



# Altair® **HyperWorks® 8.0**

LS-DYNA1 Solver Interface



## Altair Engineering Contact Information

**Web site**     [www.altair.com](http://www.altair.com)

**FTP site**     Address: [ftp.altair.com](ftp://ftp.altair.com) or [ftp2.altair.com](ftp://ftp2.altair.com) or <http://ftp.altair.com/ftp>  
Login: ftp  
Password: <your e-mail address>

Location	Telephone	e-mail
North America	248.614.2425	<a href="mailto:hwsupport@altair.com">hwsupport@altair.com</a>
China	86.21.5393.0011	<a href="mailto:support@altair.com.cn">support@altair.com.cn</a>
France	33.1.4133.0990	<a href="mailto:francesupport@altair.com">francesupport@altair.com</a>
Germany	49.7031.6208.22	<a href="mailto:hwsupport@altair.de">hwsupport@altair.de</a>
India	91.80.6629.4500 1.800.425.0234 (toll free)	<a href="mailto:support@india.altair.com">support@india.altair.com</a>
Italy	39.800.905.595	<a href="mailto:support@altairtorino.it">support@altairtorino.it</a>
Japan	81.3.5396.1341 81.3.5396.2881	<a href="mailto:support@altairjp.co.jp">support@altairjp.co.jp</a>
Korea	82.31.716.4321	<a href="mailto:support@altair.co.kr">support@altair.co.kr</a>
Scandinavia	46.46.286.2052	<a href="mailto:support@altair.se">support@altair.se</a>
United Kingdom	44.1926.468.600	<a href="mailto:support@uk.altair.com">support@uk.altair.com</a>
Brazil	55.11.4223.5733	<a href="mailto:br_support@altair.com">br_support@altair.com</a>
Australia	64.9.413.7981	<a href="mailto:anzsupport@altair.com">anzsupport@altair.com</a>
New Zealand	64.9.413.7981	<a href="mailto:anzsupport@altair.com">anzsupport@altair.com</a>

The following countries have distributors for Altair Engineering: Mexico, Romania, Russia, South Korea, Singapore, Spain, Taiwan and Turkey. See [www.altair.com](http://www.altair.com) for complete contact information.

© 2007 Altair Engineering, Inc. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated to another language without the written permission of Altair Engineering, Inc. To obtain this permission, write to the attention Altair Engineering legal department at: 1820 E. Big Beaver, Troy, Michigan, USA, or call +1-248-614-2400.

### Trademark and Registered Trademark Acknowledgments

Listed below are Altair® HyperWorks® applications. Copyright® Altair Engineering Inc., All Rights Reserved for:

HyperMesh® 1990-2006; HyperView® 1999-2006; OptiStruct® 1996-2006; HyperStudy® 1999-2006; HyperGraph® 1995-2006; HyperGraph 3D 2005-2006; MotionView® 1993-2006; MotionSolve® 2002-2006; HyperForm® 1998-2006; HyperXtrude® 1999-2006; HyperOpt® 1996-2006; HyperView Player® 2001-2006; Process Manager™ 2003-2006; HyperWeb® 2002-2004; Data Manager™ 2005-2006; Templex™ 1990-2006; Manufacturing Solutions™ 2005-2006

All other trademarks and registered trademarks are the property of their respective owners.

---

# LS-DYNA Solver Interface HyperWorks 8.0

<b>HyperView and MotionView</b> .....	<b>1</b>
Exporting MDL Models to LS-DYNA.....	1
MotionView Model API.....	5
MotionView Result API.....	13
HyperView Interfacing.....	17
HyperView LS-DYNA 3-D Results Reader.....	17
Post-processing LS-DYNA ALE Results .....	18
<b>HyperMesh</b> .....	<b>19</b>
LS-DYNA Interface Overview .....	19
Recommended Process .....	19
Supported LS-DYNA Keywords.....	22
Importing Decks .....	37
Summary Templates.....	38
Mass Calculation.....	39
LS-DYNA MAT100 Spotwelds .....	39
Duplicate Entity IDs.....	42
Components and Properties .....	43
Load Collectors .....	44
Loads, Constraints, and Boundary Conditions .....	45
HyperMesh Groups .....	46
Rigid Walls .....	49
Creating Cards .....	50
Equations .....	52
Curves .....	52
Control Volumes.....	53
Results Translation .....	54
Viewing the Results .....	61
Exporting Decks .....	61
LS-DYNA Utility Menu.....	62
Error Check.....	64
Part Info.....	65

Name Mapping.....	65
Clone Part .....	66
Create Part.....	66
Part Replacement Macro .....	66
Constrained Rgd Body.....	71
LS-DYNA Content Table.....	73
LS-DYNA Material Table .....	77
Customizing Views of the Material Table .....	77
Creating, Editing, and Loading Materials .....	78
Managing Materials.....	79

# HyperView and MotionView

## Exporting MDL Models to LS-DYNA

MotionView MDL models can be exported to an LS-DYNA .key file.

### Gravity, Units, and Solver Parameters

This section explains how MotionView works with gravity, units, and solver parameters for DYNA.

#### Gravity

Gravity is an implicit data set, meaning that its definition is created automatically by MotionView. The values for gravity can be accessed through the **Forms** panel while in the **Misc** system of a model. Default values for gravity are set in the `std_inc` file that is part of the MotionView installation. A Templex template is also included in `std_inc`. This template allows you to write the gravity to the DYNA input file.

#### Units

Although MotionView is a "unitless" interface, it is often required that the units you are working with be communicated to the solver input deck. Therefore, the definitions of mass, length, time, and force are automatically generated by MotionView. To access this, go to the **Forms** panel and select **Units** under the **Misc** system. The default values as well as the Templex template used for exporting units to DYNA are generated from the `std_inc` file.

#### Solver Parameters

Solver parameters vary considerably between different solvers and are stored in datasets. There are two ways to generate solver parameter datasets.

1. Data sets and their corresponding forms are created within the analysis task in the MDL library. If you are creating your own MDL library, you need to verify that the solver parameter datasets are defined in each analysis task.
2. No library is used to construct a model. This includes interactive model construction and the manual editing of .mdl files, or a combination of both. For this case, MotionView automatically generates the system containing the solver parameters based on a definition within the `std_inc` file.

### MDL Statement Mapping

The mapping between MDL and the corresponding DYNA entities are described below.

**Note** All property data for these entities are set in the corresponding `*Set()` statements

MDL Statement	DYNA Entity
<code>*ActionOnlyForce()</code>	None
<code>*ActionReactionForce()</code>	None
<code>*AtPointJoint()</code>	Spherical (Joint)
<code>*BallJoint()</code>	Spherical (Joint)
<code>*Beam()</code>	None
<code>*Body()</code>	Part_Inertia
<code>*Bush()</code>	Element_Discrete

*CoilSpring()	None
*ControlSISO()	None
*Coupler()	None
*Curve()	AMPLITUDE
*CVJoint()	Constant_Velocity
*CylJoint()	Cylindrical
*FixedJoint()	Rigid_Bodies
*Graphic()	None
*HookeJoint()	Universal
*InlineJoint()	Trans+D.B.+Sph
*InplaneJoint()	Planar+D.B.+Univ
*Marker()	
*Motion()	Constrained_Joint#_Motor
*OrientJoint()	Trans+D.B.+Tran+D.B.+Trans
*Output()	None
*ParallelAxisJoint()	Planar+D.B.+Trans
*PerpAxisJoint()	Cyl+D.B.+Planar
*PlanarJoint()	Planar
*Polybeam()	None
*RevJoint()	Revolute
*SetFlexbodyComplaine()	n/a
*SolverArray()	None
*SolverDiffEquation()	None
*SolverString()	None
*SolverVariables()	None
*TorsionSpring()	None
*TransJoint()	Translational
*UniversalJoint()	Universal

## MDL CommandSet Mapping

CommandSet's do not apply to the DYNA solver.

## Templex Templates and solvermode

Templex templates can be used to export syntax directly to the solver input deck, including parametric substitution if required. For the LS-DYNA solver, there is a dependency on the position of the syntax within the solver input deck. The four keywords listed below allow you to position the extra text. These keywords must be the first line of the Templex template. The remaining text of the template is written according to the position specified.

<@ABAQ/MODEL/HEAD>	Designates text that is written at the top of the model data.
<@ABAQ/MODEL/TAIL>	Indicates the end of the model data.
<@ABAQ/HISTO/HEAD>	Indicates the top of the history data.
<@ABAQ/HISTO/TAIL>	Indicates the end of the history data.

### **solvermode**

One MDL model can be used to export to more than one solver. In this case, create the instance of the Templex template using the `solvermode` reserved keyword. This can be done in two ways:

For example, an MDL model containing:

```
if( solvermode == "DYNA" )
    *Template(.....1...)
else
    *Template(.....2...)
endif
```

results in the entire template 1 being used when DYNA is selected from the **Solvers** menu. When another solver is selected, template 2 is used. When a template is used, it means that it is displayed in the interface on the **Templates** panel and is acted upon when saving the solver input deck.

To use the keyword, put the required string in the first line of the template. For example, an MDL model containing:

```
*DefineTemplate(.....)
    <@DYNA/MODEL/HEAD>
    text for dyna
*EndDefine()
```

results in "text for dyna" being exported to the input deck when you select DYNA as the solver. The same applies for the portion of template that is displayed in the user interface.

## Template Types

A Templex template can have several destinations as well as unique default behavior.

- A USER template does not get exported into any solver file but can be used to get parametrically based text into another file (by using the Templex open and close commands) or for text targeted for the GUI only.
- A SOLVER\_INPUT template results in the template text being exported to the .key file for DYNA.
- The following templates do not apply to the DYNA solver:
  - SOLVER\_PARAM
  - GRAPHICS
  - ADAMS
  - ACF

## Function Expressions

MotionView supports function expressions for many of its entities. These expressions can be a function of time and state variables. You can create function expressions that are exported directly as part of a corresponding solver entity.

The solver neutrality is somewhat limited because the solver needs to handle the syntax that MotionView exports. For the DYNA solver, supported expressions and curves must be a function of time. Expressions that are a function of an axial distance between two points are also supported. MotionView converts these to X/Y pairs for the input deck.

## Flexbodies/Substructures

Flexbodies are not supported in DYNA.

## User Subroutines

User subroutines do not apply to DYNA. Entities with a reference to user subs are not used when exporting to the solver input deck.

## Launching Solvers from MotionView

MotionView allows you to launch a process automatically after the solver input deck is exported. For LS-DYNA, the installation contains default launch sessions located in `altair/utilities/mbd/launch_scripts`. One or more of these launch sessions can be registered through the `preferences.mvw` file and then selected from the **Run** panel by using the `*RegisterSolverScript()` preference statement.

## Post-Processing

DYNA post-processing is described below:

**Animation - Transient with Rigid Bodies Only**  
**Rigid body model animation is not supported.**

### Plotting

**Plotting rigid body model results is not supported.**



## MotionView Model API

Please contact MotionView Support to obtain information about a model API example.

The Model API consists mainly of a group of *SendOff* functions that serve as the interface for passing model data from the MDL file to your solver writer program. They are activated by calling `API_MDLMoDelRead(const char *filename)`.

Model data is passed via the arguments in the *SendOff* functions. These functions are defined as *virtual* in the class "Model\_API". You must define your own class derived from "Model\_API" then handles the passed data in the derived class. The relationship is transparent in the working examples.

Presently, only one of the virtual functions is an *Inquiry Function*, i.e., `API_InquiryFlexBodySendOffOption` which returns your selection back to the API. All other API functions, except the Inquiry Function and `API_MDLMoDelRead` are one-directional, (they only send off data to the user).

The following is the Model\_API class that includes a list of the *SendOff* functions. The names of the functions are self-explanatory.

```
class MODEL_API Model_API
{
public:

    bool API_MDLMoDelRead(const char *filename);
    bool API_MDLMoDelRead(MDL *user_mdl);
    static ostream &(* apiOut)();

private:

    bool _API_MDLMoDelRead(MDL *user_mdl, const char *filename);

// #1
    virtual void API_SendOffModelUnit(const char* force_unit, const char*
length_unit,
        const char* mass_unit, const char* time_unit) {}

// #2
    virtual void API_SendOffGravityVector(const double gravity_x, const double
gravity_y,
        const double gravity_z) {}

// #3
    virtual void API_SendOffMarker(const int num_marker, const int idx,
        const int id, const int body_id, const int node_id,
        const double x, const double y, const double z,
        const double a00, const double a01, const double a02,
        const double a10, const double a11, const double a12,
        const double a20, const double a21, const double a22) {}

// #4
    virtual void API_SendOffRigidBody(const int num_rigid_body, const int idx,
        const int id, const bool IsGround,
        const int cg_id, const int im_id, const int lprf_id,
        const double mass, const double ixx, const double iyy, const double
izz,
        const double ixy, const double iyz, const double ixz,
```

```

        const double vx,    const double vy,    const double vz,
        const double wx,    const double wy,    const double wz) {}

// #5
    // option: <=0    -> global property (rigid body property only)
    //      : 1      -> SIMPACK SID_FEM file along with other info
    //      : 2      -> DADS fdf file along with other info
    //      : 3      -> ADAMS mtx file along with other info
    //      : 4      -> MADYMO dat file along with other info
    //      : >=5    -> modal matrix info (nodal mass, node position,
    //                  translational mode shapes, rotational mode shapes,
    //                  diagonal of modal stiffness matrix)
    virtual int API_InquiryFlexBodySendOffOption(const int num_flex_body, const
int idx,
        const int id) {return 999999;}

// #6
    virtual void API_SendOffFlexBodyGlobalProperty(const int num_flex_body,
const int idx,
        const int id, const int lprf_id, const double mass,
        const double cm_x, const double cm_y, const double cm_z,
        const double ixx, const double iyy, const double izz,
        const double ixy, const double iyz, const double ixz,
        const double vx,    const double vy,    const double vz,
        const double wx,    const double wy,    const double wz) {}

// #7
    virtual void API_SendOffFlexBody(const int num_flex_body, const int idx,
        const int id, const int lprf_id, const char* h3d_file,
const char* fdf_file, const int sel_mode_count,
        const int *sel_mode, const double *mode_freq, const double
*cdamp_ratio,
        const double *DMode, const double *VMode,
        const double vx,    const double vy,    const double vz,
        const double wx,    const double wy,    const double wz) {}

// #8
    virtual void API_SendOffNodeInfo(const int num_flex_body, const int idx,
        const int id, const int num_nodes, const int *node_id,
        const double *node_position_x, const double *node_position_y,
const double *node_position_z,
        const double *node_mass, const int HasNodeInertia) {}

// #9
    virtual void API_SendOffNodeInertia(const int num_flex_body, const int idx,
        const int id, const int num_nodes,
        const double *ixx, const double *iyy, const double *izz,
        const double *ixy, const double *iyz, const double *ixz) {}

// #10
    virtual void API_SendOffOrthogonalizedModeInfo(const int num_flex_body,
const int idx,
        const int id, const int num_modes, const int mode_idx, const int
mode_id, const double frequency,
        const double modal_stiffness, const double modal_mass, const int
num_nodes,
        const double *modeshape_x, const double *modeshape_y, const double
*modeshape_z,

```

```

        const double *modeshape_rx, const double *modeshape_ry, const double
*modeshape_rz) {}
// #11
    virtual void API_SendOffBallJoint(const int num_ball_joint, const int idx,
        const int id, const int i_marker_id, const int j_marker_id,
        const int body1_id, const int body2_id) {}
// #12
    virtual void API_SendOffFixedJoint(const int num_fixed_joint, const int
idx,
        const int id, const int i_marker_id, const int j_marker_id,
        const int body1_id, const int body2_id) {}
// #13
    virtual void API_SendOffRevJoint(const int num_rev_joint, const int idx,
        const int id, const int i_marker_id, const int j_marker_id,
        const int body1_id, const int body2_id) {}
// #14
    virtual void API_SendOffTransJoint(const int num_trans_joint, const int
idx,
        const int id, const int i_marker_id, const int j_marker_id,
        const int body1_id, const int body2_id) {}
// #15
    virtual void API_SendOffCylJoint(const int num_cyl_joint, const int idx,
        const int id, const int i_marker_id, const int j_marker_id,
        const int body1_id, const int body2_id) {}
// #16
    virtual void API_SendOffUnivJoint(const int num_univ_joint, const int idx,
        const int id, const int i_marker_id, const int j_marker_id,
        const int body1_id, const int body2_id) {}
// #17
    virtual void API_SendOffInlineJoint(const int num_inline_joint, const int
idx,
        const int id, const int i_marker_id, const int j_marker_id,
        const int body1_id, const int body2_id) {}
// #18
    virtual void API_SendOffOrientationJoint(const int num_orientation_joint,
const int idx,
        const int id, const int i_marker_id, const int j_marker_id,
        const int body1_id, const int body2_id) {}
// #19
    virtual void API_SendOffCVJoint(const int num_convel_joint, const int idx,
        const int id, const int i_marker_id, const int j_marker_id,
        const int body1_id, const int body2_id) {}
// #20
    virtual void API_SendOffPerpendicularJoint(const int
num_perpendicular_joint,
const int idx,
        const int id, const int i_marker_id, const int j_marker_id, const
int body1_id,
        const int body2_id) {}
// #21
    virtual void API_SendOffInplaneJoint(const int num_inplane_joint,
const int idx, const int id,

```

```

        const int i_marker_id, const int j_marker_id, const int body1_id,
const int body2_id) {}
// #22
    virtual void API_SendOffPlanarJoint(const int num_planar_joint,
const int idx, const int id,
        const int i_marker_id, const int j_marker_id, const int body1_id,
const int body2_id) {}
// #23
    virtual void API_SendOffParallelAxesJoint(const int num_par_axes_joint,
const int idx,
        const int id, const int i_marker_id, const int j_marker_id,
const int body1_id, const int body2_id) {}
// #24
    virtual void API_SendOffThreeJointCoupler(const int
num_three_joint_coupler,
const int idx, const int id,
        const int i_marker_id_joint1, const int j_marker_id_joint1,
const int body1_id_joint1, const int body2_id_joint1,
const char joint_type_joint1,
        const int i_marker_id_joint2, const int j_marker_id_joint2,
const int body1_id_joint2, const int body2_id_joint2,
const char joint_type_joint2,
        const int i_marker_id_joint3, const int j_marker_id_joint3,
const int body1_id_joint3, const int body2_id_joint3,
const char joint_type_joint3,
const double ratio1, const double ratio2) {}
// #25
    virtual void API_SendOffTwoJointCoupler(const int num_two_joint_coupler,
const int idx, const int id,
        const int i_marker_id_joint1, const int j_marker_id_joint1,
const int body1_id_joint1, const int body2_id_joint1,
const char joint_type_joint1,
        const int i_marker_id_joint2, const int j_marker_id_joint2,
const int body1_id_joint2, const int body2_id_joint2,
const char joint_type_joint2,
const double ratio) {}
// #26
    virtual void API_SendOffConstantJointMotion(const int num_expr_motion,
const int idx,
        const int id, const int joint_id, const char joint_type,
const char motion_type, const double val) {}
// #27
    virtual void API_SendOffExpressionJointMotion(const int num_expr_motion,
const int idx,
        const int id, const int joint_id, const char joint_type,
const char motion_type, const char *motion_expr) {}
// #28
    virtual void API_SendOffCurveJointMotion(const int num_curve_motion,
const int idx, const int id,
        const int joint_id, const char joint_type, const char motion_type,
const char *interpol_type, const int curve_id) {}

```

```

// #29
    virtual void API_SendOffLinearTranslationalSpringDamper(const int
num_lin_tsd,
const int idx,
                const int id, const int i_marker_id, const int j_marker_id,
                const int body1_id, const int body2_id,
                const double length, const double stiffness, const double damping,
const double preload) {}
// #30
virtual void API_SendOffGeneralTranslationalSpringDamper(
const int num_general_tspd, const int idx, const int id,
                const int i_marker_id, const int j_marker_id, const int body1_id,
const int body2_id, const double length,
                const int is_k_expression, const char *k_expression,
const int is_c_expression, const char *c_expression,
                const int is_k_curve, const char *k_interpol_type,
const char *k_indep_var, const int k_curve_id,
                const int is_c_curve, const char *c_interpol_type,
const char *c_indep_var, const int c_curve_id) {}
// #31
    virtual void API_SendOffLinearRotationalSpringDamper(const int num_lin_rsd,
const int idx,
                const int id, const int i_marker_id, const int j_marker_id,
                const int body1_id, const int body2_id,
                const double length, const double stiffness, const double damping,
const double preload) {}
// #32
    virtual void API_SendOffGeneralRotationalSpringDamper(
const int num_general_rspdp, const int idx, const int id,
                const int i_marker_id, const int j_marker_id, const int body1_id,
const int body2_id, const double length,
                const int is_k_expression, const char *k_expression,
const int is_c_expression, const char *c_expression,
                const int is_k_curve, const char *k_interpol_type,
const char *k_indep_var, const int k_curve_id,
                const int is_c_curve, const char *c_interpol_type,
const char *c_indep_var, const int c_curve_id) {}
// #33
    virtual void API_SendOffBeam(const int num_beam, const int idx,
const int id, const int i_marker_id,
                const int j_marker_id, const int body1_id, const int body2_id,
                const double length, const double E, const double G, const double
area,
                const double ixx, const double iyy, const double ASY, const double
ASZ,
const double cratio) {}

// #34
    virtual void API_SendOffLinearBushing(const int num_lin_bush,
const int idx, const int id,
                const int i_marker_id, const int j_marker_id,
const int body1_id, const int body2_id,

```

```

        const double kx, const double cx, const double preload_x,
        const double ky, const double cy, const double preload_y,
        const double kz, const double cz, const double preload_z,
        const double ktx, const double ctx, const double preload_tx,
        const double kty, const double cty, const double preload_ty,
        const double ktz, const double ctz, const double preload_tz) {}

// #35
    virtual void API_SendOffGeneralBushing(const int num_general_bush,
const int idx, const int id,
        const int i_marker_id, const int j_marker_id,
const int body1_id, const int body2_id,
        const int is_kx_expression, const char *kx_expression,
const int is_cx_expression, const char *cx_expression,
        const int is_kx_curve, const char *kx_interpol_type,
const char *kx_indep_var, const int kx_curve_id,
        const int is_cx_curve, const char *cx_interpol_type,
const char *cx_indep_var, const int cx_curve_id,
        const int is_ky_expression, const char *ky_expression,
const int is_cy_expression, const char *cy_expression,
        const int is_ky_curve, const char *ky_interpol_type,
const char *ky_indep_var, const int ky_curve_id,
        const int is_cy_curve, const char *cy_interpol_type,
const char *cy_indep_var, const int cy_curve_id,
        const int is_kz_expression, const char *kz_expression,
const int is_cz_expression, const char *cz_expression,
        const int is_kz_curve, const char *kz_interpol_type,
const char *kz_indep_var, const int kz_curve_id,
        const int is_cz_curve, const char *cz_interpol_type,
const char *cz_indep_var, const int cz_curve_id,
        const int is_ktx_expression, const char *ktx_expression,
const int is_ctx_expression, const char *ctx_expression,
        const int is_ktx_curve, const char *ktx_interpol_type,
const char *ktx_indep_var, const int ktx_curve_id,
        const int is_ctx_curve, const char *ctx_interpol_type,
const char *ctx_indep_var, const int ctx_curve_id,
        const int is_kty_expression, const char *kty_expression,
const int is_cty_expression, const char *cty_expression,
        const int is_kty_curve, const char *kty_interpol_type,
const char *kty_indep_var, const int kty_curve_id,
        const int is_cty_curve, const char *cty_interpol_type,
const char *cty_indep_var, const int cty_curve_id,
        const int is_ktz_expression, const char *ktz_expression,
const int is_ctz_expression, const char *ctz_expression,
        const int is_ktz_curve, const char *ktz_interpol_type,
const char *ktz_indep_var, const int ktz_curve_id,
        const int is_ctz_curve, const char *ctz_interpol_type,
const char *ctz_indep_var, const int ctz_curve_id) {}

// #36
    virtual void API_SendOffActionOnlyConstantForce(
const int num_action_only_const_force, const int idx,

```

```

        const int id, const int i_marker_id, const int ref_marker_id,
const int body_id, const int ref_body_id,
        const double fx, const double fy, const double fz) {}
// #37
    virtual void API_SendOffActionOnlyConstantTorque(
const int num_action_only_const_torque, const int idx,
        const int id, const int i_marker_id,
const int ref_marker_id, const int body_id, const int ref_body_id,
        const double tx, const double ty, const double tz) {}
// #38
    virtual void API_SendOffActionReactionConstantForce(
const int num_action_reac_const_force, const int idx,
        const int id, const int i_marker_id,
const int j_floating_marker_id, const int ref_marker_id,
        const int body1_id, const int body2_id, const int ref_body_id,
        const double fx, const double fy, const double fz) {}
// #39
    virtual void API_SendOffActionReactionConstantTorque(
const int num_action_reac_const_torque, const int idx,
        const int id, const int i_marker_id,
const int j_floating_marker_id, const int ref_marker_id,
        const int body1_id, const int body2_id, const int ref_body_id,
        const double tx, const double ty, const double tz) {}
// #40
    virtual void API_SendOffConstantScalarForce(
const int num_const_scalar_force, const int idx,
        const int id, const int i_marker_id, const int j_marker_id,
        const int body1_id, const int body2_id, const double force) {}
// #41
    virtual void API_SendOffConstantScalarTorque(
const int num_const_scalar_torque, const int idx,
        const int id, const int i_marker_id, const int j_marker_id,
        const int body1_id, const int body2_id, const double torque) {}
// #42
    virtual void API_SendOffActionOnlyGeneralForce(
const int num_action_only_const_force, const int idx,
        const int id, const int i_marker_id,
const int ref_marker_id, const int body_id, const int ref_body_id,
        const int is_fx_expression, const char *fx_expression,
        const int is_fy_expression, const char *fy_expression,
        const int is_fz_expression, const char *fz_expression,
        const int is_fx_curve, const char *fx_interpol_type,
const char *fx_indep_var, const int fx_curve_id,
        const int is_fy_curve, const char *fy_interpol_type,
const char *fy_indep_var, const int fy_curve_id,
        const int is_fz_curve, const char *fz_interpol_type,
const char *fz_indep_var, const int fz_curve_id) {}
// #43
    virtual void API_SendOffActionOnlyGeneralTorque(
const int num_action_only_general_torque, const int idx,
        const int id, const int i_marker_id,

```

```

const int ref_marker_id, const int body_id, const int ref_body_id,
      const int is_tx_expression, const char *tx_expression,
      const int is_ty_expression, const char *ty_expression,
      const int is_tz_expression, const char *tz_expression,
      const int is_tx_curve, const char *tx_interpol_type,
const char *tx_indep_var, const int tx_curve_id,
      const int is_ty_curve, const char *ty_interpol_type,
const char *ty_indep_var, const int ty_curve_id,
      const int is_tz_curve, const char *tz_interpol_type,
const char *tz_indep_var, const int tz_curve_id) {}
// #44
    virtual void API_SendOffActionReactionGeneralForce(
const int num_action_reac_general_force, const int idx,
      const int id, const int i_marker_id,
const int j_floating_marker_id, const int ref_marker_id,
      const int body1_id, const int body2_id, const int ref_body_id,
      const int is_fx_expression, const char *fx_expression,
      const int is_fy_expression, const char *fy_expression,
      const int is_fz_expression, const char *fz_expression,
      const int is_fx_curve, const char *fx_interpol_type,
const char *fx_indep_var, const int fx_curve_id,
      const int is_fy_curve, const char *fy_interpol_type,
const char *fy_indep_var, const int fy_curve_id,
      const int is_fz_curve, const char *fz_interpol_type,
const char *fz_indep_var, const int fz_curve_id) {}
// #45
    virtual void API_SendOffActionReactionGeneralTorque(
const int num_action_reac_general_torque, const int idx,
      const int id, const int i_marker_id,
const int j_floating_marker_id, const int ref_marker_id,
      const int body1_id, const int body2_id, const int ref_body_id,
      const int is_tx_expression, const char *tx_expression,
      const int is_ty_expression, const char *ty_expression,
      const int is_tz_expression, const char *tz_expression,
      const int is_tx_curve, const char *tx_interpol_type,
const char *tx_indep_var, const int tx_curve_id,
      const int is_ty_curve, const char *ty_interpol_type,
const char *ty_indep_var, const int ty_curve_id,
      const int is_tz_curve, const char *tz_interpol_type,
const char *tz_indep_var, const int tz_curve_id) {}
// #46
    virtual void API_SendOffGeneralScalarForce(
const int num_general_scalar_force, const int idx,
      const int id, const int i_marker_id, const int j_marker_id,
      const int body1_id, const int body2_id,
      const int is_force_expression, const char *force_expression,
      const int is_force_curve, const char *force_interpol_type,
      const char *force_indep_var, const int force_curve_id) {}
// #47
    virtual void API_SendOffGeneralScalarTorque(
const int num_general_scalar_torque, const int idx,

```



```

        const int id, const int i_marker_id, const int j_marker_id,
        const int body1_id, const int body2_id,
        const int is_torque_expression, const char *torque_expression,
        const int is_torque_curve, const char *torque_interpol_type,
        const char *torque_indep_var, const int torque_curve_id) {}

// #48
    virtual void API_SendOffXYCurveData(const int num_curve,
const int idx, const int id,
        const int num_xy_pair, const double *x_val, const double *y_val) {}
// #49
    virtual void API_SendOffTemplate(const int num_template,
const int idx, const char *text) {}
};

```

## MotionView Result API

Please contact MotionView Support to obtain information about an example.

The Model API consists mainly of a group of *mrf* functions that pass results data from the solver to the result API. Most of the functions require a pair of *Open* and *Close* functions.

Register functions should accompany the history data that can be animated, such as the position/orientation of rigid bodies or the modal participation factors of flexible bodies. If a type is registered as an animation object such as rigid body or flexible body, its component must be created using the function,

```
int mrfCreateElement(const char *element_name, const int element_id);
```

instead of its "ID-less" version:

```
int mrfCreateElement(const char *element_name, const int element_id)
```

This is because the result data and the model object are linked through IDs.

If a type is registered for rigid body animation, the first few components of that type must be specified in accordance with the order defined in the header file. The orientations of rigid bodies are defined by Euler parameters.

The following is the header file that contains a list of Result API functions. The names of the functions are self-explanatory.

```

int mrfOpenResult(const float start_time, const float end_time, int num_time_steps);
int mrfCloseResult(int flag=0);
int mrfOpenResultHeader(const char *mrf_filename, const char *abf_filename, const
char *tab_filename);
int mrfCloseResultHeader(void);
int mrfCreateDataType(const char *type_name);
int mrfOpenDataTypeByIndex(const int type_idx);
int mrfOpenDataTypeByName(const char* type_name);
int mrfRegisterForTabulatedOutput(void);
int mrfRegisterForRigidBodyAnimation(void);
/*      x,y,z,e0,e1,e2,e3
*/

```

```

int mrfRegisterNoGroundBody(void);
/* by default, the api assumes that the first rigid body is the ground body
   and its time history is discarded because they are time-invariant.
   If in your model there is no ground body or you do not want to make the
   ground body the first one in the rigid body element list, then you have
   to call this function.
*/
int mrfRegisterForModalFlexBodyAnimation(const int num_modal_coords);
/*      x,y,z,e0,e1,e2,e3
      all followed by modal coordinates q1, q2, ....
*/
int mrfRegisterForAbsNodalFlexBodyAnimation(const int num_nodes);
/*      x1,x2,...,xn,y1,y2,...,yn,z1,z2,...,zn where n=num_nodes */
int mrfCloseDataType(void);
int mrfCreateComponent(const char *component_name);
int mrfCreateElement(const char *element_name);
int mrfCreateElement(const char *element_name, const int element_id);
int mrfOpenTimeStep(const float time);
int mrfCloseTimeStep(void);
int mrfMoveToElementByIndex(const int element_idx);
int mrfMoveToElementByName(const char *element_name);
int mrfPutComponentData(const float *component_data);

```

The following is a working example and comments about the usage of the API fuctions in a simple solver.

Example solver code with the API Functions in place:

```

int main(int argc, char *argv[])
{
    //////////////// Read Model ////////////////
    model_data model;
    ReadModel(argc, argv, &model);

    //////////////// Populate Header ////////////////

    mrfOpenResult(0.0f, (float)model.tend, model.num_steps);
    mrfOpenResultHeader("balls.mrf", "balls.abf", "balls.tab");
    mrfCreateDataType("Rigid Body");
    mrfCreateDataType("System");

    mrfOpenDataTypeByName("Rigid Body");
    mrfRegisterForRigidBodyAnimation();
    mrfRegisterNoGroundBody();
    mrfRegisterForTabulatedOutput();
    mrfCreateComponent("x");
    mrfCreateComponent("y");
    mrfCreateComponent("z");
    mrfCreateComponent("e0");
    mrfCreateComponent("e1");
    mrfCreateComponent("e2");
    mrfCreateComponent("e3");
    mrfCreateComponent("xd");
    mrfCreateComponent("yd");
}

```

```

mrfCreateComponent("zd");
mrfCreateComponent("xdd");
mrfCreateComponent("ydd");
mrfCreateComponent("zdd");
mrfCloseDataType();

mrfOpenDataTypeByName("System");
mrfRegisterForTabulatedOutput();
mrfCreateComponent("Potential");
mrfCreateComponent("Kinetic");
mrfCreateComponent("Total");
mrfCloseDataType();

int body_idx;

mrfOpenDataTypeByIndex(0);
for (body_idx=0;body_idx<model.num_balls;body_idx++)
{
    char name[20];
    sprintf(name,"Ball %i",body_idx+30102);
    mrfCreateElement(name, body_idx+30102);
}
mrfCloseDataType();

mrfOpenDataTypeByIndex(1);
mrfCreateElement("Energy");
mrfCloseDataType();
mrfCloseResultHeader();

double *psi = new double [model.num_balls];
double *dis = new double [model.num_balls];
for (body_idx=0;body_idx<model.num_balls;body_idx++)
{
    dis[body_idx] = sqrt(model.x[body_idx]*model.x[body_idx]+
model.y[body_idx]*model.y[body_idx]);
    psi[body_idx] = asin(model.y[body_idx]/dis[body_idx]);
}

double step_size = model.tend/model.num_steps;
double time=0;
float q[13];
float energy[3];

for (int time_idx=0;time_idx<=model.num_steps;time_idx++,time+=step_size)
{
    memset(energy,0,3*sizeof(float));
    printf(" Time = %13.6E\n",time);

    mrfOpenTimeStep((float)time);

```

```

    mrfOpenDataTypeByIndex(0);
    for (body_idx=0;body_idx<model.num_balls;body_idx++)
    {
        mrfMoveToElementByIndex(body_idx);
        q[0] =
(float)(dis[body_idx]*cos(2*pi*model.freq*time+psi[body_idx]));
        q[1] =
(float)(dis[body_idx]*sin(2*pi*model.freq*time+psi[body_idx]));
        q[2] = (float)(model.z[body_idx]);
        q[3] = 1;
        q[4] = 0;
        q[5] = 0;
        q[6] = 0;

        q[7] = (float)(-dis[body_idx]*sin(2*pi*model.freq*time+
psi[body_idx])*2*pi*model.freq);
        q[8] = (float)( dis[body_idx]*cos(2*pi*model.freq*time+
psi[body_idx])*2*pi*model.freq);
        q[9] = 0;

        energy[0] += 0.5f*(q[3]*q[3] + q[4]*q[4] + q[5]*q[5]);
        energy[1] += 9.8f*q[2];

        q[10] = (float)(-dis[body_idx]*cos(2*pi*model.freq*time+
psi[body_idx])*2*pi*model.freq*2*pi*model.freq);
        q[11] = (float)(-dis[body_idx]*sin(2*pi*model.freq*time+
psi[body_idx])*2*pi*model.freq*2*pi*model.freq);
        q[12] = 0;
        mrfPutComponentData(q);
    }
    mrfCloseDataType();

    mrfOpenDataTypeByIndex(1);
    mrfMoveToElementByIndex(0);
    energy[2] = energy[0] + energy[1];
    mrfPutComponentData(energy);
    mrfCloseDataType();

    mrfCloseTimeStep();
}

mrfCloseResult();
return 0;
}

```

## HyperView Interfacing

When you select a model or results file, HyperView automatically detects the file type and selects the proper reader to import the data. If more than one reader claims to support the selected file format, a dialog listing those readers is displayed. From this dialog, you can select the preferred reader.

To extract only a selected set of results, use the external translator HvTrans to create an h3-D file.

### Solver Interface Support

HyperView supports result files created from the LS-DYNA 3D (solver input file is also supported, but limited to the keyword interface) and LLNL-DYNA d3plot using the `.d3plot` file format.

### Calculating the Supported Result "pressure"

For the supported result "pressure", the pressure is generated by the average of the 3 global stresses:

$$Pr = -(StressX + StressY + StressZ) / 3.0$$

## HyperView LS-DYNA 3-D Results Reader

### Supported Results

Data Type Name	Result Type	Data Type	Notes
displacement	NODE	VECTOR	
velocity	NODE	VECTOR	
acceleration	NODE	VECTOR	
temperature	NODE	SCALAR	
CFD Data	NODE	SCALAR	Supported CFD types include pressure, vorticity (x, y, z, and resultant), enstrophy, helicity, stream function, enthalpy, density, turbulent kinetic energy, and dissipation.
stress	ELEMENT	TENSOR3-D	Thick shell support is for the mid-surface only.
strain	ELEMENT	TENSOR3-D	Thick shell support is for the inner surface only.
effective plastic strain	ELEMENT	SCALAR	Thick shell support is for the mid-surface only.
extra history variables	ELEMENT	SCALAR	
thickness	ELEMENT	SCALAR	

## Notes

- HyperView supports LS-DYNA 3-D version 970
- HyperView supports implicit results
- You can request specific information in the LS-DYNA run to output results, such as beam and strain, in HyperView.

## Post-processing LS-DYNA ALE Results

ALE results are written to the results type named "Extra Solid History" as a series of the Var N results. These Var N results depend on how the LS-DYNA input deck was set up. Two LS-DYNA results can be written to Var N: ALE multi-material groupings, which are defined by \*ALE\_MULTI-MATERIAL\_GROUP cards, and the strain results.

For ALE results, the Var N results are defined as:

Var 1 = fluid density

Var 2 – Var (1 + N) = the volume fractions defined according to the number (N) of multi-material groups you have defined in the model using \*ALE\_MULTI-MATERIAL\_GROUP cards. Also refer to the remarks in the \*ALE\_MULTI-MATERIAL\_GROUP entry in the LS-DYNA Keyword User's Manual.

Var 2 + N = combined volume fraction

If it is an ALE run and you set STRFLG = 1 for the \*DATABASE\_EXTENT\_BINARY card, then:

VAR 3 + N – VAR 8 + N = correspond to x-strain, y-strain, z-strain, xy-strain, yz-strain, and zx-strain results for the solid, shell, and thick shell elements.

# HyperMesh

## LS-DYNA Interface Overview

This section describes the HyperMesh LS-DYNA interface. HyperMesh provides a complete pre-processing environment for preparing LS-DYNA data decks for analysis. HyperMesh can read existing LS-DYNA decks, create a model, display and edit LS-DYNA cards as they will look in the deck, and write a deck for analysis. Although HyperMesh also offers limited post-processing capabilities in results translation, you are strongly encouraged to exclusively use HyperView, the new and continuously updated HyperWorks post-processor. To create LS-DYNA decks in HyperMesh, you must load the LsDyna user profile with the appropriate template to access the full pre-processing capability in HyperMesh.

## Recommended Process

To take advantage of new features, you must make some changes when you set up LS-DYNA models. The recommended process for using HyperMesh's LS-DYNA interface is explained in the following sections.

### Creating welds

The LS-DYNA card `*CONSTRAINED_NODAL_RIGID_BODY` is created in HyperMesh using the **rigid** panel and it is represented in HyperMesh as rigidlink elements. To fully define this entity LS-DYNA also needs a `*SET_NODE` to store the nodes that are constrained. HyperMesh has an option that allows rigidlink elements to point to a node set. To use this feature, you must select the **Attach dependent nodes as a set** check box before you create the rigids. This creates a node set with all the secondary nodes, and attaches the set to the rigidlink. Either updating the node set or the rigidlink itself can update the weld. The node set IDs can be renumbered in the **renumber** panel. Use the **RLs with Sets** macro available in the dynakey Utility menu to convert rigids to rigidlinks with sets. Node sets that are used to define welds are written along with other entity sets (not with the `*CONSTRAINED` card). This explanation is also valid for other rigid constraints such as `*CONSTRAINED_NODE_SET` and `*CONSTRAINED_GENERALIZED_WELD`.

Mesh independent welds are supported in LS-DYNA with the configurations `*ELEMENT_BEAM` (`*ELEMENT_SOLID`) and with `*MAT_SPOTWELD`. These welds can be created using the **Realize** option in the **connector** panel. The **connector** panel can create mesh-less welds using a pre-defined master weld file or by realizing connectors.

The tool automatically creates the necessary `*CONTACT_SPOTWELD`, a `*MAT_SPOTWELD`, and the `*SECTION_BEAM` (`*SECTION_SOLID`).

Material failure and beam cross section will be automatically determined based on the thickness and material of the flanges connected, and is based on parameters set in the `weld_config.ini` file in the `~hm\scripts\connectors\` directory.

### Creating contacts

`*CONTACT` or `*RIGIDWALL` cards are created using the **interfaces** and **rigidwall** panels, respectively. To define them properly, a `*SET` (part set, segment set, element set, node set) or a `*DEFINE_BOX` are needed. The recommended process is to first create the set/box in HyperMesh and then assign them to the interface or rigidwall as needed.

Alternatively, pick the nodes and elements directly from the graphics area to have HyperMesh automatically create a set in export that stores the selected entity. However, note that you will not have control of the ID that is assigned to these sets.

To overcome the problem, an **edit** button is available in the **add** subpanel of the **interface**, **rigidwall**, **ale**, and **control volume** panels. There are two uses for this button. Consider the case of a contact. If the contact is already defined using a particular set, you can click **edit** to automatically move to the **entity set** panel. HyperMesh will highlight the displayed entities of the set so that you can update it as needed. The **edit** button can also be used for a newly-created contact or for one without master/slave. When you click **edit** the entity set panel is invoked and a new set will be created. Click **return** on the entity set panel to go to the initial interface panel where you must click **update** to confirm the selection.

### To create contacts in HyperMesh:

1. Create a contact using a \*SET\_SEGMENT:
  - In the **contactsurfs** panel on **Analysis** page, create a \*SET\_SEGMENT (**contactsurf**) using 1d elements, shell elements or faces of solid elements.
  - Create a \*CONTACT in the **interfaces** panel with the intended contact type.
  - In the **add** subpanel, select the master or slave type to **contactsurf** and select the **contactsurf** created in the first step.
  - Click **update**.
2. Create a contact using \*DEFINE\_BOX:
  - In the **blocks** panel on the **Analysis** page and create a \*DEFINE\_BOX.
  - Create a \*CONTACT from the **interfaces** panel or \*RIGIDWALL from the **rigidwall** panel with the intended contact type.
  - In to the **card image** subpanel, click **edit**
  - A card image of that contact is displayed.
  - Click on either the **SBOXID** or **MBOXID** field to convert it to a yellow box selector.
  - Click the field again and select the BOX created in first step.
  - In the **add** subpanel, select **ALL** for the master or slave type.
  - Click **update**.
3. Create a contact using \*SET\_PART or \*SET\_SHELL or \*SET\_NODE:
  - In the **entity sets** panel on the **Analysis** page, create a set of comps (SET\_PART), elements (SET\_SHELL or SET\_BEAM), or nodes (SET\_NODE).
  - Create a \*CONTACT from the **interfaces** panel or \*RIGIDWALL from the **rigidwall** panel with the intended contact type.
  - In the **add** subpanel, select set for the master or slave type and select the set created in the first step.
  - Click **add**.



4. Create a contact using exempt \*SET\_PART:
  - Perform the tasks in step 3 to create a contact using \*SET\_PART.
  - In the **card image** subpanel, click **edit**
  - A card image of that contact is displayed.
  - Clear the check box for **ExemptSlvPartSet**.
  - The **SSTYPE** field becomes 6.
  - Add a box:
    - Click card edit.
    - Click **SBOXID**.
    - Select the box.

Note that there is no on/off display of a group (interface and rigidwall) when they are defined by \*SET\_PART, \*SET\_SHELL or \*SET\_NODE. Only groups defined by nodes-elements and boxes will have a graphic representation that can be turned off using the **display** panel.

### Creating airbags

You can create control volumes using \*SET\_PART or \*SET\_SEGMENT. A contactsurf (SET\_SEGMENT) or an entity set (SET\_PART) must be created prior to the controlvol. The control volume can then be created using these contactsurfs or entity sets. A reference geometry can also be created using nodes. These nodes are independent of the nodes in the airbag and you can select any nodes in the model. The coordinates of the nodes selected at the creation of the reference geometry are stored as their reference coordinates.

### Blanks

In the card editor, all the attribute fields are supported as *Blanks*. You must click on the field and input the value. This is a change in HyperMesh 5.0 (and later) where some of the attribute fields had a default value.

### Editing an LS-DYNA Model to add cards not supported by HyperMesh

Even though HyperMesh supports most LS-DYNA cards used by the majority of users, some cards are not supported. In order to use unsupported cards with the LS-DYNA model, you can add them in HyperMesh (no need to use a text editor). Select the **unsupp\_cards** in the **control cards** panel on the **Analysis** page. You can then enter the cards in the pop-up text editor. You should exercise caution regarding formatting and card validity. Care should also be taken if any of the cards point to entities inside HyperMesh (such as cards pointing to sets, parts, etc.). HyperMesh stores these cards as text and does not consider pointers. When importing an LS-DYNA model into HyperMesh, any cards that are encountered that HyperMesh does not support are written in this section, thus they are exported along with the remaining model.

## Supported LS-DYNA Keywords

Entities can be created using the solver browser. All the supported LS-DYNA keywords appear in the tree view in the solver browser, sorted by type of input data. To display the solver browser, select **Solver Browser** from the **View** pull-down menu.

The complete list of LS-DYNA keywords that are supported in HyperMesh are listed below.

An alternate way to identify a supported card is to invoke the "create new keyword" tool in HyperMesh. This convenient selection tool can be invoked using the context-sensitive menu in the solver browser, using the **Create New** option in the **Tools** pull-down menu, or the SHIFT + N shortcut key combination.

Once you select a card of interest, HyperMesh opens the right panel and sets the necessary options.

- \*AIRBAG\_SIMPLE\_PRESSURE\_VOLUME\_(ID)
- \*AIRBAG\_SIMPLE\_AIRBAG\_MODEL\_(ID)
- \*AIRBAG\_WANG\_NEFSKE\_(ID)
- \*AIRBAG\_WANG\_NEFSKE\_POP\_(ID)
- \*AIRBAG\_WANG\_NEFSKE\_JETTING\_(ID)
- \*AIRBAG\_WANG\_NEFSKE\_JETTING\_POP\_(ID)
- \*AIRBAG\_WANG\_NEFSKE\_MULTIPLE\_JETTING\_(ID)
- \*AIRBAG\_WANG\_NEFSKE\_MULTIPLE\_JETTING\_POP\_(ID)
- \*AIRBAG\_HYDBRID\_(ID)
- \*AIRBAG\_HYDBRID\_JETTING\_(ID)
- \*AIRBAG\_ADIABATIC\_GAS\_MODEL\_(ID)
- \*AIRBAG\_INTERACTION\_(ID)
- \*AIRBAG\_LINEAR\_FLUID\_(ID)
- \*AIRBAG\_LOAD\_CURVE\_(ID)
- \*AIBAG\_HYBRID\_CHEMKIN\_(ID)
- \*AIRBAG\_REFERENCE\_GEOMETRY\_BIRTH
- \*AIRBAG\_REFERENCE\_GEOMETRY\_BIRTH\_RDT
- \*ALE\_MULTI-MATERIAL\_GROUP
- \*ALE\_REFERENCE\_SYSTEM\_SWITCH
- \*ALE\_REFERENCE\_SYSTEM\_CURVE
- \*ALE\_REFERENCE\_SYSTEM\_GROUP
- \*ALE\_REFERENCE\_SYSTEM\_NODE
- \*ALE\_SMOOTHING
- \*ALE\_TANK\_TEST
- \*BOUNDARY\_AMBIENT\_EOS
- \*BOUNDARY\_PRESCRIBED\_MOTION\_NODE\_(ID)
- \*BOUNDARY\_PRESCRIBED\_MOTION\_RIGID\_(ID)
- \*BOUNDARY\_PRESCRIBED\_MOTION\_RIGID\_LOCAL\_(ID)

\*BOUNDARY\_PRESCRIBED\_MOTION\_SET\_(ID)  
\*BOUNDARY\_SPC\_NODE\_(ID)  
\*BOUNDARY\_SPC\_SET\_(ID)  
\*BOUNDARY\_CONVECTION\_SET  
\*BOUNDARY\_RADIATION\_SET  
\*BOUNDARY\_TEMPERATURE\_NODE  
\*BOUNDARY\_TEMPERATURE\_SET  
\*CONSTRAINED\_EXTRA\_NODES\_NODE  
\*CONSTRAINED\_EXTRA\_NODES\_SET  
\*CONSTRAINED\_GENERALIZED\_WELD\_BUTT\_(ID)  
\*CONSTRAINED\_GENERALIZED\_WELD\_FILLET\_(ID)  
\*CONSTRAINED\_GENERALIZED\_WELD\_SPOT\_(ID)  
\*CONSTRAINED\_INTERPOLATION  
\*CONSTRAINED\_JOINT\_CYLINDRICAL\_(ID)  
\*CONSTRAINED\_JOINT\_CYLINDRICAL\_LOCAL(ID)  
\*CONSTRAINED\_JOINT\_CYLINDRICAL\_FAILURE(ID)  
\*CONSTRAINED\_JOINT\_CYLINDRICAL\_LOCAL\_FAILURE(ID)  
\*CONSTRAINED\_JOINT\_PLANAR(ID)  
\*CONSTRAINED\_JOINT\_PLANAR\_LOCAL(ID)  
\*CONSTRAINED\_JOINT\_PLANAR\_FAILURE(ID)  
\*CONSTRAINED\_JOINT\_PLANAR\_LOCAL\_FAILURE(ID)  
\*CONSTRAINED\_JOINT\_REVOLUTE\_(ID)  
\*CONSTRAINED\_JOINT\_REVOLUTE\_LOCAL(ID)  
\*CONSTRAINED\_JOINT\_REVOLUTE\_FAILURE(ID)  
\*CONSTRAINED\_JOINT\_REVOLUTE\_LOCAL\_FAILURE(ID)  
\*CONSTRAINED\_JOINT\_SPHERICAL\_(ID)  
\*CONSTRAINED\_JOINT\_SPHERICAL\_LOCAL(ID)  
\*CONSTRAINED\_JOINT\_SPHERICAL\_FAILURE(ID)  
\*CONSTRAINED\_JOINT\_SPHERICAL\_LOCAL\_FAILURE(ID)  
\*CONSTRAINED\_JOINT\_TRANSLATIONAL(ID)  
\*CONSTRAINED\_JOINT\_TRANSLATIONAL\_LOCAL(ID)  
\*CONSTRAINED\_JOINT\_TRANSLATIONAL\_FAILURE(ID)  
\*CONSTRAINED\_JOINT\_TRANSLATIONAL\_LOCAL\_FAILURE(ID)  
\*CONSTRAINED\_JOINT\_LOCKING(ID)  
\*CONSTRAINED\_JOINT\_LOCKING\_LOCAL(ID)  
\*CONSTRAINED\_JOINT\_LOCKING\_FAILURE(ID)  
\*CONSTRAINED\_JOINT\_LOCKING\_LOCAL\_FAILURE(ID)

\*CONSTRAINED\_JOINT\_UNIVERSAL(ID)  
 \*CONSTRAINED\_JOINT\_UNIVERSAL\_LOCAL(ID)  
 \*CONSTRAINED\_JOINT\_UNIVERSAL\_FAILURE(ID)  
 \*CONSTRAINED\_JOINT\_UNIVERSAL\_LOCAL\_FAILURE(ID)  
 \*CONSTRAINED\_JOINT\_STIFFNESS\_FLEXION-TORSION  
 \*CONSTRAINED\_JOINT\_STIFFNESS\_GENERALIZED  
 \*CONSTRAINED\_JOINT\_STIFFNESS\_TRANSLATIONAL  
 \*CONSTRAINED\_LAGRANGE\_IN\_SOLID  
 \*CONSTRAINED\_LINEAR  
 \*CONSTRAINED\_NODAL\_RIGID\_BODY  
 \*CONSTRAINED\_NODAL\_RIGID\_BODY\_INERTIA  
 \*CONSTRAINED\_NODAL\_RIGID\_BODY\_SPC  
 \*CONSTRAINED\_NODAL\_RIGID\_BODY\_INERTIA\_SPC  
 \*CONSTRAINED\_NODE\_SET(ID)  
 \*CONSTRAINED\_RIGID\_BODIES  
 \*CONSTRAINED\_RIGID\_BODY\_STOPPERS  
 \*CONSTRAINED\_RIVET(ID)  
 \*CONSTRAINED\_SPOTWELD(ID)  
 \*CONSTRAINED\_SPOTWELD\_FILTERED\_FORCE(ID)  
 \*CONSTRAINED\_TIE-BREAK  
 \*CONSTRAINED\_TIED\_NODES\_FAILURE  
 \*CONTACT\_OPTION1\_OPTION2\_OPTION3\_MPP  
 \*CONTACT\_AUTO\_MOVE  
 \*CONTACT\_AIRBAG\_SINGLE\_SURFACE(ID)  
 \*CONTACT\_AUTOMATIC\_GENERAL(ID)  
 \*CONTACT\_AUTOMATIC\_GENERAL\_INTERIOR(ID)  
 \*CONTACT\_AUTOMATIC\_NODES\_TO\_SURFACE(ID)  
 \*CONTACT\_AUTOMATIC\_ONE\_WAY\_SURFACE\_TO\_SURFACE(ID)  
 \*CONTACT\_AUTOMATIC\_SINGLE\_SURFACE(ID)  
 \*CONTACT\_AUTOMATIC\_SURFACE\_TO\_SURFACE(ID)  
 \*CONTACT\_AUTOMATIC\_SURFACE\_TO\_SURFACE\_TIEBREAK(ID)  
 \*CONTACT\_CONSTRAINT\_NODES\_TO\_SURFACE(ID)  
 \*CONTACT\_CONSTRAINT\_SURFACE\_TO\_SURFACE(ID)  
 \*CONTACT\_DRAWBEAD(ID)  
 \*CONTACT\_ERODING\_NODES\_TO\_SURFACE(ID)  
 \*CONTACT\_ERODING\_SINGLE\_SURFACE(ID)  
 \*CONTACT\_ERODING\_SURFACE\_TO\_SURFACE(ID)  
 \*CONTACT\_FORCE\_TRANSDUCER\_CONSTRAINT(ID)

\*CONTACT\_FORCE\_TRANSDUCER\_PENALTY(ID)  
\*CONTACT\_FORMING\_SURFACE\_TO\_SURFACE(ID)  
\*CONTACT\_FORMING\_ONEWAY\_SURFACE\_TO\_SURFACE(ID)  
\*CONTACT\_FORMING\_NODES\_TO\_SURFACE(ID)  
\*CONTACT\_NODES\_TO\_SURFACE(ID)  
\*CONTACT\_NODES\_TO\_SURFACE\_INTERFERENCE(ID)  
\*CONTACT\_ONE\_WAY\_SURFACE\_TO\_SURFACE(ID)  
\*CONTACT\_ONE\_WAY\_SURFACE\_TO\_SURFACE\_INTERFERENCE(ID)  
\*CONTACT\_RIGID\_NODES\_TO\_RIGID\_BODY(ID)  
\*CONTACT\_RIGID\_BODY\_ONE\_WAY\_TO\_RIGID\_BODY(ID)  
\*CONTACT\_RIGID\_BODY\_TWO\_WAY\_TO\_RIGID\_BODY(ID)  
\*CONTACT\_SINGLE\_EDGE(ID)  
\*CONTACT\_SINGLE\_SURFACE(ID)  
\*CONTACT\_SLIDING\_ONLY(ID)  
\*CONTACT\_SLIDING\_ONLY\_PENALTY(ID)  
\*CONTACT\_SPOTWELD(ID)  
\*CONTACT\_SPOTWELD\_WITH\_TORSION(ID)  
\*CONTACT\_SURFACE\_TO\_SURFACE(ID)  
\*CONTACT\_SURFACE\_TO\_SURFACE\_THERMAL(ID)  
\*CONTACT\_SURFACE\_TO\_SURFACE\_INTERFERENCE(ID)  
\*CONTACT\_TIEBREAK\_NODES\_TO\_SURFACE(ID)  
\*CONTACT\_TIEBREAK\_SURFACE\_TO\_SURFACE(ID)  
\*CONTACT\_TIED\_NODE\_TO\_SURFACE(ID)  
\*CONTACT\_TIED\_NODE\_TO\_SURFACE\_OFFSET(ID)  
\*CONTACT\_TIED\_SHELL\_EDGE\_TO\_SURFACE(ID)  
\*CONTACT\_TIED\_SHELL\_EDGE\_TO\_SURFACE\_BEAM\_OFFSET(ID)  
\*CONTACT\_TIED\_SHELL\_EDGE\_TO\_SURFACE\_OFFSET(ID)  
\*CONTACT\_TIED\_SURFACE\_TO\_SURFACE(ID)  
\*CONTACT\_TIED\_SURFACE\_TO\_SURFACE\_TITLE(ID)  
\*CONTACT\_ENTITY(ID)  
\*CONTACT\_INTERIOR(ID)  
\*CONTACT\_RIGID\_SURFACE(ID)  
\*CONTACT\_2D\_AUTOMATIC\_SURFACE\_TO\_SURFACE(ID)  
\*CONTROL\_ACCURACY  
\*CONTROL\_ADAPTIVE  
\*CONTROL\_ALE  
\*CONTROL\_BULK\_VISCOSITY

\*CONTROL\_COARSEN  
\*CONTROL\_CONTACT  
\*CONTROL\_COUPLING  
\*CONTROL\_CPU  
\*CONTROL\_DAMPING (Old LS-DYNA card for input only)  
\*CONTROL\_DYNAMIC\_RELAXATION  
\*CONTROL\_EFG  
\*CONTROL\_ENERGY  
\*CONTROL\_EXPLOSIVE\_SHADOW  
\*CONTROL\_HOURLASS  
\*CONTROL\_IMPLICIT\_AUTO  
\*CONTROL\_IMPLICIT\_BUCKLE  
\*CONTROL\_IMPLICIT\_DYNAMICS  
\*CONTROL\_IMPLICIT\_EIGENVALUE  
\*CONTROL\_IMPLICIT\_GENERAL  
\*CONTROL\_IMPLICIT\_MODES  
\*CONTROL\_IMPLICIT\_SOLUTION  
\*CONTROL\_IMPLICIT\_SOLVER  
\*CONTROL\_IMPLICIT\_STABILIZATION  
\*CONTROL\_IMPLICIT\_LINEAR (old)  
\*CONTROL\_IMPLICIT\_NONLINEAR (old)  
\*CONTROL\_OUTPUT  
\*CONTROL\_PARALLEL  
\*CONTROL\_RIGID  
\*CONTROL\_SHELL  
\*CONTROL\_SOLID  
\*CONTROL\_SOLUTION  
\*CONTROL\_SPH  
\*CONTROL\_STRUCTURED  
\*CONTROL\_STRUCTURED\_TERM  
\*CONTROL\_SUBCYCLE  
\*CONTROL\_TERMINATION  
\*CONTROL\_THERMAL\_NONLINEAR  
\*CONTROL\_THERMAL\_SOLVER  
\*CONTROL\_THERMAL\_TIMESTEP  
\*CONTROL\_TIMESTEP  
\*DAMPING\_FREQUENCY\_RANGE

\*DAMPING\_GLOBAL  
\*DAMPING\_PART\_MASS  
\*DAMPING\_PART\_STIFFNESS  
\*DAMPING\_RELATIVE  
\*DATABASE\_ABSTAT  
\*DATABASE\_AVSFLT  
\*DATABASE\_BNDOUT  
\*DATABASE\_DEFCEO  
\*DATABASE\_DEFORC  
\*DATABASE\_ELOUT  
\*DATABASE\_FORMAT  
\*DATABASE\_GCEO  
\*DATABASE\_GLSTAT  
\*DATABASE\_JNTFORC  
\*DATABASE\_MATSUM  
\*DATABASE\_MOVIE  
\*DATABASE\_MPGS  
\*DATABASE\_NCFORC  
\*DATABASE\_NODFOR  
\*DATABASE\_NODOUT  
\*DATABASE\_RBDOUT  
\*DATABASE\_RCFORC  
\*DATABASE\_RWFORC  
\*DATABASE\_SBTOUT  
\*DATABASE\_SECFORC  
\*DATABASE\_SLEOUT  
\*DATABASE\_SPCFORC  
\*DATABASE\_SPHOUT  
\*DATABASE\_SSSTAT  
\*DATABASE\_SWFORC  
\*DATABASE\_TPRINT  
\*DATABASE\_TRHIST  
\*DATABASE\_BINARY\_D3DRLF  
\*DATABASE\_BINARY\_D3DUMP  
\*DATABASE\_BINARY\_D3PLOT  
\*DATABASE\_BINARY\_D3THDT  
\*DATABASE\_BINARY\_INTFOR

\*DATABASE\_BINARY\_RUNRSF  
\*DATABASE\_BINARY\_XTFILE  
\*DATABASE\_CROSS\_SECTION\_PLANE(ID)  
\*DATABASE\_CROSS\_SECTION\_SET(ID)  
\*DATABASE\_EXTENT\_AVS  
\*DATABASE\_EXTENT\_BINARY  
\*DATABASE\_EXTENT\_MOVIE  
\*DATABASE\_EXTENT\_MPGS  
\*DATABASE\_EXTENT\_SSSTAT  
\*DATABASE\_FSI  
\*DATABASE\_HISTORY\_BEAM(ID)  
\*DATABASE\_HISTORY\_BEAM\_SET  
\*DATABASE\_HISTORY\_NODE(ID)  
\*DATABASE\_HISTORY\_NODE\_LOCAL(ID)  
\*DATABASE\_HISTORY\_NODE\_SET  
\*DATABASE\_HISTORY\_NODE\_SET\_LOCAL  
\*DATABASE\_HISTORY\_SHELL  
\*DATABASE\_HISTORY\_SHELL(ID)  
\*DATABASE\_HISTORY\_SHELL\_SET  
\*DATABASE\_HISTORY\_SOLID  
\*DATABASE\_HISTORY\_SOLID(ID)  
\*DATABASE\_HISTORY\_SOLID\_SET  
\*DATABASE\_HISTORY\_TSHELL  
\*DATABASE\_HISTORY\_TSHELL(ID)  
\*DATABASE\_HISTORY\_TSHELL\_SET  
\*DATABASE\_NODAL\_FORCE\_GROUP  
\*DATABASE\_SPRING\_FORWARD  
\*DATABASE\_SUPERPLASTIC\_FORMING  
\*DATABASE\_TRACER  
\*DEFINE\_BOX  
\*DEFINE\_BOX\_ADAPTIVE  
\*DEFINE\_BOX\_DRAWBEAD  
\*DEFINE\_BOX\_SPH  
\*DEFINE\_COORDINATE\_NODES  
\*DEFINE\_COORDINATE\_NODES\_TITLE  
\*DEFINE\_CURVE\_FEEDBACK  
\*DEFINE\_CURVE\_SMOOTH



\*DEFINE\_CURVE\_TRIM  
\*DEFINE\_CURVE\_TRIM\_3D  
\*DEFINE\_TRANSFORMATION  
\*DEFINE\_COORDINATE\_NODES(TITLE)  
\*DEFINE\_COORDINATE\_SYSTEM(TITLE)  
\*DEFINE\_COORDINATE\_VECTOR(TITLE) (converted in import)  
\*DEFINE\_CURVE(TITLE)  
\*DEFINE\_SD\_ORIENTATION(TITLE)  
\*DEFINE\_TABLE(TITLE)  
\*DEFINE\_VECTOR(TITLE)  
\*DEFORMABLE\_TO\_RIGID  
\*DEFORMABLE\_TO\_RIGID\_AUTOMATIC  
\*DEFORMABLE\_TO\_RIGID\_INERTIA  
\*ELEMENT\_BEAM  
\*ELEMENT\_BEAM\_OFFSET  
\*ELEMENT\_BEAM\_ORIENTATION\_OFFSET  
\*ELEMENT\_BEAM\_ORIENTATION  
\*ELEMENT\_BEAM\_PID  
\*ELEMENT\_BEAM\_PID\_OFFSET  
\*ELEMENT\_BEAM\_PID\_ORIENTATION  
\*ELEMENT\_BEAM\_PID\_SCALAR  
\*ELEMENT\_BEAM\_PID\_SCALR  
\*ELEMENT\_BEAM\_SCALAR  
\*ELEMENT\_BEAM\_SCALAR\_OFFSET  
\*ELEMENT\_BEAM\_SCALAR\_ORIENTATION  
\*ELEMENT\_BEAM\_SCALR  
\*ELEMENT\_BEAM\_SCALR\_OFFSET  
\*ELEMENT\_BEAM\_SCALR\_ORIENTATION  
\*ELEMENT\_BEAM\_SECTION  
\*ELEMENT\_BEAM\_SECTION\_PID  
\*ELEMENT\_BEAM\_SECTION\_OFFSET  
\*ELEMENT\_BEAM\_SECTION\_ORIENTATION  
\*ELEMENT\_BEAM\_THICKNESS  
\*ELEMENT\_BEAM\_THICKNESS\_SCALAR  
\*ELEMENT\_BEAM\_THICKNESS\_PID  
\*ELEMENT\_BEAM\_THICKNESS\_OFFSET  
\*ELEMENT\_BEAM\_THICKNESS\_ORIENTATION

\*ELEMENT\_DISCRETE  
\*ELEMENT\_INERTIA  
\*ELEMENT\_INERTIA\_OFFSET  
\*ELEMENT\_MASS  
\*ELEMENT\_PLOTEL  
\*ELEMENT\_SEATBELT  
\*ELEMENT\_SEATBELT\_ACCELEROMETER  
\*ELEMENT\_SEATBELT\_PRETENSIONER  
\*ELEMENT\_SEATBELT\_RETRACTOR  
\*ELEMENT\_SEATBELT\_SENSOR  
\*ELEMENT\_SEATBELT\_SLIPRING  
\*ELEMENT\_SHELL  
\*ELEMENT\_SHELL\_BETA  
\*ELEMENT\_SHELL\_THICKNESS  
\*ELEMENT\_SOLID  
\*ELEMENT\_SOLID\_ORTHO  
\*ELEMENT\_SPH  
\*ELEMENT\_TRIM  
\*ELEMENT\_TSHELL  
\*EOS\_GRUNEISEN  
\*EOS\_IDEAL\_GAS  
\*EOS\_IGNITION\_AND\_GROWTH\_OF\_REACTION\_IN\_HE  
\*EOS\_JWL  
\*EOS\_LINEAR\_POLYNOMIAL  
\*EOS\_LINEAR\_POLYNOMIAL\_WITH\_ENERGY\_LEAK  
\*EOS\_PROPELLANT\_DEFLAGRATION  
\*EOS\_RATIO\_OF\_POLYNOMIALS  
\*EOS\_SACK\_TUESDAY  
\*EOS\_TABULATED  
\*EOS\_TABULATED\_COMPACTION  
\*EOS\_TENSOR\_PORE\_COLLAPSE  
\*END  
\*HOURGLASS  
\*INCLUDE  
\*INCLUDE\_STAMPED\_PART (As control card and in \*Part)  
\*INCLUDE\_TRANSFORM (As control card)  
\*INITIAL\_DETONATION

\*INITIAL\_FOAM\_REFERENCE\_GEOMETRY  
\*INITIAL\_GAS\_MIXTURE  
\*INITIAL\_TEMPERATURE\_NODE  
\*INITIAL\_TEMPERATURE\_SET  
\*INITIAL\_VEHICLE\_KINEMATICS  
\*INITIAL\_VELOCITY  
\*INITIAL\_VELOCITY\_NODE  
\*INITIAL\_VELOCITY\_GENERATION  
\*INITIAL\_VOID  
\*INITIAL\_VOLUME\_FRACTION  
\*INTEGRATION\_BEAM  
\*INTEGRATION\_SHELL  
\*INTERFACE\_COMPONENT\_NODE  
\*INTERFACE\_COMPONENT\_SEGMENT  
\*INTERFACE\_LINKING\_SEGMENT  
\*INTERFACE\_LINKING\_EDGE  
\*INTERFACE\_LINKING\_DISCRETE\_NODE\_SET  
\*INTERFACE\_SPRINGBACK  
\*INTERFACE\_SPRINGBACK\_LSDYNA  
\*INTERFACE\_SPRINGBACK\_LSDYNA  
\*INTERFACE\_SPRINGBACK\_LSDYNA\_THICKNESS  
\*INTERFACE\_SPRINGBACK\_LSDYNA\_NOTHICKNESS  
\*INTERFACE\_SPRINGBACK\_NIKE3D  
\*INTERFACE\_SPRINGBACK\_NIKE3D\_THICKNESS  
\*INTERFACE\_SPRINGBACK\_NASTRAN  
\*INTERFACE\_SPRINGBACK\_NASTRAN\_THICKNESS  
\*INTERFACE\_SPRINGBACK\_NASTRAN\_NOTHICKNESS  
\*INTERFACE\_SPRINGBACK\_SEAMLESS  
\*INTERFACE\_SPRINGBACK\_SEAMLESS\_THICKNESS  
\*INTERFACE\_SPRINGBACK\_SEAMLESS\_NOTHICKNESS  
\*KEYWORD  
\*LOAD\_BLAST  
\*LOAD\_BRODE  
\*LOAD\_BODY\_GENERALIZED  
\*LOAD\_BODY\_PARTS  
\*LOAD\_BODY\_RX  
\*LOAD\_BODY\_RY

\*LOAD\_BODY\_RZ  
\*LOAD\_BODY\_X  
\*LOAD\_BODY\_Y  
\*LOAD\_BODY\_Z  
\*LOAD\_RIGID\_BODY  
\*LOAD\_NODE\_POINT  
\*LOAD\_NODE\_SET  
\*LOAD\_SEGMENT  
\*LOAD\_SEGMENT\_SET  
\*LOAD\_SHELL\_ELEMENT  
\*LOAD\_SHELL\_SET  
\*LOAD\_THERMAL\_CONSTANT  
\*LOAD\_SEGMENT\_SET  
\*LOAD\_SHELL\_SET  
\*LOAD\_NODE\_SET  
\*LOAD\_THERMAL\_CONSTANT  
\*LOAD\_THERMAL\_CONSTANT\_NODE  
\*LOAD\_THERMAL\_VARIABLE  
\*LOAD\_THERMAL\_VARIABLE\_NODE  
\*MAT\_UNSUPPORTED  
\*MAT\_ELASTIC  
\*MAT\_ELASTIC\_FLUID  
\*MAT\_ORTHOTROPIC\_ELASTIC  
\*MAT\_ANISOTROPIC\_ELASTIC  
\*MAT\_PLASTIC\_KINEMATIC  
\*MAT\_ELASTIC\_PLASTIC\_THERMAL  
\*MAT\_SOIL\_AND\_FOAM  
\*MAT\_VISCOELASTIC  
\*MAT\_BLATZ-KO\_RUBBER  
\*MAT\_HIGH\_EXPLOSIVE\_BURN  
\*MAT\_NULL  
\*MAT\_ELASTIC\_PLASTIC\_HYDRO  
\*MAT\_ISOTROPIC\_ELASTIC\_PLASTIC  
\*MAT\_SOIL\_AND\_FOAM\_FAILURE  
\*MAT\_JOHNSON\_COOK  
\*MAT\_POWER\_LAW\_PLASTICITY  
\*MAT\_STRAIN\_RATE\_DEPENDENT\_PLASTICITY

\*MAT\_RIGID  
\*MAT\_ORTHOTROPIC\_THERMAL  
\*MAT\_COMPOSITE\_DAMAGE  
\*MAT\_PIECEWISE\_LINEAR\_PLASTICITY  
\*MAT\_HONEYCOMB  
\*MAT\_MOONEY-RIVLIN\_RUBBER  
\*MAT\_RESULTANT\_PLASTICITY  
\*MAT\_FORCE\_LIMITED  
\*MAT\_SHAPE\_MEMORY  
\*MAT\_FRAZER\_NASH\_RUBBER\_MODEL  
\*MAT\_LAMINATED\_GLASS  
\*MAT\_BARLAT\_ANISOTROPIC\_PLASTICITY  
\*MAT\_BARLAT\_YLD96  
\*MAT\_FABRIC  
\*MAT\_TRANSVERSELY\_ANISOTROPIC\_ELASTIC\_PLASTIC  
\*MAT\_BLATZ-KO\_FOAM  
\*MAT\_FLD\_TRANSVERSELY\_ANISOTROPIC  
\*MAT\_BAMMAN\_DAMAGE  
\*MAT\_CLOSED\_CELL\_FOAM  
\*MAT\_ENHANCED\_COMPOSITE\_DAMAGE  
\*MAT\_LOW\_DENSITY\_FOAM  
\*MAT\_LAMINATED\_COMPOSITE\_FABRIC  
\*MAT\_COMPOSITE\_FAILURE\_MODEL  
\*MAT\_COMPOSITE\_FAILURE\_SHELL\_MODEL  
\*MAT\_COMPOSITE\_FAILURE\_SOLID\_MODEL  
\*MAT\_ELASTIC\_WITH\_VISCOSITY  
\*MAT\_KELVIN-MAXWELL\_VISCOELASTIC  
\*MAT\_VISCOUS\_FOAM  
\*MAT\_CRUSHABLE\_FOAM  
\*MAT\_RATE\_SENSITIVE\_POWERLAW\_PLASTICITY  
\*MAT\_LINEAR\_ELASTIC\_DISCRETE\_BEAM  
\*MAT\_NONLINEAR\_ELASTIC\_DISCRETE\_BEAM  
\*MAT\_NONLINEAR\_PLASTIC\_DISCRETE\_BEAM  
\*MAT\_SID\_DAMPER\_DISCRETE\_BEAM  
\*MAT\_CABLE\_DISCRETE\_BEAM  
\*MAT\_CONCRETE\_DAMAGE  
\*MAT\_LOW\_DENSITY\_VISCOUS\_FOAM

\*MAT\_ELASTIC\_SPRING\_DISCRETE\_BEAM  
\*MAT\_BILKU/DUBOIS\_FOAM  
\*MAT\_GENERAL\_VISCOELASTIC  
\*MAT\_HYPERELASTIC\_RUBBER  
\*MAT\_OGDEN\_RUBBER  
\*MAT\_PLASTICITY\_WITH\_DAMAGE  
\*MAT\_FU\_CHANG\_FOAM  
\*MAT\_MTS  
\*MAT\_PLASTICITY\_POLYMER  
\*MAT\_ACOUSTIC  
\*MAT\_ELASTIC\_DOF\_SPRING\_DISCRETE\_BEAM  
\*MAT\_INELASTIC\_SPRING\_DISCRETE\_BEAM  
\*MAT\_INELASTIC\_DOF\_SPRING\_DISCRETE\_BEAM  
\*MAT\_BRITTLE\_DAMAGE  
\*MAT\_GENERAL\_JOINT\_DISCRETE\_BEAM  
\*MAT\_SIMPLIFIED\_JOHNSON\_COOK  
\*MAT\_SIMPLIFIED\_JOHNSON\_COOK\_ORTHOTROPIC\_DAMAGE  
\*MAT\_SPOTWELD  
\*MAT\_GEPLASTIC\_SRATE\_2000a  
\*MAT\_ANISOTROPIC\_VISCOPLASTIC  
\*MAT\_FINITE\_ELASTIC\_STRAIN\_PLASTICITY  
\*MAT\_COMPOSITE\_LAYUP  
\*MAT\_GENERAL\_NONLINEAR\_DOF\_DISCRETE\_BEAM  
\*MAT\_GENERAL\_NONLINEAR\_DOF\_DISCRETE\_BEAM  
\*MAT\_MODIFIED\_PIECEWISE\_LINEAR\_PLASTICITY  
\*MAT\_PLASTICITY\_COMPRESSION\_TENSION  
\*MAT\_MODIFIED\_HONEYCOMB  
\*MAT\_ARRUDA\_BOYCE\_RUBBER  
\*MAT\_VACUUM  
\*MAT\_TRANSVERSELY\_ANISOTROPIC\_CRUSHABLE\_FOAM  
\*MAT\_DOF\_GENERALIZED\_SPRING  
\*MAT\_DESHPANDE\_FLECK\_FOAM  
\*MAT\_MODIFIED\_CRUSHABLE\_FOAM  
\*MAT\_HILL\_FOAM  
\*MAT\_VISCOELASTIC\_HILL\_FOAM  
\*MAT\_LOW\_DENSITY\_SYNTHETIC\_FOAM  
\*MAT\_LOW\_DENSITY\_SYNTHETIC\_FOAM\_ORTHO

\*MAT\_SIMPLIFIED\_RUBBER  
\*MAT\_THERMAL\_ISOTROPIC  
\*MAT\_THERMAL\_ORTHOTROPIC  
\*MAT\_THERMAL\_ISOTROPIC\_TD\_LC  
\*MAT\_SPRING\_ELASTIC  
\*MAT\_DAMPER\_VISCOUS  
\*MAT\_SPRING\_ELASTOPLASTIC  
\*MAT\_SPRING\_NONLINEAR\_ELASTIC  
\*MAT\_DAMPER\_NONLINEAR\_VISCOUS  
\*MAT\_SPRING\_GENERAL\_NONLINEAR  
\*MAT\_SPRING\_MAXWELL  
\*MAT\_SPRING\_INELASTIC  
\*MAT\_SEATBELT  
\*MAT\_HYDRAULIC\_GAS\_DAMPER\_DISCRETE\_BEAM  
\*MAT\_ORTHOTROPIC\_VISCOELASTIC  
\*MAT\_CELLULAR\_RUBBER  
\*MAT\_RESULTANT\_ANISOTROPIC  
\*MAT\_GENERAL\_SPRING\_DISCRETE\_BEAM  
\*NODE  
\*NODE\_RIGID\_SURFACE  
\*PART  
\*PART\_PRINT  
\*PART\_CONTACT  
\*PART\_CONTACT\_PRINT  
\*PART\_INERTIA  
\*PART\_INERTIA\_PRINT  
\*PART\_REPOSITION  
\*PART\_REPOSITION\_PRINT  
\*PART\_INERTIA\_CONTACT  
\*PART\_INERTIA\_CONTACT\_PRINT  
\*PART\_REPOSITION\_CONTACT  
\*PART\_REPOSITION\_CONTACT\_INERTIA  
\*PART\_MOVE  
\*RIGIDWALL\_GEOMETRIC\_FLAT(ID)  
\*RIGIDWALL\_GEOMETRIC\_FLAT\_MOTION(ID)  
\*RIGIDWALL\_GEOMETRIC\_PRISM(ID)  
\*RIGIDWALL\_GEOMETRIC\_PRISM\_MOTION(ID)

\*RIGIDWALL\_GEOMETRIC\_CYLINDER(ID)  
\*RIGIDWALL\_GEOMETRIC\_CYLINDER\_MOTION(ID)  
\*RIGIDWALL\_GEOMETRIC\_SPHERE(ID)  
\*RIGIDWALL\_GEOMETRIC\_SPHERE\_MOTION(ID)  
\*RIGIDWALL\_PLANAR(ID)  
\*RIGIDWALL\_PLANAR\_ORTHO(ID)  
\*RIGIDWALL\_PLANAR\_FORCE(ID)  
\*RIGIDWALL\_PLANAR\_MOVING(ID)  
\*RIGIDWALL\_PLANAR\_ORTHO\_FORCES(ID)  
\*RIGIDWALL\_PLANAR\_FORCES\_MOVING(ID)  
\*RIGIDWALL\_PLANAR\_FINITE(ID)  
\*RIGIDWALL\_PLANAR\_FINITE(ID)  
\*RIGIDWALL\_PLANAR\_FORCE\_FINITE(ID)  
\*RIGIDWALL\_PLANAR\_FINITE\_MOVING(ID)  
\*RIGIDWALL\_PLANAR\_ORTHO\_FINITE\_FORCES(ID)  
\*RIGIDWALL\_PLANAR\_FINITE\_FORCES\_MOVING(ID)  
\*SECTION\_BEAM(TITLE)  
\*SECTION\_DISCRETE(TITLE)  
\*SECTION\_POINT\_SOURCE(TITLE)  
\*SECTION\_POINT\_SOURCE\_MIXTURE(TITLE)  
\*SECTION\_SEATBELT(TITLE)  
\*SECTION\_SHELL(TITLE)  
\*SECTION\_SHELL\_ALE(TITLE)  
\*SECTION\_SHELL\_EFG(TITLE)  
\*SECTION\_SOLID(TITLE)  
\*SECTION\_SOLID\_ALE(TITLE)  
\*SECTION\_SOLID\_EFG(TITLE)  
\*SECTION\_SPH  
\*SECTION\_TSHELL(TITLE)  
\*SET\_BEAM(TITLE)  
\*SET\_BEAM\_GENERAL  
\*SET\_BEAM\_GENERATE(TITLE)  
\*SET\_DISCRETE(TITLE)  
\*SET\_DISCRETE\_GENERAL  
\*SET\_DISCRETE\_GENERATE(TITLE)  
\*SET\_MULTI-MATERIAL\_GROUP\_LIST(TITLE)  
\*SET\_NODE\_COLUMN



\*SET\_NODE\_GENERAL  
\*SET\_NODE\_LIST(TITLE)  
\*SET\_NODE\_LIST\_GENERATE(TITLE)  
\*SET\_PART\_LIST(TITLE)  
\*SET\_PART\_LIST\_GENERATE(TITLE)  
\*SET\_PART\_COLUMN(TITLE)  
\*SET\_SEGMENT(TITLE)  
\*SET\_SHELL\_COLUMN  
\*SET\_SHELL\_GENERAL  
\*SET\_SHELL\_LIST(TITLE)  
\*SET\_SHELL\_LIST\_GENERATE(TITLE)  
\*SET\_SOLID\_GENERAL  
\*SET\_SOLID(TITLE)  
\*SET\_SOLID\_GENERATE(TITLE)  
\*SET\_TSHELL(TITLE)  
\*SET\_TSHELL\_GENERAL  
\*SET\_TSHELL\_GENERATE(TITLE)  
\*TITLE

## Importing Decks

Versions of *keyword* files are read using the DYNAKEY input translator (or *DYNAKEY.EXE* on PCs).

HyperMesh also provides include file support for importing data decks that are organized into sub-files.

**Note**      Structured files of version 920 and higher can be read using the DYNASEQ input translators ( or *DYNASEQ.EXE* on PCs). This translator handles both regular and large format decks. Thermal control cards 27 through 30 are not supported.

### To import data with the input translator:

1. Select the *files* panel.
2. Select the *import* subpanel.
3. Click *FE*.
4. Select a translator using the toggle.
5. Select read include files or skip include files.
6. Select FE overwrite or no FE overwrite.
7. Click *import...* and select a file from the *Open file* browser.
8. Click *Open*.

## Summary Templates

To obtain a concise list of entities in the current model, use the **template file =** option on the **summary** panel. This creates an ASCII text file with the summary information and displays the information on the screen.

LS-DYNA has five summary template files: *elements*, *properties*, *center\_of\_gravity*, *mom\_of\_inertia*, and *errorcheck*. They are located in the `template/summary/ls-dyna` directory.

The *elements* summary template provides a listing of element types and the number of each type contained in the current model. It also indicates the total number of elements in the model.

The *properties* summary template describes the components and their properties contained in the model. The template lists the name, ID, and material ID for each component. Components that have the *\*PART* card image loaded also list the Section ID, the card option (for example *\*PART\_INERTIA*), and the mass contained in that component. The mass is calculated using the same procedure as the *mass* panel. See Mass Calculation.

The *center\_of\_gravity* summary template calculates the model's center of gravity based on the procedure described in the *Mass Calculations* section.

The *mom\_of\_inertia* summary template calculates the model's moment of inertia based on the procedure described in the *Mass Calculations* section.

There are also two templates that perform error checking on LS-DYNA models: *errorcheck* and *sharedrigids*.

- Use the *errorcheck* template in the **summary** panel. This template gives a brief overview of problems that exist in the model.
- Use the *sharedrigids* template in the **element check** panel. This template highlights the elements with nodes that are shared among rigids.

When either of the error-checking templates check for shared rigids, the template highlights only the second element that uses a node. If an element in a rigid material is sharing a node with another rigid, only the first element in that component is highlighted.

Possible error messages:

- Elements in components pointing to a *\*SECTION* card of the wrong type
- Zero or negative thickness
- Rigid elements, components, or interfaces sharing nodes
- Components do not have a *\*SECTION* card selected
- Components do not have a necessary *\*PART* card defined
- Components do not have a material selected

Note that a more advanced error check tool is available on the **Tools**Utility menu of the **Utility** tab. This tool checks components, rigids, joints, properties, boundary conditions, and other model entities and displays the results in a user-friendly format that enables you to select an error and go directly to the relevant part of the model to make corrections.

## Mass Calculation

The mass of each element is the calculation of density \* volume. When calculating model mass, several assumptions are made for the element volume. Densities are retrieved from the material associated with the element's component.

A \*PART\_INERTIA card uses the supplied mass instead of calculating the mass based on the individual elements.

Mass calculations include the mass supplied on the \*CONSTRAINED\_NODAL\_RIGID\_BODY\_INERTIA cards.

A shell element thickness is one of the following:

- the thickness on the first node for uniform thickness shells
- the average of three or four nodes for non-uniform thickness shells

The values come from the \*SECTION\_SHELL card, unless a \*ELEMENT\_SHELL\_THICKNESS card is defined for an element. If an \*ELEMENT\_SHELL\_THICKNESS card is defined, the element values override the \*SECTION\_SHELL values.

Integrated beams have an area equal to the average of the two end areas. Resultant beams use the area entered on the \*SECTION\_BEAM card. The volume is calculated by multiplying the length of the beam by the \*SECTION\_BEAM card area. Discrete beams use the volume supplied by the \*SECTION\_BEAM card. In all cases, if an \*ELEMENT\_BEAM\_THICKNESS card is defined for an element, then the element values override the \*SECTION\_BEAM values.

Only element masses are considered. Other mass specifications, such as on a *rigid wall* card, are ignored.

Mass calculation is also supported for node structural mass for beam and shell elements.

## LS-DYNA MAT100 Spotwelds

MAT100 spot welds are a mesh independent representation of spotwelds for the LS-DYNA solver. The spotwelds are modeled as beam elements (type 9) or hexa elements placed between any two deformable shell elements and tied with constrained contact. The beam or hexa elements are assigned a special material MAT\_SPOTWELD also known as MAT100. The tied contact eliminates the need for shared nodes between the spotweld and shell elements, thus making it mesh independent.

MAT100 welds can be created and managed in HyperMesh using connectors. Once a connector is created, they can be realized as MAT100 spotwelds as follows:

1. Load the LsDyna user profile from the **user prof...** panel on the **Geom** or **Tools** page.
2. Make sure that the connectors are created at each of the weld locations along with connecting parts information.
3. Make sure all the connecting parts are assigned SECTION\_SHELL property with correct thicknesses.
4. Select the connectors to be realized as MAT100 in the **fe realize** panel of **connectors** module on the **1D** page.
5. Choose **custom** element config and select **type = dyna 100 mat100** for MAT100 beams or **type = dyna 101 mat100 (hexa)** for MAT100 hexa representation. The appropriate **property** script is automatically loaded for the selected **type**.
6. Set the appropriate tolerance (**proj tol=**) value.

7. Make sure the **attach to shell** and **snap to node** options are turned off in **fe options...**
8. For **type = dyna 101 mat100 (hexa)** , select a DvsT file, which determines the size of the hexa based on the thicknesses of the components being connected. If no DvsT file is selected, hexas are created with weld nugget diameter =1.0.
9. Click **realize**.

For MAT100 Beams, the following actions are performed automatically:

\*Element\_Beam\_PID of type 9 is created for each layer at every connector's location. The created elements are organized into their corresponding components (C\_^\_1W\_234\_12 is the name of \*PART card that contains MAT100 beams that connect between part IDs 234 and 12).

Plot elements (\*Element\_Plottel) are created to connect the beams to the nodes of the surrounding shell. Use this connection to find elements attached to beams.

\*Section\_Beam card for every pair of connected parts is created (P\_^\_1W\_234\_12 is the name of \*SECTION property card assigned to MAT100 beams that connect between part IDs 234 and 12). The appropriate beam properties are calculated based on the connecting components thickness and the thickness lookup table in the weld\_config.ini file. The thickness lookup table contains the range of thickness values and the corresponding TS1 (and TS2) values. If the minimum of two connecting components thicknesses fall in a specific range below, that corresponding TS value is assigned to the SECTION\_BEAM.

```
# SAMPLE MATERIAL THICKNESS LOOKUP TABLE:
# NUMBER INDICATES NUMBER OF LEVELS
#LAST LINE:
*THICKNESS      5
0    0.1  0.125
0.1  0.5  0.25
0.5  1.0  0.75
1.0  1.5  1.25
1.5  2.0  1.75  2.0
```

The remaining entries of the section card are set to default.

A \*Mat\_Spotweld is created for every pair of components connected (M\_^\_1W\_234\_12 is the name of \*MATERIAL\_SPOTWELD card assigned to beams that connect between part IDs 234 and 12). The various failure parameters (NRR, NRT and NRS) are calculated based on the Yield strength of the connecting materials. The function uses the material lookup table in the weld\_config.ini file to determine these values. A sample table is shown below:

```
#MATERIAL STRENGTH LOOKUP TABLE
# FIRST NUMBER INDICATES NUMBER OF LEVELS
# SECOND NUMBER INDICATES MULTIPLIER FOR SIGY OF NUGGET
#MIN      MAX      k      n      a      b
#LAST LINE:
# MIN      MAX      k      n      a      b      NUMBER
*SIGY      4      1.85
0          0.20      4.0000  1.9000  10.500  -4.0000
```

0.20	0.40	4.2000	1.9500	12.000	-3.0000	
0.40	0.80	4.5000	1.9500	14.200	-2.0000	
0.80	0.90	4.7000	1.9900	16.500	-1.0000	9.00

$$\text{If } Bound_{lowerj} < SIGY_{base_k} \leq Bound_{upperj}, \text{ then } \begin{aligned} NRR_k &= k_j t_{min}^{n_j} \\ NRS_k &= NRT_k = a_j t_{min} + b_j \end{aligned}$$

$$\text{If } SIGY_{base_k} > Bound_n, \text{ then } NRR_k = NRS_k = NRT_k = number$$

A single \*Contact\_Spotweld is defined for the entire model as follow - the master is defined using a \*Set\_Part\_List with all the parts that are welded to the beam elements. Slave is defined with a \*Set\_Node\_List with all the nodes of the beam used in the contact.

For MAT100 Hexa the following actions are performed:

\*Element\_SOLID is created for each layer at every connector's location. The created elements are organized into their corresponding components (C\_^\_1W\_234\_12 – is the name of \*PART card that contains mat100 beams that connect between part ids 234 and 12). The size of the solid is based on the DvsT file.

Plot elements (\*Element\_Plottel) are created to connect the hexas to the nodes of the shell surrounding. Use this connection to find elements attached to beams.

\*Section\_SOLID is created for each of the weld parts containing the weld elements. (P\_^\_1W\_234\_12 – is the name of \*SECTION property card assigned to mat100 hexas that connect between part ids 234 and 12).

A \*Mat\_Spotweld is created for every pair of components connected (M\_^\_1W\_234\_12 – is the name of \*MATERIAL\_SPOTWELD card assigned to beams that connect between part ids 234 and 12). The various failure parameters (NRR, NRT and NRS) are calculated based on the Yield strength of the connecting materials. The function uses the material lookup table in the weld\_config.ini file to determine these values. A sample table is shown below:

```
#MATERIAL STRENGTH LOOKUP TABLE
# FIRST NUMBER INDICATES NUMBER OF LEVELS
# SECOND NUMBER INDICATES MULTIPLIER FOR SIGY OF NUGGET
#MIN      MAX      k      n      a      b
#LAST LINE:
# MIN      MAX      k      n      a      b      NUMBER
*SIGY      4      1.85
0          0.20    4.0000  1.9000  10.500  -4.0000
0.20      0.40    4.2000  1.9500  12.000  -3.0000
0.40      0.80    4.5000  1.9500  14.200  -2.0000
0.80      0.90    4.7000  1.9900  16.500  -1.0000  9.00
```

$$\text{If } Bound_{lowerj} < SIGY_{base_k} \leq Bound_{upperj}, \text{ then } \begin{aligned} NRR_k &= k_j t_{min}^{n_j} \\ NRS_k &= NRT_k = a_j t_{min} + b_j \end{aligned}$$

$$\text{If } SIGY_{base_k} > Bound_n, \text{ then } NRR_k = NRS_k = NRT_k = number$$

A single \*Contact\_Spotweld is defined for the entire model as follow - the master is defined using a \*Set\_Part\_List with all the parts that are welded to the hexa elements. Slave is defined with a \*Set\_Node\_List with all the nodes of the hexas used in the contact.

## Duplicate Entity IDs

IDs that are duplicated within element and property groups for the LS-DYNA user profile are supported in HyperMesh beginning with version 8.0. For example, elements like beam and shell can now be imported without renumbering even if they have the same ID. Although it is not recommended to use duplicate IDs, this support has been implemented for greater flexibility for users who need to import models that already contain duplicate IDs. This support helps to preserve data integrity between HyperMesh and the solver by mimicking the solver's data rules. This support is provided by several means.

LS-DYNA data decks that have duplicated IDs are no longer automatically renumbered by HyperMesh. The ID numbers for elements and properties are preserved regardless of duplication. However, renumbering is still in effect for other entity types.

On an FE input, if the **FE overwrite** option is selected, IDs are overwritten only within the same ID group.

On the **options** panel, a new option named **allow duplicate IDs** can be selected, which enables the duplication of IDs across pools of new elements or properties you create in HyperMesh. For example, ID 1 can be used for a shell element and a solid element. However ID 1 cannot be used for two shell elements. By default, the option is not selected and HyperMesh never creates entities with the same ID during renumbering or meshing, and uses a sequential numbering scheme. With the exception of importing decks, the HyperMesh support of duplicate IDs must be manually activated.

When you attempt to select properties or elements for editing or deletion and you choose the **by id** option, you now must select an ID group (such as ELEMENT\_SHELL or ELEMENT\_BEAM) in addition to the ID, if the ID is present in more than one group. This selection is required to identify specific IDs if there is duplication across ID groups. For example, if both beam and shell have element ID 1, and you select an element by ID=1, a pop-up window appears that allows you to select between the two available groups that contain ID 1.

## Components and Properties

The following list contains information about translating components and properties:

Define components and properties so that valid LS-DYNA cards are generated. For example, shells and solid elements cannot be defined in the same component.

Mass, rigid, and weld elements are placed into separate components during input.

The recommended order for creating components, properties, and materials is shown below:

1. Load Curves
2. Materials
3. Properties
  - Integration rules
  - Hour Glass definitions
  - Equations of State
  - Section properties
4. Components

This is the most efficient way to create the HyperMesh collectors. By following this order and using the **Create/Edit** function, you have to view and edit each card only once.

The material associated with a component is controlled by HyperMesh functionality and cannot be changed in the card previewer.

Component card hourglass information in Structured does not translate to Keyword decks.

When you edit component cards with a section property selected, an image of that card is displayed at the end of the component card image.

### To create the LS-DYNA Key for \*CONSTRAINED\_RIGID\_BODIES in HyperMesh:

1. Create an entity set of comps that are to be slave for a single master.
2. In the **collectors** panel:
3. Select **create**.
4. Click **create/edit** and select the master comp.  
or
5. Select card image.
6. Click **load/edit** and select the master comp.
7. Select RigidBodyMerge in the card image panel.
8. Click **Slave\_SID** and select the entity set with slave comps.
9. Load the LsDyna user profile to access the **Constrained Rgd Body** macro, available on the **Tool** page.

This macro allows you to graphically review master and slave components, master component IDs, slave component IDs, and slave set names (IDs).

## Load Collectors

Load collector information is specified with a required \$HMNAME comment card and an optional \$HMCOLOR comment card. If an input translator encounters one of these comments while reading a load card, a new load collector is created. For the comments to be valid, they must follow a load keyword or the last line of the previous *Structured* block. The loads that follow a \$HMNAME LOADCOLS comment are read into that collector. If there is a new *Keyword* or *Structured* block, HyperMesh ignores the previous load collector information.

For non-HyperMesh generated input decks, loads are divided into collectors based on classification. The following load collectors are created:

- Mechanical loads for forces and moments
- Constraints/Displacements
- Velocities
- Accelerations
- Pressures

If translational or rotational constraints are defined in the input model, they are placed in a separate load collector named ***Nodal Constraints***

Load collectors are not used by LS-DYNA, but are useful for visualization in HyperMesh. Additional load collectors can be defined to describe other entities.

HyperMesh	Keyword	Structured	Notes
InitialVel	*INITIAL_VELOCITY	Card 30	This card changes the INITV definition on Control Card 11. Only the first card defined is valid for <i>Structured</i> .
	*INITIAL_VELOCITY_GENERATION		
LoadBody	*LOAD_BODY_X	Card 39	Activate the <b><i>proper</i></b> option and enter the data. Only the first card defined is valid for <i>Structured</i> .
	*LOAD_BODY_Y	Card 40	Activate the <b><i>proper</i></b> option and enter the data. Only the first card defined is valid for <i>Structured</i> .
	*LOAD_BODY_Z	Card 41	Activate the <b><i>proper</i></b> option and enter the data. Only the first card defined is valid for <i>Structured</i> .
	*LOAD_BODY_RX	Card 42	Activate the <b><i>proper</i></b> option and enter the data. Only the first card defined is valid for <i>Structured</i> .
	*LOAD_BODY_RY	Card 43	Activate the <b><i>proper</i></b> option and enter the data. Only the first card defined is valid for <i>Structured</i> .



	*LOAD_BODY_RZ	Card 44	Activate the <i>proper</i> option and enter the data. Only the first card defined is valid for Structured.
	*LOAD_BODY_PARTS	Card 45	Select component set.
LoadBodyGen	*LOAD_BODY_ GENERALIZED	Card 46	
LoadRbody	*LOAD_RIGID_BODY		
PrcribRgd	*BOUNDARY_PRESCRIBED _MOTION_RIGID		RIGID_LOCAL and _SET options are supported.
Dform2Rigid	*DEFORMABLE_TO_RIGID *DEFORMABLE_TO_RIGID AUTOMATIC		Select an arraycount for the PSID and MRB pairs.  Change the option to automatic and card edit. In the D2R fields enter the number of PIDs that need to be converted to Rigid. Create an entity set of comps of the slave PIDs and select the set.
Dform2RgdInertia	*DEFORMABLE_TO_RIGID_ INERTIA		
BoundSPCset	*BOUNDARY_SPC_SET		

## Loads, Constraints, and Boundary Conditions

Several HyperMesh load types cause three cards to be output for x, y, and z components. During input, HyperMesh groups these into one load. Loads cannot be applied to sets, components, or boxes. Load curves are input and output. Use the Card Editor to select load curves.

HyperMesh Panel	Type	Keyword	Structured	Notes
<b>Constraints</b>	1	*BOUNDARY_SPC	Card 13 SPC	
	2	*BOUNDARY_ PRESCRIBED _MOTION_NODE	Card 26 VAD = 2	DOF 4, -4, 8, -8, 9, -9, 10, -10, 11, -11 are not supported
<b>Forces</b>	1	*LOAD_NODE_POINT	Card 23 Point Loads	LS-DYNA Load Configs 1, 2, 3 and 4
<b>Moments</b>	1	*LOAD_NODE_POINT	Card 23 Point Loads	LS-DYNA Load Configs 5, 6 7 and 8.
<b>Pressures</b>	2	*LOAD_SHELL_PRESSURE	Card 24 Pressure BC	
	1	*LOAD_SEGMENT		

<b>Velocities</b>	1	*BOUNDARY_PRESCRIBED_MOTION_NODE	Card 26 VAD = 0	DOF 4, -4, 8, -8, 9, -9, 10, -10, 11, -11 are not supported
	2	*INITIAL_VELOCITY	Card 30 INITV = 3	For Structured output, global velocity is set to 0.0. For Structured input, non-zero values for INITV = 1 or INITV = 5 create HyperMesh velocities. INITV values of 2, 4, 6, and 7 are ignored.
<b>Acceleration</b>	1	*BOUNDARY_PRESCRIBED_MOTION_NODE	Card 26 VAD = 1	DOF 4, -4, 8, -8, 9, -9, 10, -10, 11, -11 are not supported

## HyperMesh Groups

HyperMesh groups are created from the *interfaces* and *rigid wall* panels. An LS-DYNA entity that utilizes a \*SET\_ [NODE, SHELL, PART, etc.] Keyword card belongs to a HyperMesh group, with the exception of Rigid Bodies/RBE2's.

**REVIEW** allows you to efficiently visualize the entities defining master and slave. A transparency mode as well as the ability to turn on/off master and slave entities is also available. In review mode, **review opts** allows you to customize the graphical review of the interfaces.

The *interface* panel allows you to define groups with HyperMesh configurations of 1, 2, 3, and 4. The difference among these configurations is the type of entities contained within the group.

- Config 1** Holds master and slave elements.
- Config 2** Holds master elements and slave nodes.
- Config 3** Holds slave elements.
- Config 4** Holds slave nodes.

The *rigid wall* panel allows you to define a group with the HyperMesh configuration 5. This group configuration holds the additional geometric data for LS-DYNA **rigid wall** definitions.

### Sliding Interfaces

- Accessed via the interfaces panel.
- The Keyword \_TITLE option is supported. The \_THERMAL(IREAD==3) option is not supported.
- Use the additional cards option in Keyword decks to select number of lines of data. If this is on, two additional cards are available.
- In Structured, additional cards are controlled by using the IREAD variable. Valid values are 0, 1, and 2.
- Boxes, part sets, and sets are supported.
- The \$HMNAME fields are used for names. When using the \_TITLE option, the 70-character field is considered a comment.

- If the line following the keyword (No TITLE option), or the first line of the Structured card contains \$HM\_NAME, the name supplied is read and used as the group's name. If the string \$HM\_ID also exists, this is used as the group's ID. NAME is 16 characters, starting in Column 9. ID field is 8 characters, starting in Column 35.
- The HyperMesh interface type defines the general type of the LS-DYNA Sliding Interface. Use the card previewer to make changes to the LS-DYNA type.

HyperMesh	Option	Keyword *CONTACT_	Structured
SlidingOnly	Defines a *CONTACT_SLIDING_ONLY_option card.		
	Off	<nothing>	Type 1
	On	PENALTY	Type p1
SurfaceToSurface	Defines a *CONTACT_option_SURFACE_TO_SURFACE card.		
	None	<nothing>	Type 3
	Automatic	AUTOMATIC_	Type a3
The <b>None</b> and <b>Automatic</b> options have an additional option to define a <i>OneWayInterface</i> . If this option is on, the following cards are created.			
	None and OneWay	ONE_WAY_	Type 10
	Automatic and OneWay	AUTOMATIC_ONE_WAY_	Type a10
	Constraint	CONSTRAINT_	Type 17
	Eroding	ERODING_	Type 14
	TieBreak	TIEBREAK_	Type 9
The Tied option has an additional option to define OFFSET.			
	Tied	TIED	Type 2
	Tied and offset	TIED_OFFSET	
	Tiedshell	TIED_SHELL_EDGE	
	Tiedshell and offset	TIED_SHELL_EDGE_OFFSET	

NodesToSurface	Defines a *CONTACT_option_NODES_TO_SURFACE card.		
	None	<nothing>	Type 5
	Automatic	AUTOMATIC	Type a5
	Constraint	CONSTRAINT_	Type 18
	Eroding	ERODING_	Type 16
	TieBreak	TIEBREAK_	Type 8
	The Tied option has an additional option to define OFFSET.		
	Tied	TIED_	Type 6
	Tied and offset	TIED_OFFSET	
SingleSurface	Defines a *CONTACT_option_SINGLE_SURFACE card.		
	none	<nothing>	Type 4
	Automatic	AUTOMATIC_	Type 13
	Airbag	AIRBAG_	Type a13
	Eroding	ERODING_	Type 15
RgdBodyToRgd Body	Defines a *CONTACT_RIGID_BODY_option_TO RIGID_BODY card		
	Off	ONE_WAY_	Type 21
	On	TWO_WAY_	Type 19
RgdNodeToRgd Body	N/A	RIGID_NODES_TO_ RIGID BODY	Type 20
DrawBead	N/A	DRAW_BEAD	Type 23
Interior	Defines a *CONTACT_INTERIOR card.		
AutomaticGeneral	Defines a *CONTACT_AUTOMATIC_GENERAL card.		
ForceTransducer	Defines a *CONTACT_FORCE_TRANSDUCER_option card		
	On	PENALTY	
	Off	CONSTRAINT	

ContactSpotweld	Defines a *CONTACT_SPOTWELD card  none  Torsion	WITH_TORSION
ContactSingEdge	Defines a *CONTACT_SINGLE_EDGE card	

**To define a Type a5/\*CONTACT\_AUTOMATIC\_NODES\_TO\_SURFACE interface:**

1. Select the **NodeToSurface** type.
2. Click **create**.
3. Edit the card.
4. Click Automatic.

Note that the `Keyword` or `Type` field will change.

## Rigid Walls

The **rigid wall** panel supports planes (finite/infinite), prisms (finite/infinite), cylinders, and spheres. Motion options are available for all geometric configurations. For finite rigidwalls the negative numbers for LengthL and LengthM are treated as Infinite in the LS-DYNA solver. If defined negative, HyperMesh moves the coordinates of the base node to make LengthL and LengthM positive during export.

The described geometry determines the completion of the keywords and/or structured options.

HyperMesh	Keyword	Structured	Notes
XsectionPlane	*DATABASE_CROSS_ SECTION_PLANE		Previously on the <b>interfaces</b> panel.
RWGeometric	*RIGIDWALL_ GEOMETRIC	Stonewalls Card 27 LIMIT = 4, 5, 6, and 7	
RWPlanar	*RIGIDWALL_ PLANAR  The <code>_FORCES</code> option is also available.	Stonewalls Card 27 LIMIT = 1 and 2	ORTHO, FINITE, MOVING options supported.
ContEntity	*CONTACT_ENTITY	Geometric Contact Entity Card 58	Support for GEOTYPs 1, 2, 3, 6, 7, and 10.

# Creating Cards

## Control Cards

\*DATABASE\_OPTION cards in *Keyword* are listed on the *DBOpt* control card in HyperMesh. An active field is output as the appropriate individual card in the data deck.

A control card can be in one of three states:

State	Color	Explanation
Undefined	Gray	The control card was either never created or was deleted.
Defined (See <b>Note</b> )	Green	Any control card viewed in the card previewer is activated.
Inactive	Red	A card that has been defined may be disabled. The attributes for that card remain; however, the control card is not output.

**Note** Those control cards that are defined (green in the control card editor) are output.

Default values for attributes are common throughout the card previewer. A default value field has one of the following states:

State	Description
Default = ON	In this state, the field label color is yellow and no data entry is allowed.
Default = OVERRIDDEN	To override a default value field, pick the yellow field label. When you override a default value field, the label text color changes to cyan and you can enter data in the field.

## Systems

The systems cards can be previewed, but not edited.

## Nodes

These cards can be previewed, but not edited.

## Elements

1. To edit these cards, select **Card Editor** from the **Setup** pull-down menu.
2. Select the **elems** data type from the extended entity selection menu.
3. Select the elements to edit.
4. If more than one config or type of element is selected, enter the Config and Type to in the fields provided. Only one config and type of element can be edited simultaneously.

### **\*CONSTRAINED\_SPOTWELD/Card 13**

Normal and shear failure values can be edited.

### **\*CONSTRAINED\_GENERALIZED\_WELD/Card 36**

Spot(default)/type 1, Fillet/type 2, and Butt/type 3 failure modes are supported. Failure information is based on weld type selected. Coordinate System ID can be selected.

No Failure/Type 0 Card 36 entities are defined as \*CONSTRAINED\_NODAL\_RIGID\_BODIES in *Keyword*. They are a separate element type in HyperMesh.

### **\*ELEMENT\_DISCRETE/ Card 50**

Scale factor, printing flags, and offset values can be edited.

### **\*ELEMENT\_BEAM / Card 8**

Thickness option can be added. This allows you to edit the parameters based on the element formulation in the property to which the beam points.

### **\*ELEMENT\_SHELL / Card 9**

Thickness and beta options can be added singularly or together. This allows you to edit the thickness and material angles to override the SECTION card.

## Components, Properties, and Materials

These are edited using the **card image** subpanel on the **collectors** panel.

## Loads

1. To edit these cards, select **Card Editor** from the **Setup** pull-down menu.
2. Select the **loads** data type from the extended entity selection menu.
3. If more than one config or type of load is selected, enter the Config and Type to edit in the fields provided. Only one config and type of load can be edited simultaneously.

### **Editable Fields**

If a load is not mentioned below, it does not have any editable fields. However, that load can still be displayed in the card previewer. See the *Loads* section above for a complete list of the LS-DYNA entities that HyperMesh treats as loads.

### **\*LOAD\_NODE\_POINT / Card 23**

A load curve can be selected for these loads.

### **\*BOUNDARY\_SPC\_NODE / Card 13**

A local coordinate system can be selected.

### **\*BOUNDARY\_PRESCRIBED\_MOTION\_NODE / Card 26**

A loading condition release time can be specified. A load curve can also be specified.

## Curves

Edit by selecting **Card Editor** from the **Setup** pull-down menu. Editable values include stress initialization, offset values, and data type.

## Groups

Edit using the **card image** subpanel on the **Interfaces or Rigid Walls** panels.

## Output Blocks

Use the card previewer to view block entity IDs. Data on LS-DYNA output blocks cannot be edited.

## Sets

Use the card previewer to view the set entity IDs. The default LS-DYNA attribute values for the set can be edited. Individual values cannot be edited.

## Equations

Equations are used to define linear constraints in local and global coordinate systems.

HyperMesh	Keyword	Structured
Equations	CONSTRAINED_LINEAR	Section 66

## Curves

Output of curves creates a \*DEFINE\_CURVE or Structured Card 22 using Option 0

\*DEFINE\_CURVE can be changed to \*DEFINE\_TABLE in the card previewer. When DEFINE\_CURVE is changed to DEFINE\_TABLE, the number of curves the table should contain depends on the HyperMesh XY curves that are referenced by a load, material, component, property, and so on are output

Upon input, the \*DEFINE\_CURVE and \*DEFINE\_TABLE/Card 22 cards are read and placed in a plot called **LS-DYNA Load Curves**

Upon input, references to curves are preserved and are output along with the card, such as material, component, property, load and so on

Curves that are not referenced by HyperMesh entities can be output using the `curves.key/curves.seq` templates. These curves can then be pasted into the deck created by the `dyna936` templates. The `curves` templates output the curves that exist in the current model. By using the **output displayed** option and the **display** panel, a precise set of load curves can be created. This method allows you to create new curves or modify existing curves for LS-DYNA entities that HyperMesh does not support.



## Control Volumes

The **control volume** panel allows you to control volume objects within a model. The \*AIRBAG\_OPTION card is output from this panel. The POP option is supported for WANG\_NEFSKE options.

The following options are supported:

SIMPLE\_PRESSURE\_VOLUME

SIMPLE\_AIRBAG

WANG\_NEFSKE

WANG\_NEFSKE\_JETTING

HYBRID

HYBRID\_JETTING

**To create the above airbag cards in the *control vol* panel:**

1. Select the **airbag** subpanel.
2. Type a name for the airbag.
3. Select card image = airbag.
4. Select entities that are defined in the airbag.

There are three different ways to select entities that are defined in an airbag.

- Create a contact surf (\*SET\_SEGMENT) from the **contact surfs** panel and select that contact surf.
- Create an entity set of components (\*SET\_PART) in the **entity sets** panel and select that comp set.
- Select elements. This also exports at \*SET\_SEGMENT whose id is defined during export using the LS-DYNA keyword template.

5. Click **create**.
6. Once an airbag is created with the selected entities, click **edit** to bring up the card image.
7. Select the required OPTION of \*AIRBAG\_OPTION.

The following reference geometry card is supported:

\*AIRBAG\_REFERENCE\_GEOMETRY\_OPTION\_OPTION

BIRTH

\_RDT

**To create a reference geometry card in the *control vol* panel:**

1. Select the **reference geometry** subpanel.
2. Enter a name for the reference geometry.
3. Select card image = airbagRefGeom.
4. Select the nodes that define the reference geometry.
5. Click **create**.

After you create the reference geometry, you can manipulate the original mesh (i.e., rotate, deform, scale, or translate the mesh to simulate airbag folding or other conditions). When you review the reference geometry, it displays the original nodal locations at the time that reference geometry was created. In this way, the reference geometry definition can be used to preserve IMM locations of the nodes. Once the reference geometry is created, click on **edit** to bring up the card image. Select the required OPTION of `_BIRTH` or/and `_RDT`

## Results Translation

`hmdyna` translates files from the `force` database and `state` database to HyperMesh binary results files. To translate `force` database files, use the command line option `-force`. To translate `state` database files, use the command line option `-state`. `-state` is the default and is used for translating `d3plot` files. If neither `-force` nor `-state` is specified in the command line, `state` database files are translated by default. The syntax to run the translator is:

```
hmdyna [arguments] <input file> <output file> <model file>
```

The following options can be used in conjunction with `-state`.

Flag	Meaning
-t	Temperatures
-d	Displacements
-v	Velocities
-a	Accelerations
-xx	Sigma xx
-yy	Sigma yy
-zz	Sigma zz
-xy	Sigma xy
-yz	Sigma yz
-zx	Sigma zx
-von	von Mises stress
-ps	Plastic Strains
-ebv	Extra brick variables
-esv	Extra shell variables
-epxx	Epsilon xx
-epyy	Epsilon yy
-epzz	Epsilon zz
-epxy	Epsilon xy
-epyz	Epsilon yz
-epzx	Epsilon zx
-af	Axial force

-srs	Shear Resultant-s
-srt	Shear Resultant-t
-bms	Bending Moment-s
-bmt	Bending Moment-t
-tr	Torsional Resultant
-bmxx	Bending Moment-mxx
-bmyy	Bending Moment-myy
-bmxy	Bending Moment-mxy
-qxx	Shear Resultant-qxx
-qyy	Shear Resultant-qyy
-nxx	Normal Resultant-nxx
-nyy	Normal Resultant-nyy
-nxy	Normal Resultant-nxy
-th	Thickness
-eld1	Element Dependent Variable 1
-eld2	Element Dependent Variable 2
-ie	Internal Energy
-lower	Lower Surface
-upper	Upper Surface
-mid	Mid Surface
-max	Maximum of top and bottom surface values
--max	Option to turn off Maximum
-902	Version 902
-float	Float format
-3s1	3-D Principal Strain 1 (Minimum 3-D Principal Strain)
-3s2	3-D Principal Strain 2 (Maximum 3-D Principal Strain)
-3s3	3-D Principal Strain 1 (Second 3-D Principal Strain)
-3sh	3-D Maximum Shear Strain
-3S1	3-D Principal Stress 1 (Minimum 3D Principal Stress)
-3S2	3-D Principal Stress 2 (Maximum 3D Principal Stress)
-3S3	3-D Principal Stress 3 (Second Principal Stress 3D)
-3Sh	3-D Maximum Shear Stress

- stepN            Step increment. If N is 1, translation is performed for steps 1, 2, 3, and so on. If N is 2, translation is performed for steps 2, 4, 6, and so on.
- h3d             Outputs file to an H3D file instead of to an `hmresults` file. The file includes model and results information that was translated. The model must contain geometry for it to be output to an H3D file.

The following options can be used in conjunction with `-force`. `-force` must be specified as an argument and is used for the `iff` file from LS-DYNA. The `iff` file needs to be requested from LS-DYNA using the command line syntax `S=iff`.

<b>Flag</b>	<b>Meaning</b>
-d	Displacements
-v	Velocities
-nip	Normal Interface Pressure
-miss	Maximum Interface Shear Stress
-ssr	Shear stress in local r-direction of segment
-sss	Shear stress in local s-direction of segment
-xfe	X-Force on element (4-noded elements only)
-yfe	Y-Force on element (4-noded elements only)
-zfe	Z-Force on element (4-noded elements only)

The following options are common to both the `-state` and `-force` options:

<b>Flag</b>	<b>Meaning</b>
-stepN	Step increment. If N is 1, translation is performed for steps 1, 2, 3, etc. If N is 2, translation is performed for steps 2, 4, 6, and so on.
-disk	Translation is performed on disk (default off)
-size	Number of entities (10000 default)
-file	Scratch file name (default off)

The following parameters are also available when the results translation is not performed on the analysis machine. One of these parameters may need to be specified to indicate where the analysis result file was created:

<b>Parameter</b>	<b>Analysis File Created On</b>
-cray	Cray
-dec	Dec 5000
-decalpha	Dec Alpha
-hp	Hewlett Packard
-ibm	IBM RS\6000
-pc	PC
-sgi	SGI
-sun	Sun

When the number of integration points is more than 3, in the case of shell elements, LS-DYNA outputs as many sets of stresses as the number of integration points. The first three sets of these stresses are the midsurface, lower surface and upper surface stresses. The remaining stresses in the d3plot files are output in the following form in HyperMesh:

<stress var>, from setN, where <stress var> can be one of Sigmaxx, Sigmayy, Sigmazz, Sigmaxy, Sigmayz, and Sigmaxz. N varies from 4 to MAXINT (number of integration points), with an increment of 1.

The same rule applies to the plastic strains and additional history variables for shell elements.

The supported results for the `state` database include:

#### **nodal**

Temperatures

Displacements

Velocities

Accelerations

#### **bricks**

Sigma - xx (mid)

Sigma - yy (mid)

Sigma - zz (mid)

Sigma - xy (mid)

Sigma - yz (mid)

Sigma - zx (mid)

von Mises Stress (mid)

Effective plastic strain (mid)

History variables (mid)

Epsilon - xx (mid)

Epsilon - yy (mid)

Epsilon - zx (mid)  
Epsilon - xy (mid)  
Epsilon - yz (mid)  
Epsilon - zx (mid)  
Principal Strain 3D1 (mid)  
Principal Strain 3D2 (mid)  
Principal Strain 3D3 (mid)  
Maximum Shear Strain 3D (mid)  
Principal Stress 3D1 (mid)  
Principal Stress 3D2 (mid)  
Principal Stress 3D3 (mid)  
Maximum Shear Stress 3D (mid)

#### **brick shells**

Sigma - xx (lower, upper, mid and max)  
Sigma - yy (lower, upper, mid and max)  
Sigma - zz (lower, upper, mid and max)  
Sigma - xy (lower, upper, mid and max)  
Sigma - yz (lower, upper, mid and max)  
Sigma - zx (lower, upper, mid and max)  
von Mises stress (lower, upper, mid and max)  
Effective Plastic Strain (lower, upper, mid and max)  
Additional history variables (lower, upper and mid)  
Epsilon - xx (lower and upper)  
Epsilon - yy (lower and upper)  
Epsilon - zz (lower and upper)  
Epsilon - xy (lower and upper)  
Epsilon - yz (lower and upper)  
Epsilon - zx (lower and upper)  
Principal Strain 3D1 (lower and upper)  
Principal Strain 3D2 (lower and upper)  
Principal Strain 3D3 (lower and upper)  
Maximum Shear Strain 3D (lower and upper)  
Principal Stress 3D1 (lower, upper and mid)  
Principal Stress 3D2 (lower, upper and mid)  
Principal Stress 3D3 (lower, upper and mid)  
Maximum Shear Stress 3D (lower, upper and mid)

## **beams**

Axial Force

Shear resultant - s

Shear resultant - t

Bending moment - s

Bending moment - t

Torsional resultant

## **4-noded shells**

Sigma - xx (lower, upper, mid, max and at integration points 4, 5 etc up to MAXINT)

Sigma - yy (lower, upper, mid, max and at integration points 4, 5 etc. up to MAXINT)

Sigma - zz (lower, upper, mid, max and at integration points 4, 5 etc. up to MAXINT)

Sigma - xy (lower, upper, mid, max and at integration points 4, 5 etc. up to MAXINT)

Sigma - yz (lower, upper, mid, max and at integration points 4, 5 etc. up to MAXINT)

Sigma - zx (lower, upper, mid, max and at integration points 4, 5 etc. up to MAXINT)

von Mises Stress (lower, upper, mid and max)

Effective Plastic Strain (lower, upper, mid, max and at integration points 4, 5 etc up to MAXINT)

Additional history variables (lower, upper, mid and at integration points 4, 5 etc. up to MAXINT)

Bending moment - mxx

Bending moment - myy

Bending moment - mxy

Shear resultant - qxx

Shear resultant - qyy

Normal resultant - nxx

Normal resultant - nyx

Normal resultant - nxy

Thickness

Element Dependent Variable1

Element Dependent Variable2

Internal Energy

Epsilon - xx (lower and upper)

Epsilon - yy (lower and upper)

Epsilon - zz (lower and upper)

Epsilon - xy (lower and upper)

Epsilon - yz (lower and upper)

Epsilon - zx (lower and upper)

Principal Strain 3D1 (lower and upper)

Principal Strain 3D2 (lower and upper)

Principal Strain 3D3 (lower and upper)  
 Maximum Shear Strain 3D (lower and upper)  
 Principal Stress 3D1 (lower, upper and mid)  
 Principal Stress 3D2 (lower, upper and mid)  
 Principal Stress 3D3 (lower, upper and mid)  
 Maximum Shear Stress 3D (lower, upper and mid)

The supported results for the *force* database include the following:

**nodal**

Displacements

Velocities

**4-noded elements**

Normal interface pressure

Maximum interface shear stress

Shear stress in local r-direction of segment

Shear stress in local s-direction of segment

X-force on element

Y-force on element

Z-force on element

**3-noded elements**

Normal interface pressure

Maximum interface shear stress

Shear stress in local r-direction of segment

Shear stress in local s-direction of segment

The following table lists the LS-DYNA elements and their HyperMesh equivalents for the *state* database:

<b>LS-DYNA</b>	<b>HyperMesh</b>
Bricks	Hex8
Brick Shells	Hex8
Beams	Plot
4-noded shells	Quad4

The following table lists the LS-DYNA elements and their HyperMesh equivalents for the *force* database:

<b>LS-DYNA</b>	<b>HyperMesh</b>
4-noded elements	Quad4
3-noded elements	Tria3



## Viewing the Results

If the model file option is selected, an ASCII model file is created. You can use this ASCII model file to view the model in HyperMesh.

### To import the model file and view the results:

1. Select the *files* panel.
2. Select the *results* subpanel.
3. Click *browse...* and select a file from the *Open file* browser
4. Click *Open*.
5. Select the *import* subpanel.
6. Click *FE*.
7. Select a translator using the toggle.
8. Select read include files or skip include files.
9. Select FE overwrite or no FE overwrite.
10. Click *import...* and select a file from the *Open file* browser.
11. Go to the *Post* page, which contains the *contour* and the *transient* panels.

The results can be viewed as a contour or assign plot, or as a transient animation.

## Exporting Decks

HyperMesh can output the following LS-DYNA files:

- LS-DYNA v970 and v960 input files in *Keyword* format
- LS-DYNA v936 input files in *Structured* format
- By default, the LS-DYNA user profile outputs v970 *.key* files.
- To output *Keyword* decks, use the `feoutput/ls-dyna960/dyna.key` for the 960 format or the `/feoutput/ls-dyna/dyna.key` template file for the newest LS-DYNA 970.
- To output regular format *Structured* decks, use the `dyna.seq` template file
- To output large format *Structured* decks, use the `dyna.lrg` template file
- Two templates are also provided to output the defined curves in the database:
- To output curves in *Keyword* format, use the `curves.key` template
- To output curves in *Structured* format, use the `curves.seq` template

**Note:** When converting an LS-DYNA model to the RADIOSS format, check and update the element types before writing the converted model to file. Some formulations may not map correctly, such as RADIOSS SHELL 3N to LS-DYNA ACCEL.

### To create an analysis deck via a template:

1. Select the **files** panel.
2. Select the **export** subpanel.
3. Click **TEMPLATE**.
4. Select **all** or **displayed**.
5. Click **load...** and select a template using the **Open file** browser.
6. Click **Open**.
7. Click **write as...** and designate a file using the **Save file** browser.
8. Click **Save**.

## LS-DYNA Utility Menu

The LS-DYNA Utility menu on the **Utility** tab of the project explorer is automatically loaded when you select the LsDyna user profile, and contains shortcuts and tools that can help simplify LS-DYNA tasks. Set the user profile from the **User Profiles...** option of the **Preferences** pull-down menu.

The LsDyna user profile sets the FE input reader to DYNA KEY and loads the `dyna.key` (ver 970) FE output template and LS-DYNA Utility menu. Also, the graphical user interface becomes LS-DYNA focused, renaming or removing some panels and/or options. The entire **ALE Setup** is available only when the LsDyna user profile is loaded.

### Tools Menu

The LS-DYNA Utility menu contains a **Tools** menu in addition to the standard HyperMesh Utility menu. This menu includes special time-saving setup macros and other features that are specific to an LS-DYNA analysis. The following macros are available:

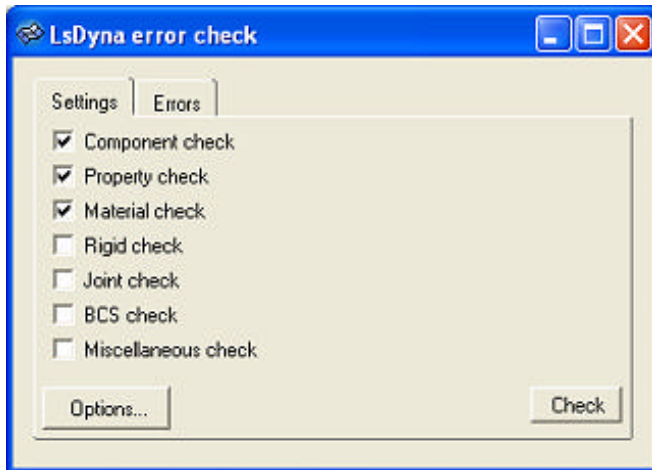
<b>Error Check</b>	Checks the LS-DYNA data deck for errors.
<b>Part Info</b>	Displays statistics of a selected part.
<b>Name Mapping</b>	Converts differing part names to either the HyperMesh name or the LS-DYNA name.
<b>Clone Part</b>	Creates a new part from the properties of an existing part.
<b>Create Part</b>	Creates a new component quickly.
<b>Part Replacement</b>	This macro allows you to replace the elements in an existing component (*PART) with new elements.
<b>Constrained Rgd Body</b>	This macro allows you to view master components, master component IDs, slave component IDs, and slave set names (IDs).

<b>Convert To Rigid</b>	<p>This macro converts a selected portion of elements to rigid. It performs the following:</p> <ul style="list-style-type: none"> <li>• Organizes elements to rigid components</li> <li>• Creates and assigns the required *MAT_RIGID cards</li> <li>• Converts welds to *CONSTRAINED_EXTRA_NODES</li> <li>• See Convert To Rigid Flow Chart.</li> </ul> <p><b>How do I...</b> Use the Convert To Rigid macro</p>
<b>Find free</b>	<p>Finds the welds (*Constained_Spotweld), rigids (*Constrained_Node_Sets &amp; *Constrained_Nodal_RigidBody), and rigidlinks (*Constrained_Node_Sets and *Constrained_Nodal_RigidBody), and checks if any of its nodes are free (not connected to any other entities). The display is cleared and then only <i>free</i> 1d elements are displayed.</p>
<b>Find Fix Free</b>	<p>Finds the welds, rigids, and rigidlinks that are free as described above (<b>Find free</b> macro) and corrects them. These elements are corrected as follows:</p> <p>All 2-noded rigid and weld elements that have one free node are deleted. For the rigidlink elements that have free nodes, those nodes are removed from the rigidlink element. A check is performed for any rigidlinks with only one node and they are deleted.</p>
<b>Fix Incorrect</b>	<p>Finds:</p> <ul style="list-style-type: none"> <li>• Rigid elements (rigids, welds) that are connected to other rigids and combines them into one rigid element.</li> <li>• Rigid elements that are connected to other xtra_nodes_to_rigidbodies and converts them to xtra_nodes.</li> <li>• Rigid elements connected directly to rigid component (MAT 20) will be converted to xtra_nodes.</li> </ul>
<b>RLs With Sets</b>	<p>The macro, <b>RLs with Sets</b>, finds all the rigid and rigidlink elements that are not attached to a set and converts them so that they are attached to a set.</p>
<b>Content Table</b>	<p>Displays a tabular list of all the components that exist in the model.</p>
<b>Material Table</b>	<p>Allows you to easily create and edit materials.</p>
<b>C-Interfto50</b>	<p>Converts all the contacts that are defined using node sets in the <b>entity sets</b> panel or segment sets to master and slave elements in groups so that they can be easily displayed on/off.</p>

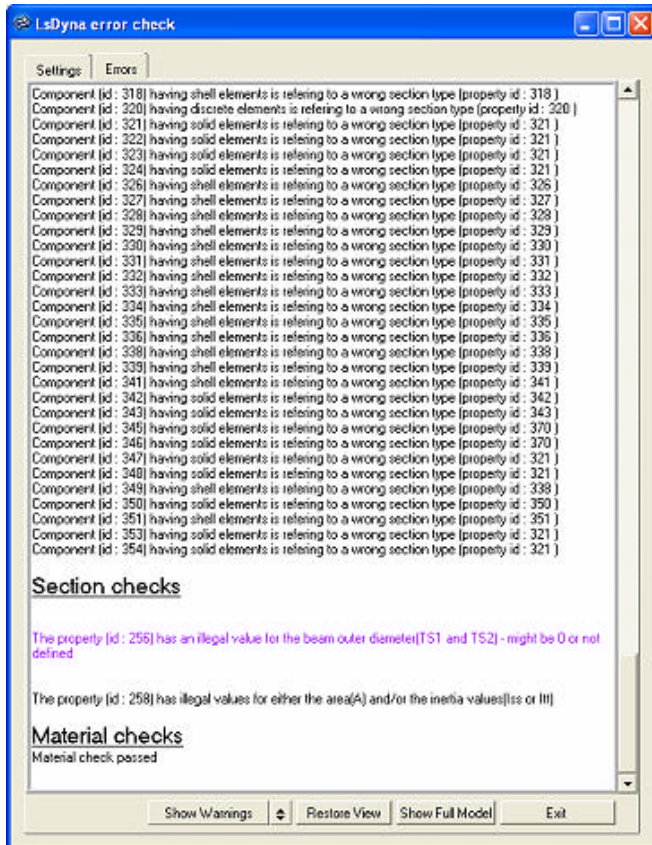
## Error Check

The **Error check** macro checks your LS-DYNA deck for potential problems with components, properties, materials, rigids, joints, boundary conditions, and other entities and reports them on-screen. The report identifies the problem entity by ID, describes the error, and then enables you to isolate the entity in the model and quickly make changes.

Click **Error check** on the LS-DYNA Utility menu to open the macro. The macro opens in a dialog as shown below:



Select the types of errors for which you want to search and click **Check**. When the check is complete, the results appear on the **Errors** tab of the dialog, as shown below:



Each error in the list is a hyperlink that, when clicked, highlights the affected visualizations in the model and opens the relevant card image or panel for correcting the error.

You can systematically click on each error in the list, correcting them as you go. On the **Settings** tab, click **Check** again to verify that the errors were corrected.

If you want to restore the full view of the model including all components, click the **Show full model** button on the **Errors** tab of the dialog. To return to the previous view, click **Restore View**.

Use the **Options** menu button to update or saving settings for the **Error Check** macro. You can specify minimum and maximum values for the material check and a maximum value for the distance of a constrained extra node to its part. To save the current settings, choose **Save Settings...** from the **Options** menu button and specify a file name and location. You can also load previously-saved error check settings.

Click the **Exit** button to close the error checking tool's results.

## Part Info

The **Part Info** macro summarizes a part's statistics in a dialog.

To start the macro, click **Part Info** on the Utility menu. Click **component** on the main menu area to select a component or click a component in the graphics area to select it. Then click **proceed**. The **Part Information** dialog appears, which lists the part ID, name, thickness, and material type. To view additional statistics about the part, click **More Detail>>**. To display statistics for a different part, select the part in the graphics area or the **components** selector and click **proceed** again.

**Tip** Click the middle mouse button instead of the **proceed** button to quickly select components.

## Name Mapping

LS-DYNA and HyperMesh maintain separate names for parts. To make the names consistent, you can run the **Name Mapping** macro, which provides the ability to change names for various entity types to either the HyperMesh name or the LS-DYNA name. This macro can be accessed by clicking **Name Mapping** on the **DYNA Tools** macro page when the LsDyna user profile is loaded.

Select whether you want to convert the HyperMesh names to LS-DYNA names or vice-versa by choosing the corresponding radio button at the top of the dialog. Then select the entity group(s) you want to update by clicking its row in the entity list. Click **Convert selected**; the names for all the entities that exist in the selected groups are automatically changed to either the HyperMesh or LS-DYNA format, depending on which setting is active.

The **Custom...** option provides the ability to change individual entities instead of an entire entity group. A new dialog appears when you click the **Custom...** button. All the entities of that type are listed in a new table, from which you can select individual entities and click **Edit** to open the card image and manually change the name or click **Apply** to automatically match names based on the current setting of the main **Name Mapping** dialog box.

**Note** If there is no card image available, the LS-DYNA name does not appear in the **Custom...** table and name mapping is not available.

## Clone Part

The **Clone Part** macro enables you to quickly create a new part from the properties of an existing part. It can be accessed by clicking **Clone Part** on the **DYNA Tools** macro page when the LsDyna user profile is loaded.

Select the existing part on which to model the new part by clicking the ... button, which opens a dialog listing all the existing components. Select a component from the list and click **OK**.

Type a name for the new part in the **New Part** field and click the color icon to select a color for the component.

Select whether to duplicate the material and section properties or to re-use the original material and section properties. **Duplicate** means that a new material and section is created (the name is suffixed with .n version numbers and new IDs are used) with the same properties, while **Reuse** refers to the same material and section as the original.

Click **Create>>** to either create or create and edit the card.

## Create Part

The **Create Part** macro enables you to create components on-the-fly. It can be accessed by clicking **Create Part** on the **DYNA Tools** macro page when the LS-DYNA user profile is loaded.

Type a name for the new component in the **Part name** field and select a color by clicking the adjacent color icon.

Select a section in the **Section** field by choosing **Create New** (create a new section), **Same As** (create a new section based on an existing section), or **Model...** (select an existing section) from the selection menu.

Select a material for the component in the **Material** field by the same method as described above for the **Section** field.

Click **Create>>** to either create or create and edit the card.

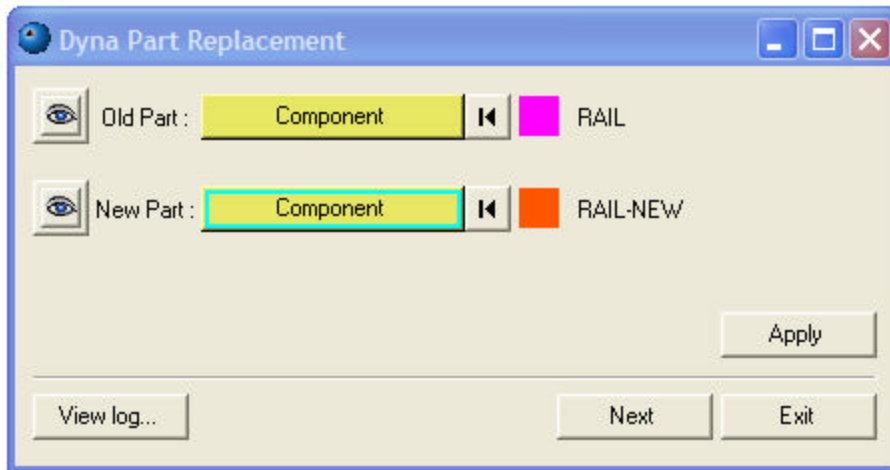
## Part Replacement Macro


The **Part Replacement** macro allows you to replace the elements in an existing component (\*PART) with new elements; typically replacing a similar part remeshed or slightly reshaped. It can be accessed in the **Tool** macro page when the LsDyna user profile is loaded.

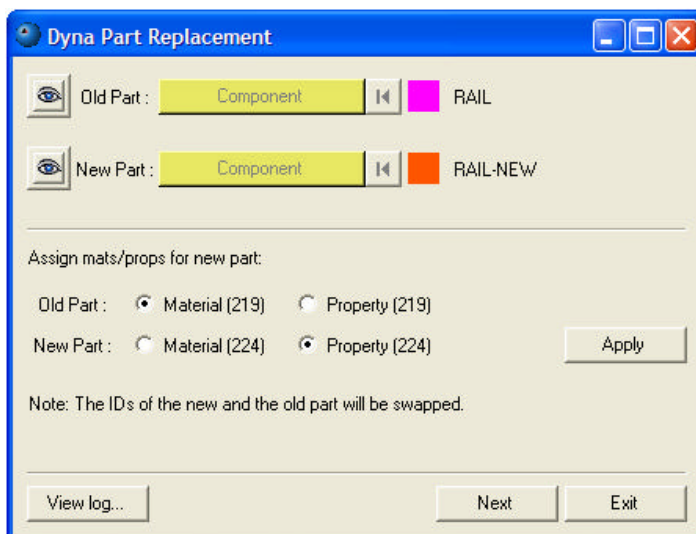
This macro not only replaces nodes and elements between parts, it also restores the referenced items in the original model to the new part, e.g. 1-D connections, distributed mass, contacts, loads, and database history. A message log is provided, which lists the entities being replaced and reconnected as well as cases that required or will require user interaction.

### To replace parts with the Part Replacement macro:

1. Select the old and new part. Both parts must be available in the HyperMesh database.
  - Identify the **Old Part** and **New Part**. The name and color of the components are reported once the parts are selected.
  - Click **Apply**.



- Click the icon, , to turn on/off the corresponding part from the graphics area.
  - Click **View log...** anytime during the part replacement process to view a list of events.
2. Assign the material and property.



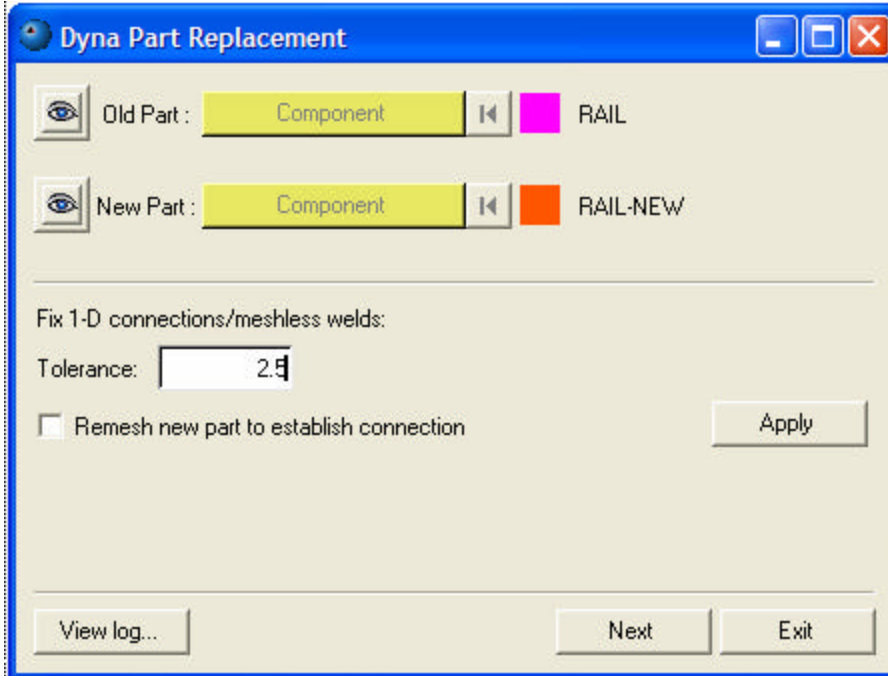
- Click **Apply** to accept the selection or click **Next** to skip this step and proceed with the part replacement process.

Note that the IDs of the new and old part will be swapped. This automatically preserves any LS-DYNA card that refers to this part ID directly or through a set of parts.

3. Fix 1-D connections and mesh-less welds.

This step offers both an automatic and interactive reconnection to the new part for 1-D elements (e.g. beams, rigids, and springs) and mesh-less welds (beam type 9 and hexa).

- For **Tolerance**:, specify a tolerance value for the reconnection attempt.



- Check **Remesh new part to establish connection** to allow a local remesh of the new part to restore connection.

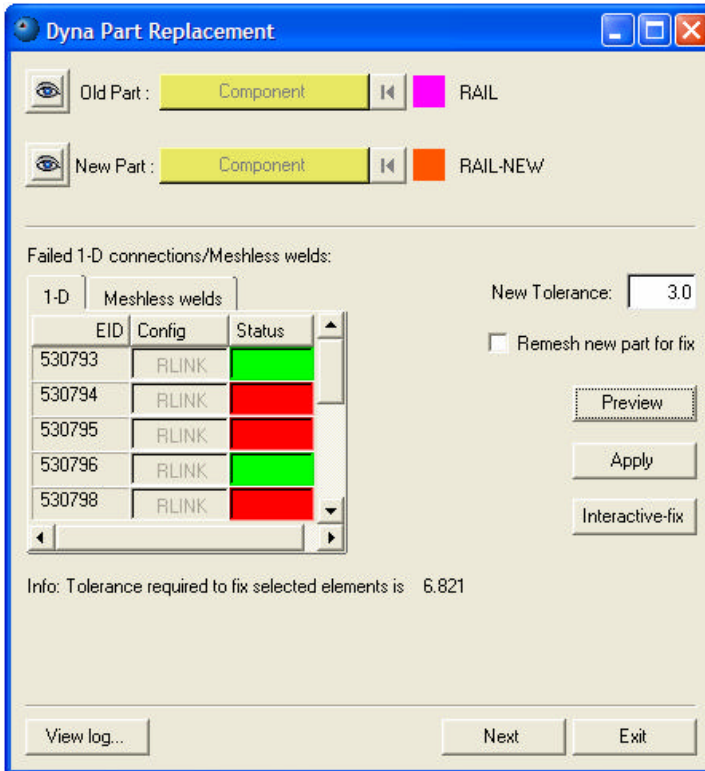
If you select **Remesh new part to establish connection**, HyperMesh will locally remesh the new part to establish the connection. In this case, the tolerance specified is the projection distance between the end node of the 1-D and the closest element in the new part. A new element is created and the 1-D connection may be restored with a smaller tolerance value.

If you do not select **Remesh new part to establish connection**, the value specified for the research tolerance will be the nodal distance between the end node of the 1-D element and the closest node in the new part. In this case, the 1-D element keeps its original ID and properties; only the node previously connected to the old part will be moved.

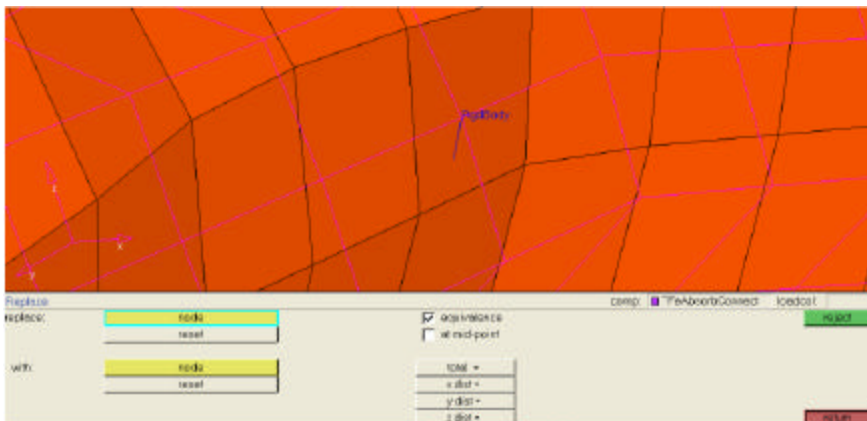
- Click **Apply** to replace the 1-D connection and the mesh-less welds within that tolerance and display the elements that cannot be fixed in red.



- Click the **EID** field to select the remaining elements, increase the tolerance, and preview the effect of the increased value on the 1-D elements.



- Click **Apply** to use the defined tolerance to fix the elements displayed in green.
  - Note:** A message reports the tolerance required to fix the selected elements. This tolerance is used to fix all the 1-D connections.
- Use a higher tolerance value to fix all 1-D elements that are still reported as failing or select one or more 1-D elements and click **Interactive-fix**.
  - Interactive-fix** is recommended for cases where you want to directly monitor the nodes being connected and is only available for 1-D elements replaced using nodal tolerance.
- Unnecessary entities will be masked and the **replace** panel will be opened. The 1-D element requiring an interactive fix will have one end already detached and a node of the new part can be selected as needed. You must select the 1-D end node as the first node and a node of the new part as the second node.



- Use the **Meshless welds** tab to replace beam type 9 or a hexa used in a mesh-less connection. The same preview functionality described for regular 1-D connections is also available. **Interactive-fix** and the **Remesh new part for fix** are disabled since they do not apply to this type of connection.

A contact spotweld, materials, and properties for the mesh-less welds will also be created if the new part shows a different thickness or material information.

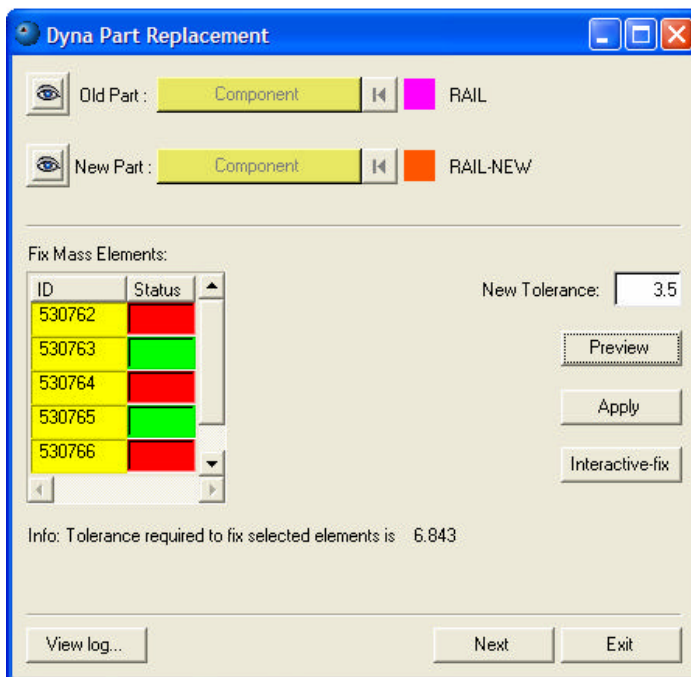
- Review the log file created during the part replacement to determine if any connections remain unfixed.

4. Fix mass elements. Masses attached to the old part can be connected to a new part using steps similar to the ones previously illustrated for 1-D elements.

- Specify a tolerance value for the mass element reconnection when prompted.

The value specified for research tolerance will be the nodal distance between the node of the old part where the mass was originally located and the closest node in the new part.

- Click **Apply** to replace the masses within that tolerance and display the elements that could not be fixed in red.
- Click the **EID** field to select the remaining elements, increase the tolerance, and preview the effect of the increased value on the mass elements.



- Click **Apply** to use the defined tolerance to fix the mass elements displayed in green.
- Use a higher tolerance value to fix all 1-D elements that are still reported as failing or select one or more 1-D elements and click **Interactive-fix**.

A message reports the tolerance required to fix the selected elements. This tolerance is used to fix all the mass connections.

## Additional Entities

The **Part Replacement** macro not only replaces elements, it also restores the referenced items in the original model to the new part.

### Contact and Rigidwall

Since the IDs of the new and old part are swapped at the beginning of the part replacement process, most of the common contact definition will be automatically preserved since that part ID does not change and will still be unchanged in any \*set\_part\_list.

In some instances, a contact or rigidwall may be defined by a set of nodes, set of elements, or set of segments. Consider the case of a contact node-to-surface, where the slave entity is defined using a set of nodes. The contact will be updated only when the contact contains every node of the old part. When this condition is not met, HyperMesh will report that the contact was not updated and that user interaction is required. A similar approach is used when a contact is defined using a set of elements or set segment (contactsurf).

### Database History

HyperMesh detects and fixes Database\_history\_nodes and Database\_History\_shell.

To fix a history\_node or history\_shell, all the nodes/shells must follow the tolerance that you have specified. For shells, the tolerance will not be a nodal distance between nodes but the distance between the element centroid in the old part and its projection (if any) to the elements of the new part.

### Constrained\_extra\_node

HyperMesh detects and fixes constrained\_extra\_node and constrained\_extra\_node\_set.

To fix a constrained\_extra\_node all its nodes must follow the tolerance that you have specified. Click **Preview** to identify the tolerance value required to fix a particular xtrinode.

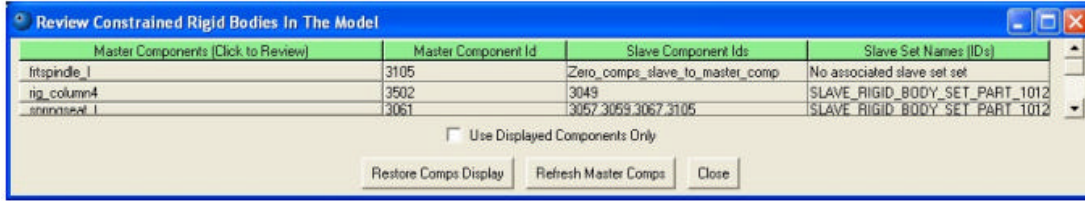
### Boundary Condition

HyperMesh can detect and fix the following individual loads: temperature, moments, constraints, and forces.

## Constrained Rgd Body

The **Rigid Body Review** macro is located in the **Tools** menu of the LS-DYNA Utility menu. It allows you to view the following:

<b>Master Components</b>	Click on a master name to display only this part and its slave parts. In the list, the master's row is highlighted. In the graphics area, the master part is highlighted in white.
<b>Master Component Id</b>	Master rigid body part (component collector) IDs.
<b>Slave Component Ids</b>	Corresponding slave rigid body part IDs.
<b>Slave Set Names (IDs)</b>	Corresponding HyperMesh entity set names/IDs for the slave rigid body part IDs.



The Constrained Rgd Body macro

The following functions are available in the **Constrained Rgd Body** macro.

<b>Use Displayed Components Only</b>	This check box allows you to view a list of rigid bodies from only the displayed component collectors.
<b>Restore Comps Display</b>	This button restores the graphical display to what it was when the macro was opened.
<b>Refresh Master Comps</b>	This button refreshes the list display after you have modified rigid bodies.
<b>Close</b>	This button closes the macro.

#### To use the Convert To Rigid macro:

This macro is used to convert deformable parts of an LS-DYNA model to rigid.

1. From the **Geom** or **Tool** page click **user prof...**
2. From the **User Profile** dialog, select **LsDyna**.
3. Click **OK**.
4. Click **Tools** in the macro panel.
5. Click **Convert To Rigid**.
6. Click **OK** from the pop up window.
7. Select the elements to convert to rigid.
8. Click **return**.

The **Convert To Rigid** macro performs the following steps when the selected elements are converted to rigid.

- For the selected elements, a check is performed on the comps for rigid (MAT\_RIGID or matl20) or deformable materials (all, except matl20). If deformable materials exist, rigid materials (MAT\_RIGID) are created with the properties from the original deformable materials. A check is performed for rigid materials that are already defined. If rigid materials are found, the comps and rigid materials are retained.
- Comps located partially within the window are split into two comps. The new comp has the same property (section ID) but new material (Material ID). For example, if A-pillar is partially within the window, then a new comp A-pillar\_rig is created. A-pillar\_rig is updated with newly created material.
- All the spotwelds and rigids located entirely within the window are removed. For example, \*CONSTRAINED\_NODAL\_RIGID\_BODY\_option, \*CONSTRAINED\_NODE\_SET, \*CONSTRAINED\_SPOTWELD, and \*CONSTRAINED\_GENERALIZED\_WELD\_option.
- For spotwelds that are connected from the deformable body to the rigid body, an extra node is created and referenced by the master rigid body.

- A check is performed to detect joints located partially or entirely within the window. Detected joints are deleted.
- A check is performed to detect springs located partially or entirely within the window. Detected springs are deleted.
- A check is performed to detect seatbelt elements (seatbelt elements, Retractor, Pretensioner) located partially or entirely within the window. Detected seatbelt elements are deleted.
- Master and slave comps are defined (for example, CONSTRAINED\_RIGID\_BODIES). You are prompted to select a comp for master rigid body. A slave set is created with the newly created rigid bodies (except the master rigid body comp).
- A message is displayed when the conversion is complete.

## LS-DYNA Content Table

The LS-DYNA Content Table is an interactive tabular list used to represent LS-DYNA components with associated properties and materials. It is accessed by loading the LsDyna user profile and clicking the **Content Table** button on the LS-DYNA **Tools**Utility menu.

Part name	Part id	Material name	Material id	Material type	Thick	Section name	Section id	Section type
Frame2Blw MTS RH 417	500024	MATL24_100275	502758	MATL24	4.17	SectShll_100417	500417	SectShll
LeafSpring LH Shck Frt 381	500025	MATL24_100275	502758	MATL24	3.81	SectShll_100381	500381	SectShll
LeafSpring LH Shck Rear 375	500026	MATL24_100423	504236	MATL24	3.75	SectShll_100375	500375	SectShll
LeafSpring RH Shck Frt 381	500027	MATL24_100275	502758	MATL24	3.81	SectShll_100381	500381	SectShll
LeafSpring RH Shck Rear 375	500028	MATL24_100423	504236	MATL24	3.75	SectShll_100375	500375	SectShll
Rail LH A-Arm Mt Up 6004	500029	MATL24_100275	502758	MATL24	6.004	SectShll_106004	506004	SectShll
Rail LH A-Arm Mt Lwrf 303	500030	MATL24_100275	502758	MATL24	3.03	SectShll_100303	500303	SectShll
Rail LH A-Arm Mt Lwrf 606	500031	MATL24_100275	502758	MATL24	6.06	SectShll_100606	500606	SectShll
Rail LH Crush INI 303	500032	MATL24_102756	502756	MATL24	4.0	SectShll_100400	500400	SectShll
Rail LH Double gage 500 top	500033	MATL24_100275	502758	MATL24	5.0	SectShll_100500	500500	SectShll
Rail LH Double gage 606 BOT	500034	MATL24_100275	502758	MATL24	7.1	SectShll_100710	500710	SectShll
Rail LH Double gage 606 TOP	500035	MATL24_100275	502758	MATL24	7.1	SectShll_100710	500710	SectShll
Rail LH Inner 303	500036	MATL24_100275	502758	MATL24	4.0	SectShll_100400	500400	SectShll
Rail LH Outer 303	500037	MATL24_100275	502758	MATL24	4.0	SectShll_100400	500400	SectShll
Rail RH A-Arm Mt Up 6004	500038	MATL24_100275	502758	MATL24	6.004	SectShll_106004	506004	SectShll

The table contains a variety of tools that allow you to review, edit, and update the model. The essential features are:

- LS-DYNA components with various associated properties and materials are listed in separate columns.
- You can select the column types from a set of available options.
- There are two modes of operation: review and editable. The review mode allows you to quickly review the component information without changing any values. The editable mode, allows you to change values for the selected components.
- There are enhanced selection, review, display, and filter options for components.
- Components can be sorted according to any available column.
- The current configuration is saved automatically to a file at the end of a session and recalled on reload. You can also save and load a configuration file.
- The table data can be export in CSV and HTML formats.
- Right click on the table to display menu options. All pull-down menu options are also available using a right click.

- Columns can be moved or swapped by holding the left mouse button on a column title and dragging it to the desired location.
- Columns can be resized by positioning the cursor along a column border, pressing the left or right mouse button, and dragging the border to a new position.
- The SHIFT or CTRL key combined with a left click can be used to select multiple rows.

The following tools are available in the LS-DYNA Content Table:

#### Table

<b>Refresh</b>	Regenerates the table with all the parts in the model
<b>Editable</b>	Sets the table mode to editable mode, allowing you to change values for the selected components
<b>Filter</b>	Enables the filtering GUI
<b>Configure</b>	Allows you to specify the number and type of columns listed in the table
<b>Save</b>	Saves the information listed in the table in CSV or HTML format

#### Selection

<b>All</b>	Selects all rows or parts
<b>None</b>	Selects none or deselects parts/rows that were previously selected
<b>Reverse</b>	Reverses the selection
<b>Displayed</b>	Selects the rows or the displayed parts
<b>User</b>	User graphic interaction to select parts

#### Display

By default, the table is invoked with only the displayed parts. You can refresh the table to show a new part being displayed or use one of the following display commands.

<b>All</b>	Displays all the components in the model
<b>None</b>	Turns off every component displayed
<b>Reverse</b>	Reverses the display of the part
<b>Show selection</b>	Displays the components of the selected rows
<b>Show only Selection</b>	Displays only the components of the selected rows
<b>Hide selection</b>	Hides the components of the selected rows from the display

#### Action

<b>Delete Selection</b>	Deletes selected rows (parts) from the model
-------------------------	--

## User

<b>Set MatDB Path...</b>	Opens a dialog on which you can set the location of an external database of material definitions.
<b>Refresh Material List</b>	Updates the list of available materials in the Content Table.

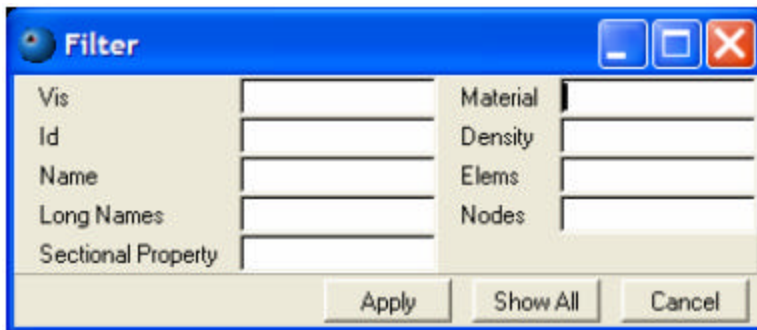
## Editable Mode

The editable mode in the Content Table allows you to change values for all selected components at the same time. Select the **Table→Editable** option to open the Content Table in editable mode. Cells with a white background can be manually edited. When you click on an editable cell, it is selected with a cursor. Once a cell is selected, enter a value and press ENTER.

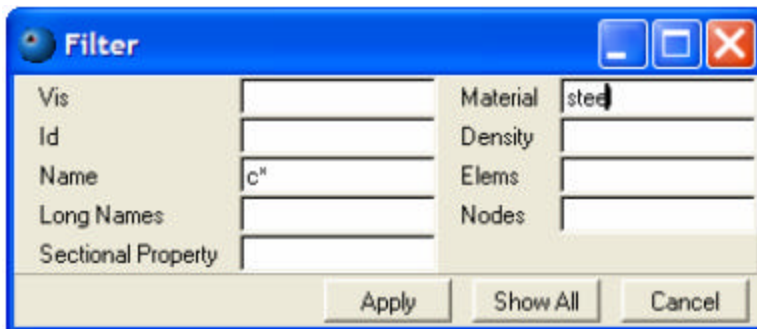
If you want to assign the same value to multiple components at once, select the column type and value from the **Assign Values:** pull-down menu and click **Set**. All the selected components will be updated with the assigned values.

## Filter

The Content Table supports advanced filtering based on available columns. The **Table→Filter...** menu option opens the **Filter** dialog as shown below.



You can write any valid string with a wildcard (\*) in any of the available column types and click **Apply** to filter the table. For example, if you want to show all components that start with letter 'c' and use material 'steel', you can use the dialog as shown below. Note that the filter strings are case-sensitive.

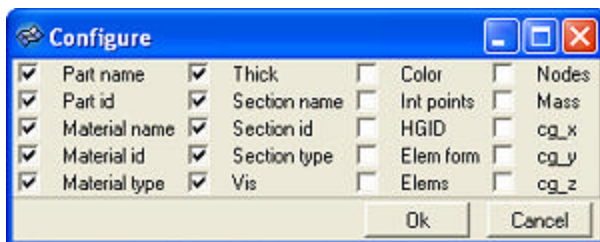


**Show All** turns off the filtering and displays all the components. Select the **Table → Configure → Filter on top** option to keep the **Filter** dialog posted after clicking **Apply** or **Show All**. Otherwise, it closes.

## Configure Columns

Column types can be selected from the **Table → Configure → Columns...** menu option. The table displays only the selected columns. The available columns types are:

<u>Title</u>	<u>Description</u>
<b>Part name</b>	HyperMesh name of the component (maximum 32 characters)
<b>Part id</b>	HyperMesh ID of the component
<b>Material name</b>	Material name associated with the component
<b>Material id</b>	Material ID associated with the component
<b>Material type</b>	Material type associated with the component
<b>Thick</b>	Thickness of elements specified in *section_shell
<b>Section name</b>	*Section name associated with the component
<b>Section id</b>	*Section ID associated with the component
<b>Section type</b>	Type of the *Section associated with the component
<b>Vis</b>	Visualization status. 1 = display on, 0 = display off
<b>Color</b>	Component color
<b>Int points</b>	Number of integration points specified for the *Section_shell
<b>HGID</b>	Hourglass ID associated with component
<b>Elem form</b>	Element formulation for the *section of the component
<b>Elms</b>	Number of elements in the component
<b>Nodes</b>	Number of nodes in the component
<b>Mass</b>	Total mass of the component
<b>cg_x</b>	Center of gravity for the x coordinate
<b>cg_y</b>	Center of gravity for the y coordinate
<b>cg_z</b>	Center of gravity for the z coordinate



## Components All or Displayed mode

The Content Table lists components in two modes: **All** or **Displayed**. If **All** is selected from the **Table → Configure → Components** menu, the table will list all the components in the model. If **Displayed** is selected, only the visible components will be shown. Blank components are not shown in the **Displayed** mode even though their display status is on.

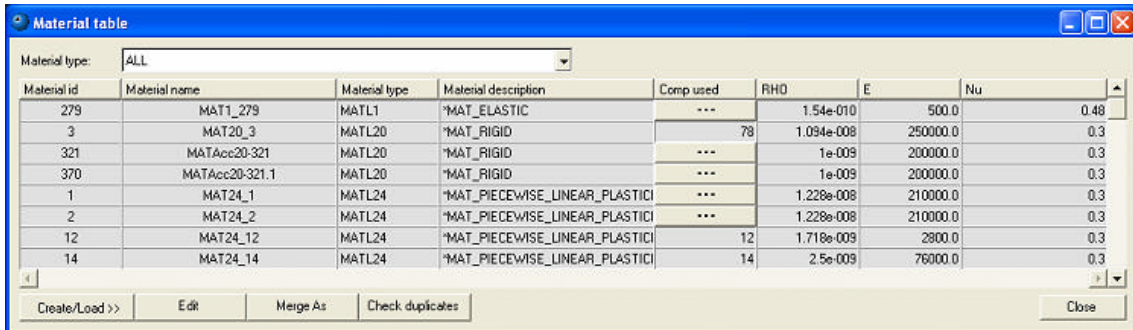


## LS-DYNA Material Table

The LS-DYNA Material Table enables you to easily create and edit materials in HyperMesh for LS-DYNA. To access the Material Table, open HyperMesh, load the LsDyna user profile, import an LS-DYNA model, and click **Material Table** on the Utility menu **Tools** page. All the existing materials are retrieved and populated in the table.

From the Material Table, you can also merge identical materials, search for duplicate materials, and change the properties of materials.

When you first display the Material Table, all materials are listed in the table, showing the material's ID, name, type, description, list of components in which it is used, and the RHO, E, and Nu values. An example is shown below.



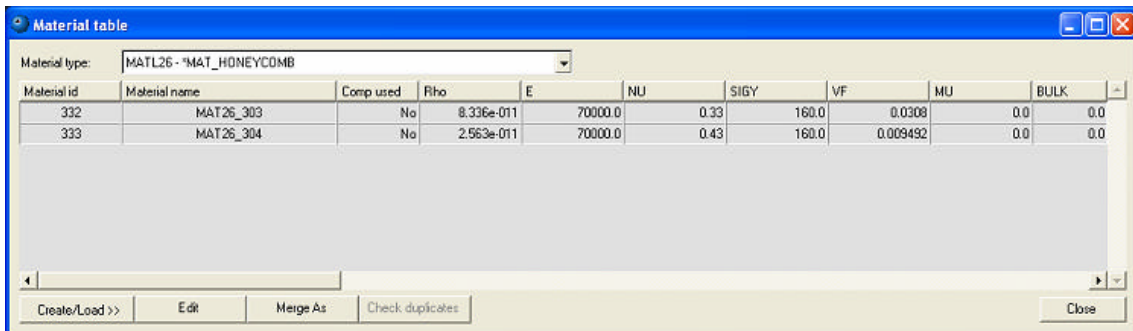
Material id	Material name	Material type	Material description	Comp used	RHO	E	Nu
279	MAT1_279	MATL1	*MAT_ELASTIC	---	1.54e-010	500.0	0.48
3	MAT20_3	MATL20	*MAT_RIGID	79	1.094e-008	250000.0	0.3
321	MATAcc20-321	MATL20	*MAT_RIGID	---	1e-009	200000.0	0.3
370	MATAcc20-321.1	MATL20	*MAT_RIGID	---	1e-009	200000.0	0.3
1	MAT24_1	MATL24	*MAT_PIECEWISE_LINEAR_PLASTICI	---	1.228e-008	210000.0	0.3
2	MAT24_2	MATL24	*MAT_PIECEWISE_LINEAR_PLASTICI	---	1.228e-008	210000.0	0.3
12	MAT24_12	MATL24	*MAT_PIECEWISE_LINEAR_PLASTICI	12	1.718e-009	2800.0	0.3
14	MAT24_14	MATL24	*MAT_PIECEWISE_LINEAR_PLASTICI	14	2.5e-009	76000.0	0.3

Materials in the table can be selected by clicking the row, which is then highlighted in blue. Many functions are performed by selecting materials in the table and choosing an option from the context menu or clicking a button below the table. SHIFT+click and CTRL+click can be used to select multiple rows. Refer to the links below for details about using the Material Table.

## Customizing Views of the Material Table

The Material Table initially lists all existing materials, but you can sort and filter the list to more easily identify materials that you want to work with. Each of the columns in the table can be used to sort the list. Click the column heading to sort by that characteristic, such as ID number or material type. To view only materials of a particular type, select that type in the **Material type** drop-down at the top of the window. For example, if you want to identify materials that are not used so you can delete them, you can click the **Comp used** column heading to quickly group together all materials that contain the value "No", which indicates that none of the components use the material.

**Note** To view all material properties in the table, select a material type from the drop-down. When all material types are shown in the table, only the **RHO** (density), **E** (Young's modulus), and **Nu** (Poisson's ratio) properties appear. However, when a particular material type is displayed, all the relevant properties for that material type also appear in the table, as shown in the image below.



Material id	Material name	Comp used	Rho	E	NU	SIGY	VF	MU	BULK
332	MAT26_303	No	8.336e-011	70000.0	0.33	160.0	0.0308	0.0	0.0
333	MAT26_304	No	2.563e-011	70000.0	0.43	160.0	0.009492	0.0	0.0

The material table also enables you to view the model's components based on the material used. These options are available by selecting **Display** from the menu that appears when you right-click anywhere in the table. Options include:

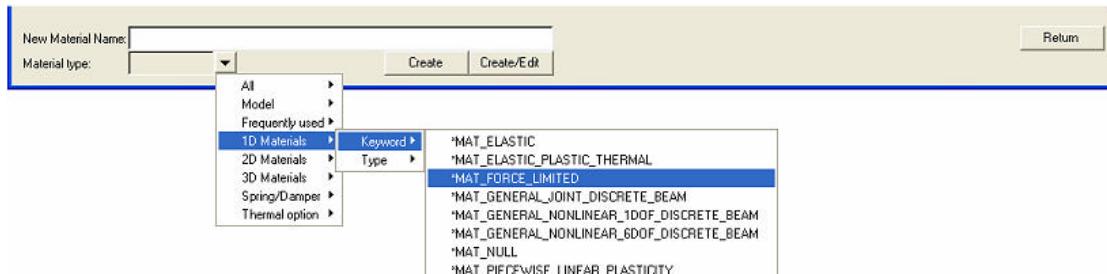
- viewing only the selected materials
- hiding the selected materials
- viewing all or none of the materials
- adding the selected materials to the current display
- reversing the current display option.

Once you make your selection, the corresponding components appear or become hidden in the graphics area of HyperMesh.

## Creating, Editing, and Loading Materials

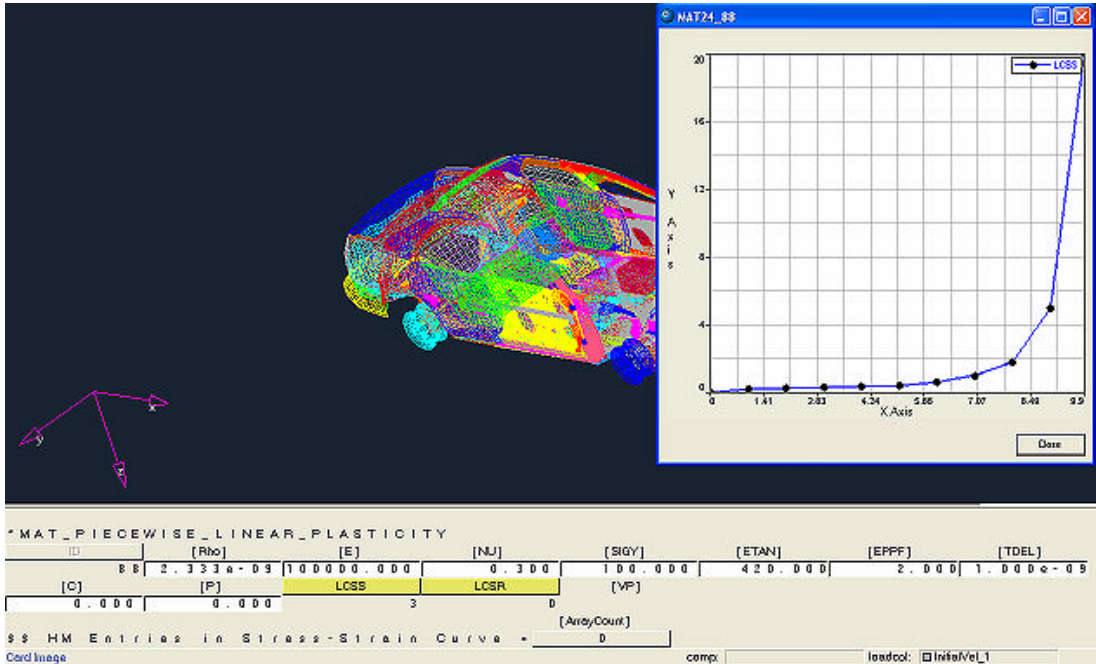
You can create, edit and load materials all from within the Material Table. Materials can be added or modified with the **Create/Load** and **Edit** buttons or by selecting the same options in the menu that appears when you right-click anywhere inside the table. To save time, you can choose the **Same As** selection to begin creating a material with the same properties as the currently-selected material in the table.

When you create a new material, you specify a name and the type of material. The materials are conveniently organized into categories, including groups of recently used materials and only materials that exist in the model. These categories are further listed by the LS-DYNA keyword or type identifier, as shown in the following image.



You can add the material to the table immediately by clicking **Create** or go to the card image panel to specify its properties by clicking **Create/Edit**.

At any time you can select a material in the table and click the **Edit** button to open the material's card image in HyperMesh. In the card image, you can modify values for the keyword's variables. In addition, the material's load curve appears in a pop-up graph, as shown below.

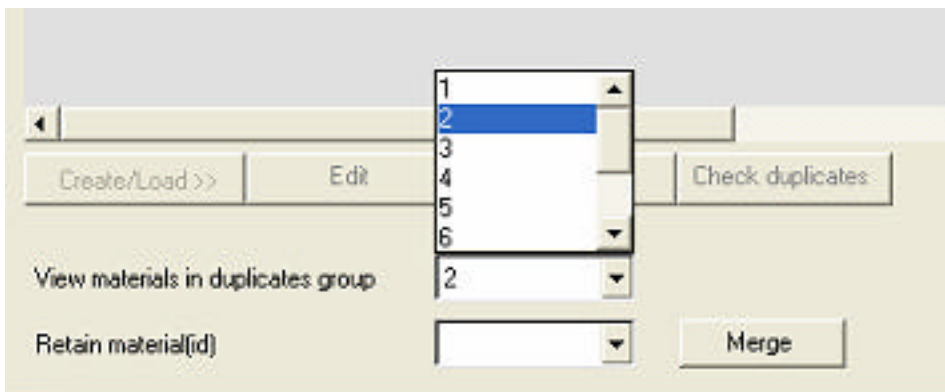


## Managing Materials

In addition to viewing, creating, modifying, and deleting materials, you can also identify duplicate materials, merge like materials into one, and rename materials.

The names of materials and the material IDs can be edited directly in the table. (All other values must be edited with the **Edit** button, which opens the card image in HyperMesh.)

Materials that have the same properties can be identified using the **Check duplicates** button. This feature, which is only available when all materials are displayed in the table, finds all materials that have identical properties and returns them in result sets. You can then select each result set to view the matching materials. Optionally, you can merge the duplicate materials into one material using the **Merge** button, which is the same feature as described in the following paragraph.



When you select multiple materials from the table, you can merge them into one of the selected materials using the **Merge As** button. Typically this action is performed on materials with like properties to simplify a model, although it can be performed on dis-similar materials with all selected materials taking on the properties of one of the materials. When materials are merged into one, the remaining materials still exist and appear in the table, but do not have any components assigned to them.

#### To sort materials:

1. Click the column heading of the criteria by which you want to sort.
2. Click the column heading again to list the materials in reverse order.

See Customizing Views of the Material Table to learn about other ways to filter the list of materials in the table.

#### To create a new material:

You can create a new material, or create a new material based on an existing material. Both procedures are described below.

##### To create a new material:

1. Click **Create/Load** and select **New...** from the menu. New fields appear at the bottom of the Material Table.
2. Type a name for the material in the **New Material Name** field.
3. Select a material type from the drop-down list. The list expands to categories of material types, and also sorts them by keyword or material ID. You can view the complete list of material types under the **All** category.
4. Click **Create/Edit** to open the material card image in HyperMesh to specify the properties, or click **Create** to add the material to the table without immediately specifying any properties.
5. Click **Return** to exit the Create/Load mode.

##### To create a new material based on an existing material:

1. Select a material in the table that you want to use as the basis for a new material.
2. Click **Create/Load** and select **Same as...** from the menu. New fields appear at the bottom of the Material Table. The material you selected appears in the **Selected material** field.
3. Type a name for the material in the **New Material Name** field.
4. Click **Create/Edit** to open the material card image in HyperMesh to specify the properties, or click **Create** to add the material to the table without immediately specifying any properties.
5. Click **Return** to exit the Create/Load mode.

##### To edit a material's properties:

1. Select a material in the table that you want to edit.
2. Click the **Edit** button. The card image for the material appears in HyperMesh and, if applicable, the load curve appears in a pop-up window.
3. Modify values in the card image and click **return** to go back to the Material Table.

**To merge materials:**

1. In the table, select the materials you want to merge. (Use SHIFT+click to select multiple, consecutive rows and CTRL+click to select non-consecutive rows.)
2. Click **Merge As**. The Material Table expands to include new fields for merging materials.
3. Select the material ID to use as the new material in the **Retain material(id)** field.
4. Click the **Merge** button. The components for each of the selected materials are merged into the material you selected. The remaining materials still exist and are listed in the table, but they are not assigned to any components.

**To find duplicate materials:**

1. Ensure that ALL is selected in the **Material type** field.
2. Click the **Check duplicates** button. The Material Table expands to include new fields for handling duplicate materials.
3. Choose a group number from the **View materials in duplicate group** field. The materials in that group appear in the table. (The results for the duplicates check are divided into consecutively numbered groups of the same material type.)

You can easily merge the duplicate materials using the **Merge** button. See How Do I Merge Materials? for steps on using the merge feature.

To view another result group of duplicate materials, select another group number from the **View materials in duplicate groups** field. That group's list of duplicate materials appears in the table.

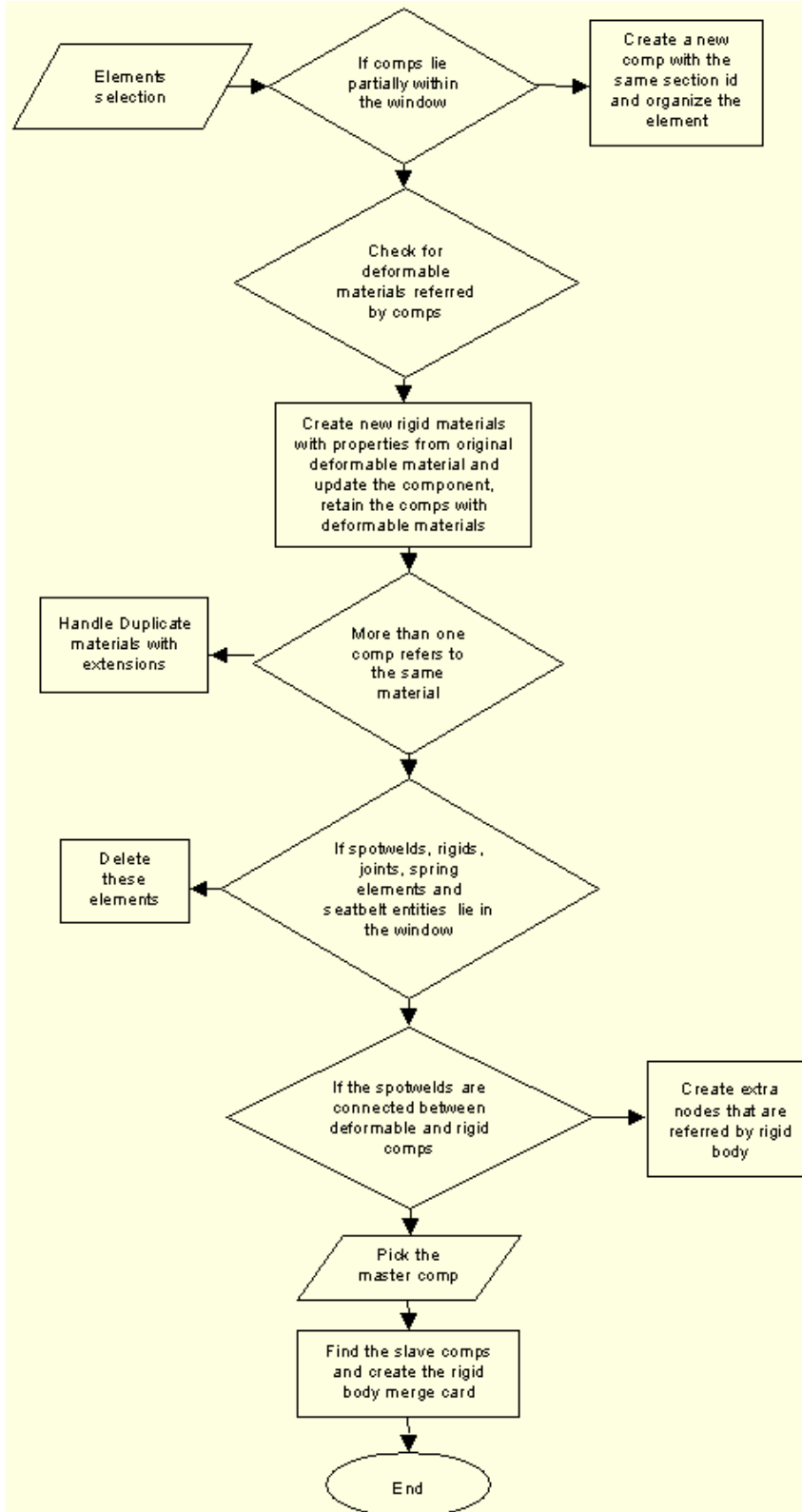
**To see the load curve for a material:**

1. Select a material in the table for which a load curve ID has been defined.
2. Click the **Edit** button. The load curve appears in a pop-up window.

**To export data from the material table:**

1. Select a material type or ALL from the **Material type** field to export only materials of a particular type or all materials, respectively.
2. Right-click anywhere in the table and select **Save** and then **CSV** for comma- or semicolon-separated values or **HTML** for an HTML-based table. The **Select output file** dialog appears.
3. Browse for or type a name in the **File name** field and click **Save**. The file containing material data is saved in the location you specified.

## Convert To Rigid Flow Chart



### To read load curves from disk:

Load curves can be read from disk by reading a LS-DYNA file that contains only load curve data (\*DEFINE\_CURVE) via the `feinput` translator.

1. Select the **xy plots** panel on the Post page.
2. Select the **read curves** panel.

The data must be in the following format:

Format	Example
XYDATA, <curve name>	XYDATA, curve1
<x1>,<y1>	0.000000, 2.000000
<x2>,<y2>	1.000000, 3.000000
<x3>,<y3>	2.000000, 4.000000
ENDDATA	ENDDATA

### To write load curves to disk:

Follow these steps to output a data curve as an XY file (XYDATA ... ENDDATA).

1. Select the **xy plots** panel on the **Post** page.
2. Select the **simple math** panel.
3. In the **plot =**, **1<sup>st</sup> curve =**, and **2<sup>nd</sup> curve =** fields select the plot and curve to be written to disk.
4. Select the **external** operation to select an external filter to be applied to the curve.
5. In the **target =** field, select the same curve as the one to be written to disk.

This results in the curve being overwritten by itself. Ignore the warning message when **execute** is selected.

6. Select the **todisk** filter in the **filter** box or use a copy command such as `/bin/cp` (the path must be present).
7. Enter the name of the output file in the **params** field.
8. Click **execute**.
9. Click **OK** to overwrite the curve in the **1<sup>st</sup> curve =** field.