# VS Database Upgrade Guidelines

The Upgrade Process ........ 1
- Conversion Issues ........ 2
- Upgrade Options ........ 2
- Validating the Upgrade ........ 5

Archiving Old Data ........ 5
Known Issues That Require Manual Editing ........ 6
- New data fields and libraries ........ 6
- Custom Settings in Miscellaneous Yellow Fields ........ 6
- External Software ........ 7
- Features that Always Require Replacements ........ 9
- Powertrain Modularity ........ 9
- Errors Fixed in Specific Screens ........ 9

End of Support ........ 10

This document provides guidelines and recommendations for upgrading databases from older versions of CarSim, TruckSim, and BikeSim into the newest release. It describes some issues that might exist and provides methods for dealing with them.

There is a product-specific companion document accessible from the Help menu **Help > Release Notes** > *product* **Backward Compatibility** (where *product* is CarSim, TruckSim, or BikeSim). This is a record of bug fixes and backward compatibility issues occurring at each release, going back to version 9.0 (2014 for CarSim, 2015 for TruckSim and BikeSim).

## The Upgrade Process

VS product upgrades have two parts:

1. installation of new static resources such as programs, libraries, animator shape files, and help documents, all located in the folder *product*`_Prog`, and

2. conversion of existing databases.

Installation of new programs is straightforward and is performed by the installer program for the release.

Starting with 2021.1, there are hundreds of archived examples in *product*`_Prog` that can be assembled to build any number of databases as needed that are compatible with the new version.

If databases exist that were used with older versions, they may be updated by opening them in the new version of the VS Browser (*product*`.exe`). When the VS Browser updates a database, it reads and then rewrites all the data files (called Parsfiles).

**Note** At any time, you can convert old databases to the new version. However, this is a one-way process. Databases from a new version cannot be converted by the VS Browser to work in an older version. For this reason, it is a good idea to make a copy of the original and then convert the copy.

## Conversion Issues

Most changes made to the software involve adding new features. Old databases do not have settings for the new features, but in general do not cause conflicts. Occasionally, changes are made to names of parameter or variables to provide better consistency given new capabilities of the software. When names of parameters are written automatically by the Browser, the new version of the VS Browser is usually able to recognize both old and new names, maintaining compatibility.

However, sometimes old databases contain data that cannot be handled automatically. Typical conversion issues involve expansion to support new features and/or the removal of previous ones.

Many of these issues are described in the *Backward Compatibility* release note for each product, such as:

1. Old screens that have been removed.
2. VS Math Model Import variables, export variables, and state variables that have been renamed or eliminated.
3. VS Commands that have more stringent error checking and will give errors that were not reported in earlier versions.
4. Bug fixes change model behavior, introducing differences in simulation results.

Other issues that have been encountered in previous updates and internal testing at Mechanical Simulation are described in a later *Known Issues* section (page 6).

## Upgrade Options

The main choices for upgrading older databases are:

1. Folder or Consolidated Parsfile (CPAR) archive:
    a. Make a copy of the original database folder and convert the copy by opening the folder with the new browser.
    b. Or, import one or more CPAR archive files made from older version(s) into a new database.
2. One step or multiple steps:
    a. Convert directly from the old database to the newest version in one step.
    b. Or, convert in multiple steps, going first to an intermediate version, then to a newer version, and so on.

## *Converting copies of database folders*

When a database folder from an older version of a VS product is opened with a newer version, the end-user will be prompted with an option to create a backup copy of the database. If this option is selected, a backup copy is made and the conversion to the new version is applied to the originally selected database folder. However, if the option to create a backup is not selected, the database will be converted for use with the new version and no backup database will exist.

Mechanical Simulation recommends using Windows File Explorer to manually copy the database folder from the old version before opening the original in the new version. This eliminates risk of clicking the wrong button if distracted.

## *Using CPAR archive files*

The CPAR archive option is useful if you want to archive sets of simulation runs rather than an entire database. Importing the CPAR archive files into an existing database will place your older datasets among the newly released examples, making this option useful if the goal is to quickly modify older examples for use with the latest tools and techniques.

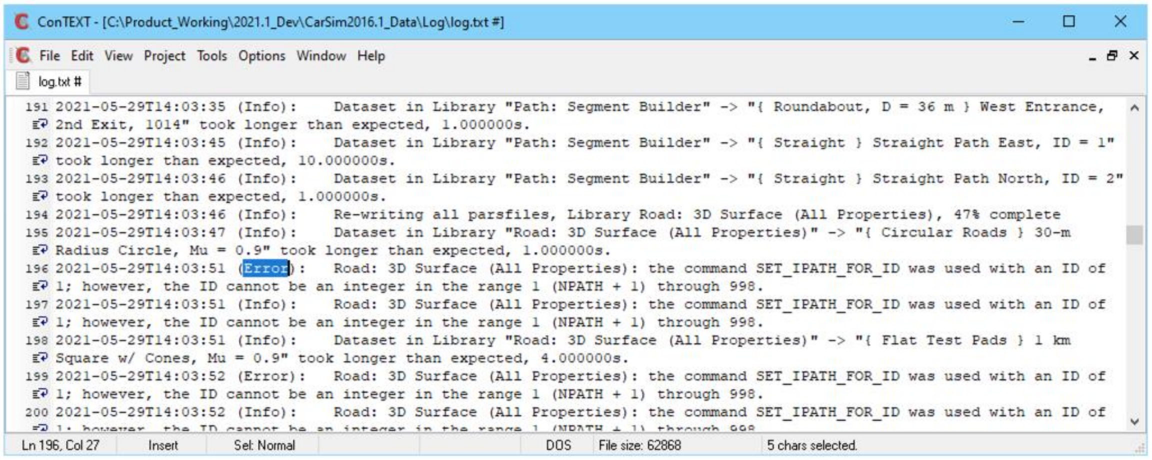
To export a CPAR archive, use the **File** menu on a given screen (e.g., **Run Control)** and select the option **Export Consolidated Parsfile**. To make CPAR archives with more data, use the Library Tool (**Tools > Library Tool**). The Library Tool allows you to export anywhere from a single dataset to all the datasets in a database. It also offers the option to append data to an existing CPAR file. The Append option is very convenient for making a few CPAR files with large collections of data.

## *Watch for error messages when updating*

When a VS Browser re-writes a database for a new version, it will show popup warnings if invalid settings are found, or if deprecated libraries are used. (Click the OK button in the popup window to continue or press the Return key.)

After the VS Browser finishes re-writing the datasets in the updated database, use the menu command **Tools > Show Log File** if there were any Error or Warning popups. The log file includes information about re-writing every library and shows all warnings and error messages. For example, Figure 1 shows part of a log file written during the re-writing of a CarSim 2016.1 database by CarSim 2021.1. Note the highlighted (Error) indicator on line 196. It is followed by two lines related to the error; the second (line 198) indicates there was a delay in handling a dataset (due to the time the error popup was displayed), and it identifies the library, category, and title of the dataset that needs to be fixed.

It the rewriting only has a few errors, they can be identified and fixed, using the Log file for guidance. On the other hand, if there are many occurrences, it might be more efficient to make another copy of the old database, open it in the old version (CarSim 2016.1 in this example), and modify the datasets there so they will be compatible when imported in the new version.


ConTEXT - [C:\Product_Working\2021.1_Dev\CarSim2016.1_Data\Log\log.txt #]
File Edit View Project Tools Options Window Help
log.txt #
Ln 196, Col 27
Insert
Sel: Normal
DOS
File size: 62868
5 chars selected.

```
2021-05-29T14:03:35 (Info):    Dataset in Library "Path: Segment Builder" -> "{ Roundabout, D = 36 m } West Entrance,
2nd Exit, 1014" took longer than expected, 1.000000s.
2021-05-29T14:03:45 (Info):    Dataset in Library "Path: Segment Builder" -> "{ Straight } Straight Path East, ID = 1"
took longer than expected, 10.000000s.
2021-05-29T14:03:46 (Info):    Dataset in Library "Path: Segment Builder" -> "{ Straight } Straight Path North, ID = 2"
took longer than expected, 1.000000s.
2021-05-29T14:03:46 (Info):    Re-writing all parsfiles, Library Road: 3D Surface (All Properties), 47% complete
2021-05-29T14:03:47 (Info):    Dataset in Library "Road: 3D Surface (All Properties)" -> "{ Circular Roads } 30-m
Radius Circle, Mu = 0.9" took longer than expected, 1.000000s.
2021-05-29T14:03:51 (Error):   Road: 3D Surface (All Properties): the command SET_IPATH_FOR_ID was used with an ID of
1; however, the ID cannot be an integer in the range 1 (NPATH + 1) through 998.
2021-05-29T14:03:51 (Info):    Road: 3D Surface (All Properties): the command SET_IPATH_FOR_ID was used with an ID of
1; however, the ID cannot be an integer in the range 1 (NPATH + 1) through 998.
2021-05-29T14:03:51 (Info):    Dataset in Library "Road: 3D Surface (All Properties)" -> "{ Flat Test Pads } 1 km
Square w/ Cones, Mu = 0.9" took longer than expected, 4.000000s.
2021-05-29T14:03:52 (Error):   Road: 3D Surface (All Properties): the command SET_IPATH_FOR_ID was used with an ID of
1; however, the ID cannot be an integer in the range 1 (NPATH + 1) through 998.
2021-05-29T14:03:52 (Info):    Road: 3D Surface (All Properties): the command SET_IPATH_FOR_ID was used with an ID of
1; however, the ID cannot be an integer in the range 1 (NPATH + 1) through 998
```

*Figure 1. Log file after importing a 2016.1 database into version 2021.1.*

## *Converting in one step (old version directly to new version)*

The most direct way to convert an old database is to select the old database from the new version. If most of the results seem OK but there are just a few problems, there are two possible solutions. Both involve opening the original (old) database in the old VS Browser, side-by-side with the new VS Browser that is using the converted database. The two options are:

1. In the converted database, visit each dataset that was mentioned in the log file and fix it in the converted database.
2. Or, in the converted database, visit each dataset that was mentioned in a warning during the conversion. Visit the same screen in the old version, address the issues, and then convert the just-modified old database to the new version, effectively replacing the previously converted database.

The advantage of the first option is that it is fast; you will not have to repeat the import process. The advantage of the second option is that the old database is now compatible with the new version and may be imported more times without any editing.

## *Converting in multiple steps*

If the old version is more than a year or two older than the new version, there may be many conflicts, and some might be difficult to diagnose.

Before converting, look at the first section in the *Backward Compatibility* document, and check the table in the first section listing the library screens that have been removed recently. The same document includes a list of discontinued variables for each release. Check your database in the old version to see if those libraries were used. If so, rework the datasets to use the replacement libraries.

If the database is very old, it is possible that the preferred (i.e., new) library screen did not exist at that time. In this case, you should identify a version of the VS product that still supported both GUI screens and install that version. You can import your old data to this intermediate version, copy the data from the deprecated screen to the new one, and then convert the intermediate database to the current version. For example, multiple libraries were removed in 2019.1. If your old database made use of some of those libraries, you could install version 2019.0, import the database into 2019.0, edit as needed to avoid the use of the deprecated screens, and then import a copy of the 2019.0 database into the new version.

### Validating the Upgrade

It is sometimes necessary to compare simulation results in the new version of to those of an older version. This validation step, sometimes referred to as version-to-version comparison, is described in the document *Validation of VS Vehicle Models* from the **Help > Technical Memos** submenu. It provides information and tips for comparing results from different versions.

In some cases, examples that ran in previous versions no longer run in the new version. Sometimes this can be due to changes in the model behavior that make it more susceptible to certain input data, while in other cases fundamental changes in the software mean that certain examples are no longer supported. Details concerning this are covered in a following section.

## Archiving Old Data

There are several steps that can be taken to archive data, making it easier to work with later.

- Make copious use of the Notes fields for important datasets. This way, the methods, data sources, assumptions, and limitations of the simulation are recorded and available later.
- Keep the original installer program that was delivered with the release, e.g., `Setup_`*product*`_2021.0_r{123}.exe`. This makes it possible to recreate a pristine copy of that release at any time. If you do not have a copy, this is a good time to download one from the Users section in www.carsim.com.
- Keep a copy of the *product*`_Prog` folder that is the same version as the archived databases. Clearly identify the version of the archived programs folder, as well as the corresponding archived database(s), e.g., *product*`2021_Prog` and *product*`2021_Data`.

As noted earlier, there are at least two approaches for archiving the data: Archiving folders and exported CPAR files.

### *Archive database folders*

Archive entire databases simply by copying the folder and its contents. To save space, you can make a ZIP file. Using this method, all datasets in the database will be available for use with the new version following the database conversion. Simulation results between the new and old databases can be compared via the individual screen parsfiles, the Echo files, and/or using the Overlay option on the Run Control screen. Within Mechanical Simulation, these methods serve as part of our diagnostic methods for investigating compatibility from one release to another.

### *Archive CPAR files*

Archive specific datasets (and their links) using the **File** drop-down menu on a given screen (e.g., **Run Control)** and selecting the option **Export Consolidated Parsfile**. To make CPAR archives with more data, use the **Library Tool** (**Tools > Library Tool**). The **Library Tool** allows you to export anywhere from one to all other datasets in a database. It also offers the option to append data to an existing CPAR file. The **Append** option is very convenient for making a few CPAR files with large collections of selected data.

## Known Issues That Require Manual Editing

Most simulations set up in versions 9.0 and later will work as before when imported to the newest version; however, some will fail due to major changes made in the VS Math Model capability or the VS Browser/GUI. Following are some cases where changes will probably be needed when converting older databases to the latest version.

### New data fields and libraries

Some of the newer model features are represented with new data fields on the GUI screens. Once a database is converted, the new data fields introduced in that new version will be initially blank, with the intention that if left as-is, those fields will have no effect on the simulation. Most of the time this is true, but not always.

Consider the example of the Brake/Bearing Friction on the **Brake System** screens in CarSim and TruckSim, added to the model with version 2018.0. If these fields are left blank, the default data for this math model functionality will be used: 0.5 N-m of brake torque will be applied to each wheel on the vehicle. If you are trying to replicate old results (prior to 2018.0), specify a value of zero in these fields to disable the effect of the brake/bearing friction.

**Note** Leaving a data field blank or not linking a dataset to a required library link means a default value is used, and the only place to see that default value is in the Echo file. Parameters that were not set by reading a value from an input file have descriptions that begin with the prefix `[D]`.

Mechanical Simulation does not recommend leaving data fields blank, nor do we recommend unlinking necessary datasets. Many math model parameters have non-zero default values that can only be found in the Echo file. In cases of unlinked datasets, default options will be used for certain features (e.g., suspension options) that may not be what is intended. When in doubt, refer to shipping examples that make use of these features to see how they should be set up.

### Custom Settings in Miscellaneous Yellow Fields

Many Browser library windows include Miscellaneous yellow data fields for advanced users to provide VS Commands. They can also be used to set parameter values or provide any valid inputs for the VS Math Model.

When updating datasets that include content in Miscellaneous fields, the contents are left “as-is.” No changes are made automatically by the Browser when rewriting these fields.

In general, the VS Math Models will recognize renamed parameters and output variables. However, many changes have been made involving state variables: some have been removed while others have been added. Runs that contain VS Commands involving state variables will fail if they refer to state variables that no longer exist.

### *VS Command parsing improvements*

In the past few years, major updates have been made to the parsing of VS Commands that are typically placed in Miscellaneous yellow fields. This means that errors in VS Commands which may not have been caught before (e.g., spaces in variable names, missing parentheses) are now flagged when a simulation is made. As noted above, the contents of Miscellaneous yellow fields are left “as-is” during a database conversion. If the contents generate errors, they will have to be identified and corrected manually.

### *Read-only parameters*

Recent versions include a set of read-only parameter-type variables that have keywords such as: `PI` (π = 3.141592654…) , `ZERO`, `G` (gravity constant, 9.80665), and others. In some older versions, examples were provided with `ZERO` defined with a `DEFINE_PARAMETER` command. If updated to a new version, that command will cause an error. It is corrected by simply deleting the command to create `ZERO` but keeping any equations that make use of it.

## External Software

Screens used to connect with external software (Models and I/O screens) are often used to support custom setups. The layouts of the Models screens (**Models: Simulink**, **Models: Self-Contained Solvers**, etc.) were modified in 2018.0 and 2018.1 to support more interface options. Datasets for these screens that are imported from older versions may need modification to work as originally intended.

Installations that do not involve the VS Browser for most or all operations cannot be updated by the built-in database conversion process. If, for example, your installation uses MATLAB scripts and API function calls to load and run simulations, you must update those files manually. If you access, set, or modify VS Math Model parameters by name, you must check that the parameter names are current with the new version.

### *I/O channels and I/O arrays*

In some updates, import and/or export variables were either renamed or removed. Linked external software such as MATLAB Simulink will no longer work with the old variable names.

If simulations involving external tools no longer work, there are several things to check.

- Check the *Backward Compatibility* document and look for import and/or export variables that have been renamed or removed. Search for those variables in the database as accessed in the old version and take note of where they are used. In the converted database, update the relevant I/O lists with the new variables.
- Another place to check for Import variable issues is at the end of the Echo files as made with both the old and new versions. In all recent versions, the Echo file lists all I/O variables activated for import and export. Compare the Echo files for the old and new version to see what is different.

- In some cases, the simulation terminates before the end of the Echo file gets written. If this happens, use the drop-down control in the lower-right corner of the Run Control screen to select the option Echo file with current data (Figure 2). Then click the **View** button. The Browser will cause the VS Math Model to generate an Echo file without making a run, using the current data. Inspect the Echo file, checking names of Import variables, Export variables, and other names in the Echo file. Compare these with the Echo file that exists in the old database copy that was not updated.

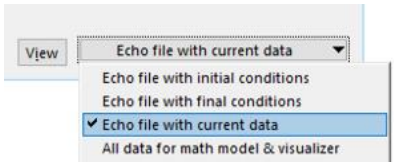

View
Echo file with current data
Echo file with initial conditions
Echo file with final conditions
✓ Echo file with current data
All data for math model & visualizer

*Figure 2. Drop control next to View button on Run Control screen.*

### Embedded Python

The Embedded Python capabilities introduced in version 2018.1 did not allow direct access to VS variables. The code has redone for 2018.1 to allow this access. However, examples from 2018.1 will not run if imported into any newer version.

### Simulink models

If an existing Simulink model shows a broken link rather than the CarSim S-Function(s), the problem is usually a machine-specific issue with Read/Write restrictions or running with an incompatible MATLAB/Simulink version. Some things to check:

- Ensure the chosen version of MATLAB/Simulink is compatible with the version of CarSim you are working with (**Help > Release Notes > System Compatibility**).
- Close MATLAB/Simulink. From the CarSim **Run Control** screen for a simulation set up to run with Simulink, click the **Send to Simulink** button. Once Simulink launches, open the Simulink Library Browser and see if the CarSim S-Function blocks can be selected.

### Simulink Library Browser

If the CarSim S-Functions do not appear in the Simulink Library Browser, the most common reason is that MATLAB was launched directly rather than via CarSim.

- If the button **Send to Simulink** was not clicked from the CarSim **Run Control** screen, close MATLAB, then click the button **Send to Simulink**. Check again to see if the CarSim S-Function blocks show up in the Simulink Library Browser.
- If the CarSim S-Functions still do not appear in the Simulink Library Browser, make sure the MATLAB installation can properly access Load Library functions, and that there are no Read/Write permission restrictions.

### *Simulink examples with memory blocks*

Timing is critical when extending a VS Math Model with Simulink. To avoid an error message from Simulink regarding an algebraic loop, it is sometimes helpful to insert a memory block in the Simulink model (typically before the S-Function), thus adding an artificial 1-step delay.

In versions 2018.0 and 2018.1, timing was improved for connecting with Simulink and other external workspaces (FMI, LabVIEW, etc.). This timing improvement means that in some cases, the memory blocks are no longer needed in the Simulink models, and they should be removed. A new parameter `OPT_SKIP_TSTART` was introduced in 2019.1 that reverts to the old timing for improved backward compatibility support. For some VS/Simulink model combinations, this allows the memory blocks to remain in the Simulink models following a VS database conversion.

## Features that Always Require Replacements

Some features are not intended to be compatible across revisions.

- Encrypted expanded Parsfiles are essentially frozen to the version in which they were created. If encrypted data will be used over several versions, the user who encrypts the data must maintain a backup copy of the original and be prepared to make new encrypted expanded Parsfiles for future versions.
- ETAS LabCar (RT system). Simulink models for ETAS LabCar must be updated for every new release.

## Powertrain Modularity

All powertrain subsystems to be used for the run (engine, transmission, differentials, electric motors, etc.) must be installed before the run starts. Past versions had some examples where the powertrain was changed via an Event (e.g., switching from 2WD to 4WD). In other examples, I/O datasets used for Simulink models intended to support multiple powertrain configurations (e.g., both 2WD and 4WD) specified 4WD import variables that do not exist in 2WD powertrains (e.g., `IMP_CLUTCH_D3`).

These methods do not work in newer versions. The VS Command that installs the powertrain is `INSTALL_POWERTRAIN` (it was OPT_PT in older versions). The command must be applied before the simulation starts, and it cannot be repeated.

New version of CarSim and TruckSim support multiple vehicles, and powertrain parameters are indexed as are other system-related parameters. Also, as support has been extended for electric and hybrid systems, single powertrains might have multiple components (e.g., motors) that require indexing that was not needed in older versions.

## Errors Fixed in Specific Screens

A few screens have been changed to fix mistakes. However, the fixes can cause issues for datasets imported from older versions.

### *Speed Controller: Incorrect units for brake system performance*

A bug fix was made in 2018.1 for the units of the brake performance parameter `BK_PERF_SC` used by the speed controller in CarSim and TruckSim. In most cases the effect was minor. However, in some cases, datasets from 2018.0 and older will have a value of `BK_PERF_SC` that may not be sufficient to brake the vehicle.

When working with datasets from 2018.0 and earlier, either multiply the value by 9.80665, or use the checkbox (added in 2019.1) to use the old value with the old units.

This issue is mentioned here because there is not an automatic correction to the contents of the yellow data field. If the value of `BK_PERF_SC` was set to a value tuned for a vehicle in a version prior to 2018.1, then it should be reevaluated following the database conversion.

### *Control: Shifting (Closed Loop) screen*

In versions prior to 2019.1, the screen **Control: Shifting (Closed Loop)** allowed a link to be made to a dataset from the **Control: Shifting (Open Loop)** library. This was a mistake; if the intent is to use open-loop shifting data, then a link should be made to a **Control: Shifting (Open Loop)** dataset from either a **Procedure** or **Event** dataset. In newer versions, it is not possible to make this bad link.

If old datasets are imported that have such links, they will appear red and be identified as broken. However, the Parsfiles still include the pathnames to the originally linked datasets, and they will be used if a new run is made. In these cases, we recommend deleting any **Control: Shifting (Closed Loop)** datasets with bad links, then modifying the affected **Procedure** or **Event** datasets to instead link directly to the **Control: Shifting (Open Loop)** datasets.

### *Spline Plots in the GUI*

In versions of the VS Browser prior to 2017.1, the wrong algorithm was used to show lines for 2D tables. This was corrected in 2017.1 to match the 2D spline algorithm used in the VS Solvers.

In version 2020.0, the VS Browser showed bad plots for some datasets with 1D Spline interpolation and extrapolation. This was fixed in 2020.1.

## End of Support

Supported versions of VS products are available for download from www.carsim.com. If a version is not available for download, then it is not supported. At this time (June 2022), supported versions go back to 9.0. A schedule for the planned end of support is also available on the web site at www.carsim.com/contactus/endofsupport.php. The schedule is based roughly on stopping support for software seven years after the initial release and is consistent with other engineering software for Windows.

Databases from versions that are nearing the end of support should be imported into a newer version before support is stopped. Given that support for 9.0.*x* ends in December 2022, data from 9.0.*x* should be imported into something newer to maintain a useful archive. Older versions of the VS Browser might need to run on older computers and Windows versions, which might be difficult to manage in future years.