Chapter 6.
Classes

Topic: Ignore

The class interface is a set of C++ classes, including their public and protected data and methods (member functions), that an application can use directly to interact with ACIS. Developers may also derive their own classes from these classes to add application-specific functionality and data. Refer to the *3D ACIS Online Help User's Guide* for a description of the fields in the reference template.

DM_act_icon

Class: Deformable Modeling

Purpose: This class is used for drawing area constraint tag objects in deformable modeling.

Derivation: DM_act_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmacticon.hxx

Description: The DM_act_icon is part of the dmicon library. This icon uses a DM_dbx_icon to depict an area constraint on a deformable surface.

Related classes: The DM_def_icon_cmd_args and the DM_def_icon_query_args encapsulate many of the methods inherited from the DM_default_icon interface. Thus the Set_state and Query methods can be used to perform casting, when different libraries of icons are mixed. The DM_def_icon_draw_args and the DM_def_icon_cmd_args can be broadcast to icons using the deformable modeling interface methods, such as DM_draw_all_icons. A DM_act_icon owns a DM_dbx_icon, providing an example of icon aggregation.

Limitations: None

References: DS DM_dbx_icon

Data:

None

Constructor:

```
protected: DM_act_icon::DM_act_icon (
    const DM_act_icon&      // icon
    );
```

Copy constructor; call Make_copy and Set_owner instead.

```
public: DM_act_icon::DM_act_icon ();
```

Public constructor.

Destructor:

```
protected: virtual DM_act_icon::~DM_act_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_act_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args&// draw-command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_act_icon::Get_colors (
    DM_dbl_array&           // data array
    ) const;
```

Query color data.

```
public: virtual void DM_act_icon::Get_grid (
    int[2]                  // density
    ) const;
```

Query discretization grid density.

```
public: virtual void DM_act_icon::Invalidate ();
```

Mark cached data as bad.

```
public: virtual DM_icon* DM_act_icon::Make_copy (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_act_icon::Set_colors (
    const double*,          // color data (r,g,b)
    int                     // array size
                            // number of doubles
    );
```

Set color data.

```
public: virtual void DM_act_icon::Set_draw_option (
    int                     // option
    );
```

Set draw option.

```
public: virtual void DM_act_icon::Set_grid (
    const int[2]            // density
    );
```

Set discretization grid density.

```
public: virtual void DM_act_icon::Set_icon_width (
    double                  // width
    );
```

Set icon draw width.

```
public: virtual void DM_act_icon::Set_owner (
    Spatial_abs_hurler&,    // error handler
    DS_dmod* new_dmod,      // icon owner ID
    int new_tag             // icon owner ID
    );
```

Notification of owner completion, for icon initialization.

```
public: virtual void DM_act_icon::Tag_object_changed
(
    Spatial_abs_hurler&     // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_act_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DM_ald_icon

Class: Deformable Modeling

Purpose: This class is used for drawing area load tag objects in deformable modeling.

Derivation: DM_ald_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmaldicon.hxx

Description: The DM_ald_icon is part of the dmicon library. This icon depicts an area load on a deformable surface. The method DM_add_area_C0_load is used to create an area load.

Limitations: None

References: DS DM_dbl_array

Related Fncs:

None

DM_att_icon

Class: Deformable Modeling

Purpose: This class is used for drawing point attractor tag objects in deformable modeling.

Derivation: DM_att_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmatticon.hxx

Description: The DM_att_icon is part of the dmicon library. This icon updates a marker to depict the current position of an attractor.

Related classes: The DM_def_icon_cmd_args and the DM_def_icon_query_args encapsulate many of the methods inherited from the DM_default_icon interface. Thus the Set_state and Query methods can be used to perform casting, when different libraries of icons are mixed. The DM_def_icon_draw_args and the DM_def_icon_cmd_args can be broadcast to icons using the deformable modeling interface methods, such as DM_draw_all_icons.

Limitations: None

References: None

Data:

None

Constructor:

```
protected: DM_att_icon::DM_att_icon (
    const DM_att_icon&      // icon
    );
```

Copy constructor; call Make_copy and Set_owner instead.

```
public: DM_att_icon::DM_att_icon ();
```

Default constructor.

Destructor:

```
protected: virtual DM_att_icon::~DM_att_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_att_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args& // draw-command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_att_icon::Get_colors (
    DM_dbl_array&           // data array
    ) const;
```

Query color data.

```
public: virtual DM_icon* DM_att_icon::Make_copy (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_att_icon::Set_colors (
    const double*,          // color data (r,g,b)
    int                     // array size
                            // number of doubles
    );
```

Set color data.

```
public: virtual void DM_att_icon::Tag_object_changed
(
    Spatial_abs_hurler&     // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_att_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DM_cct_icon

Class: Deformable Modeling

Purpose: This class is used for drawing curve constraint tag objects in deformable modeling.

Derivation: DM_cct_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmccticon.hxx

Description: The DM_cct_icon is part of the dmicon library. This icon updates a polyline to depict the current state of a curve constraint.

Related classes: The DM_def_icon_cmd_args and the DM_def_icon_query_args encapsulate many of the methods inherited from the DM_default_icon interface. Thus the Set_state and Query methods can be used to perform casting, when different libraries of icons are mixed. The DM_def_icon_draw_args and the DM_def_icon_cmd_args can be broadcast to icons using the deformable modeling interface methods, such as DM_draw_all_icons.

Limitations: None

References: DS DM_dbl_array

Data:

None

Constructor:

```
protected: DM_cct_icon::DM_cct_icon (
    const DM_cct_icon&      // icon
    );
```

Copy constructor; call Make_copy and Set_owner instead.

```
public: DM_cct_icon::DM_cct_icon ();
```

Public constructor.

Destructor:

```
protected: virtual DM_cct_icon::~DM_cct_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_cct_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args& // draw-command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_cct_icon::Get_colors (
    DM_dbl_array&           // data array
    ) const;
```

Query color data.

```
public: virtual void DM_cct_icon::Get_grid (
    int[2]                  // density
    ) const;
```

Query discretization grid density.

```
public: virtual DM_icon* DM_cct_icon::Make_copy (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_cct_icon::Set_colors (
    const double*,          // color data (r,g,b)
    int                     // array size
                            // number of doubles
    );
```

Set color data.

```
public: virtual void DM_cct_icon::Set_grid (
    const int[2]            // density
    );
```

Set discretization grid density.

```
public: virtual void DM_cct_icon::Tag_object_changed
(
    Spatial_abs_hurler&     // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_cct_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DM_cld_icon

Class: Deformable Modeling

Purpose: This class is used for drawing curve load tag objects in deformable modeling.

Derivation: DM_cld_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmcldicon.hxx

Description: The DM_cld_icon is part of the dmicon library. This icon updates a polyline to depict the current state of a curve load.

Related classes: The DM_def_icon_cmd_args and the DM_def_icon_query_args encapsulate many of the methods inherited from the DM_default_icon interface. Thus the Set_state and Query methods can be used to perform casting, when different libraries of icons are mixed. The DM_def_icon_draw_args and the DM_def_icon_cmd_args can be broadcast to icons using the deformable modeling interface methods, such as DM_draw_all_icons.

Limitations: None

References: DS DM_dbl_array

Data:

None

Constructor:

```
protected: DM_cld_icon::DM_cld_icon (
    const DM_cld_icon&      // icon
    );
```

Copy constructor; call Make_copy and Set_owner instead.

```
public: DM_cld_icon::DM_cld_icon ();
```

Default constructor.

Destructor:

```
protected: virtual DM_cld_icon::~DM_cld_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_cld_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args& // draw-command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_cld_icon::Get_colors (
    DM_dbl_array&           // data array
    ) const;
```

Query color data.

```
public: virtual void DM_cld_icon::Get_grid (
    int[2]                  // density
    ) const;
```

Query discretization grid density.

```
public: virtual DM_icon* DM_cld_icon::Make_copy (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_cld_icon::Set_colors (
    const double*,          // color data (r,g,b)
    int                     // array size
                            // number of doubles
    );
```

Set color data.

```
public: virtual void DM_cld_icon::Set_grid (
    const int[2]            // density
    );
```

Set discretization grid density.

```
public: virtual void DM_cld_icon::Tag_object_changed
(
    Spatial_abs_hurler&     // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_cld_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DM_cpt_icon

Class: Deformable Modeling

Purpose: The DM_cpt_icon draws an array of control points.

Derivation: DM_cpt_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmcpticon.hxx

Description: The DM_cpt_icon draws an array of control points. It is used in the dmicon library as a “helper” icon, owned by other icons. It is used to depict spline surface and curve control points. As an owned icon, it provides an example of icon aggregation.

Related classes: The DM_srf_icon, DM_crv_icon, and the ADM_srf_icon each own DM_cpt_icons.

Limitations: None

References: by DS DM_crv_icon, DM_srf_icon

Data:

None

Constructor:

```
protected: DM_cpt_icon::DM_cpt_icon (
    const DM_cpt_icon&      // icon
    );
```

Copy constructor; call Make_copy and Set_owner instead.

```
public: DM_cpt_icon::DM_cpt_icon ();
```

Default constructor.

Destructor:

```
protected: virtual DM_cpt_icon::~DM_cpt_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_cpt_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args& // draw–command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_cpt_icon::Get_colors (
    DM_dbl_array&            // data array
    ) const;
```

Query color data.

```
public: virtual DM_icon* DM_cpt_icon::Make_copy (
    Spatial_abs_hurler&      // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_cpt_icon::Set_colors (
    const double*,           // color data (r,g,b)
    int                      // array size
                             // number of doubles
    );
```

Set color data.

```
public: virtual void DM_cpt_icon::Tag_object_changed
(
    Spatial_abs_hurler&      // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_cpt_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DM_crv_icon

Class: Deformable Modeling

Purpose: This class is used for drawing deformable curve tag objects in deformable modeling.

Derivation: DM_crv_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmcrvicon.hxx

Description: The DM_crv_icon is part of the dmicon library. This icon updates a polylines to depict the current state of a deformable curve. Control points, element boundaries, and a curvature comb can also be displayed.

Related classes: The DM_def_icon_cmd_args and the DM_def_icon_query_args encapsulate many of the methods inherited from the DM_default_icon interface. Thus the Set_state and Query methods can be used to perform casting, when different libraries of icons are mixed. The DM_def_icon_draw_args and the DM_def_icon_cmd_args can be broadcast to icons using the deformable modeling interface methods, such as DM_draw_all_icons.

Limitations: None

References: DS DM_cpt_icon, DM_dbl_array

Data:

None

Constructor:

```
protected: DM_crv_icon::DM_crv_icon (
    const DM_crv_icon&      // icon
    );
```

Copy constructor; call Make_copy and Set_owner instead.

```
public: DM_crv_icon::DM_crv_icon ();
```

Default constructor.

Destructor:

```
protected: virtual DM_crv_icon::~DM_crv_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_crv_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args&// draw-command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_crv_icon::Get_colors (
    DM_dbl_array&            // data array
    ) const;
```

Query color data.

```
public: double DM_crv_icon::Get_comb_gain () const;
```

Return the comb gain.

```
public: virtual void DM_crv_icon::Get_grid (
    int[2]                   // density
    ) const;
```

Query discretization grid density.

```
public: virtual void DM_crv_icon::Invalidate ();
```

Update cached data.

```
public: virtual DM_icon* DM_crv_icon::Make_copy (
    Spatial_abs_hurler&      // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_crv_icon::Set_colors (
    const double*,           // color data (r,g,b)
    int                      // array size
                             // number of doubles
    );
```

Set color data.

```
public: void DM_crv_icon::Set_comb_gain (
    double                  // gain
    );
```

Set the comb gain.

```
public: virtual void DM_crv_icon::Set_draw_option (
    int                     // option
    );
```

Set draw options.

```
public: virtual void DM_crv_icon::Set_grid (
    const int[2]            // density
    );
```

Set discretization grid density.

```
public: virtual void DM_crv_icon::Set_icon_width (
    double                  // width
    );
```

Set icon draw width.

```
public: virtual void DM_crv_icon::Set_owner (
    Spatial_abs_hurler&,    // error handler
    DS_dmod* new_dmod,      // icon owner ID
    int new_tag             // icon owner ID
    );
```

Notification of owner completion, for icon initialization.

```
public: virtual void
    DM_crv_icon::Tag_object_changed (
    Spatial_abs_hurler&     // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_crv_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DM_dbl_array

Class: Deformable Modeling

Purpose: This class is a pseudo–read–only container class for double type supporting deformable modeling geometry query methods.

Derivation: DM_dbl_array : –

SAT Identifier: None

Filename: ds/dshusk/dskernel/dm_dbl_array.hxx

Description: Some deformable modeling geometry queries return a DM_dbl_array populated with doubles. Thus, clients needn't allocate or destroy heap memory. Queries to curve–based tag object loads and constraints are supported. The DM_dbl_array can also be used as an expandable array, using the DM_set_array_size method and the DM_dbl_array::Set_elem method.

Related methods: A DM_dbl_array is returned by the DM_eval_crv_src, DM_eval_crv_tgt, ATTRIB_DM2ACIS::Eval_crv_src, and ATTRIB_DM2ACIS::Eval_crv_tgt methods. Cached data (array elements) can be reset with Set_elem methods, and can be resized with DM_set_array_size.

Related classes: DM_dbl_array access or set methods taking a Spatial_error_hurler will check array boundaries.

Limitations: None

References: by DS DM_cct_icon, DM_cld_icon, DM_crv_icon, DM_dbx_icon, DM_grd_icon, DM_lct_icon, DM_lld_icon, DM_sps_icon

Data:

None

Constructor:

```
public: DM_dbl_array::DM_dbl_array ();
```

Default constructor.

```
public: DM_dbl_array::DM_dbl_array (
    int nelems              // size
    );
```

Public constructor with non–zero size.

Destructor:

```
public: DM_dbl_array::~DM_dbl_array ();
```

Default destructor.

```
public: operator const
    DM_dbl_array::double* () const;
```

Cast to read–only double*.

Methods:

```
public: double DM_dbl_array::Get_elem (
    Spatial_abs_hurler&,    // error handler
    int                     // index
    ) const;
```

Access array element with bounds checking.

```
public: double DM_dbl_array::operator[] (
    int                     // index
    ) const;
```

Access array element with no bounds checking.

```
public: void DM_dbl_array::Set_elem (
    int,                    // start index
    const double*,          // values
    int                     // number of elements
    );
```

Sets a number of consecutive array elements to the specified values with no bounds checking.

```
public: void DM_dbl_array::Set_elem (
    int,                    // index
    double                  // value
    );
```

Sets one array element to the specified value with no bounds checking.

```
public: void DM_dbl_array::Set_elem (
    Spatial_abs_hurler&,     // error handler
    int,                     // start index
    const double*,           // values
    int                      // number of elements
    );
```

Sets a number of consecutive array elements to the specified values with no bounds checking.

```
public: void DM_dbl_array::Set_elem (
    Spatial_abs_hurler&,     // error handler
    int,                     // index
    double                   // value
    );
```

Sets one array element to the specified value with bounds checking.

```
public: int DM_dbl_array::Size () const;
```

Array size == number of elements.

Related Fncs:

None

DM_dbx_icon

Class: Deformable Modeling

Purpose: The DM_dbx_icon draws and manages an outline corresponding to the image of a parameter sub–rectangle on a deformable surface.

Derivation: DM_dbx_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmdboxicon.hxx

Description: The DM_dbx_icon is used in the dmicon library as a “helper” icon, to depict the image of a parameter sub–rectangle on a deformable surface. As an owned icon, it provides an example of icon aggregation.

Related classes: The DM_dpr_icon and the DM_act_icon both own DM_dbx_icons.

Limitations: None

References:

DS	DM_dbl_array
by DS	DM_act_icon, DM_dpr_icon

Data:

None

Constructor:

```
protected: DM_dbx_icon::DM_dbx_icon (
    const DM_dbx_icon&      // icon
    );
```

Copy constructor; call Make_copy and Set_owner instead.

```
public: DM_dbx_icon::DM_dbx_icon ();
```

Default constructor.

Destructor:

```
protected: virtual DM_dbx_icon::~DM_dbx_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_dbx_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args& // draw-command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_dbx_icon::Get_colors (
    DM_dbl_array&            // data array
    ) const;
```

Query color data.

```
public: virtual void DM_dbx_icon::Get_grid (
    int[2]                   // density
    ) const;
```

Query discretization grid density.

```
public: virtual DM_icon* DM_dbx_icon::Make_copy (
    Spatial_abs_hurler&      // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_dbx_icon::Set_colors (
    const double*,           // color data (r,g,b)
    int                      // array size
                             // number of doubles
    );
```

Set color data.

```
public: virtual void DM_dbx_icon::Set_grid (
    const int[2]             // density
    );
```

Set discretization grid density.

```
public: void DM_dbx_icon::Set_max_min (
    const double*,           // maximum
    const double*            // minimum
    );
```

Set parameter space maximum and minimum.

```
public: virtual void DM_dbx_icon::Tag_object_changed
(
    Spatial_abs_hurler&      // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_dbx_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DM_default_icon

Class: Deformable Modeling

Purpose: Base class for the icon objects in the dmicon and admicon libraries.

Derivation: DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dm_default_icon.hxx

Description: This class provides an interface for common queries and commands. As an optimization, the default icons cache the geometry data needed for drawing. A Tag_object_changed notification sets a validation flag, marking the cache as invalid. When a draw request is received, the validation flag is checked, and if marked invalid, the geometry cache is refreshed before drawing (via tag object queries).

Because of data caching, clients must request redraws after executing certain commands which change geometry. These are the add tag object commands, set tag object behavior commands (e.g., toggle a constraint), solve, and the change–domain commands: split domain, extrapolate, and change degree.

Limitations: None

References: DS DS_dmod

Data:

```
protected DS_dmod* m_dmod;
```
User must initialize.

```
protected double m_icon_width;
```
Icon width.

```
protected int m_draw_option;
```
Draw seams, constraints, or loads.

```
protected int m_on;
```
On or off state.

```
protected int m_tag;
```
User must initialize.

```
protected int m_valid;
```
Force rebuilding before drawing to ensure a valid state.

Constructor:

```
protected: DM_default_icon::DM_default_icon ();
```

Default constructor.

```
protected: DM_default_icon::DM_default_icon (
    const DM_default_icon&  // icon
    );
```

Copy constructor.

Destructor:

```
public: virtual void DM_default_icon::Lose ();
```

Lose method for self–destruction.

```
protected: virtual DM_default_icon::~DM_default_icon
();
```

Destructor; use the Lose method.

Methods:

```
public: virtual void DM_default_icon::Get_colors (
    DM_dbl_array&           // data array
    ) const;
```

Query color data.

```
public: virtual int DM_default_icon::Get_draw_option
() const;
```

Returns the current draw option.

```
public: virtual void DM_default_icon::Get_grid (
    int[2]                  // density
    ) const;
```

Returns 0 by default. Query discretization grid density.

```
public: virtual double
DM_default_icon::Get_icon_width () const;
```

Returns the icon width.

```
public: virtual int DM_default_icon::Get_on_off ()
const;
```

Returns the current on or off state.

```
public: virtual void DM_default_icon::Get_owner (
    DS_dmod*& dmod,          // icon owner ID
    int& tag                 // icon owner ID
    ) const;
```

Icon owner query.

```
public: virtual void DM_default_icon::Invalidate ();
```

Update cached data.

```
public: virtual int DM_default_icon::Is_valid ()
const;
```

Returns TRUE if the cache is valid.

```
public: virtual void DM_default_icon::Query (
    Spatial_abs_hurler&,     // error handler
    DM_icon_query_args&      // query object
    ) const;
```

Query and return the result in args.

```
public: virtual void DM_default_icon::Set_colors (
    const double*,           // color data (r,g,b)
    int                      // array size
                             // number of doubles
    );
```

Set color data.

```
public: virtual void
    DM_default_icon::Set_draw_option (
    int                      // option
    );
```

Set the option to draw seams, constraints, or loads.

```
public: virtual void DM_default_icon::Set_grid (
    const int[2]             // density
    );
```

Set discretization grid density. Does nothing by default.

```
public: virtual void DM_default_icon::Set_icon_width
(
    double                  // width
    );
```

Sets the icon draw width.

```
public: virtual void DM_default_icon::Set_on_off (
    int                     // new state
    );
```

Set the on or off state.

```
public: virtual void DM_default_icon::Set_owner (
    Spatial_abs_hurler&,    // error handler
    DS_dmod* new_dmod,      // icon owner ID
    int new_tag             // icon owner ID
    );
```

Notification of owner completion, for icon initialization.

```
public: virtual void DM_default_icon::Set_state (
    Spatial_abs_hurler&,    // error handler
    const DM_icon_cmd_args& // command object
    );
```

Execute the command.

```
public: virtual void DM_default_icon::Validate ()=0;
```

Update cached data.

Related Fncs:

None

DM_default_icon_factory

Class: Deformable Modeling

Purpose: This is an example of the icon class factory implementation.

Derivation: DM_default_icon_factory : DM_icon_factory : –

SAT Identifier: None

Filename: ds/dmicon/dm_default_icon_factory.hxx

Description: To replace icons, a user can replace the cxx files in the DM_icon library, or add new ones and replace DM_default_icon_factory.cxx.

Limitations: None

References: DS DM_icon

Data:

None

Constructor:

```
public:
DM_default_icon_factory::DM_default_icon_factory ();
```

Default constructor.

Destructor:

```
public: virtual void DM_default_icon_factory::Lose
();
```

Lose method for self–destruction.

```
protected: virtual
DM_default_icon_factory::~DM_default_icon_factory ();
```

Destructor; use Lose method.

Methods:

```
public: void DM_default_icon_factory::Initialize (
    Spatial_abs_hurler&     // error handler
    );
```

Set up a DM_default_icon_factory with the default values.

```
public: virtual DM_icon*
    DM_default_icon_factory::Make_area_cstrn_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make an area constraint icon.

```
public: virtual DM_icon*
    DM_default_icon_factory::Make_area_load_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make an area load icon.

```
public: virtual DM_icon*
    DM_default_icon_factory::Make_attractor_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make an attractor icon.

```
public: virtual DM_icon_factory*
    DM_default_icon_factory::Make_copy (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Deep copy this factory.

```
public: virtual DM_icon*
DM_default_icon_factory::Make_crv_cstrn_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a curve constraint icon.

```
public: virtual DM_icon*
    DM_default_icon_factory::Make_crv_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a curve icon.

```
public: virtual DM_icon*
    DM_default_icon_factory::Make_crv_load_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a curve load icon.

```
public: virtual DM_icon*
    DM_default_icon_factory::Make_dist_press_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a distributed pressure icon.

```
public: virtual DM_icon*
    DM_default_icon_factory::Make_link_cstrn_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a link constraint icon.

```
public: virtual DM_icon*
    DM_default_icon_factory::Make_link_load_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a link load icon.

```
public: virtual DM_icon*
    DM_default_icon_factory::Make_point_press_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a point pressure icon.

```
public: virtual DM_icon*
    DM_default_icon_factory::Make_pt_cstrn_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a point constraint icon.

```
public: virtual DM_icon*
    DM_default_icon_factory::Make_spring_load_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a spring load icon.

```
public: virtual DM_icon*
    DM_default_icon_factory::Make_spring_set_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a spring set icon.

```
public: virtual DM_icon*
    DM_default_icon_factory::Make_srf_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a surface icon.

```
public: virtual DM_icon*
    DM_default_icon_factory::Make_vector_load_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a vector load icon.

```
public: virtual void
DM_default_icon_factory::Replace_area_cstrn_icon (
    Spatial_abs_hurler&,    // error handler
    DM_icon*&               // icon to change
    );
```

Replace methods change what the factory makes.

```
public: virtual void
DM_default_icon_factory::Replace_area_load_icon (
    Spatial_abs_hurler&,    // error handler
    DM_icon*&               // icon to change
    );
```

Replace methods change what the factory makes.

```
public: virtual void
DM_default_icon_factory::Replace_attractor_icon (
    Spatial_abs_hurler&,    // error handler
    DM_icon*&               // icon to change
    );
```

Replace methods change what the factory makes.

```
public: virtual void
DM_default_icon_factory::Replace_crv_cstrn_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon to change
    );
```

Replace methods change what the factory makes.

```
public: virtual void
DM_default_icon_factory::Replace_crv_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon to change
    );
```

Replace methods change what the factory makes.

```
public: virtual void
DM_default_icon_factory::Replace_crv_load_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon to change
    );
```

Replace methods change what the factory makes.

```
public: virtual void
DM_default_icon_factory::Replace_dist_press_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon to change
    );
```

Replace methods change what the factory makes.

```
public: virtual void
DM_default_icon_factory::Replace_link_cstrn_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon to change
    );
```

Replace methods change what the factory makes.

```
public: virtual void
DM_default_icon_factory::Replace_link_load_icon  (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon to change
    );
```

Replace methods change what the factory makes.

```
public: virtual void
DM_default_icon_factory::Replace_point_press_icon  (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon to change
);
```

Replace methods change what the factory makes.

```
public: virtual void
DM_default_icon_factory::Replace_pt_cstrn_icon  (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon to change
    );
```

Replace methods change what the factory makes.

```
public: virtual void
DM_default_icon_factory::Replace_spring_load_icon  (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon to change
    );
```

Replace methods change what the factory makes.

```
public: virtual void
DM_default_icon_factory::Replace_spring_set_icon  (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon to change
    );
```

Replace methods change what the factory makes.

```
public: virtual void
DM_default_icon_factory::Replace_srf_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon to change
    );
```

Replace methods change what the factory makes.

```
public: virtual void
DM_default_icon_factory::Replace_vector_load_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon to change
    );
```

Replace methods change what the factory makes.

Related Fncs:

None

DM_def_icon_cmd_args

Class: Deformable Modeling

Purpose: This class provides a common interface (an Execute method) to the DM_default_icons for encapsulating state change commands.

Derivation: DM_def_icon_cmd_args : DM_icon_cmd_args : –

SAT Identifier: None

Filename: ds/dmicon/dm_def_icon_args.hxx

Description: Concrete DM_def_icon_cmd_args encapsulate a state change command by holding “input” data and implementing the Execute method. The Execute method is called by the DM_default_icon::Set_state method.

Limitations: None

References: None

Data:

None

Constructor:

None

Destructor:

```
public: virtual
    DM_def_icon_cmd_args::~DM_def_icon_cmd_args ();
```

Default destructor.

Methods:

```
public: virtual void DM_def_icon_cmd_args::Execute (
    DM_default_icon*         // icon
    ) const =0;
```

Concrete args must implement override base class cast method.

```
public: virtual const DM_def_icon_cmd_args*
    DM_def_icon_cmd_args::Spatial_cast () const;
```

Reserved for dmicon library.

Related Fncs:

None

DM_def_icon_draw_args

Class: Deformable Modeling

Purpose: This class provides a common interface between ACIS rendering libraries.

Derivation: DM_def_icon_draw_args : DM_icon_draw_args : –

SAT Identifier: None

Filename: ds/dmicon/dm_def_icon_args.hxx

Description: The interface consists of casting methods allowing a concrete DM_draw_engine to decide whether to process or ignore a draw request. Any draw engine library interfacing to the dmicon library must override DM_def_icon_draw_args.

Concrete DM_def_icon_draw_args must provide drawing data when received by the DM_draw_engine::Draw method (for example, which view to draw into). Concrete DM_def_icon_draw_args provide draw data which passes through the DM_icon::Draw method to a concrete DM_draw_engine.

Related classes: The ADM_giicon_draw_args in the admgi_control library are an example of a concrete DM_def_icon_draw_args. The DM_icon_draw_args are the base class for the DM_def_icon_draw_args. Each level of derivation of the DM_icon_draw_args introduces another level of casting methods, for RTTI.

Limitations: None

References: None

Data:

None

Constructor:

None

Destructor:

```
public: virtual
DM_def_icon_draw_args::~DM_def_icon_draw_args ();
```

Default destructor.

Methods:

```
public: virtual const ADM_giicon_draw_args*
    DM_def_icon_draw_args::GI_cast () const;
```

Args for GI draw engine override this.

```
public: virtual const DM_def_icon_draw_args*
    DM_def_icon_draw_args::Spatial_cast () const;
```

Reserved for dmicon library.

```
public: virtual const VM_icon_draw_args*
    DM_def_icon_draw_args::VM_cast () const;
```

Args for VM draw engine override this.

Related Fncs:

None

DM_def_icon_query_args

Class: Deformable Modeling

Purpose: This class provides a common interface to the DM_default_icons for encapsulating query commands.

Derivation: DM_def_icon_query_args : DM_icon_query_args : –

SAT Identifier: None

Filename: ds/dmicon/dm_def_icon_args.hxx

Description: Concrete DM_def_icon_query_args encapsulate a state change command by holding “input” and “output” data and implementing the Execute method. The Execute method is called by the DM_default_icon::Query method.

Limitations: None

References: None

Data:

None

Constructor:

None

Destructor:

```
public: virtual
DM_def_icon_query_args::~DM_def_icon_query_args ();
```

Public destructor.

Methods:

```
public: virtual void DM_def_icon_query_args::Execute
(
    const DM_default_icon*  // icon
    )=0;
```

Concrete args must implement override base class cast method.

```
public: virtual DM_def_icon_query_args*
DM_def_icon_query_args::Spatial_cast ();
```

Reserved for dmicon library.

Related Fncs:

None

DM_dpr_icon

Class: Deformable Modeling

Purpose: This class is used for drawing distributed pressure tag objects in deformable modeling.

Derivation: DM_dpr_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmdpricon.hxx

Description: The DM_dpr_icon is part of the dmicon library. This icon uses a DM_dbx_icon to depict a distributed pressure on a deformable surface.

Related classes: The DM_def_icon_cmd_args and the DM_def_icon_query_args encapsulate many of the methods inherited from the DM_default_icon interface. Thus the Set_state and Query methods can be used to perform casting, when different libraries of icons are mixed. The DM_def_icon_draw_args and the DM_def_icon_cmd_args can be broadcast to icons using the deformable modeling interface methods, such as DM_draw_all_icons. A DM_act_icon owns a DM_dbx_icon, providing an example of icon aggregation.

Limitations: None

References: DS DM_dbx_icon

Data:

None

Constructor:

```
protected: DM_dpr_icon::DM_dpr_icon (
    const DM_dpr_icon&      // icon
    );
```

Copy constructor; call Make_copy and Set_owner instead.

```
public: DM_dpr_icon::DM_dpr_icon ();
```

Public constructor.

Destructor:

```
protected: virtual DM_dpr_icon::~DM_dpr_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_dpr_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args& // draw-command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_dpr_icon::Get_colors (
    DM_dbl_array&           // data array
    ) const;
```

Query color data.

```
public: virtual void DM_dpr_icon::Get_grid (
    int[2]                  // density
    ) const;
```

Query discretization grid density.

```
public: virtual void DM_dpr_icon::Invalidate ();
```

Update cached data.

```
public: virtual DM_icon* DM_dpr_icon::Make_copy (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_dpr_icon::Set_colors (
    const double*,          // color data (r,g,b)
    int                     // array size
                            // number of doubles
    );
```

Set color data.

```
public: virtual void DM_dpr_icon::Set_draw_option (
    int                     // option
    );
```

Set draw options.

```
public: virtual void DM_dpr_icon::Set_grid (
    const int[2]            // density
    );
```

Set discretization grid density.

```
public: virtual void DM_dpr_icon::Set_icon_width (
    double                  // width
    );
```

Set icon draw width.

```
public: virtual void DM_dpr_icon::Set_owner (
    Spatial_abs_hurler&,    // error handler
    DS_dmod* new_dmod,      // icon owner ID
    int new_tag             // icon owner ID
    );
```

Notification of owner completion, for icon initialization.

```
public: virtual void DM_dpr_icon::Tag_object_changed
(
    Spatial_abs_hurler&     // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_dpr_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DM_draw_engine

Class: Deformable Modeling

Purpose: This class provides an abstract draw primitive interface between the deformable modeling icons and the view–controller.

Derivation: DM_draw_engine : –

SAT Identifier: None

Filename: ds/dmicon/dm_draw_engine.hxx

Description: The DM_draw_engine interface classes provide abstract draw primitive methods to icons Draw_point and Draw_polyline. This decouples the icon library from rendering–context specifics, for example, OpenGL or DirectX commands.

Related classes: The draw engine uses the data encapsulated in a DM_draw_args object for drawing. The DM_draw_args object is passed through from the original view manager draw request. Icons only obtain concrete DM_draw_engine objects from the (global) DM_draw_engine_mgr.

Limitations: None

References: None

Data:

None

Constructor:

```
protected: DM_draw_engine::DM_draw_engine ();
```

Default constructor.

Destructor:

```
public: virtual void DM_draw_engine::Lose ()=0;
```

Lose method for self–destruction.

```
protected: virtual DM_draw_engine::~DM_draw_engine
();
```

Destructor; use the Release_this method.

Methods:

```
public: virtual void*
    DM_draw_engine::Customer_cast ();
```

Casting operator; used for overriding.

```
public: virtual void DM_draw_engine::Draw_marker (
    Spatial_abs_hurler&,     // error handler
    DS_dmod*,                // icon owner ID
    int tag,                 // icon owner ID
    const double*,           // marker location (xyz)
    double,                  // color (r,g,b)
    double,                  // color (r,g,b)
    double,                  // color (r,g,b)
    double,                  // line width
    const DM_def_icon_draw_args&// draw args
    ) const =0;
```

Add a marker.

```
public: virtual void DM_draw_engine::Draw_polyline (
    Spatial_abs_hurler&,     // error handler
    DS_dmod*,                // icon owner ID
    int tag,                 // icon owner ID
    int,                     // number of points
    const double*,           // array of points
    double,                  // color (r,g,b)
    double,                  // color (r,g,b)
    double,                  // color (r,g,b)
    double,                  // line width
    const DM_def_icon_draw_args&// draw args
    ) const =0;
```

Add a polyline. The color values must be between 0 and 1.

```
public: virtual void DM_draw_engine::Mark_dirty (
    Spatial_abs_hurler&      // error handler
    )=0;
```

Notify the draw engine of changes which may affect cached data.

```
public: virtual ADM_draw_engine*
    DM_draw_engine::Spatial_cast ();
```

Casting operator; reserved for DM_default_icons.

Related Fncs:

None

DM_draw_engine_mgr

Class: Deformable Modeling

Purpose: This class manages the singleton DM_draw_engine, providing global access to the unique instance.

Derivation: DM_draw_engine_mgr : –

SAT Identifier: None

Filename: ds/dmicon/dm_draw_engine_mgr.hxx

Description: The DM_draw_engine_mgr is used by the dmicon library as its access to the singleton DM_draw_engine. If the admgi_draweng library is initialized, it creates a concrete DM_draw_engine and loads it into the DM_draw_engine_mgr using the Replace_draw_engine method. Users replacing the dmicon library should do the same. If no concrete DM_draw_engine is loaded into the DM_draw_engine_mgr, DM_default_icons will not draw.

Related classes: Access to a concrete DM_draw_engine is provided by the DM_draw_engine_mgr::Instance method.

Limitations: None

References: None

Data:

None

Constructor:

```
public: DM_draw_engine_mgr::DM_draw_engine_mgr ();
```

Public constructor. Users should not call this; the icon library creates a single static instance.

Destructor:

```
public: DM_draw_engine_mgr::~DM_draw_engine_mgr ();
```

Destructor; use the Lose method instead.

Methods:

```
public: static DM_draw_engine*
DM_draw_engine_mgr::Instance ();
```

Request a concrete DM_draw_engine object.

```
public: static void
DM_draw_engine_mgr::Replace_draw_engine (
    Spatial_abs_hurler&,     // error handler
    DM_draw_engine*&         // owner
    );
```

Road a draw engine. The owner argument takes ownership of a concrete DM_draw_engine. This is the prototype for future requests.

Related Fncs:

None

DM_grd_icon

Class: Deformable Modeling

Purpose: This class draws and manages a rectangular grid of lines.

Derivation: DM_grd_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmgridicon.hxx

Description: The DM_grd_icon draws a rectangular grid of lines. It is used in the dmicon library as a “helper” icon, to depict spline surface element boundaries, and to depict a surface as an adjustable density grid of polylines. As an owned icon, it provides an example of icon aggregation.

Related classes: The DM_srf_icon and the ADM_srf_icon both own DM_grd_icons.

Limitations: None

References:

DS	DM_dbl_array
by DS	DM_srf_icon

Data:

None

Constructor:

```
protected: DM_grd_icon::DM_grd_icon (
    const DM_grd_icon&       // icon
    );
```

Copy constructor; call Make_copy and Set_owner instead.

```
public: DM_grd_icon::DM_grd_icon ();
```

Public constructor.

Destructor:

```
protected: virtual DM_grd_icon::~DM_grd_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_grd_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args&// draw-command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_grd_icon::Get_colors (
    DM_dbl_array&            // data array
    ) const;
```

Query color data.

```
public: virtual void DM_grd_icon::Get_grid (
    int[2]                   // density
    ) const;
```

Query discretization grid density.

```
public: void DM_grd_icon::Get_npts_per_elem (
    int&,                    // u density
    int&                     // v density
    ) const;
```

Query density for display mode 0.

```
public: virtual DM_icon* DM_grd_icon::Make_copy (
    Spatial_abs_hurler&      // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_grd_icon::Set_colors (
    const double*,          // color data (r,g,b)
    int                     // array size
                            // number of doubles
    );
```

Set color data.

```
public: void DM_grd_icon::Set_display_mode (
    int                     // mode
    );
```

Set display mode:
0==> use auxiliary “element” data
!=0 ==> use grid data

```
public: virtual void DM_grd_icon::Set_grid (
    const int[2]            // density
    );
```

Set discretization grid density.

```
public: void DM_grd_icon::Set_npts_per_elem (
    int,                    // u density
    int                     // v density
    );
```

Set density for display mode 0.

```
public: virtual void DM_grd_icon::Tag_object_changed
(
    Spatial_abs_hurler&     // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_grd_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DM_ica_draw_option

Class: Deformable Modeling

Purpose: This class is a DM_def_icon_cmd_args that encapsulates a command to set a DM_default_icon draw options, e.g., which tag objects to draw.

Derivation: DM_ica_draw_option : DM_def_icon_cmd_args : DM_icon_cmd_args : –

SAT Identifier: None

Filename: ds/dmicon/dm_def_icon_cmd_args.hxx

Description: The DM_ica_draw_option class is a concrete DM_def_icon_cmd_args, providing an example of command encapsulation for the dmicon library.

Related classes: The base class DM_def_icon_cmd_args derive from DM_icon_cmd_args. The DM_icon_cmd_args have a cast interface, allowing icons to decide whether to process or ignore the command; this allows an application to mix icons from different libraries. The class DM_rend_options is used by the DM_default_icons to encapsulate draw options.

Limitations: None

References: None

Data:

None

Constructor:

```
public: DM_ica_draw_option::DM_ica_draw_option (
    int opt                 // option
        = 0
    );
```

Constructor, defaults to null option.

Destructor:

None

Methods:

```
public: virtual void DM_ica_draw_option::Execute (
    DM_default_icon*        // icon
    ) const;
```

Process this icon.

Related Fncs:

None

DM_ica_grid

Class: Deformable Modeling

Purpose: This class is a DM_def_icon_cmd_args that encapsulates a command to set a DM_default_icon draw grid density.

Derivation: DM_ica_grid : DM_def_icon_cmd_args : DM_icon_cmd_args : –

SAT Identifier: None

Filename: ds/dmicon/dm_def_icon_cmd_args.hxx

Description: The DM_ica_grid class is a concrete DM_def_icon_cmd_args, providing an example of command encapsulation for the dmicon library.

Related classes: The base class DM_def_icon_cmd_args derive from DM_icon_cmd_args. The DM_icon_cmd_args have a cast interface, allowing icons to decide whether to process or ignore the command. This allows an application to mix icons from different libraries.

Limitations: None

References: None

Data:

None

Constructor:

```
public: DM_ica_grid::DM_ica_grid (
    int grid1,              // density
    int grid2               // density
    );
```

Constructor for 2D anisotropic.

```
public: DM_ica_grid::DM_ica_grid (
    int grid                // density
        = 5
    );
```

Constructor for 1D or 2D isotropic.

Destructor:

None

Methods:

```
public: virtual void DM_ica_grid::Execute (
    DM_default_icon*        // icon
    ) const;
```

Process this icon.

Related Fncs:

None

DM_ica_on_off

Class: Deformable Modeling

Purpose: This class is a DM_def_icon_cmd_args that encapsulates a command to switch an icon on or off (draw or don't draw).

Derivation: DM_ica_on_off : DM_def_icon_cmd_args : DM_icon_cmd_args : –

SAT Identifier: None

Filename: ds/dmicon/dm_def_icon_cmd_args.hxx

Description: The DM_ica_on_off class is a concrete DM_def_icon_cmd_args, providing an example of command encapsulation for the dmicon library.

Related classes: The base class DM_def_icon_cmd_args derive from DM_icon_cmd_args. The DM_icon_cmd_args have a cast interface, allowing icons to decide whether to process or ignore the command. This allows an application to mix icons from different libraries.

Limitations: None

References: None

Data:

None

Constructor:

```
public: DM_ica_on_off::DM_ica_on_off (
    int on_off              // on or off
        = 0
    );
```

Constructor, defaults to off.

Destructor:

None

Methods:

```
public: virtual void DM_ica_on_off::Execute (
    DM_default_icon*         // icon
    ) const;
```

Process this icon.

Related Fncs:

None

DM_ica_width

Class: Deformable Modeling

Purpose: This class is a DM_def_icon_cmd_args that encapsulates a command to set a DM_default_icon draw width.

Derivation: DM_ica_width : DM_def_icon_cmd_args : DM_icon_cmd_args : –

SAT Identifier: None

Filename: ds/dmicon/dm_def_icon_cmd_args.hxx

Description: The DM_ica_grid class is a concrete DM_def_icon_cmd_args, providing an example of command encapsulation for the dmicon library.

Related classes: The base class DM_def_icon_cmd_args derive from DM_icon_cmd_args. The DM_icon_cmd_args have a cast interface, allowing icons to decide whether to process or ignore the command. This allows an application to mix icons from different libraries.

Limitations: None

References: None

Data:

None

Constructor:

```
public: DM_ica_width::DM_ica_width (
    double width             // width
        = 1.0
    );
```

Public constructor.

Destructor:

None

Methods:

```
public: virtual void DM_ica_width::Execute (
    DM_default_icon*         // icon
    ) const;
```

Process this icon.

Related Fncs:

None

DM_icon

Class: Deformable Modeling

Purpose: This class provides abstract notify methods to the deformable modeling kernel for drawing services.

Derivation: DM_icon : –

SAT Identifier: None

Filename: ds/dshusk/dskernel/dmicon.hxx

Description: The contract with the DM_icon class gives the deformable modeling kernel three responsibilities. First, DM_icon objects are owned by tag objects: they are created by the tag object CTOR and destroyed by the tag object DTOR. This is supported by the DM_icon Make_copy and Lose methods.

Second, the deformable modeling kernel calls the icon Set_owner method when the tag object is fully constructed. This allows the icon to initialize itself, and retain knowledge of its owner. Third, the kernel calls the icon Tagobject_changed method to notify the icon when the tag object state has changed, e.g., geometry or behavior. The icon can then redraw, or set a flag for lazy update.

The deformable modeling kernel also defines a command object interface which allows an application to broadcast commands to all or selected icons in a deformable modeling hierarchy. These are supported by the DM_icon Draw and Set_state methods.

Related classes: Implementations derived from the DM_icon class must support the DM_icon_factory class by overriding the Lose and Make_copy methods. Implementations overriding the Draw method, Set_state method, and the Query method must also override the DM_icon_draw_args, the DM_icon_cmd_args, and the DM_icon_query_args classes respectively.

Limitations: None

References: by DS DM_default_icon_factory, DS_dmod

Data:

None

Constructor:

```
protected: DM_icon::DM_icon ();
```

Default constructor.

Destructor:

```
public: virtual void DM_icon::Lose ()=0;
```

Public destructor.

```
protected: virtual DM_icon::~DM_icon ();
```

Destructor.

Methods:

```
public: virtual void DM_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args& // draw-command object
    ) const=0;
```

Draw this icon.

```
public: virtual void DM_icon::Get_owner (
    DS_dmod*& dmod,          // icon owner ID
    int& tag                 // icon owner ID
    ) const=0;
```

Icon owner query.

```
public: virtual DM_icon* DM_icon::Make_copy (
    Spatial_abs_hurler&      // error handler
    ) const=0;
```

Makes a copy of the icon.

```
public: virtual void DM_icon::Query (
    Spatial_abs_hurler&,     // error handler
    DM_icon_query_args&      // query object
    ) const=0;
```

Query and return the result in the arguments.

```
public: virtual void DM_icon::Set_owner (
    Spatial_abs_hurler&,     // error handler
    DS_dmod* new_dmod,       // icon owner ID
    int new_tag              // icon owner ID
    )=0;
```

Notification of owner completion. Notification of owner completion, for icon initialization.

```
public: virtual void DM_icon::Set_state (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_cmd_args& // command object
    )=0;
```

Execute command.

```
public: virtual void DM_icon::Tag_object_changed (
    Spatial_abs_hurler&      // error handler
    )=0;
```

Notification of owning tag object state change.

Related Fncs:

None

DM_icon_cmd_args

Class: Deformable Modeling

Purpose: This interface class provides a command object to forward client requests through the DM_icon::Set_state method to the DM_icon.

Derivation: DM_icon_cmd_args : –

SAT Identifier: None

Filename: ds/dshusk/dskernel/dm_icon_args.hxx

Description: The DM_icon_cmd_args command object can forward through the deformable modeling interface, providing DS_dmod hierarchy broadcast capabilities. Implementations overriding the DM_icon_cmd_args should override the customer_cast method to return a pointer to their derived object. Clients must then downcast the returned void pointer. Overriding the Spatial_cast method is reserved for the example dmicon library.

Related methods: DM_icon_cmd_args command objects can forward through the DM_setstate_icon and ATTRIB_DM2ACIS::Setstate_icon interface methods to provide DS_dmod hierarchy broadcast capabilities.

Limitations: None

References: None

Data:

None

Constructor:

```
public: DM_icon_cmd_args::DM_icon_cmd_args ();
```

Default constructor.

Destructor:

```
public: virtual DM_icon_cmd_args::~DM_icon_cmd_args
();
```

Default destructor.

Methods:

```
public: virtual const void*
    DM_icon_cmd_args::Customer_cast () const;
```

Implementations should override this method as {return (void*)this;}

```
public: virtual const DM_def_icon_cmd_args*
    DM_icon_cmd_args::Spatial_cast () const;
```

Reserved for the example dmicon library.

Related Fncs:

None

DM_icon_draw_args

Class: Deformable Modeling

Purpose: This interface class provides a command object to forward client requests through the DM_icon::Draw method to the DM_draw_engine.

Derivation: DM_icon_draw_args : –

SAT Identifier: None

Filename: ds/dshusk/dskernel/dm_icon_args.hxx

Description: The DM_icon_cmd_args command object can forward through the Deformable Modeling interface, providing DS_dmod hierarchy broadcast capabilities. A typical example is the particular view or views to draw into: the DM_icon can tell DM_draw_engine what geometry to draw, and the DM_icon_draw_args can be set up to tell the DM_draw_engine what canvas to draw onto.

Implementations overriding the DM_icon_draw_args should override the customer_cast method to return a pointer to their derived object. Clients must then downcast the returned void pointer. Overriding the Spatial_cast method is reserved for the example dmicon library.

Related methods: DM_icon_draw_args command objects can forward through the DM_draw_icon and ATTRIB_DM2ACIS::Draw_icon interface methods, to provide DS_dmod hierarchy broadcast capabilities.

Limitations: None

References: None

Data:

None

Constructor:

```
public: DM_icon_draw_args::DM_icon_draw_args ();
```

Default constructor.

Destructor:

```
public: virtual DM_icon_draw_args::~DM_icon_draw_args
();
```

Default destructor.

Methods:

```
public: virtual const void*
    DM_icon_draw_args::Customer_cast () const;
```

Implementations should override this method as {return (void*)this;}

```
public: virtual const DM_def_icon_draw_args*
    DM_icon_draw_args::Spatial_cast () const;
```

Reserved for the example dmicon library.

Related Fncs:

None

DM_icon_factory

Class: Deformable Modeling

Purpose: The DM_icon_factory class provides an abstract interface to the deformable modeling kernel for creating and destroying DM_icon objects.

Derivation: DM_icon_factory : –

SAT Identifier: None

Filename: ds/dshusk/dskernel/dmicon_factory.hxx

Description: For each type of deformable modeling tag object (DS_dmod, DS_load, or DS_cstrn), the DM_icon_factory has a Replace_XXX_icon and a Make_XXX_icon method. The deformable modeling kernel uses the Make_XXX_icon methods to attach an icon to a tag object, upon tag object creation. The Replace_XXX_icon methods are intended to determine what subsequent calls to the Make_XXX_icon method will return. Thus tag object icons can be changed at runtime.

Related classes: The class DM_icon_factory_mgr provides the deformable modeling kernel with a singleton DM_icon_factory object. Implementations overriding the DM_icon_factory must initialize the DM_icon_factory_mgr. A sample implementation of the DM_icon_factory is the class DM_default_icon_factory in the dmicon library.

Limitations: None

References: None

Data:

None

Constructor:

```
protected: DM_icon_factory::DM_icon_factory ();
```

Default constructor.

Destructor:

```
public: virtual void DM_icon_factory::Lose ()=0;
```

Public destructor.

```
protected: virtual DM_icon_factory::~DM_icon_factory
();
```

Destructor.

Methods:

```
public: virtual DM_icon*
    DM_icon_factory::Make_area_cstrn_icon (
    Spatial_abs_hurler&      // error handler
    ) const;
```

Make a new icon of this type.

```
public: virtual DM_icon*
DM_icon_factory::Make_area_load_icon (
    Spatial_abs_hurler&      // error handler
    ) const;
```

Make an area load icon.

```
public: virtual DM_icon*
DM_icon_factory::Make_attractor_icon (
    Spatial_abs_hurler&      // error handler
    ) const;
```

Make a new icon of this type.

```
public: virtual DM_icon_factory*
    DM_icon_factory::Make_copy (
    Spatial_abs_hurler&      // error handler
    ) const=0;
```

Make a deep copy of this factory.

```
public: virtual DM_icon*
    DM_icon_factory::Make_crv_cstrn_icon (
    Spatial_abs_hurler&      //error handler
    ) const;
```

Make a new icon of this type.

```
public: virtual DM_icon*
    DM_icon_factory::Make_crv_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a new icon of this type.

```
public: virtual DM_icon*
    DM_icon_factory::Make_crv_load_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a new icon of this type.

```
public: virtual DM_icon*
    DM_icon_factory::Make_dist_press_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a new icon of this type.

```
public: virtual DM_icon*
    DM_icon_factory::Make_link_cstrn_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a new icon of this type.

```
public: virtual DM_icon*
    DM_icon_factory::Make_link_load_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a new icon of this type.

```
public: virtual DM_icon*
    DM_icon_factory::Make_point_press_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a new icon of this type.

```
public: virtual DM_icon*
    DM_icon_factory::Make_pt_cstrn_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a new icon of this type.

```
public: virtual DM_icon*
    DM_icon_factory::Make_spring_load_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a new icon of this type.

```
public: virtual DM_icon*
    DM_icon_factory::Make_spring_set_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a new icon of this type.

```
public: virtual DM_icon*
    DM_icon_factory::Make_srf_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a new icon of this type.

```
public: virtual DM_icon*
    DM_icon_factory::Make_vector_load_icon (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Make a new icon of this type.

```
public: virtual void
    DM_icon_factory::Replace_area_cstrn_icon (
    Spatial_abs_hurler&,    // error handler
    DM_icon*&               // icon
    );
```

Replace methods change what the factory makes.

```
public: virtual void
    DM_icon_factory::Replace_area_load_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon
    );
```

Replace methods change what the factory makes.

```
public: virtual void
    DM_icon_factory::Replace_attractor_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon
    );
```

Replace methods change what the factory makes.

```
public: virtual void
    DM_icon_factory::Replace_crv_cstrn_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon
    );
```

Replace methods change what the factory makes.

```
public: virtual void
    DM_icon_factory::Replace_crv_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon
    );
```

Replace methods change what the factory makes.

```
public: virtual void
    DM_icon_factory::Replace_crv_load_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon
    );
```

Replace methods change what the factory makes.

```
public: virtual void
    DM_icon_factory::Replace_dist_press_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon
    );
```

Replace methods change what the factory makes.

```
public: virtual void
    DM_icon_factory::Replace_link_cstrn_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon
    );
```

Replace methods change what the factory makes.

```
public: virtual void
    DM_icon_factory::Replace_link_load_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon
    );
```

Replace methods change what the factory makes.

```
public: virtual void
    DM_icon_factory::Replace_point_press_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon
    );
```

Replace methods change what the factory makes.

```
public: virtual void
    DM_icon_factory::Replace_pt_cstrn_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon
    );
```

Replace methods change what the factory makes.

```
public: virtual void
    DM_icon_factory::Replace_spring_load_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon
    );
```

Replace methods change what the factory makes.

```
public: virtual void
    DM_icon_factory::Replace_spring_set_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon
    );
```

Replace methods change what the factory makes.

```
public: virtual void
    DM_icon_factory::Replace_srf_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon
    );
```

Replace methods change what the factory makes.

```
public: virtual void
    DM_icon_factory::Replace_vector_load_icon (
    Spatial_abs_hurler&,     // error handler
    DM_icon*&                // icon
    );
```

Replace methods change what the factory makes. a

Related Fncs:

None

DM_icon_factory_mgr

Class: Deformable Modeling

Purpose: This class manages the singleton DM_icon_factory, providing global access to the unique instance.

Derivation: DM_icon_factory_mgr : –

SAT Identifier: None

Filename: ds/dshusk/dskernel/dmicon_factory_mgr.hxx

Description: The DM_icon_factory_mgr is used by the deformable modeling kernel as its access to the singleton DM_icon_factory. If the dmicon library is initialized, it creates a DM_default_icon_factory and loads it into the DM_icon_factory_mgr using the Replace_factory method. Users replacing the dmicon library should do the same. If no concrete DM_icon_factory is loaded into the DM_icon_factory_mgr, no icons will be created by the deformable modeling kernel.

Related classes: Access to a concrete DM_icon_factory is provided by the DM_icon_factory_mgr::Instance method.

Limitations: None

References: None

Data:

```
protected static DM_icon_factory* m_factory;
```

A static instance to clean up the DM_icon_factory on program termination.

Constructor:

```
public: DM_icon_factory_mgr::DM_icon_factory_mgr ();
```

Public constructor.

Destructor:

```
public: DM_icon_factory_mgr::~DM_icon_factory_mgr ();
```

Destructor; should not be called.

Methods:

```
public: static DM_icon_factory*
    DM_icon_factory_mgr::Instance ();
```

Provide global access to the singleton icon factory.

```
public: static void
    DM_icon_factory_mgr::Replace_factory (
    Spatial_abs_hurler&,     // error handler
    DM_icon_factory*&        // icon factory
    );
```

Take ownership and null the pointer.

Related Fncs:

None

DM_icon_query_args

Class: Deformable Modeling

Purpose: This interface class provides a common interface between icon libraries.

Derivation: DM_icon_query_args : –

SAT Identifier: None

Filename: ds/dshusk/dskernel/dm_icon_args.hxx

Description: The common interface provided by this class consists of casting methods allowing a concrete DM_icon to decide whether to process or ignore a command. DM_icon_query_args are for querying an icon's state.

Concrete DM_icon_query_args encapsulate commands for querying an icon's state. The DM_icon_query_args query–command object passes through the deformable modeling interface, giving a common look and feel to icon draw, set state, and query commands. Implementations overriding the DM_icon_query_args should override the Customer_cast method to return a pointer to their derived object; clients must then downcast the returned void pointer.

Derived classes can implement cast methods, allowing icons to decide whether to process or ignore the args. Overriding the Spatial_cast method is reserved for the example dmicon library.

Related methods: DM_icon_query_args query–command objects can pass through the DM_query_icon and ATTRIB_DM2ACIS::Query_icon deformable modeling interface methods, giving a common look and feel to icon draw, set state, and query commands.

Limitations: None

References: None

Data:

None

Constructor:

```
public: DM_icon_query_args::DM_icon_query_args ();
```

Public constructor.

Destructor:

```
public: virtual
    DM_icon_query_args::~DM_icon_query_args ();
```

Public destructor.

Methods:

```
public: virtual void*
    DM_icon_query_args::Customer_cast ();
```

Customer implementations should override this method as {return (void*)this;}

```
public: virtual DM_def_icon_query_args*
    DM_icon_query_args::Spatial_cast ();
```

Reserved for the example dmicon library.

Related Fncs:

None

DM_icq_draw_grid

Class: Deformable Modeling

Purpose: This class is a DM_def_icon_query_args that encapsulates a command to query DM_default_icon draw grid density.

Derivation: DM_icq_draw_grid : DM_def_icon_query_args : DM_icon_query_args : –

SAT Identifier: None

Filename: ds/dmicon/dm_def_icon_query_args.hxx

Description: The DM_icq_draw_grid class is a concrete DM_def_icon_query_args, providing an example of command encapsulation for the dmicon library.

Related classes: The base class DM_def_icon_query_args derives from DM_icon_query_args. The DM_icon_cmd_args have a cast interface, allowing icons to decide whether to process or ignore the command; this allows an application to mix icons from different libraries.

Limitations: None

References: None

Data:

None

Constructor:

```
public: DM_icq_draw_grid::DM_icq_draw_grid ();
```

Public constructor.

Destructor:

None

Methods:

```
public: virtual void DM_icq_draw_grid::Execute (
    const DM_default_icon*  // icon
    );
```

Query this icon.

```
public: const int*
    DM_icq_draw_grid::Get_grid () const;
```

Return stored data – used after processing a query.

Related Fncs:

None

DM_icq_draw_option

Class: Deformable Modeling

Purpose: This class is a DM_def_icon_query_args that encapsulates a command to query DM_default_icon draw options, e.g., which tag objects to draw.

Derivation: DM_icq_draw_option : DM_def_icon_query_args : DM_icon_query_args : –

SAT Identifier: None

Filename: ds/dmicon/dm_def_icon_query_args.hxx

Description: The DM_icq_draw_option class is a concrete DM_def_icon_query_args, providing an example of command encapsulation for the dmicon library.

Related classes: The base class DM_def_icon_query_args derives from DM_icon_query_args. The DM_icon_cmd_args have a cast interface, allowing icons to decide whether to process or ignore the command; this allows an application to mix icons from different libraries. The class DM_rend_options is used by the DM_default_icons to encapsulate draw options.

Limitations: None

References: None

Data:

None

Constructor:

```
public: DM_icq_draw_option::DM_icq_draw_option ();
```

Public constructor.

Destructor:

None

Methods:

```
public: virtual void DM_icq_draw_option::Execute (
    const DM_default_icon*  // icon
    );
```

Query this icon.

```
public: int DM_icq_draw_option::Get_option () const;
```

Return stored data; used after processing a query.

Related Fncs:

None

DM_lct_icon

Class: Deformable Modeling

Purpose: This class is used for drawing curve link constraint tag objects in deformable modeling.

Derivation: DM_lct_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmlcticon.hxx

Description: The DM_lct_icon is part of the dmicon library. This icon updates a polyline to depict the current state of a curve link constraint.

Related classes: The DM_def_icon_cmd_args and the DM_def_icon_query_args encapsulate many of the methods inherited from the DM_default_icon interface. Thus the Set_state and Query methods can be used to perform casting, when different libraries of icons are mixed.

The DM_def_icon_draw_args and the DM_def_icon_cmd_args can be broadcast to icons using the deformable modeling interface methods, such as DM_draw_all_icons.

Limitations: None

References: DS DM_dbl_array

Data:

None

Constructor:

```
protected: DM_lct_icon::DM_lct_icon (
    const DM_lct_icon&      // icon
    );
```

Copy constructor; call Make_copy and Set_owner instead.

```
public: DM_lct_icon::DM_lct_icon ();
```

Public constructor.

Destructor:

```
protected: virtual DM_lct_icon::~DM_lct_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_lct_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args&// draw-command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_lct_icon::Get_colors (
    DM_dbl_array&           // data array
    ) const;
```

Query color data.

```
public: virtual void DM_lct_icon::Get_grid (
    int[2]                  // density
    ) const;
```

Query discretization grid density.

```
public: virtual DM_icon* DM_lct_icon::Make_copy (
    Spatial_abs_hurler&      // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_lct_icon::Set_colors (
    const double*,           // color data (r,g,b)
    int                      // array size
                             // number of doubles
    );
```

Set color data.

```
public: virtual void DM_lct_icon::Set_grid (
    const int[2]             // density
    );
```

Set discretization grid density.

```
public: virtual void DM_lct_icon::Tag_object_changed
(
    Spatial_abs_hurler&      // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_lct_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DM_lld_icon

Class: Deformable Modeling

Purpose: This class is used for drawing curve link load tag objects in deformable modeling.

Derivation: DM_lld_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmlldicon.hxx

Description: The DM_lld_icon is part of the dmicon library. This icon updates a polyline to depict the current state of a curve link load.

Related classes: The DM_def_icon_cmd_args and the DM_def_icon_query_args encapsulate many of the methods inherited from the DM_default_icon interface. Thus the Set_state and Query methods can be used to perform casting, when different libraries of icons are mixed.

The DM_def_icon_draw_args and the DM_def_icon_cmd_args can be broadcast to icons using the deformable modeling interface methods, such as DM_draw_all_icons.

Limitations: None

References: DS DM_dbl_array

Data:

None

Constructor:

```
protected: DM_lld_icon::DM_lld_icon (
    const DM_lld_icon&      // icon
    );
```

Copy constructor; call Make_copy and Set_owner instead.

```
public: DM_lld_icon::DM_lld_icon ();
```

Public constructor.

Destructor:

```
protected: virtual DM_lld_icon::~DM_lld_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_lld_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args& // draw-command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_lld_icon::Get_colors (
    DM_dbl_array&           // data array
    ) const;
```

Query color data.

```
public: virtual void DM_lld_icon::Get_grid (
    int[2]                  // density
    ) const;
```

Query discretization grid density.

```
public: virtual DM_icon* DM_lld_icon::Make_copy (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_lld_icon::Set_colors (
    const double*,          // color data (r,g,b)
    int                     // array size
                            // number of doubles
    );
```

Set color data.

```
public: virtual void DM_lld_icon::Set_grid (
    const int[2]            // density
    );
```

Set discretization grid density.

```
public: virtual void DM_lld_icon::Tag_object_changed
(
    Spatial_abs_hurler&     // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_lld_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DM_pct_icon

Class: Deformable Modeling

Purpose: This class is used for drawing point constraint tag objects in deformable modeling.

Derivation: DM_pct_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmpcticon.hxx

Description: The DM_pct_icon is part of the dmicon library. This icon updates markers and polylines to depict the current state of a point constraint.

Related classes: The DM_def_icon_cmd_args and the DM_def_icon_query_args encapsulate many of the methods inherited from the DM_default_icon interface. Thus the Set_state and Query methods can be used to perform casting, when different libraries of icons are mixed.

The DM_def_icon_draw_args and the DM_def_icon_cmd_args can be broadcast to icons using the deformable modeling interface methods, such as DM_draw_all_icons.

Limitations: None

References: None

Data:

None

Constructor:

```
protected: DM_pct_icon::DM_pct_icon (
    const DM_pct_icon&      // icon
    );
```

Copy constructor; call Make_copy and Set_owner instead.

```
public: DM_pct_icon::DM_pct_icon ();
```

Public constructor.

Destructor:

```
protected: virtual DM_pct_icon::~DM_pct_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_pct_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args& // draw-command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_pct_icon::Get_colors (
    DM_dbl_array&            // data array
    ) const;
```

Query color data.

```
public: virtual DM_icon* DM_pct_icon::Make_copy (
    Spatial_abs_hurler&      // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_pct_icon::Set_colors (
    const double*,           // color data (r,g,b)
    int                      // array size
                             // number of doubles
    );
```

Set color data.

```
public: virtual void DM_pct_icon::Set_owner (
    Spatial_abs_hurler&,     // error handler
    DS_dmod* new_dmod,       // icon owner ID
    int new_tag              // icon owner ID
    );
```

Notification of owner completion, for icon initialization.

```
public: virtual void DM_pct_icon::Tag_object_changed
(
    Spatial_abs_hurler&      // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_pct_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DM_ppr_icon

Class: Deformable Modeling

Purpose: This class is used for drawing point pressure tag objects in deformable modeling.

Derivation: DM_ppr_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmppricon.hxx

Description: The DM_ppr_icon is part of the dmicon library. This icon updates a marker to depict the current position of a point pressure.

Related classes: The DM_def_icon_cmd_args and the DM_def_icon_query_args encapsulate many of the methods inherited from the DM_default_icon interface. Thus the Set_state and Query methods can be used to perform casting, when different libraries of icons are mixed.

The DM_def_icon_draw_args and the DM_def_icon_cmd_args can be broadcast to icons using the deformable modeling interface methods, such as DM_draw_all_icons.

Limitations: None

References: None

Data:

None

Constructor:

```
protected: DM_ppr_icon::DM_ppr_icon (
    const DM_ppr_icon&      // icon
    );
```

Copy constructor; call Make_copy and Set_owner instead.

```
public: DM_ppr_icon::DM_ppr_icon ();
```

Public constructor.

Destructor:

```
protected: virtual DM_ppr_icon::~DM_ppr_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_ppr_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args&// draw-command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_ppr_icon::Get_colors (
    DM_dbl_array&           // data array
    ) const;
```

Query color data.

```
public: virtual DM_icon* DM_ppr_icon::Make_copy (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_ppr_icon::Set_colors (
    const double*,          // color data (r,g,b)
    int                     // array size
                            // number of doubles
    );
```

Set color data.

```
public: virtual void DM_ppr_icon::Tag_object_changed
(
    Spatial_abs_hurler&     // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_ppr_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DM_rend_options

Class: Deformable Modeling

Purpose: This class is used internally by the dmicon library for encapsulating draw options.

Derivation: DM_rend_options : –

SAT Identifier: None

Filename: ds/dmicon/dmrend_opt.hxx

Description: The DM_rend_options is part of the dmicon library, and is only used internally by the dmicon library. DM_rend_options are constructed from a bitfield defined by the DM_DRAW_XXX macros defined in the file dmapinum.hxx.

Related classes: DM_ica_draw_option and DM_icq_draw_option encapsulate set and query draw option commands using the DM_rend_options class. These derive from the DM_def_icon_cmd_args and DM_def_icon_query_args base classes.

The DM_def_icon_cmd_args and the DM_def_icon_query_args encapsulate many of the methods inherited from the DM_default_icon interface. Thus the Set_state and Query methods can be used to perform casting, when different libraries of icons are mixed.

The DM_def_icon_draw_args and the DM_def_icon_cmd_args can be broadcast to icons using the deformable modeling interface methods, such as DM_draw_all_icons.

Limitations: None

References: None

Data:

None

Constructor:

```
public: DM_rend_options::DM_rend_options ();
```

Public constructor.

```
public: DM_rend_options::DM_rend_options (
    int bitfield            // bitfield
    );
```

Copy constructor; call Make_copy and Set_owner instead.

Destructor:

None

Methods:

```
public: int DM_rend_options::Draw_cpts () const;
```

Query draw control points option.

```
public: int DM_rend_options::Draw_cstrns () const;
```

Query draw constraints option.

```
public: int
    DM_rend_options::Draw_curve_comb () const;
```

Query draw curvature comb option.

```
public: int DM_rend_options::Draw_elems () const;
```

Query draw element boundary option.

```
public: int DM_rend_options::Draw_loads () const;
```

Query draw loads option.

```
public: int DM_rend_options::Draw_seams () const;
```

Query draw seams option.

```
public: void DM_rend_options::Set_cpts_off ();
```

Set draw control points option off.

```
public: void DM_rend_options::Set_cpts_on ();
```

Set draw control points option on.

```
public: void DM_rend_options::Set_cstrns_off ();
```

Set draw constraints option off.

```
public: void DM_rend_options::Set_cstrns_on ();
```

Set draw constraints option on.

```
public: void DM_rend_options::Set_curve_comb_off ();
```

Set draw curvature comb option off.

```
public: void DM_rend_options::Set_curve_comb_on ();
```

Set draw curvature comb option on.

```
public: void DM_rend_options::Set_elems_off ();
```

Set draw element boundary option off.

```
public: void DM_rend_options::Set_elems_on ();
```

Set draw element boundary option on.

```
public: void DM_rend_options::Set_loads_off ();
```

Set draw loads option off.

```
public: void DM_rend_options::Set_loads_on ();
```

Set draw loads option on.

```
public: void DM_rend_options::Set_seams_off ();
```

Set draw seams option off.

```
public: void DM_rend_options::Set_seams_on ();
```

Set draw seams option on.

Related Fncs:

None

DM_rtnerr_hurler

Class: Deformable Modeling

Purpose: This class provides a protocol for handling exceptions across interfaces.

Derivation: DM_rtnerr_hurler : Spatial_abs_hurler : –

SAT Identifier: None

Filename: ds/dshusk/dskernel/dm_rtnerr_hurler.hxx

Description: Interface routines taking a Spatial_abs_hurler agree to trap all exceptions, translate them to an integer code, and then call the Spatial_abs_hurler rethrow_error method with the integer code. The DM_rtnerr_hurler::rethrow_error method simply stores the error code, which can later be queried; the int cast operator allows treating DM_rtnerr_hurler objects as ints.

Limitations: None

References: None

Data:

None

Constructor:

```
public: DM_rtnerr_hurler::DM_rtnerr_hurler ();
```

Public constructor.

Destructor:

```
public: DM_rtnerr_hurler::~DM_rtnerr_hurler ();
```

Public destructor.

Methods:

```
public: operator DM_rtnerr_hurler::int& ();
```

int cast; this method can treat a DM_rtnerr_hurler object as an int.

```
public: virtual void
    DM_rtnerr_hurler::rethrow_error (
    int                     // code
    );
```

Encapsulate the error handling method.

Related Fncs:

None

DM_spr_icon

Class: Deformable Modeling

Purpose: This class is used for drawing point spring load tag objects in deformable modeling.

Derivation: DM_spr_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmspricon.hxx

Description: The DM_spr_icon is part of the dmicon library. This icon updates markers and polylines to depict the current position of a point spring load and its target.

Related classes: DM_ica_draw_option and DM_icq_draw_option encapsulate set and query draw option commands using the DM_rend_options class. These derive from the DM_def_icon_cmd_args and DM_def_icon_query_args base classes.

The DM_def_icon_cmd_args and the DM_def_icon_query_args encapsulate many of the methods inherited from the DM_default_icon interface. Thus the Set_state and Query methods can be used to perform casting, when different libraries of icons are mixed.

Limitations: None

References: None

Data:

None

Constructor:

```
protected: DM_spr_icon::DM_spr_icon (
    const DM_spr_icon&      // icon
    );
```

Copy constructor; call Make_copy and Set_owner instead.

```
public: DM_spr_icon::DM_spr_icon ();
```

Public constructor.

Destructor:

```
protected: virtual DM_spr_icon::~DM_spr_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_spr_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args&// draw-command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_spr_icon::Get_colors (
    DM_dbl_array&           // data array
    ) const;
```

Query color data.

```
public: virtual DM_icon* DM_spr_icon::Make_copy (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_spr_icon::Set_colors (
    const double*,          // color data (r,g,b)
    int                     // array size
                            // number of doubles
    );
```

Set color data.

```
public: virtual void DM_spr_icon::Tag_object_changed
(
    Spatial_abs_hurler&     // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_spr_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DM_sps_icon

Class: Deformable Modeling

Purpose: This class is used for drawing point spring set load tag objects in deformable modeling.

Derivation: DM_sps_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmspsicon.hxx

Description: The DM_sps_icon is part of the dmicon library. This icon updates markers and polylines to depict the current positions of a point spring set load and its targets.

Related classes: DM_ica_draw_option and DM_icq_draw_option encapsulate set and query draw option commands using the DM_rend_options class. These derive from the DM_def_icon_cmd_args and DM_def_icon_query_args base classes.

The DM_def_icon_cmd_args and the DM_def_icon_query_args encapsulate many of the methods inherited from the DM_default_icon interface. Thus the Set_state and Query methods can be used to perform casting, when different libraries of icons are mixed.

Limitations: None

References: DS DM_dbl_array

Data:

None

Constructor:

```
protected: DM_sps_icon::DM_sps_icon (
    const DM_sps_icon&      // icon
    );
```

Copy constructor; call Make_copy and Set_owner instead.

```
public: DM_sps_icon::DM_sps_icon ();
```

Public constructor.

Destructor:

```
protected: virtual DM_sps_icon::~DM_sps_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_sps_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args&// draw-command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_sps_icon::Get_colors (
    DM_dbl_array&           // data array
    ) const;
```

Query color data.

```
public: virtual DM_icon* DM_sps_icon::Make_copy (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_sps_icon::Set_colors (
    const double*,          // color data (r,g,b)
    int                     // array size
                            // number of doubles
    );
```

Set color data.

```
public: virtual void DM_sps_icon::Tag_object_changed
(
    Spatial_abs_hurler&     // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_sps_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DM_srf_icon

Class: Deformable Modeling

Purpose: This class is used for drawing deformable surface tag objects in standalone deformable modeling.

Derivation: DM_srf_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmsrficon.hxx

Description: The DM_srf_icon is part of the dmicon library. This icon updates a grid of polylines to depict the current state of a deformable surface. Control points and element boundaries can also be displayed

Related classes: DM_ica_draw_option and DM_icq_draw_option encapsulate set and query draw option commands using the DM_rend_options class. These derive from the DM_def_icon_cmd_args and DM_def_icon_query_args base classes.

The DM_def_icon_cmd_args and the DM_def_icon_query_args encapsulate many of the methods inherited from the DM_default_icon interface. Thus the Set_state and Query methods can be used to perform casting, when different libraries of icons are mixed.

Limitations: None

References: DS DM_cpt_icon, DM_grd_icon

Data:

None

Constructor:

```
public: DM_srf_icon::DM_srf_icon ();
```

Public constructor.

Destructor:

```
protected: virtual DM_srf_icon::~DM_srf_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_srf_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args& // draw-command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_srf_icon::Get_colors (
    DM_dbl_array&            // data array
    ) const;
```

Query color data.

```
public: virtual void DM_srf_icon::Get_grid (
    int[2]                   // density
    ) const;
```

Query discretization grid density.

```
public: virtual void DM_srf_icon::Invalidate ();
```

Update cached data.

```
public: virtual DM_icon* DM_srf_icon::Make_copy (
    Spatial_abs_hurler&      // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_srf_icon::Set_colors (
    const double*,          // color data (r,g,b)
    int                     // array size
                            // number of doubles
    );
```

Set color data.

```
public: virtual void DM_srf_icon::Set_draw_option (
    int                     // option
    );
```

Set draw options.

```
public: virtual void DM_srf_icon::Set_grid (
    const int[2]            // density
    );
```

Set discretization grid density.

```
public: virtual void DM_srf_icon::Set_icon_width (
    double                  // width
    );
```

Set icon draw width.

```
public: virtual void DM_srf_icon::Set_owner (
    Spatial_abs_hurler&,    // error handler
    DS_dmod* new_dmod,      // icon owner ID
    int new_tag             // icon owner ID
    );
```

Notification of owner completion, for icon initialization.

```
public: virtual void DM_srf_icon::Tag_object_changed
(
    Spatial_abs_hurler&     // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_srf_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DM_syserr_hurler

Class: Deformable Modeling

Purpose: This class provides a protocol for handling exceptions across interfaces.

Derivation: DM_syserr_hurler : Spatial_abs_hurler : –

SAT Identifier: None

Filename: ds/dshusk/dskernel/dm_syserr_hurler.hxx

Description: This class encapsulates the DM_sys_error method. Interface routines taking a Spatial_abs_hurler agree to trap all exceptions, translate them to an integer code, and then call the Spatial_abs_hurler rethrow_error method with the integer code. The DM_syserr_hurler::rethrow_error method calls DM_sys_error on non–zero error codes, and ignores the zero error code.

Limitations: None

References: None

Data:

None

Constructor:

```
public: DM_syserr_hurler::DM_syserr_hurler ();
```

Public constructor.

Destructor:

```
public: DM_syserr_hurler::~DM_syserr_hurler ();
```

Public destructor.

Methods:

```
public: virtual void DM_syserr_hurler::rethrow_error
(
    int                     // code
    );
```

Encapsulate DM_sys_error.

Related Fncs:

None

DM_tag_array

Class: Deformable Modeling

Purpose: The DM_tag_array class supports tag query methods and icon broadcast methods.

Derivation: DM_tag_array : –

SAT Identifier: None

Filename: ds/dshusk/dskernel/dm_tag_array.hxx

Description: This is a read–only container class for deformable modeling tags. Deformable modeling tag query methods return a DM_tag_array populated with tag object tags. Thus, clients needn't allocate or destroy heap memory. Queries on a single DS_dmod or a DS_dmod hierarchy are supported. The returned DM_tag_array can be passed as–is to the deformable modeling icon broadcast draw and set_state methods; the compiler will automatically use the provided const int* const cast method.

Related methods: A DM_tag_array object is returned by the DM_get_tags, DM_get_dmod_tags, ATTRIB_DM2ACIS::Get_tags, and ATTRIB_DM2ACIS::Get_dmod_tags methods. A DM_tag_array can be passed in place of a const int* to the icon broadcast methods DM_draw_icon, DM_draw_dmod_icons, ATTRIB_DM2ACIS::Draw_icon, ATTRIB_DM2ACIS::Draw_dmod_icons, and the corresponding XXX_setstate_icon, XXX_setstate_dmod_icons methods.

Limitations: None

References: None

Data:

None

Constructor:

```
public: DM_tag_array::DM_tag_array ();
```

Public constructor.

Destructor:

```
public: DM_tag_array::~DM_tag_array ();
```

Public destructor.

Methods:

```
public: operator const DM_tag_array::int* () const;
```

Read–only cast operator.

```
public: int DM_tag_array::operator[] (
    int                     // pointer
    ) const;
```

Used for data access.

```
public: int DM_tag_array::Size () const;
```

Number of array elements.

Related Fncs:

None

DM_vec_icon

Class: Deformable Modeling

Purpose: This class is used for drawing vector load tag objects in deformable modeling.

Derivation: DM_vec_icon : DM_default_icon : DM_icon : –

SAT Identifier: None

Filename: ds/dmicon/dmvecicon.hxx

Description: The DM_vec_icon is part of the dmicon library. This icon uses a polyline to depict the current direction of a vector load.

Related classes: The DM_def_icon_cmd_args and the DM_def_icon_query_args encapsulate many of the methods inherited from the DM_default_icon interface. Thus the Set_state and Query methods can be used to perform casting, when different libraries of icons are mixed.

The DM_def_icon_draw_args and the DM_def_icon_cmd_args can be broadcast to icons using the deformable modeling interface methods, such as DM_draw_all_icons.

Limitations: None

References: None

Data:

None

Constructor:

```
protected: DM_vec_icon::DM_vec_icon (
    const DM_vec_icon&      // icon
    );
```

Copy constructor; call Make_copy and Set_owner instead.

```
public: DM_vec_icon::DM_vec_icon ();
```

Public constructor.

Destructor:

```
protected: virtual DM_vec_icon::~DM_vec_icon ();
```

Destructor; use the Lose method instead.

Methods:

```
public: virtual void DM_vec_icon::Draw (
    Spatial_abs_hurler&,     // error handler
    const DM_icon_draw_args& // draw-command object
    ) const;
```

Draw this icon.

```
public: virtual void DM_vec_icon::Get_colors (
    DM_dbl_array&           // data array
    ) const;
```

Query color data.

```
public: virtual DM_icon* DM_vec_icon::Make_copy (
    Spatial_abs_hurler&     // error handler
    ) const;
```

Clone method.

```
public: virtual void DM_vec_icon::Set_colors (
    const double*,          // color data (r,g,b)
    int                     // array size
                            // number of doubles
    );
```

Set color data.

```
public: virtual void DM_vec_icon::Tag_object_changed
(
    Spatial_abs_hurler&     // error handler
    );
```

Notification of owning tag object state change.

```
public: virtual void DM_vec_icon::Validate ();
```

Update cached data.

Related Fncs:

None

DS_dmod

Class: Deformable Modeling

Purpose: Pointer to this class acts as an handle.

Derivation: DS_dmod : –

SAT Identifier: None

Filename: ds/dshusk/dskernel/dsdmod.hxx

Description: Pointers to this SDM interface class DS_dmod should be considered as handles – they should always be forward–referenced, and never dereferenced. They are created, copied, and deleted by DM api's, and only used as handle (tag) arguments to DM api's. For example, the functions DM_make_bspline_curve and DM_make_bspline_surface will make a curve or surface DS_pfunc from arrays of knots and control points. The resulting DS_pfunc is then used to make a DS_dmod with one of the functions DM_make_dmod_curve or DM_make_dmod_surface; the resulting DS_dmod object assumes ownership of the DS_pfunc. After completing the desired DM api calls, the user must call DM_delete_dmod on the DS_dmod pointer.

Limitations: None

References: DS DM_icon, DS_pfunc
by DS DM_default_icon

Related Fncs:

None

DS_pfunc

Class: Deformable Modeling

Purpose: Pointer to this class acts as an handle.

Derivation: DS_pfunc : –

SAT Identifier: None

Filename: ds/dshusk/dskernel/dspfunc.hxx

Description: Pointers to this SDM interface class DS_pfunc should be considered as handles – they should always be forward–referenced, and never dereferenced. They are created, copied, and deleted by DM api's, and only used as handle (tag) arguments to DM api's. For example, the functions DM_make_bspline_curve and DM_make_bspline_surface will make a curve or surface DS_pfunc from arrays of knots and control points. The resulting DS_pfunc is then used to make a DS_dmod with one of the functions DM_make_dmod_curve or DM_make_dmod_surface; the resulting DS_dmod object assumes ownership of the DS_pfunc. After completing the desired DM api calls, the user must call DM_delete_dmod on the DS_dmod pointer.

Limitations: None

References: by DS DS_dmod

Related Fncs:

DS_call_segment_curve_by_pfunc_isolines

DS_poly_zone

Class: Deformable Modeling

Purpose: The DS_poly_zone is used for building a DS_area_load.

Derivation: DS_poly_zone : DS_zone : –

SAT Identifier: None

Filename: ds/dshusk/dskernel/dszone.hxx

Description: A DS_poly_zone delineates a piecewise linear subarea of a deformable surface. The DM interface function DM_build_poly_zone is used to create a DS_poly_zone and the DM interface function DM_delete_zone is used to delete a DS_poly_zone.

Limitations: None

References: DS DS_dmod

Related Fncs:

None

DS_zone

Class: Deformable Modeling

Purpose: Pointer to this class acts as an handle.

Derivation: DS_zone : –

SAT Identifier: None

Filename: ds/dshusk/dskernel/dszone.hxx

Description: Pointers to this SDM interface class DS_zone should be considered as handles – they should always be forward–referenced, and never dereferenced. They are created, copied, and deleted by DM api's, and only used as handle (tag) arguments to DM api's. For example, the functions DM_make_bspline_curve and DM_make_bspline_surface will make a curve or surface DS_pfunc from arrays of knots and control points. The resulting DS_pfunc is then used to make a DS_dmod with one of the functions DM_make_dmod_curve or DM_make_dmod_surface; the resulting DS_dmod object assumes ownership of the DS_pfunc. After completing the desired DM api calls, the user must call DM_delete_dmod on the DS_dmod pointer.

Limitations: None

References: None

Related Fncs:

None

SDM_options

Class: Deformable Modeling

Purpose: Provides an optional argument to all DM_ interface functions.

Derivation: SDM_options : –

SAT Identifier: None

Filename: ds/dshusk/dskernel/sdm_options.hxx

Description: The SDM_options class provides an optional argument to all DM_ interface functions. Currently it is used only for algorithmic versioning of boundary loads to pre or post R10 behaviors.

SDM_options must be created with the interface function DM_make_SDM_options, and must be deleted with the interface function DM_delete_SDM_options. An example is provided below. Refer to the respective function documentation for more information.

```
    #include ”dshusk/dskernel/dmapi.hxx”
                 // Create a DS_dmod object and a curve constraint DS_pfunc
object
                 DS_dmod* dmod = ...;
                 DS_pfunc* src_C_pfunc = ...;
                 // add a curve constraint to the DS_dmod object
                 int rerr=0;
                 SDM_options * sdo = DM_make_SDM_options(8,0,0);// for pre–R10
behavior
                 // if (rerr!=0) error_condition()
                 DM_add_curve_load(rerr, dmod, 2, NULL, 0, src_C_pfunc, NULL,
NULL, NULL, NULL,ds_bound_cstrn, DM_POS_FIXED, 1.e+6, –1);
                 // if (rerr!=0) error_condition()
                 DM_delete_SDM_options(rerr,sdo);
                 sdo = NULL:
                 // if (rerr!=0) error_condition()
```

Limitations: None

References: BASE AcisVersion

Related Fncs:

None

Spatial_abs_hurler

Class: Deformable Modeling

Purpose: This class provides a protocol for handling exceptions across interfaces.

Derivation: Spatial_abs_hurler : –

SAT Identifier: None

Filename: ds/dshusk/dskernel/spatial_abs_hurler.hxx

Description: Exception handling across interfaces is problematic, because different libraries may have different exception handling methods, e.g., C++ try–catch or C setjmp–longjmp. Interface routines taking a Spatial_abs_hurler agree to trap all exceptions, translate them to an integer code, and then call the Spatial_abs_hurler rethrow_error method with the integer code. Implementations overriding the Spatial_abs_hurler must have the rethrow_error method non–operational when passed a 0 argument.

Limitations: None

References: None

Data:

None

Constructor:

```
public: Spatial_abs_hurler::Spatial_abs_hurler ();
```

Default constructor.

Destructor:

```
public: virtual
    Spatial_abs_hurler::~Spatial_abs_hurler ();
```

Default destructor.

Methods:

```
public: virtual void
    Spatial_abs_hurler::rethrow_error (
    int                     // resignal
    )=0;
```

A method for resignaling an exception. It must be non–operational when passed a 0 argument.

Related Fncs:

None