VS Connect S-function

This memo describes the VS Connect Simulink block, which uses the VS Connect s-function to provide inter-process communication, co-simulation, and time synchronization within Simulink. This block is available in the s-functions provided with BikeSim, CarSim, and TruckSim. Figure 1 shows the block provided with CarSim.

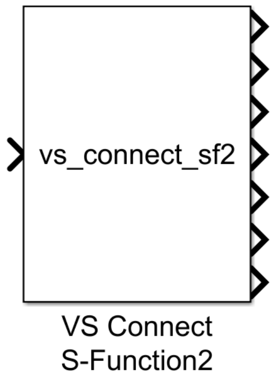


Figure 1. Simulink s-function block using the VS Connect MEX file.

The VS Connect S-function is located within the MEX files `vs_connect_sf2.mexw32` and `vs_connect_sf2.mexw64`, which can be found in the CarSim installation directory (`CarSim_Prog\Programs\solvers\Matlab84+`). These MEX files depend on VS Connect DLL's vs_connect_32.dll, and vs_connect_64.dll, respectively, which can also be found in the CarSim installation directory (`CarSim_Prog\Programs`).

Related Material

The following documents contain additional material related to VS Connect and can be found on the CarSim or TruckSim **Help** menu:

- Help > Reference Manuals > VS Connect API
- Help > Technical Memos > VS Connect S-Function
- Help > Guides and Tutorials > VehicleSim Dynamics plugin for Unreal

Mechanical Simulation provides some VS Connect enabled programs that can be communicated with using VS Connect, including:

- **VS Connect S-function for MATLAB/Simulink -** Provides a VS Connect client block within MATLAB/Simulink that can be configured with a JSON configuration file to specify the data to be sent and received via VS Connect. This can be found in the vs_connect section of the VS SDK: `VS_SDK\Libraries\vs_connect\bin`

- **Example C code projects** – Example projects using the VS Connect API can be found in the VS SDK as well:

  `VS_SDK\Libraries\vs_connect\Example`

- **VehicleSim Dynamics plugin for Unreal** - Provides both a VS Connect server, and facilities to integrate CarSim and/or TruckSim vehicle dynamics solvers into an Unreal Engine simulation. The plugin, example projects, and additional documentation can be accessed from the Unreal Marketplace:

  https://www.unrealengine.com/marketplace/carsim-vehicle-dynamics

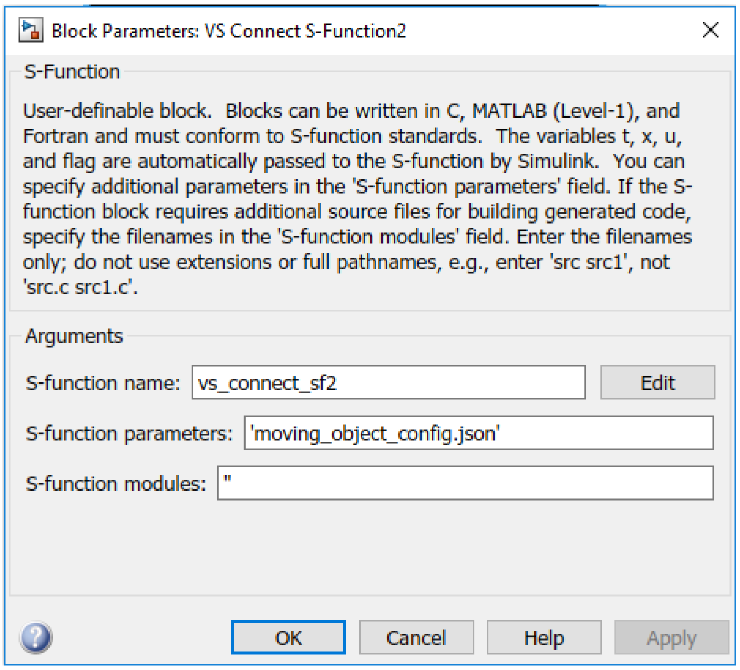
Note that the VS Connect S-function for Simulink and the VehicleSim Dynamics Plugin for Unreal Engine facilitate communication, co-simulation, and time synchronization between MATLAB/Simulink and Unreal Engine without the need for custom C programming.

Operation Overview

The s-function acts as a client which connects to a VS Connect server (e.g. the Unreal plugin). VS Connect utilizes the UDP network communications protocol to send information back and forth. This allows the s-function and server to exist on the same computer, or on different computers.

S-function Block Parameters

To use the VS Connect block, specify the name of a JSON configuration file enclosed in single quotes in the `S-function parameters` field of the Block Parameters dialog (Figure 2).



Block Parameters: VS Connect S-Function2

S-Function

User-definable block. Blocks can be written in C, MATLAB (Level-1), and Fortran and must conform to S-function standards. The variables t, x, u, and flag are automatically passed to the S-function by Simulink. You can specify additional parameters in the 'S-function parameters' field. If the S-function block requires additional source files for building generated code, specify the filenames in the 'S-function modules' field. Enter the filenames only; do not use extensions or full pathnames, e.g., enter 'src src1', not 'src.c src1.c'.

Arguments

S-function name: vs_connect_sf2

S-function parameters: 'moving_object_config.json'

S-function modules: ''

Figure 2. CarSim VS Connect Block Parameters.

S-function Input and Output Ports

The first input port will always be the Connection Directive (Figure 3). If this is zero, the block will disconnect from the peer (if currently connected) and remain unconnected. If this port is non-zero, the block will remain connected to the remote server, and perpetually attempt to reconnect if it becomes disconnected. If this port is not connected, it is treated as if its value is always 1 (the block will always attempt to connect/reconnect and stay connected).

The first output port will always be the Connection Status, an integer value of the connection status between the s-function and the server.

The following are the descriptions for each connection status value.

0 = Status unknown/undefined

1 = Unconnected (not trying to connect)

2 = Attempting to connect, or in the process of negotiating the connection

3 = Connected

4 = Error state (Not connected, not trying to connect)

5 = In the process of disconnecting

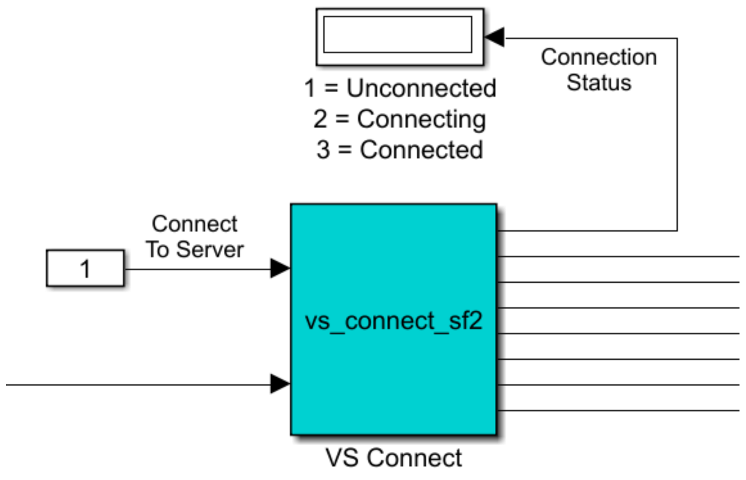


Figure 3. Connect Directive input and Connection Status output.

There may be any number of additional input and/or output ports, which are dynamically created and configured based on the contents of the configuration file.

Configuration File Setup and Parameters

The config file for a VS Connect S-function is a JSON format file. JSON files can be viewed and edited in any text editor. The configuration file specifies the connection parameters for the remote peer (a VS Connect server). The configuration file also defines the incoming and outgoing simulation data (VS Connect Contracts) that will be exchanged with the remote peer, and the corresponding s-function ports which connect these to the rest of the Simulink model.

An example Simulink model and configuration file can be accessed via the Unreal Marketplace page for the VehicleSim Dynamics plugin:

https://unrealengine.com/marketplace/carsim-vehicle-dynamics

Configuration File Structure

JSON uses *objects* and *arrays* to organize data. See http://json.org for details. Listing 1 describes the structure of the configuration file. For this listing, note the following:

- "Comment" text beginning with "←" indicates the structure that opening delimiter ({, [ ) is introducing.
- Text enclosed in brackets **<*like this*>** indicates the types of parameters that can be included at that location.
- Locations where the preceding structure can be repeated are indicated with "**<…>**".

Listing 1. S-function configuration file structure.

```
{ 🡐 Top-level JSON document object.
  "version": 2.0,
  <S-function settings>
  <connection details>

  "inputs": [ 🡐 Array of "port groups".
    { 🡐 Port group object.
      <port group parameters, scheduling details>
      "ports": [ 🡐 Array of "ports".
        [ 🡐 "Port" object.
          { 🡐 "Field" object.
            <Field details>
          }
          <...>
        ]
        <...>
      ]
      <...>
    }
    <...>
  ],
  "outputs": [ 🡐 Array of "port groups".
    { 🡐 Port group object.
      <port group parameters, scheduling details>
      "ports": [ 🡐 Array of "ports".
        [ 🡐 "Port" object.
          { 🡐 "Field" object.
            <Field details>
          }
          <...>
        ]
        <...>
      ]
      <...>
    }
    <...>
  ]}
```

The top-level **JSON Document Object** contains configuration information such as S-function settings (e.g. timeDilation), and the connection details that identify how to connect to the remote VS Connect Server. This section may also include an “inputs” and/or “outputs” array.

The **“inputs”** and **“outputs”** arrays contain one or more **Port Groups**.

Port Groups contain an array of one or more ports and the parameters that are common to them, including scheduling data that says how frequently the data should be updated. Port Groups can be one of two types: `"contract"`, for Port Groups that convey data from the simulations, or `"commData"` for groups that convey only data about the networking/communication system itself. Port Groups of type `contract` produce VS Connect Contracts, while those of type `commData` do not.

Ports may contain an array of one or more **Field** objects. Ports are a convenience that allow related fields to be combined into a single block output in the Simulink model.

Fields identify what the individual elements of data are.

Table 1. S-function settings and connection details

Name	**Data Type**	**Description**
`version`	float	Config file format version. This should be set to 2 for this version.
`timeDilation`	number	Controls throttling of the Simulink simulation rate. timeDilation less than or equal to 0 will run Simulink without any time throttling.
`remoteIP`	string	VS Connect server IP address. By default, it is “localhost” aka “127.0.0.1”.
`remotePort`	integer	VS Connect server port. By default, it is 4367.
`inputs`	array	Array of groups of inputs to the s-function, which correspond to VS Connect Outgoing Contracts (data sent to the remote VS Connect server).
`outputs`	array	Array of groups of outputs from the s-function, which correspond to VS Connect Incoming Contracts (data received from the remote VS Connect server).
`remoteObject`	string	Default remote object name. This is used for individual field parameters if they do not define their own remoteObject.

Table 2. Port group parameters, scheduling details

Name	**Data Type**	**Description**
`groupType`	`"contract"` or `"commData"`	Optional, defaults to `"contract".` Specifies the class of data contained in the enclosed group of inputs/outputs.
`updatePeriod`	number	Rate at which the data should be updated, in seconds, e.g. 0.05 (50 milliseconds, 20 Hz).
`updatePeriodMs`	number	Rate at which the data should be updated, in milliseconds, e.g. 8.333333 (120 Hz).
`timeSyncMode`	string	The time synchronization mode. Valid values include `"absolute"`. and `"none"`. Default is `"none"`.
`overdueDataTimeoutMs`	number	How long (milliseconds) to wait for expected data before considering the associated Contract, Link, and Node blocked.
`blockTimeThresholdMs` (deprecated)		
`overdueDataTimeout`	number	How long (seconds) to wait for expected data before considering the associated Contract, Link, and Node blocked.
`blockTimeThreshold` (deprecated)		
`ports`	array	Array of ports. Each item represents an input or output port in the s-function block.

Table 3. Field details

Name	Data Type	Description
`remoteObject`	string	The remoteObject that this field is associated with. If this is omitted, the top-level remoteObject is used.
`propertyName`	string	proptertyName is a generic representation of a particular variable, function, or solver item.
`propertyType`	`"contract_field"` or `"meta"`	Optional, defaults to `"contract_field"`. for groups of type `"contract"`, and `"meta"` for groups of type `"commData"`.
`dataType`	string	Only available as float type.
`dataSize`	integer	Size of the data of the field parameter.
`arraySize`	integer	
`initialValue`	number	The value reported from an output before valid data has been received. Default is 0.0.
`invalidValue`	number	The value when no valid data is available. Default is 0.0.
`params`	string	A parameter passed to a function. E.g. if remoteObject is “VS Vehicle 0”, params is “Moving_Car”, and propertyName is “VSAP_DISTANCE”. VSAP_DISTANCE points to a function that calculates the distance between the remoteObject and the params (Moving_Car).

Table 4. Valid `propertyName`’s for `meta` fields

propertyName	Required groupType	Description
`time`	`contract`	The remote simulation time that the associated data was generated.
`receive_time_offset`	`contract`	The difference between the local simulation time that the associated data was received and the remote simulation time that it was generated.
`serial_number`	`contract`	The serial number of the latest incoming data update received.
`ping_results`	`commData`	Detailed ping results of the most recently received ping response.

Inter-process Synchronization

A VS Connect server that the s-function can connect to can be built by the user. The custom server code defines what data it can accept (from the s-function block inputs), and what data it can provide (to be output from the s-function block). The s-function’s configuration file specifies which inputs and outputs it is interested in, and the frequency of updates.

A simple server program example (with source code) is included in the VS SDK:

```
VS_SDK\Libraries\vs_connect\Example
```

VS Connect is a generic communication system that enables the VS Connect s-function to communicate with any VS Connect enabled application. The s-function provides a mechanism to send any data from the Simulink model to the remote peer, call functions in the remote environment, and integrate data from the remote environment back into the Simulink model.

In this document, the **VehicleSim Dynamics Plugin for Unreal Engine** ("the plugin"), with its embedded **CarSim** solver, will serve as an example peer that the s-function can communicate with.

The plugin is also an example of a VS Connect Server. Any number of data pointers can be defined within the plugin. These pointers can represent data available in the Unreal Engine simulation environment or variables that exist in the CarSim vehicle solver. They can be simple variables or functions that return multiple variables, such as the vehicle position defined by X, Y, and Z.

With asynchronous communication between different applications, there will likely be differences in the update rates that each client can process information. The Simulink model and CarSim solver may be able to run at 1000 Hz, but an Unreal Engine application may run at a rate that is much slower. The Unreal Engine refresh rate otherwise known as the FPS (frames per second) is how quickly a computer can process and render a video frame. The refresh rate at which an Unreal Engine simulation instance runs will always depend on how it was created, how many special effects are implemented, the number of actors, objects, and polygon counts on models.

Note The traditional target for an Unreal Engine application is 60 FPS or 60 Hz to match standard refresh rates on most computer monitors. This is not a limit; it is simply a target to create a smooth visual experience for standard monitors. If a computer has a powerful CPU and GPU, frame rates can exceed 60 FPS, and in some cases exceed 140 or 200 FPS(Hz).

For the Unreal plugin, FPS is an important factor in how often the data is served to its clients from the simulation environment. The VS Connect server that is included with the plugin is processed within the Unreal Engine simulation loop, which means that VS Connect is able to service the network and process incoming and outgoing data once per video frame. The FPS that results from the graphical and computational complexity of the Unreal application determines how quickly data can be sent to the other clients and how frequently incoming data can be applied to objects in the Unreal simulation.

Input and Output Data Relationships

The inputs and outputs of the s-function block are defined by the config file, except for the first input port and the first output port. The first input port is reserved for the Connection Directive (zero = disconnect, non-zero = connect). The first output port is reserved for the connection status of the s-function block and the VS Connect server. The number and size of the other ports is defined by the config file.

Note

Simulink Block Input = VS Connect Outgoing Data

Simulink Block Output = VS Connect Incoming Data

S-function block inputs and outputs are inversely related to VS Connect's Outgoing Contracts and Incoming Contracts. Outgoing Contracts include data from the local simulation to be send to the remote peer, which correspond to Simulink block inputs. Incoming Contracts contain information received from the remote peer, which are conveyed to the Simulink model as outputs from the block.

Config File Parameter Details

version

The configuration file format number. This document describes version 2.

timeDilation

The s-function contains internal high-precision timing. This parameter specifies how fast the Simulink model is allowed to run. A setting of 1.0 means that the Simulink model will be throttled to run in "real time". A setting of 0.5 will cause the model to run in slow-motion at ½ of "real time" (1 second of simulation time will take 2 seconds of real-world time to execute). Note that an Incoming Contract with Time Sync enabled will override local simulation time throttling and the `timeDilation` setting.

remoteIP

IP address of the VS Connect server to connect to. By default, it is "localhost" aka "127.0.0.1".

remotePort

VS Connect server network port to connect to. If omitted, the default port (4367) will be used.

remoteObject (top-level parameter)

Default remote object name. This is used for individual field parameters if they do not define their own remoteObject.

inputs

Contracts containing data processed by the Simulink model and sent to VS Connect server. Represented as input on the s-function block. Each Contract has an updatePeriodMs and a list of fields.

outputs

Contracts containing data being received from the VS Connect server and processed by Simulink model. Represented as output from the s-function block.

groupType

Specifies what type of data is contained within the port group. Can be `"contract"` or `"commData"`. “Contract” data is data from or about the simulation, such as an object’s location, or the simulation time. “CommData” is data about the communication system, such as latency and transport time statistics (ping stats).

updatePeriod or updatePeriodMs

Define the maximum rate at which the Contract will update its data, in seconds or milliseconds, respectively. These parameters can be set to any decimal value greater than 0.

timeSyncMode

Enables time synchronization for the Contract. The only currently valid values for this parameter are `"none"` (disabled) or `"absolute"`. A Contract’s Time Sync Mode defaults to `"none"` if this keyword is not present. Only one Contract can have Time Sync enabled.

When Time Sync is enabled for an incoming Contract, the contract will enter the blocking state whenever the remote simulation time of latest data received is less than or equal to the local Simulink simulation time. This will cause the VS Connect S-function to pause the Simulink simulation until new data arrives from the remote simulation. This locks the progression of time in the local Simulink simulation to that of the remote simulation absolutely. In this case, the remote simulation is the **Time Sync Master**.

When Time Sync is enabled for an outgoing Contract, VS Connect will instruct the remote Peer that it should lock its clock to that of Simulink. If the remote application properly supports Time Sync, it will throttle and/or pause its local simulation to keep its simulation time synchronized with the Simulink simulation time. In this case, Simulink is the Time Sync Master.

The **VehicleSim Plugin for Unreal Engine** (version 2019.1) supports Absolute Time Sync, so when co-simulating with this s-function and the Unreal plugin, the user may optionally configure either Unreal or Simulink to be the Time Sync Master by specifying `"timeSyncMode": "absolute"` on either an incoming or outgoing Contract, respectively.

See the following **overdueDataTimeout** section for more information about Contract blocking.

Using both `timeSyncMode` and `overdueDataTimeout` options simultaneously can lead to unexpected results and is not recommended.

overdueDataTimeout or overdueDataTimeoutMs

Specifies an amount of time beyond the expected arrival of update data after which an Incoming Contract is considered to be in the “blocking” state. If this parameter is not specified, the port/Contract will never be in the “blocking” state.

For example: consider a Contract that has an `updatePeriod` of 0.1 seconds and a `overdueDataTimeout` of 0.5 seconds. If an update is received at time t=10.0, VS Connect expects another update to arrive at approximately 10.1. If an update is not received by time t=10.6 (the expected time 10.1 + 0.5 threshold), VS Connect will report that the contract is blocking.

Both the Simulink s-function and the Unreal plugin will force the local simulation to pause when a contract is in the blocking state.

Care must be taken to ensure that a dead-lock does not result from two peers both blocking while awaiting data from the other. To avoid a dead-lock situation, it is recommended to specify `overdueDataTimeout` only on inputs or outputs, but not both.

When communicating with the plugin for Unreal, it is preferred to use `overdueDataTimeout` on inputs/Outgoing Contracts rather than outputs/Incoming Contracts. This will cause the Unreal plugin to pause the Unreal simulation when the Simulink model is paused.

Using both `timeSyncMode` and `overdueDataTimeout` options simultaneously can lead to unexpected results and is not recommended.

ports

A list of ports. Where each item in the list will represent one or more values on the port of the s-function block. Each port can contain a list of fields where the number of fields define the width of the port.

remoteObject (field parameter)

Define the remoteObject for a parameter. The parameter remoteObject can be the same or different as the remoteObject defined in the beginning of the config file. If it is undefined, the main remoteObject will be used. If it is different, it will override the main remoteObject. This is used to identify different objects within in a scene and access their properties.

propertyType

Specifies the type/class of data that the field contains. If not specified, this defaults to `contract_field` for groups of type `contract`, and `meta` for groups of type `commData`.

For fields in groups of type `contract`, this may be set to `"contract_field"` (for simulation data), or `"meta"` for metadata about the simulation itself. For Contract metadata, there are three options that can be specified for `propertyName`: `"time"`, `"receive_time_offset"`, or `"serial_number"`.

For fields in groups of type `commData`, this must be set to `"meta"`. In this case, the only valid setting for `propertyName` is `"pingResults"`.

propertyName

proptertyName is a generic representation of a variable, function, or solver item. The properties available for a given object are defined by the creator of the VS Connect server.

dataType

The computer type representation of the parameter. Currently, the only supported type is `float`.

dataSize

Size of the data of the field parameter. For the `float` type, this specifies the size in bits. Currently, the default and only supported `dataSize` for `float` is `64`, which is the size of a double-precision floating point number in C code.

params

When `propertyName` specifies a "function" that accepts one or more parameters, the `params` parameter is used to specify the parameter(s) to be used for that function. For example, the VS Connect server implemented in the VehicleSim Dynamics Plugin for Unreal defines a "function" property called `VSAP_DISTANCE` which reports the distance between two scene objects (*Actors* in UE4 parlance). To receive this information, the VS Connect client specifies the VS Connect Name of the first Actor as the `remoteObject`, `VSAP_DISTANCE` as the property, and the VS Connect Name of the second Actor as `params`. As seen in Listing 2.

Listing 2. A field describing a "function" property and its parameter.

```
{
    "remoteObject": "Vehicle_A",
    "propertyName": "VSAP_DISTANCE",
    "params": "Vehicle_B"
}
```

For those familiar with C++ or Java, this can be thought of like calling a method on an object instance:

```
remoteObject->propertyName( params );
```

or

```
Vehicle_A->VSAP_DISTANCE( Vehicle_B );
```