



From Includes to Model Browser

Table of Contents

- 1. Introduction 3
- 2. Execution 3
 - 2.1. Defintion 4
 - 2.1.1. Identification of include characteristics 6
 - 2.1.2. Modifying the initial include structure 8
 - 2.2. Settings 9
 - 2.3. Simulation properties..... 11
 - 2.3.1. Extracting simulation attributes from the main include file name using a custom user script..... 12
 - 2.4. Subsystems properties..... 12
 - 2.4.1. Extracting properties from the subsystem include names using a custom user script13
 - 2.5. Library files properties 14
 - 2.6. Preview 16
 - 2.7. Automatic execution 17
- 3. Report and resulting model 19
 - 3.1. Handling of parameters 23
 - 3.2. Handling of unsupported content 24

1. Introduction

From Includes to Model Browser is a tool available in ANSA as a plugin. Its purpose is to enable the migration of a model that contains Include entities to a model based on Model Browser Containers in ANSA.

As a result, the initial model is represented with Subsystems, Simulation Models, Library Files, Loadcases and Simulation Runs in the Model Browser and the different nodules are stored in the data repository of ANSA DM. From this point on, the users can create new iterations of the simulation in the Model Browser, using ANSA DM as a data management backbone.

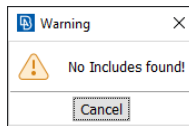
2. Execution

The tool is automatically loaded and available in *Tools > Plugins > From Includes to Model Browser*.

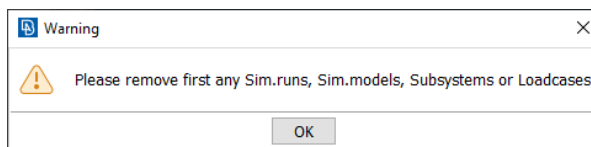
! Note: For the time being, the plugin works only for Nastran and LsDyna decks.

There are two main pre-requisites for the plugin to be executed:

- The model opened in ANSA should be organized with Includes. If no includes are found, the plugin cannot be executed and a respective warning window appears.



- The model should not be already organized with Model Browser Containers. If any Subsystems, Simulation Models, Loadcases and Simulation Runs are detected, a relevant warning window appears and the process cannot continue unless these entities are removed.

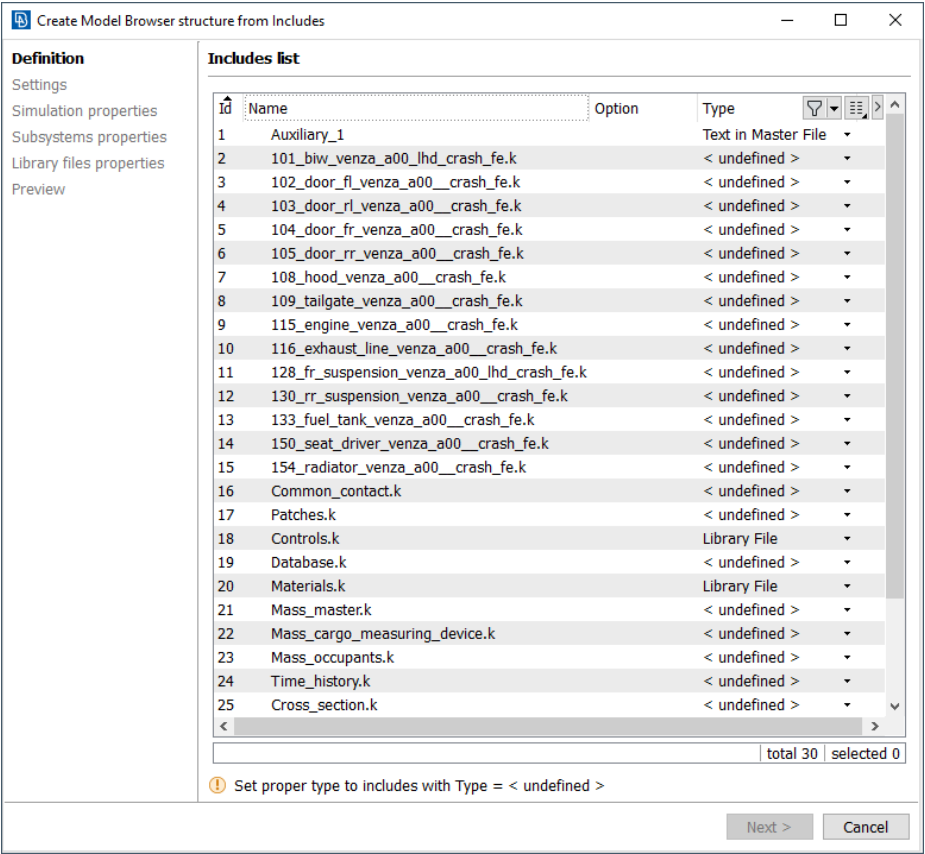


When the model loaded in ANSA satisfies both of the pre-requisites above, the plugin can be executed and the *Create Model Browser structure from Includes* window will appear. This window is a wizard that consists of several pages that guide the user through the creation of a Model Browser structure for the loaded model.

The paragraphs below provide a detailed description of each page of the wizard.

2.1. Defintion

The first page of the wizard contains a list of all the includes present in the model. The main target here is to categorize the includes into model-specific structural includes (Subsystems and Connecting Subsystems) and library files.



Leaving the mouse cursor over an include in the list, a tooltip appears listing all the keywords found within this include. In this way, the operator can be assisted in the decision process of types for each include.

Includes list

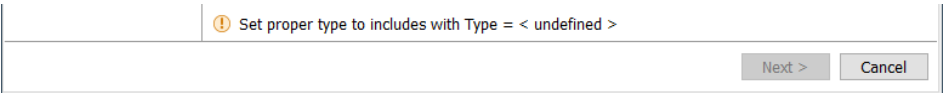
Id	Name	Option	Type
1	Auxiliary_1		Text in Master File
2	101_biw_venza_a00_lhd_crash_fe.k		< undefined >
3	102_door_fl_venza_a00_crash_fe.k		< undefined >
4	103_door_rl_venza_a00_crash_fe.k	CONSTRAINED_EXTRA_NODES_NODE	defined >
5	104_door_fr_venza_a00_crash_fe.k	CONSTRAINED_GENERALIZED_WELD	defined >
6	105_door_rr_venza_a00_crash_fe.k	CONSTRAINED_NODAL_RIGID_BODY	defined >
7	108_hood_venza_a00_crash_fe.k	CONSTRAINED_RIGID_BODIES	defined >
8	109_tailgate_venza_a00_crash_fe.k	CONSTRAINED_SPOTWELD	defined >
9	115_engine_venza_a00_crash_fe.k	CONTACT	defined >
10	116_exhaust_line_venza_a00_crash_fe.k	ELEMENT_SHELL	defined >
11	128_fr_suspension_venza_a00_crash_fe.k	ELEMENT_SOLID	defined >
12	130_rr_suspension_venza_a00_crash_fe.k	HOURLGLASS	defined >
13	133_fuel_tank_venza_a00_crash_fe.k	NODE	defined >
14	150_seat_driver_venza_a00_crash_fe.k	SECTION_SHELL	defined >
		SECTION_SOLID	defined >
		SET	defined >
			< undefined >

The following columns exist in the *Includes list*:

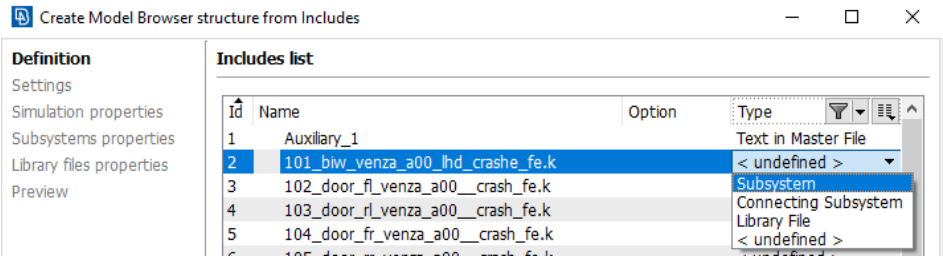
Id	It displays the Id of the respective include in the ANSA database. It is a read-only column, automatically filled in by ANSA.
Name	It displays the name of the respective include in the ANSA database. Its value is automatically filled in by ANSA, however it can be modified if needed.
Option	It displays the option value of the keyword of the respective include in the ANSA database (e.g. _TRANSFORM). This is only available for when the current Deck is LS-DYNA. It is a read-only column, automatically filled in by ANSA.
Type	It displays the type of the respective include, which controls the Model Browser Container that will be generated. Its value is automatically filled in by ANSA, where possible; however it can be modified if needed.

The following values can be found in the *Type* column in the *Definition* page of the wizard:

- **Subsystem**: Used for model-specific includes containing structural parts of the model, which will be handled as individual files.
- **Connecting Subsystem**: Used for model-specific includes containing connecting entities, which connect subsystems with each other or with library files.
- **Library File**: Used for includes containing additional entities and solver keywords, which will be handled as individual Library Files.
- **Subsystem Group**: Used for grouping includes containing structural parts of the model and possibly connecting entities too. Each sub-include will be stored individually as a Subsystem or Connecting Subsystem, and the parent include will be stored as a Group Subsystem.
- **Library File Group**: Used for grouping includes containing additional entities and solver keywords. Each sub-include will be stored individually as a Library File or Connecting Subsystem
- **Text in Master File**: Used for Auxiliary includes marked as inline, which contain entities that will be written as keywords in the main file. These includes are eventually removed and their contents are added to the Simulation Run as Model Setup Entities.
- **< undefined >**: Used when ANSA was unable to automatically detect and fill the include type. <undefined> is not an accepted value and the wizard cannot advance to the next page, unless one of the types above are defined for each include



To change the type of an include in the *Definition* page, the user can select one of the available values from the drop down list.



Note that the eligible types depend on the structure level of each include. For example, a nested include (i.e. an include that contains other includes), can be marked as a Subsystem Group or Library File Group or Subsystem or Library File. However, a base level include can only become a Subsystem or a Library file. Moreover, in the case of a nested include marked as Subsystem or Library File, their sub-includes will be automatically marked as “-” in the *Type* column.

Using the option **Edit** from the context menu, the edit card of the selected include can be opened in ANSA.

Finally, using the options **Show**, **Hide** and **Show Only**, the user control the visibility of each include in the ANSA drawing area.

2.1.1. Identification of include characteristics

As mentioned previously, the plugin will attempt to automatically fill the type of the detected includes in the *Definition* page of the wizard. This is done based on the identification of include characteristics on two levels.

- First, an attempt will be made to detect the include type based on the include's contents (for LsDyna deck only).
- If the include type is not defined using its contents, then a second attempt will be made, based on the include's name, using some generic default settings which can be customized according to each team's naming conventions.

In case the include type fails to be detected with both of the two methods described above, the *<undefined>* value will be assigned and the user will have to define the include type manually.

! Note: It is advisable to review the auto-detected include types since the default settings of the tool attempt to cover the most usual naming rules, but exceptions may apply.

Content-based detection of include type

When the current Deck is LS-DYNA, ANSA will check the include contents in order to detect and assign the type value in the *Definitions* page.

For individual includes, the following three cases are examined:

Include contents (exclusive)	Auto-detected include type
*DEFINE_TRANSFORMATION	Library File
*CONSTRAINED_RIGID_BODY	Connecting Subsystem
*DEFINE_CURVE *DEFINE_TABLE MATERIAL	Library File

An additional case of content-based detection of the include type is that of stamped parts. Whenever the plugin detects an include marked with the *_STAMPED_PART* option, the user is required to place it under an existing include marked as Subsystem. This action will automatically assign the “-” value to the stamped part include *Type*.

For grouping-includes, ANSA will define their type based on the type of their children:

Include Type of Children	Auto-detected include type of parent
Subsystem	Subsystem Group
Subsystem Connecting Subsystem	
Library File	Library File Group
Library File Connecting Subsystem	

! Note: For each include whose type is successfully retrieved from its contents, ANSA does not proceed to the name-based detection of its type.

Name-based detection of include type

ANSA will attempt to detect the include type based on its name when the contents-based type identification has failed.

The name-based detection of the include type relies on the identification of certain tags in the include name. These tags are defined in the ANSA defaults settings and can be customized to match the customer needs. A characteristic example of these tags can be seen below.

```
# ***User Defined Keywords #

***consider_numeric_include_identifiers = false
***connecting_includes_identifier       = patch
***mat_includes_identifier              = mat
***prop_includes_identifier             =
***subsystem_includes_identifier        =
***project_identifiers                  = venza,proj1,proj2,proj3
***release_identifiers                  = a00,a01,a02,a03,a04,a05
***library_file_identifiers             =
***generic_mat_includes_identifiers      = matdbase,mat,material
***generic_library_file_identifiers      = nhtsa,xsection,50th,5th,90th,95th,_occ_,barrier,wall,friction,control,
                                         odb,accel,measurement,addmass,add_mass,mass,occupant,sensor,fmvss
```

ANSA tries to detect the include type in two stages. First, the following ANSA defaults keywords are used to tag an include as *Subsystem*, *Connecting Subsystem* or *Library File*:

- consider_numeric_include_identifiers*

It takes a Boolean value (*true*, *false*) and controls whether the identification of the include type will be based on numeric ranges. It determines whether the ANSA.defaults keys below will expect numbers and numeric ranges or keywords.
- connecting_includes_identifier*

It takes comma-separated values that can be numbers and numeric ranges (e.g. 100, 800-899), or keywords (e.g. cnct, connect). If found in the name of an include, its type will be defined as *Connecting Subsystem*.
- mat_includes_identifier*

It takes comma-separated values that can be numbers and numeric ranges (e.g. 100, 800-899), or keywords (e.g. matdbase). If found in the name of an include, its type will be defined as *Library File*.
- prop_includes_identifier*

It takes comma-separated values that can be numbers and numeric ranges (e.g. 100, 800-899), or keywords (e.g. sectiondbase).. If found in the name of an include, its type will be defined as *Library File*.

subsystem_includes_identifier

It takes comma-separated values that can be numbers and numeric ranges (e.g. 100, 800-899), or keywords (e.g. module). If found in the name of an include, its type will be defined as *Subsystem*.

generic_mat_includes_identifiers

It takes comma-separated keywords. If found in the name of an include, its type will be defined as *Library File*.

If an include type fails to be detected using the keywords described above, ANSA will try to characterize each include as *Project-specific*, *Release-specific* or *Loadcase-specific* based on the following keywords:

project_identifiers

It takes comma-separated list of project names that may be found in the include name. If found in the name of an include, it is characterized as *Project-specific*.

release_identifiers

It takes comma-separated list of release names that may be found in the include name. If found in the name of an include, it is characterized as *Release-specific*.

library_file_identifiers

It takes comma-separated values that can be numbers and numeric ranges (e.g. 0-99, 900-999), or keywords (e.g. control, contact, header).. If found in the name of an include, it is characterized as *Loadcase-specific*.

generic_library_file_identifiers

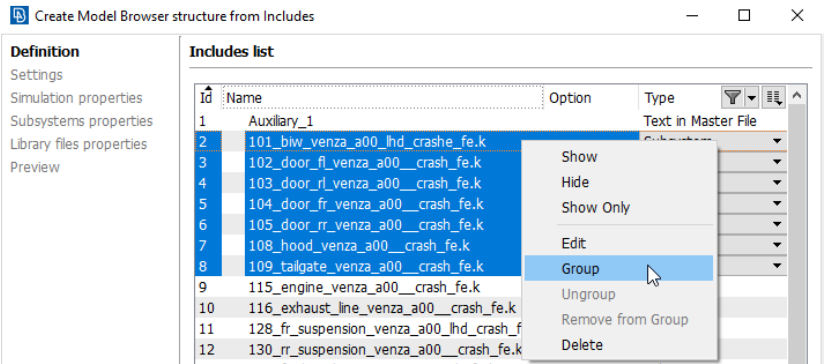
It takes comma-separated keywords. If found in the name of an include, it is characterized as *Loadcase-specific*.

After characterizing the includes as *Project-specific*, *Release-specific* or *Loadcase-specific*, the include input type will be defined as follows:

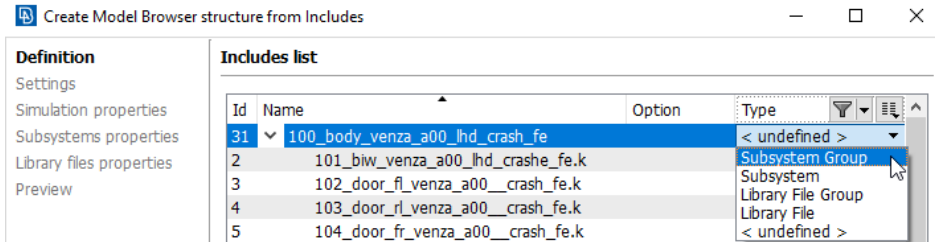
- If an include is *Project-specific* and *Release-specific*, but not *Loadcase-specific*, then its type will be defined as *Subsystem*.
- If an include is *Loadcase-specific*, then its type will be defined as *Library File*.
- If none of the above conditions are satisfied, then the include will be marked as *<undefined>*.

2.1.2. Modifying the initial include structure

To manually create a grouping include, the user can select the individual includes that should be added under the parent include, activate the context menu using the right mouse button and select the option **Group**.



Once the grouping include has been created, the user can modify its name and type accordingly.



To add an include to an existing grouping include, the user can drag and drop the include onto the group.

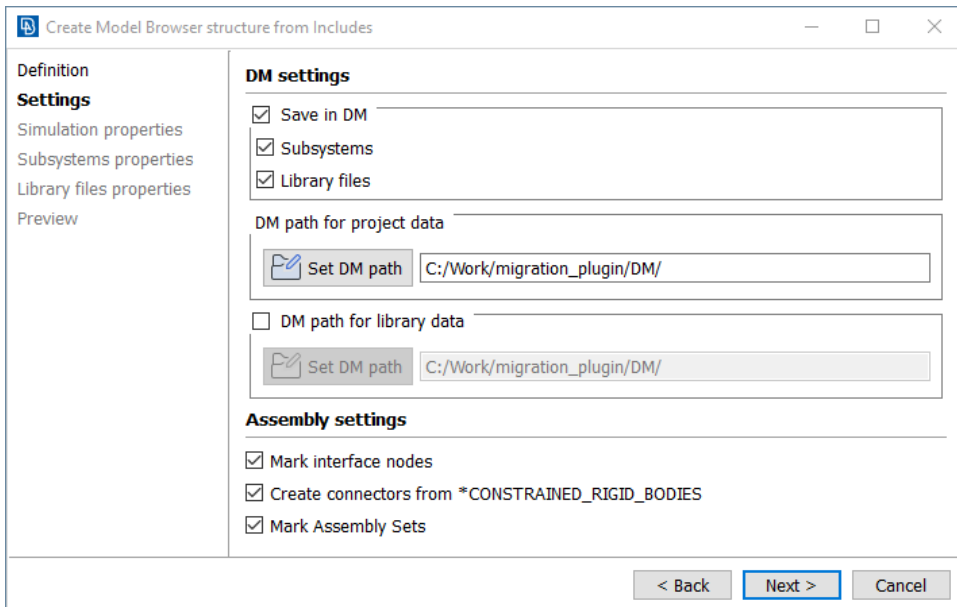
To ungroup a grouping include, the option **Ungroup** can be selected from its context menu, while to remove one or more sub-includes from a grouping include, the option **Remove from Group** can be selected from the context menu of the sub-includes, respectively.

To exclude an include from the conversion to Model Browser Containers, the user can select the option **Delete**, which will remove it from the *Include list* of the *Definition* page of the wizard.

! Note: Any actions that modify the include structure in the plugin wizard are also reflected in the Includes list in ANSA.

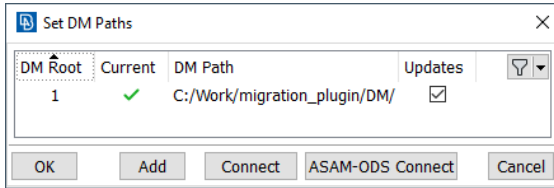
2.2. Settings

The second page of the plugin wizard is the *Settings* page, where the user can define various data management and model assembly related settings for the plugin. The page is split in two sections: the *DM settings* and the *Assembly settings*, respectively.

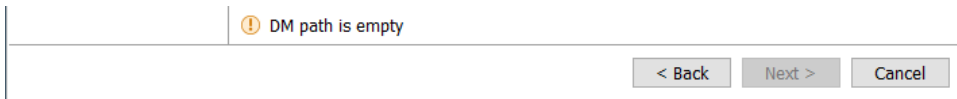


In the **DM Settings** section, the user can control whether the migration of the include-based model to a model organized with Model Browser Containers will be combined with data management functionality, facilitating the management of the model in a modular manner with a data management backbone. If the **Save in DM** checkbox is activated, by activating the respective checkboxes, it is possible to control if **Subsystems** and/or **Library files** will be saved in DM.

Using the **Set DM Path** button, the *Set DM Paths* window opens where the user can select to **Add** a file-based DM or **Connect** to an SPDRM server, using the respective buttons. It is also possible to define two separate data management repositories, one for project data and one for library data.



! Note: If the **Save in DM** checkbox is activated, the wizard cannot advance to the next page unless a valid DM Path is specified.



To take full advantage of the data management functionality, the plugin should be executed with a DM configured with a data model that supports Library Item definitions. In this case, the tool enables the comprehensive management of the model by differentiating between structural includes and library files, saving both to the data repository with the appropriate metadata.

This data model definition can be either the default ANSA data model or a custom data model definition. Using the *DM Schema Editor*, accessed through the respective option from the DM group in the ANSA Lists menu, the user can define a custom data model, which needs to be saved as a data model configuration file called *dm_structure.xml*, in the DM directory. This file contains all the required definitions of structural model entities and library files, allowing the customization of their attribution. For a detailed description of the data management customization capabilities, please refer to paragraph 30.8. *Customizing ANSA DM* of the ANSA User's Guide.

In the **Assembly Settings** section in the *Settings* page of the wizard, the user can control the various model assembly settings that are available in the tool.

The **Mark interface nodes** checkbox is enabled only if *Connecting Subsystem* includes have been defined in the *Definitions* page of the wizard. If the checkbox is activated, ANSA will identify all interface nodes between the includes and mark them with Assembly Points based on the connectivity of the model. ANSA will consider as interface nodes those nodes that belong to includes marked as Subsystems or Library files, but are used by elements that belong to Connecting Subsystems.

The **Create connectors from *CONSTRAINED_RIGID_BODIES** checkbox is enabled only if *Connecting Subsystem* includes have been defined in the *Definitions* page of the wizard and the current deck is LS-DYNA. If the checkbox is activated, *CONSTRAINED_RIGID_BODIES will be converted to ANSA Connector entities.

Finally, if the **Mark assembly sets** checkbox is activated, ANSA will identify all sets that contain other sets that belong to a different include than their parent and mark them as *assembly sets*. The contained sets are marked as *interface sets* and an ANSA Domain Finder is created and added to the assembly set, replacing its original content.

2.3. Simulation properties

The third page of the wizard is the *Simulation properties* page, where the properties of the Simulation Run that will be created and saved in DM can be defined.

Depending on the definition of the Simulation Run in the data model configured through the *dm_structure.xml*, some of the properties may already be filled by ANSA using the default property value, others may have a set of accepted values from which the user can select and others may be free-text fields.

For any of the data model properties that are common between the Simulation Run and other data model entities, the values defined here for the Simulation Run will also be set to the other entities in the following wizard pages.

Property name	Values
Discipline	crash
Loadcase Id	front_offset
Model Id	crash_assembly
Model Variant	crash_assembly
Project	pedestrian_assembly
Release	body
	seat_dummy

! Notes:

- If the user has selected to save the entities in a DM not configured with a custom data model definition, the Properties that will appear in this page will be the default ANSA properties for Simulation Runs.
- If any of the properties are undefined, the wizard cannot advance to the next page.

Undefined properties detected

2.3.1. Extracting simulation attributes from the main include file name using a custom user script

The path of the user script and the function that will be executed are defined in the ANSA defaults settings using the keywords *interpret_attributes_script_file_path* and *interpret_sim_run_attributes_script_func_name*, respectively.

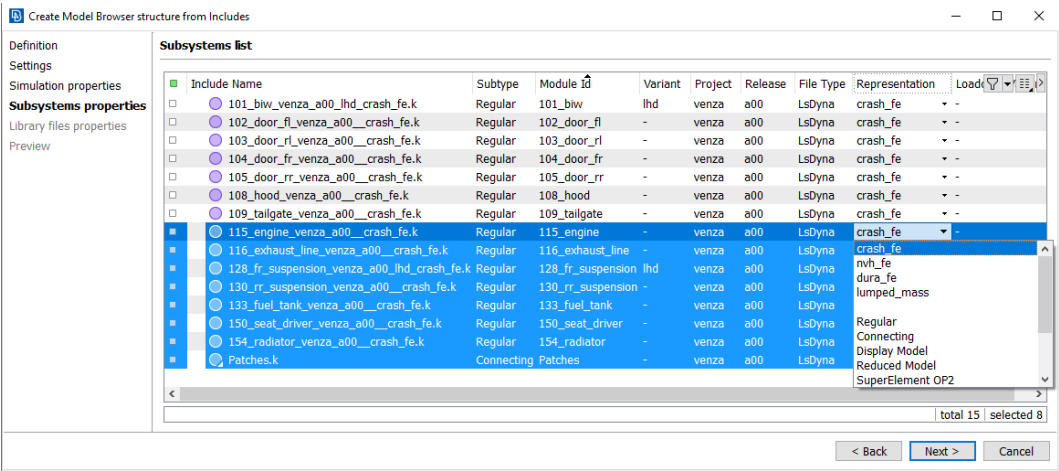
```
# ***User Defined Keywords #  
  
***interpret_attributes_script_file_path= C:\Users\demo\user_scripts\InterpretMBContainerAttributes_generic.py  
***interpret_subsystem_attributes_script_func_name = extractSimRunAttributes
```

The user script function takes as input argument an the main include filename, from which the simulation attributes will be extracted; and must return a dictionary with keys the attribute names and values the extracted attribute values.

```
def extractSimRunAttributes(main_include_file):  
    ...  
    include_props = {}  
    ...  
    return include_props
```

2.4. Subsystems properties

The fourth page of the wizard is the *Subsystems properties* page, where all the Subsystem entities that will be created based on the includes characterized as *Subsystems*, *Connecting Subsystems* and *Subsystem Groups* are listed and their properties can be defined.



Depending on the definition of the Subsystem in the data model configured through the *dm_structure.xml*, some of the properties may have a set of accepted values from which the user can select and others may be free-text fields. In addition, any properties that are common between the Subsystem and the Simulation Run will be populated with the values that were set by the user in the *Simulation properties* page of the wizard.

The tool offers the possibility to automatically extract certain properties from the include names using a custom user script. More information on this functionality is given in paragraph 2.4.1.

Subsystems list

 | | _iw_venza_a00_lhd_crash_fe.k | ☒ | | _door_fl_venza_a00_crash_fe.k | ☐ | | 103_door_rl_venza_a00_crash_fe.k | ☐ | | 104_door_fr_venza_a00_crash_fe.k | ☒ |"/>

A check is performed to identify entities that already exist in the data management repository in order to skip saving them again. The check is based on the properties of each entity and the DM status is displayed in the respective column of the *Subsystems list*.

! Notes:

- If the user has selected to save the entities in a DM not configured with a custom data model definition, the Properties that will appear in this page will be the default ANSA properties for Subsystems.
- If any of the entities have identical primary attributes (i.e. when attributes are erroneously extracted using a custom user script), they will appear in bold font in the list and the wizard cannot advance to the next page.

- If any of the properties are undefined, the wizard cannot advance to the next page.

2.4.1. Extracting properties from the subsystem include names using a custom user script

The path of the user script and the function that will be executed are defined in the ANSA defaults settings using the keywords *interpret_attributes_script_file_path* and *interpret_subsystem_attributes_script_func_name*, respectively.

```
# ***User Defined Keywords #  
***interpret_attributes_script_file_path= C:\Users\demo\user_scripts\InterpretMBContainerAttributes_generic.py  
***interpret_subsystem_attributes_script_func_name = extractSubsystemAttributes
```

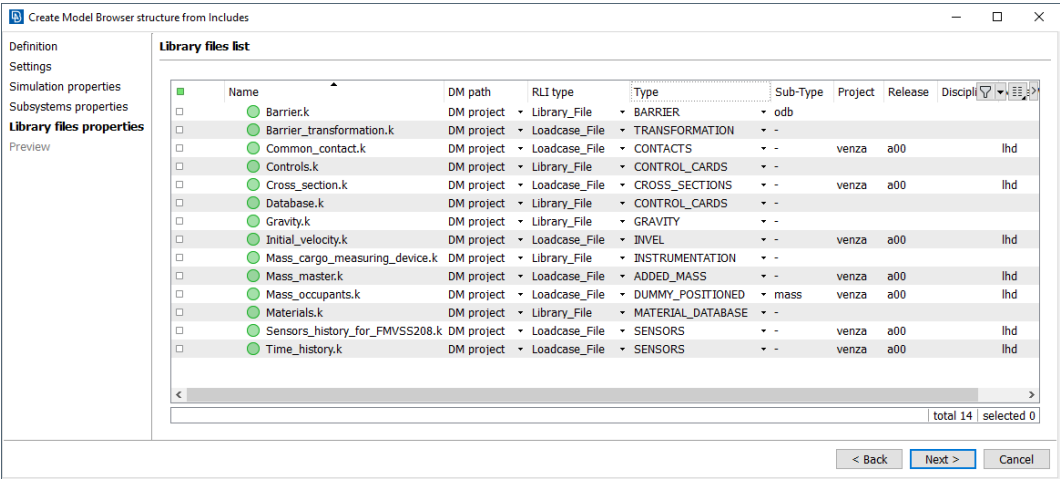
The user script function takes as input argument an include entity, for which the properties will be extracted from its name; and must return a dictionary with keys the properties names and values the extracted property values.

```
def extractSubsystemAttributes(include_ent):  
    ...  
    include_props = {}  
    ...  
    return include_props
```

2.5. Library files properties

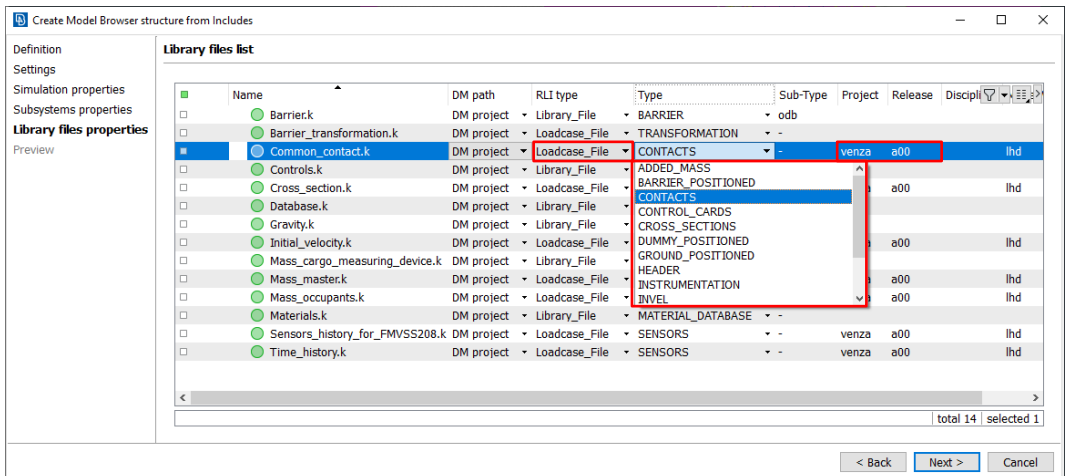
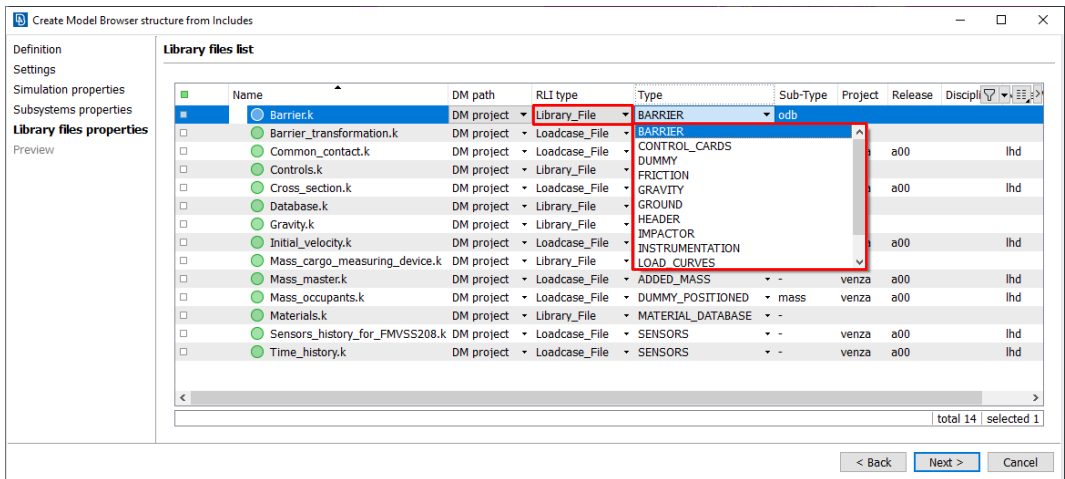
The fifth page of the wizard is the *Library files properties* page, where all the Library items that will be created based on the includes characterized as *Library Files* are listed and their properties can be defined.

Depending on the definition of Library Items in the data model configured through the *dm_structure.xml*, some of the properties may have a set of accepted values from which the user can select and others may be free-text fields. ANSA will automatically detect includes that contain transformation and material keywords only and pre-define their Type as *TRANSFORMATION* and *MATERIAL_DATABASE*, respectively, that is, of course, if the attribute *Type* and the values *TRANSFORMATION* and *MATERIAL_DATABASE* are among the accepted values of the *dm_structure.xml* used.

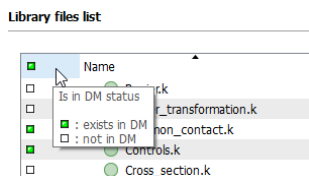


Any properties that are common between the Library Items and the Simulation Run will be populated with the values that were set by the user in the *Simulation properties* page of the wizard. As described in the previous section, the tool also offers the possibility to automatically extract certain properties from the include names using a custom user script. More information on this functionality is given in paragraph 2.4.1.

The columns that appear in the *Library files list* include all the properties of all the Library Items that have been defined in the data model. However, the relationship between the properties columns and the *RLI type* column is dynamic. The definition of each property value is required only when the respective property is defined in the data model for the selected RLI type.

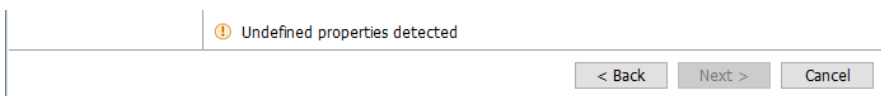


A check is performed to identify entities that already exist in the data management repository in order to skip saving them again. The check is based on the properties of each entity and the DM status is displayed in the respective column of the *Library files list*.



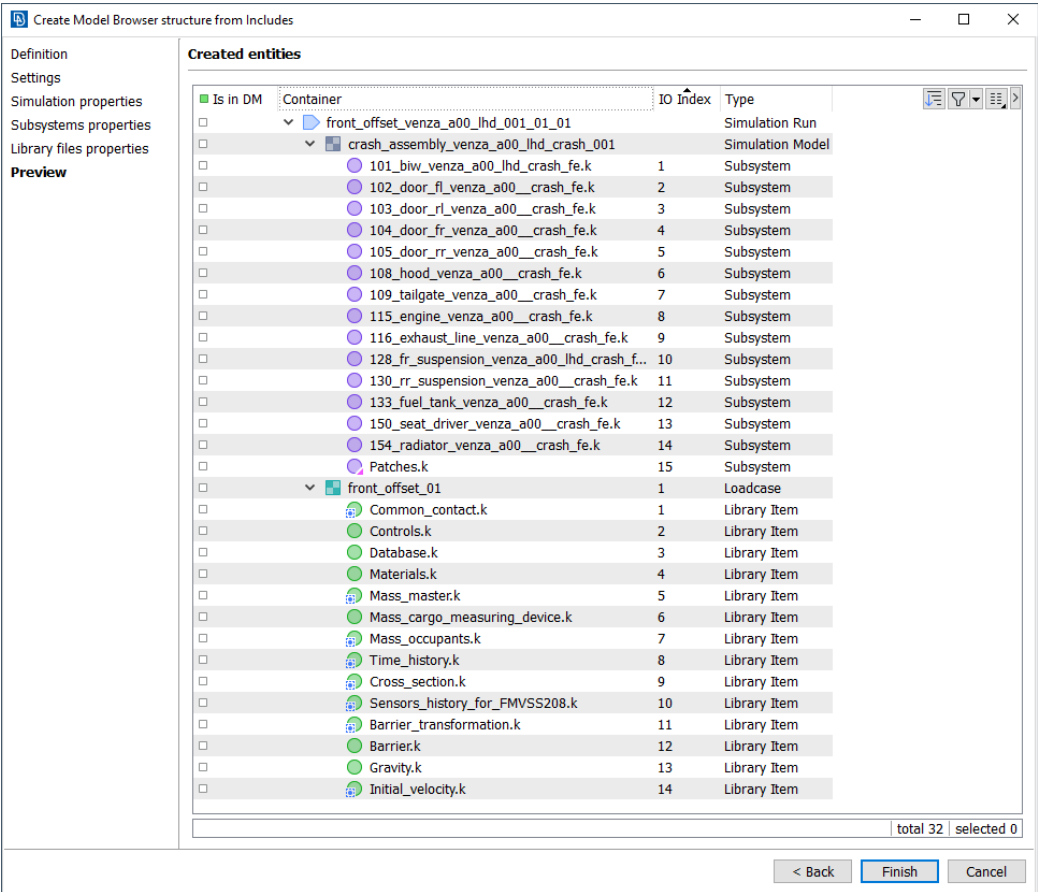
! Notes:

- If the user has selected to save the entities in a DM not configured with a custom data model definition, the Properties that will appear in this page will be the default ANSA properties for Library Files.
- If any of the properties are undefined, the wizard cannot advance to the next page.

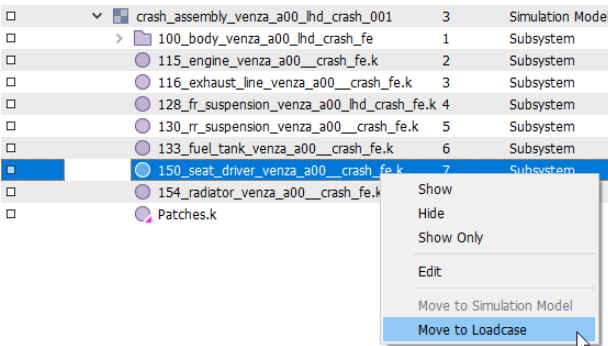


2.6. Preview

The final page of the wizard is the *Preview* page, where the *Created entities* list offers a preview of the Model Browser Containers that will be created.



The user is able to rearrange the position of the Model Browser Containers, moving Subsystems from the Simulation Model to the Loadcase or Library files from the Loadcase to the Simulation Model, using the respective context menu option

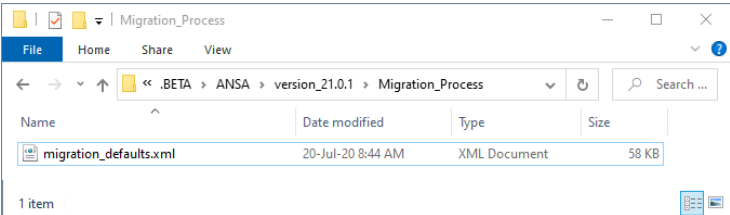


In this list the *I/O Index* of each container can be defined, using the respective column, in order to control the sequence in which the respective data will be written in the main file of the Simulation Run and the solver files of the Loadcase and the Simulation Model.


A check is performed to identify if any of the generated Model Browser Containers are already saved in DM. The DM status of each container can be seen in the *Is in DM* column. As mentioned previously, for Subsystems and Library files that already exist in DM, saving will be skipped. On the other hand, for the Simulation Model, the Loadcase and the Simulation Run, if matches are found in DM, the tool will check if the content of the generated entities is the same or different from the entities that exist in DM. If the content is the same, saving will be skipped, whereas if the content is different, new versions of these Model Browser Containers will be saved in DM.

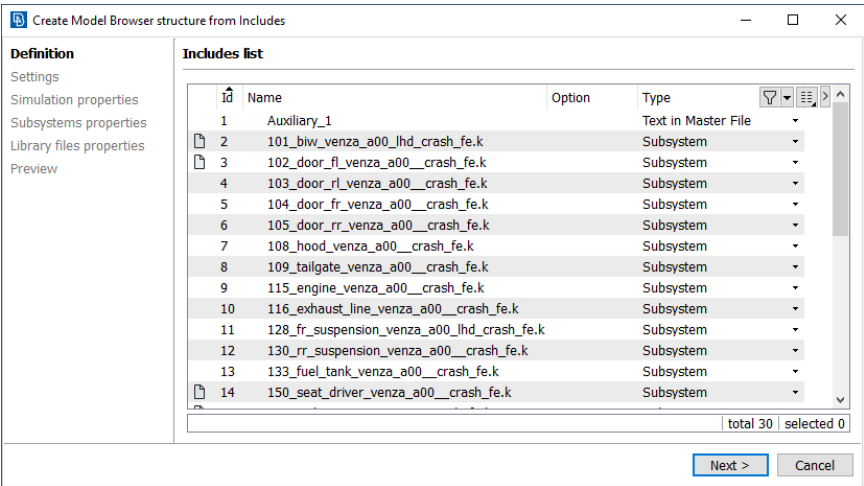
2.7. Automatic execution

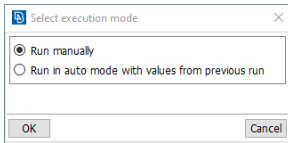
When the plugin is executed for a given model, ANSA keeps a log of the execution settings in the user's *.BETA* directory, in order to facilitate the automatic execution of the plugin for the same model again. The execution settings are saved in the *migration_defaults.xml* file, within the *Migration_Process* directory.



The *migration_defaults.xml* file stores for every model the settings defined in all pages of the plugin wizard, such as the definition of the includes types, the DM and Assembly settings, as well as the Simulation Run, Subsystems and Library Items properties. The settings are stored in two separate sections. In the section called *models_list* the plugin saves the absolute path of the migrated model as well as the settings of all the wizard pages. At the same time, in the section called *includes_library*, the plugin saves for each of the includes contained in the model, the include name as well as the individual include settings.

When the plugin is executed using a model containing includes that are also used in other models, previously migrated using the *From Includes to Model Browser* plugin, the plugin wizard is launched with settings automatically defined using the migration defaults file. The identification of the common includes is based on the same include name. In such cases, a characteristic icon  appears in front of the settings that are automatically defined in the plugin wizard.

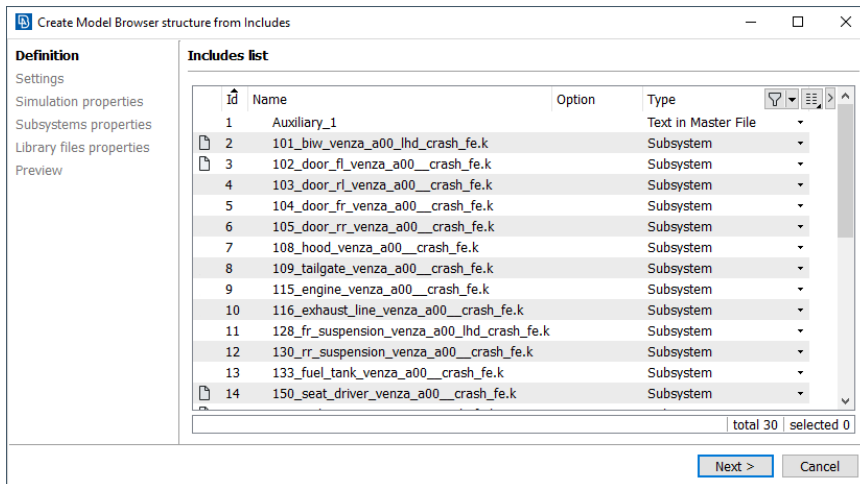




Moreover, when a previously migrated model is input in ANSA again, if its absolute file path matches the path of a model stored in the migration defaults file, the execution of the plugin is available in two modes.

Run manually

In this mode, the plugin wizard is launched with pre-defined values for all settings, as read from the migration defaults file. A characteristic icon will appear in front of the automatically defined settings. The user can modify any of the wizard settings and proceed with the plugin execution.



Run in auto mode with values from previous run

In this mode, the plugin is automatically executed in batch, with all settings taken from the *migration_defaults.xml* file. No user input is required and the plugin wizard is not launched.

3. Report and resulting model

At the end of the execution of the plugin, a *Report* window is displayed offering an overview of the migration result.

Container	Save in DM	A_POINTS	Connectors	A_SETS	Interface Sets	Upload Interfaces	More info
front_offset_venza_a00_lhd_001_01_01	✓						
crash_assembly_venza_a00_lhd_crash_001	✓						
101_biw_venza_a00_lhd_crash_fe.k	✓	262	0	0	1	✓	
102_door_fl_venza_a00_crash_fe.k	✓	3	0	0	1	✓	
103_door_rl_venza_a00_crash_fe.k	✓	3	0	0	1	✓	
104_door_fr_venza_a00_crash_fe.k	✓	3	0	0	1	✓	
105_door_rr_venza_a00_crash_fe.k	✓	3	0	0	1	✓	
108_hood_venza_a00_crash_fe.k	✓	7	0	0	1	✓	
109_tailgate_venza_a00_crash_fe.k	✓	5	0	0	1	✓	
115_engine_venza_a00_crash_fe.k	✓	2	0	0	1	✓	
116_exhaust_line_venza_a00_crash_fe.k	✓	4	0	0	0	✓	
128_fr_suspension_venza_a00_lhd_crash_fe.k	✓	160	0	0	1	✓	
130_rr_suspension_venza_a00_crash_fe.k	✓	15	0	0	1	✓	
133_fuel_tank_venza_a00_crash_fe.k	✓	4	0	0	1	✓	
150_seat_driver_venza_a00_crash_fe.k	✓	4	0	0	0	✓	
154_radiator_venza_a00_crash_fe.k	✓	4	0	0	1	✓	
Patches.k	✓	0	84	0	0	-	
front_offset_01	✓						
Barrier.k	✓	0	0	0	0	-	
Barrier_transformation.k							
Common_contact.k	✓	0	0	1	0	-	
Controls.k	✓	0	0	0	0	-	
Cross_section.k	✓	0	0	0	0	-	
Database.k	✓	0	0	0	0	-	
Gravity.k	✓	0	0	0	0	-	
Initial_velocity.k	✓	0	0	0	0	-	
Mass_cargo_measuring_device.k	✓	1	0	0	0	✓	
Mass_master.k	✓	0	0	0	0	-	
Mass_occupants.k	✓	17	0	0	1	✓	
Materials.k	✓	0	0	0	0	-	
Sensors_history_for_FMVSS208.k	✓	4	4	0	1	✓	
Time_history.k	✓	0	0	0	0	-	
total 32 selected 0							Close

The **Save in DM** column displays the status of the save operation for each container, with a characteristic icon for success (✓) and failure (✗).

The **A_POINTS** column displays the number of interface points that were created for each container.

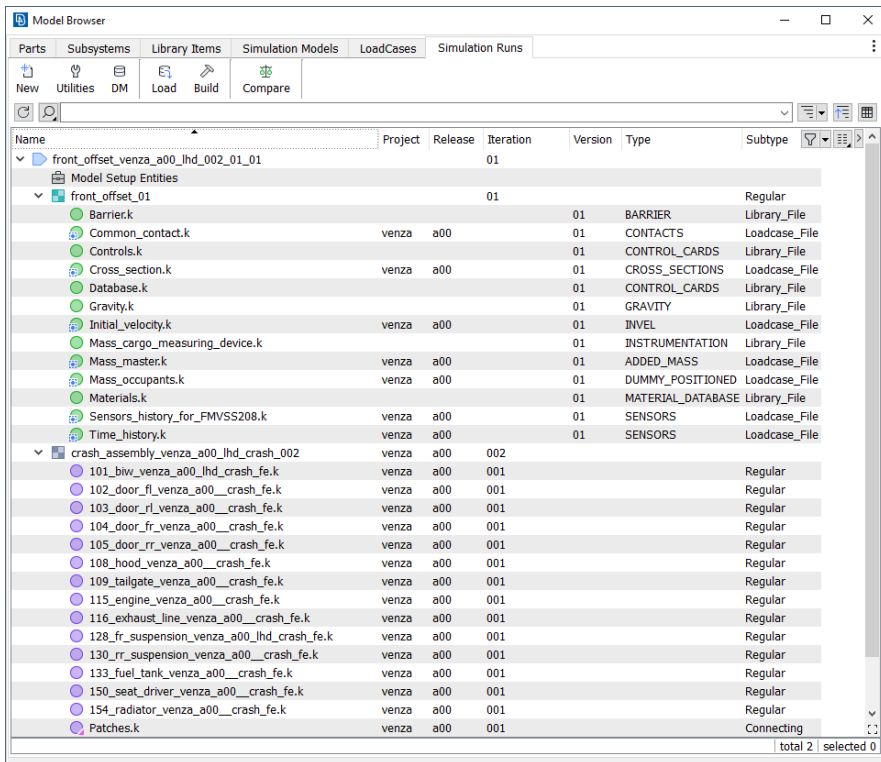
The **Connectors** column displays the number of ANSA Connectors that were created from *CONSTRAINED_RIGID_BODIES for each container.

The **A_SETS** column displays the number of assembly sets that were created for each container.

The **Interface Sets** column displays the number of interface sets that were marked in each container.

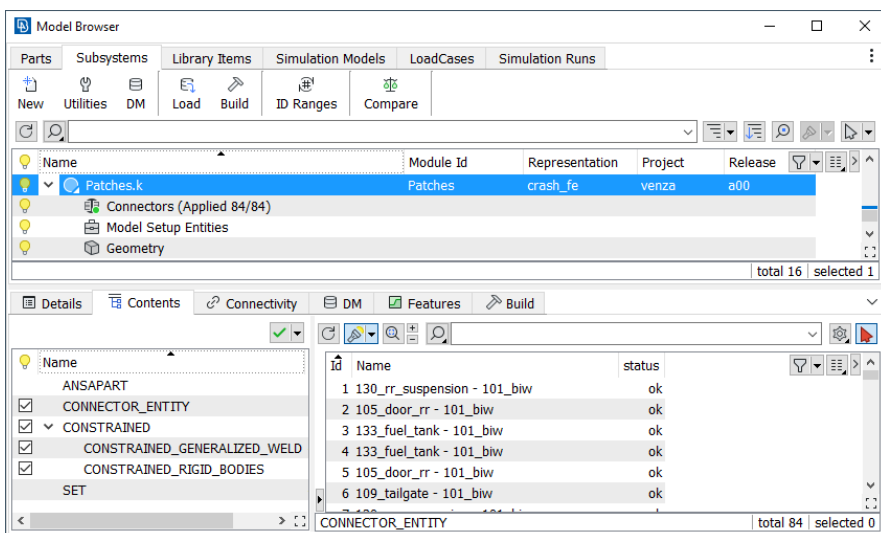
The **Upload Interfaces** displays the status of the save operation for the interface points of each container, with a characteristic icon for success (✓) and failure (✗), and a "-" for containers that contain no interface points.

After the migration of the model, the user can open the Model Browser window and find the model organized with Subsystems, Library Files, Simulation Model, Loadcases and Simulation Runs, in the respective Model Browser tabs.

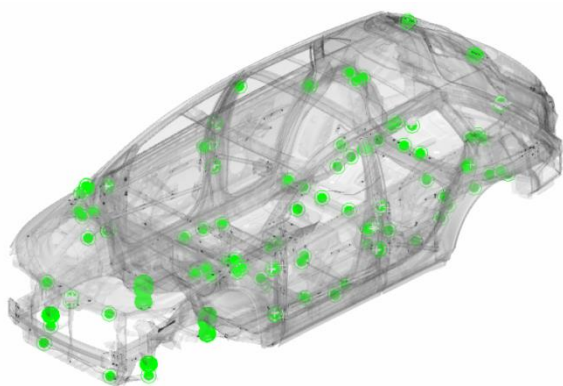


Note that Auxiliary includes have been removed and their contents are added to the Simulation Run as Model Setup Entities

In cases where the user selected mark interface points and create connectors from *CONSTRAINED_RIGID_BODIES, these can be seen in the migrated model. For each Subsystem or Library file, the *Contents* tab of the Model Browser will display all entities belonging to the specific Model Browser Container. Any ANSA Connectors that were generated from *CONSTRAINED_RIGID_BODIES will be collected under the *Connectors* container, while interface points will be collected under the *Interfaces* container. For a more detailed description of the Model Browser, please refer to Chapter 5 of the ANSA User's Guide.



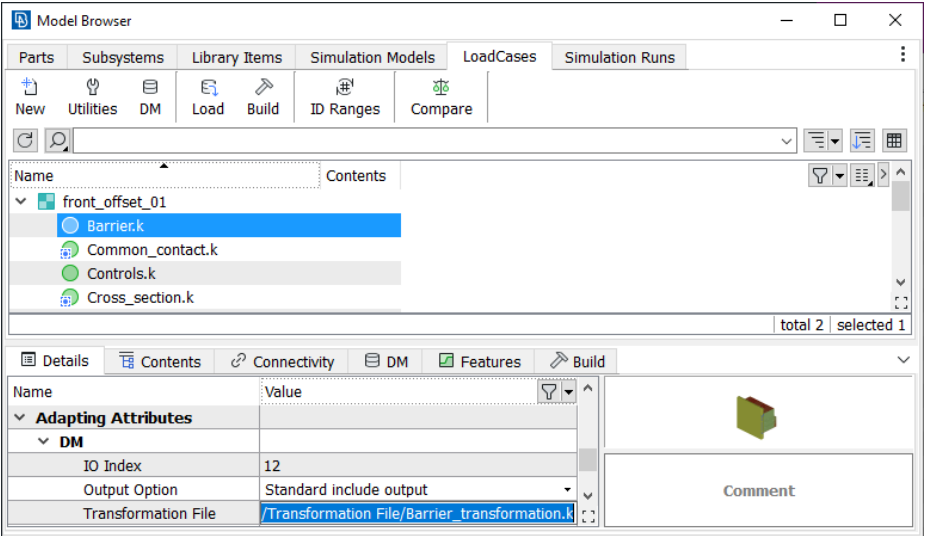
A characteristic view of interface points marked in the migrated model, as taken from the ANSA drawing area, can be seen below.

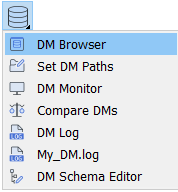


When Library files containing transformation keywords only, i.e. `*DEFINE_TRANSFORMATION`, these will be converted in the migrated model to adapting attributes of the includes that will be transformed. No Model Browser Container is generated from these includes, which are not saved in DM as separate entities and their original content is deleted from the migrated model. In the *Report* window, these includes will appear crossed out.

▼ front_offset_01	✓						
○ Barrier.k	✓	0	0	0	0	-	
✚ Barrier_transformation.k							
✚ Common_contact.k	✓	0	0	1	0	-	
○ Controls.k	✓	0	0	0	0	-	
✚ Cross_section.k	✓	0	0	0	0	-	
○ Database.k	✓	0	0	0	0	-	
○ Gravity.k	✓	0	0	0	0	-	
✚ Initial_velocity.k	✓	0	0	0	0	-	

In the Model Browser, such include files can be seen in the *Details* tab, as *Adapting attributes* of other Library items.





When the tool is combined with the data management functionality offered by ANSA and the generated Model Browser Containers are saved in DM, the user can access the data management repository using the ANSA *DM Browser*, with the option *Lists>DM>DM Browser* or using the respective button

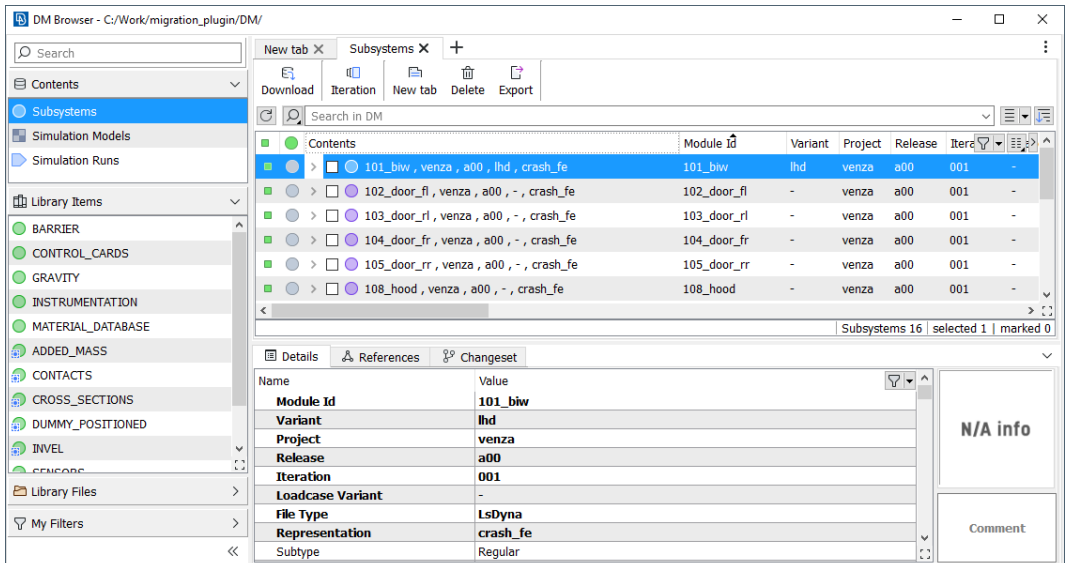
The initial include files were copied from their initial location and saved into the DM, with the only addition being the DM header that is prepended in their content ,which describes the Model Browser Container that has been generated.

```

1  $=====DM INFO BEGIN=====
2  $ANSA_DM_INFO_PRE_FORMAT;
3  $
4  $Entity Type      : Subsystem
5  $Module Id       : 116_EXHAUST_LINE
6  $Variant         : -
7  $Project         : VENZA
8  $Release        : A00
9  $Iteration       : 001
10 $Representation  : CRASH_FE
11 $Loadcase Variant :
12 $File Type       : LsDyna
13 $Name           : 116_EXHAUST_LINE_VENZA_A00_CRASH_FE_001
14 $User           : s.tzamtzis
15 $Comment        : FE representation for DM purposes
16 $
17 $END_ANSA_DM_INFO_PRE_FORMAT;
18 $=====DM INFO END=====
19 $
20 $
21 $
22 *KEYWORD
23 *NODE
24 25200092      3876.461      431.1968      142.1042      0      0
25 25200093      3876.5042      430.52176     148.3575      0      0
26 25200227      887.85126     -79.49479     32.030354     0      0

```

In the *DM Browser* window, all entities that were saved in DM can be seen under different containers.



3.1. Handling of parameters

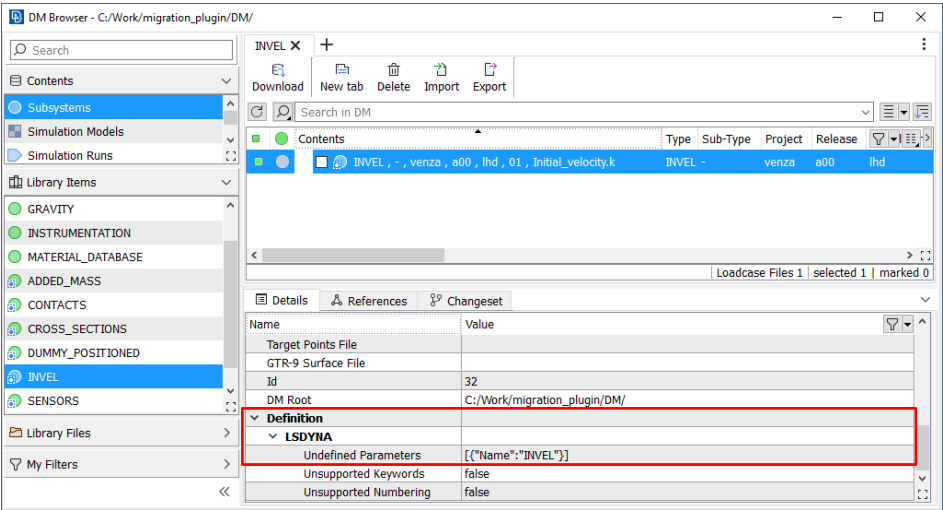
```

77 *PARAMETER
78 R FS 0.2
79 R FD 0.2
80 R INVEL -15646.
81 *INCLUDE
82 101_biw_venza_a00_lhd_crashe_fe.k
83 *INCLUDE
84 102_door_fl_venza_a00_crash_fe.k

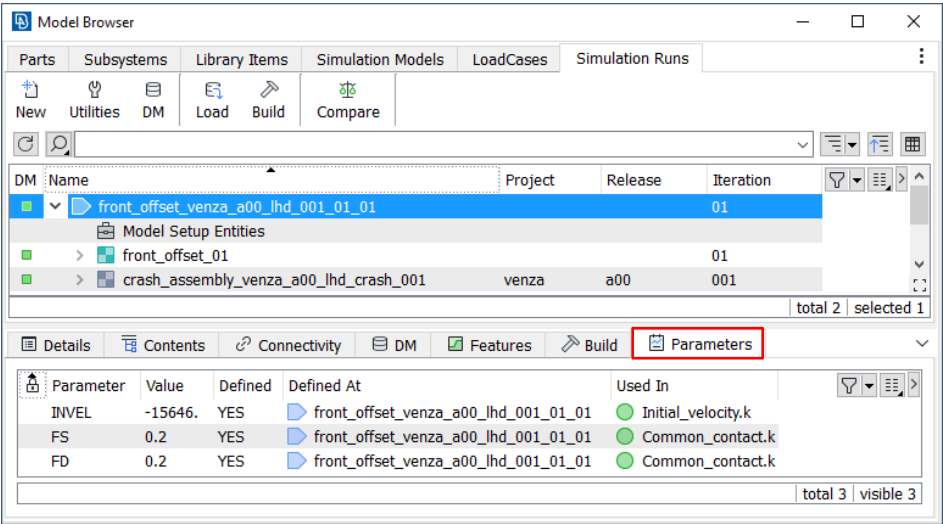
```

The plugin effectively handles parameter keywords referenced in the individual includes or defined in the model main file.

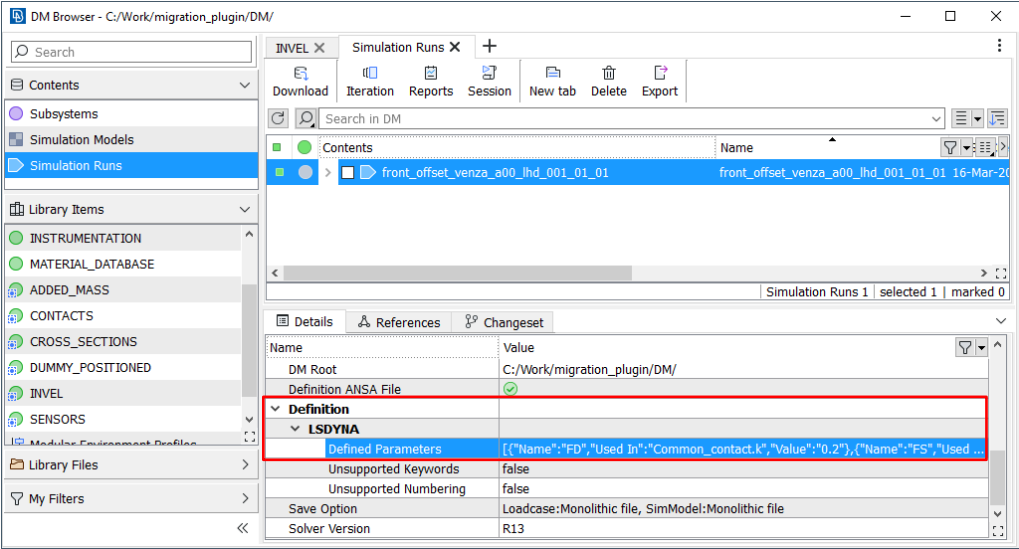
When a main file that contains parameter keywords is input, ANSA entities of type *A_PARAMETER* are created. During the saving of the individual includes in DM, information about their defined and undefined parameters is extracted from the indexing process and stored as metadata of the generated Model Browser Containers.



With the execution of the plugin, any of the parameters found in auxiliary includes of the main file will be converted in the migrated model to *MBParameter* entities and will be displayed in the *Parameters* tab of the Model Browser. Selecting the Simulation Run in the Model Browser and switching to the *Parameters* tab, the user can have a full overview of the parameter definitions, their values and where each parameter is used in the model.

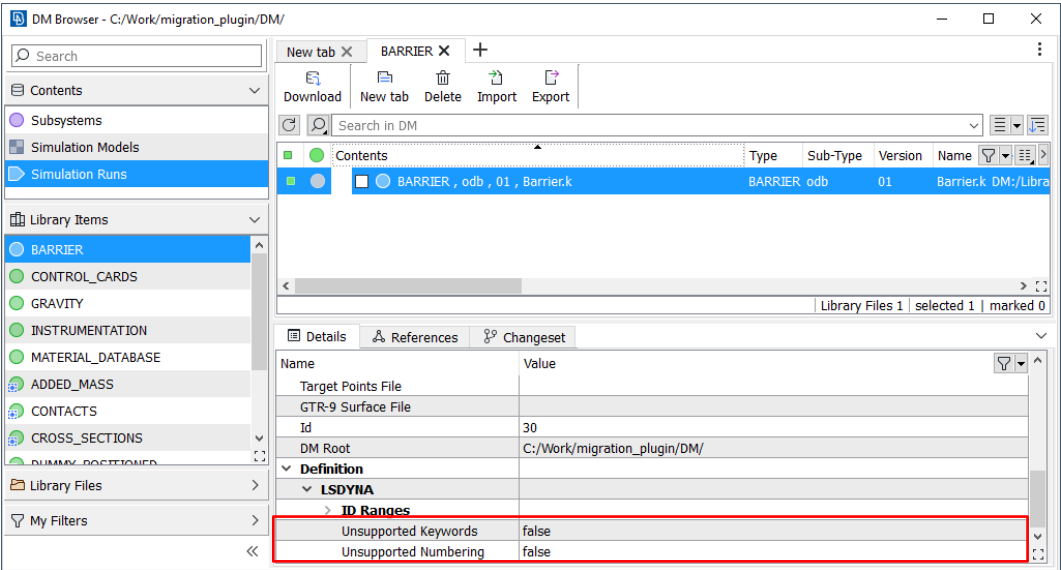


Once the generated Simulation Run is saved in DM, the parameters are indexed. The Definition Attributes metadata of the Simulation Run provide information about the defined parameters, such as the parameters names and their values.



3.2. Handling of unsupported content

The plugin also effectively handles cases where any of the original includes referenced by the main file input to ANSA contain unsupported keywords or an unsupported numbering scheme. During the saving of the Subsystem or Library Item includes in the DM, the indexing captures the existence of *unsupported keywords* or *unsupported numbering* and marks this in the definition attributes of the generated Model Browser Container, in the respective attributes. Moreover, the actual numbering ranges defined in the include file are indexed and stored as metadata of the generated Model Browser Container



In the migrated model, the Model Browser Containers generated from includes that contain unsupported content are marked as *Read Only*. This ensures that they are protected from any subsequent save operations from the Model Browser and that when the execution of their Build process is triggered, the Build Actions take into consideration the definition attributes of the Model Browser Container in the DM.

