

VS SDK

The VehicleSim Software Development Kit

Introduction	1
Documentation	2
Solvers.....	2
Libraries	3
Utilities	6
Configuration.....	7

Introduction

Mechanical Simulation Corporation produces and distributes software tools for simulating and analyzing the dynamic behavior of motor vehicles in response to inputs from steering, braking, throttle, road, and aerodynamics. The simulation packages are organized into families of products named BikeSim®, CarSim®, SuspensionSim®, and TruckSim®. All are based on the simulation architecture named VehicleSim®.

A *VS Solver* is a program in a VehicleSim product that reads input files, writes output files, and calculates variables from an internal math model. Although each specific VS Solver is intended to represent a different kind of vehicle or other simulated system, all operate the same way when it comes to reading and writing files, solving differential equations, and communicating with other software.

In general, a VS Solver is initialized and controlled using the *VS Browser*, a GUI application that manages a database of vehicle and test data and runs simulation models. However, the VS Browser is not the only method of interacting with the VehicleSim simulation architecture. The *VS SDK* is a collection of useful tools that enable you to do just that.

The VS SDK is a *Software Development Kit*. This means that it includes all the tools, libraries, documentation, and example projects necessary to get working on a project with as little configuration possible. With nothing but the contents of the SDK, a valid license for a VehicleSim product, and a development environment for your language of choice you can get to work building and extending various aspects of the simulation architecture.

Example Source code and projects are included for the various libraries and APIS bundled within the VS SDK. The use and operation of these are not detailed within this manual, but instead the documents related to each of those libraries or examples individually.

To obtain a copy of the VS SDK, visit the User Section of the CarSim website. The URL for the download is https://www.carsim.com/users/vs_sdk/vs_sdk_download.php.

Documentation

In order to get the most out of the included features of the VS SDK a collection of relevant documentation has been included for reference.

All the documentation relevant to contents of the VS SDK can be found within the “Documentation” folder at the root level of the unpacked VS SDK. Within this directory can be found various subdirectories that organize the documentation into areas of interest.

General

This directory contains general documentation regarding information not pertaining to any one part of the VS SDK or VehicleSim software suite, but instead important information that will be useful regardless of your intent with the VS SDK contents.

Such documents include Licensing Agreements, System Requirements, a glossary of vehicle dynamics terms, and more.

It is highly recommended that you read and familiarize yourself with the contents of all the documents within this directory before beginning to use the VS SDK.

Libraries

This directory contains documentation relating to the libraries and APIs included with the VS SDK.

Use of the libraries without the proper documentation will prove difficult. It is recommended that you read through and follow the examples contained within the documents for each API you plan to use.

If the APIs are updated in a future version of the VS SDK the explanations of new features and changes pertaining to that API can be found in the document for that API.

Licensing

This directory contains documentation which describes how to get your system up and running with a license. Without a license the VS Solver will not run.

The documents contained within can assist in setting up licensing from a dongle, a network server, or a node-lock license file.

Solvers






This directory contains information describing the use of the VS Solvers and topics related to it.

It is highly recommended that all users of the VS SDK be familiar with the contents of these documents, as the intricacies of the VS Solver may be required knowledge depending on the goals of the programmer.

Solvers

Within the “Solvers” folder of the parent directory of the VS SDK exist a collection of solvers for different platforms.

The solvers in the VS SDK are included for the following platforms.

	CarSim	TruckSim	BikeSim
Windows 32bit (x86) and 64bit (x64)			
Ubuntu, CentOS			

When running under a Microsoft Windows operating system, the VS Solver is a dynamically linked library (DLL) file with a set of VS API functions.

Note The acronym DLL is used throughout this document to refer to a dynamically linked library on any operating system, even though the extension might not have the standard Windows OS extension `.dll`. For example, on Linux system, a DLL is called a “shared object” and is identified with the file extension `.so`.

The solvers are found within a logical hierarchy of directories, each one more specific than its parent as to which version/product is of interest.

Libraries

Perhaps the most important aspect of the VS SDK is the inclusion of various libraries that provide access to functions and data from within VehicleSim. These libraries each serve a distinctly different purpose and can be used by themselves or alongside one another to create powerful automation tasks and projects centered around the VehicleSim family of products.

In addition to the libraries themselves, example projects have been included to help get you started with some of the common features of the libraries. The use of these examples is explained in the documentation associated with each library in the “Documentation” folder within the VS SDK.

VS API

The VS API is a set of functions that allow for direct interaction with a VS Solver instance. An incredibly powerful tool, it can be used to extend or wrap the solver using several important methods:

- Adding equations at runtime, through the employment of VS Commands
- Exchange of internal and external variables into the simulation

- Direct access of variables within the VS Math Model. Operations performed on these variables within the wrapper program can then be integrated with the calculations performed within the VS Math Model.
- Saving and restoring VS Math Model state
- Applying callbacks to the VS Math Model

Versions of the API are included with the VS SDK for its use in C/C++, MATLAB, Python, and Visual Basic application. Each supported language comes with a collection of examples that demonstrate their use.

In general, each language comes with an example showcasing the VS API in use for generating a simple run as well as creating a steering controller. Careful study of the examples for your language will help when starting your own projects. Examples exist for CarSim, TruckSim, and BikeSim in the Examples directory of the API.

To use the `Run_all.par` and `simfile`, you must use the correct executable/script alongside these two files found within the “Examples” directory (e.g., `C\Examples`). The executables and scripts can be found within either the “Code” or “Projects” directories for each language. Proper configuration is detailed within the documentation for each API.

The folder “Consolidated_Parsfiles” contains `.cpar` archives for CarSim, TruckSim, and BikeSim. These `.cpar` files contain files and database examples for use with the VS API. To run the examples contained within the `.cpar` files, a Windows installation of CarSim, TruckSim, and/or BikeSim must be on your computer. The `.cpar` files would then either be imported into an existing database, or the `.cpar` files can be used to create new databases. In the latter case, the new databases would contain only the examples in the `.cpar` files.

VS Connect API

The VS Connect API is a library used for the facilitation of communications between external programs and VehicleSim products. The primary use for it is the synchronization of simulations between multiple programs for co-simulation. These programs may be on a single computer or separated by a network.

The VS Connect API is a C library, and as such can only be used in C/C++ applications without effort. It is possible to wrap the API in the language of your choosing, however that is an advanced technique outside the scope of this manual. An S-function has been provided that creates a VS Connect client within MATLAB/Simulink.

Two examples are included with the API. One is a project that gives an example of a simple VS Connect Server, and the other project showcases a client implementation. Both can be used as starting points for development of your own projects.

Further information on the API, as well as descriptions of the example projects can be found within the documentation for the VS Connect API bundled with the VS SDK.

VS Shared Buffer API

The VS Shared Buffer API facilitates interprocess transfer of image data. Utilizing this API, VS Visualizer provides rendered data (color, depth, and surface normal vectors) to the VS Camera Sensor S-function within Simulink. VS Visualizer uses this API to create shared buffers and update

them continuously with data. The VS Camera Sensor S-function uses this API to access and read the data which VS Visualizer is sharing.

Using this API, custom applications can be written to either provide shared data to other processes, read shared data provided by other processes, or both. The reader could, for example, use this API to write their own application which reads and processes image data generated by VS Visualizer.

For more detailed information about the available features, see the header files in the include directory.

VS Output API

Upon the completion of a simulation the results are compiled into a header and a data file. The header files (.vs or .erd) are used to define the parameters which have been output, and the data files (.vsb or .bin) contain the values to those defined parameters.

Although several tools exist to consume and visualize this data, it is likely that some need may arise wherein manipulating the output data programmatically is required. For this purpose, the VS Output API was created.

The VS Output API allows a programmer to access, edit, and create VS Output data for specialized use in external projects.

Instructions for use, as well as descriptions of the various output file formats, can be found within the documentation for the VS Output API.

VS Table API

A major component of many VehicleSim simulations is the use of nonlinear relationships represented within a table structure. These relationships are stored in a standard VS Table Format within a table file (.vstb).

In order to access and manipulate the data within one of these tables programmatically, the VS Table API can be used. The API allows for creating, editing, and accessing both 1-D and 2-D tables.

Although Tables can exist through VS Command definitions, if a table is required with very large amounts of data, it is recommended to use the VS Table API.

Descriptions of the VS Table API use can be found in the documentation for the API. This documentation is located within the same document as the VS Output API.

VS Vehicle Module

The VS Vehicle module is a wrapper for the VehicleSim solvers' native interface (the VS API). It is intended to simplify the integration of VehicleSim solvers into other simulation environments. VS Vehicle provides a high-level interface to the programmer while automatically handling the intricacies of the VS API. VS Vehicle also facilitates simulating multiple vehicle instances, using one or more solvers, within the same simulation environment.

VS Terrain Library

The VS Terrain Library is a standalone Windows DLL that can be used to generate VS Terrain files. Geometry can be fed in from the source application, attaching friction and rolling resistance

values to each triangle face. This data can then be optimized and built to be used directly by the VS Solvers. An example python script is included in the SDK, and the VS Terrain document includes an example written in C.

Utilities

Included with the VS SDK are a collection of utilities to make interacting with the VehicleSim products as simple as possible. Most of these are also bundled with a full installation of a VehicleSim product. Their inclusion here is so that, unless necessary, a copy of a VehicleSim browser is not required.

HostID

The HostID application is used for determining the unique Host ID of a machine that will be using a VS Solver. This is only important when attempting to retrieve a Node-Lock License.

Instructions for use will be provided during correspondence with Mechanical Simulation when requesting a Node-Lock License.

LicenseManager

Multiple License Managers have been included with the VS SDK for use with a VS Solver. Generally, the licensing is managed by a license manager embedded within an instance of the VS Browser, however the use of the VS SDK does not require this.

Once your licensing has been properly configured, as described in the Licensing Documentation, the proper license manager must be active before attempting to run the VS Solver.

SolverWrapper

In order to facilitate development of applications that interact with and consume the output of a VS Solver, a command-line wrapper has been provided for basic use.

This wrapper application loads a VS Solver dll and uses it to perform a simulation, given a valid simulation input.

Instructions for its use can be found by executing the following command within the directory of the solver wrapper:

On Windows:

```
VS_SolverWrapper_CLI_32.exe -help
```

On Linux:

```
./VS_SolverWrapper_CLI.bin -help
```

TableTool

The `vs_table_tool` utility provides a way for users to create VS Table header and binary files. VS Table files provide binary table data to the VS Solver; which is useful in situations where the table

data is large or it is impractical to add the table using the GUI. Additional documentation is provided in the Documentation folder.

VS Terrain Utility

It is provided so you can create and test drivable surfaces from your own shape files. Additional documentation is provided in the Documentation folder.

Configuration

Although we have tried to make the SDK as easy to use and “Out of the box” as possible, some configuration *may* be required depending on your environment and wishes. This chapter attempts to enumerate these as well as possible.

x86 and x64

On Windows, we provide solvers that are compatible with x86 and x64 systems. It is recommended that if you are running an x64 system that you should use that solver. The examples are set up to use the x64 solvers, and thus must be modified to work with the x86 solvers.

In order to make the modifications, all that must be done is to edit the simfile for the relevant example. Simply open the simfile in your editor of choice and edit the line with the keyword DLLFILE to point to the solver you would like to use.

Steer Controller for TruckSim

One of the examples bundled with the VS API library is a steer controller, which can be used with both CarSim and TruckSim. In order to use this with TruckSim properly however, a small edit must be made.

The controller uses different values for some of the parameters to work properly with TruckSim. The settings in the files are defaulted for CarSim operation. The parameters involved are L_FORWARD, LAT_TRACK, and GAIN_STEER_CNTRL. Look at the comment area near where these parameters are set to get the correct values for the TruckSim Steer Controller. Similarly, these values (or equivalent values) need to be changed when using the Simulink or Visual Basic Steer Controllers for TruckSim as well. The C files where these settings are made and the comment area can be found are external_steer_control.c and solver_vs_cmd.c.

Parameters for getting the Steer Controller to work with BikeSim are also provided, however, the preferred code would be to use the Lean Controller, which was created to be used with BikeSim. The Steer Controller and Lean Controller are very similar in operation and use the same basic equations.