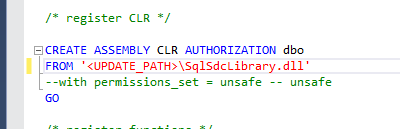


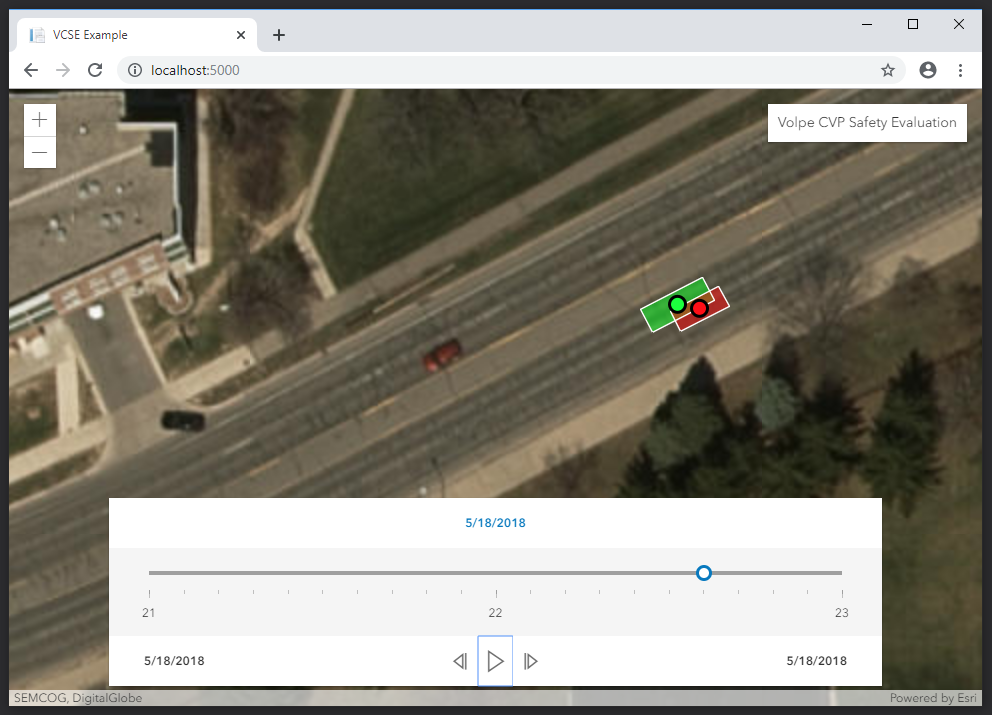
Figure 1: The path to update within the register.sql script.

You can verify the registration of the DLL by running the following SQL command to show the current version of the DLL:

|  |
| --- |
| SELECT dbo.SqlClrDllVersion() |

# Web prototype

The GitHub project also contains an ASP .NET Core 3.x prototype for displaying a single event from the database on a map. The solution can be found in the WebPrototype folder, named Prototype.sln.



It is driven by the wwwroot\website.htm page and use the DbController to retrieve data from the database through an API (AJAX) call from within the website.htm. The website.htm uses the Esri Javascript API 4.13.

<end of document>