



MYRIAD 3D Reader Interface Details v. 5.0

This document provides brief descriptions of the properties, methods, and events used to interface with the M3DR.dll ActiveX control for MYRIAD 3D Reader. Example code for the Model Tree and Cross Section functions are included at the bottom of this document.

Contents:

Rendering Interface	4
Properties	4
BackColor:	4
DrawViewType	4
ProjectionType	4
RenderType	4
ShowOrientationCue	5
Mouse Tool Interface	5
LeftMouseButtonTool	5
Model\View Manipulation Interface	7
Properties	7
FileName	7
AnimateViewChanges	7
CurrentViewState	7
CurrentCameraPosition	7
UpAxis	8
ViewAtExtents():	8
Methods	8
SelectPredefinedView	8
Model Part\Entity Manipulation Interface.....	9
Methods	9
SetFileDisplayName	9
GetRootEntityID	9
GetChild	9
GetEntityDisplayed.....	9
GetEntityName.....	9
GetNumberOfChildren.....	9
SetEntityBoundingBox	10
SetEntityDisplayed.....	10
GetEntityBoundingBoxMinValues	10
GetEntityBoundingBoxMaxValues	10
GetEntitySurfaceArea	10
GetEntityVolume	10
GetEntityCenterOfMass	11
View Output Interface	11
Methods	11
ExportViewAsJPG	11

GenerateDIB	11
Measurement Interface	11
Properties	11
MeasurementUnits	11
MeasurementType.....	12
NumberOfMeasurements	12
FirstPickType	12
Methods	13
RemoveMeasuegmentCues	13
RemoveMeasurement.....	13
UnSelectAllMeasurements();.....	13
RemoveSelectedMeasurement();	13
SuspendMeasurement();.....	13
RestartMeasurement();.....	13
GetMeasurementValue.....	14
Cut/Cross-section Interface	14
Properties	14
NumberOfCrossSections.....	14
RemoveCrossSections.....	14
Methods	14
InitializeCCSection	14
GetCCSectionPlane.....	15
FinalizeCCSection.....	15
CancelCCSection	16
Get and Set CCParam.....	16
Explode Interface	16
Properties	16
ShowExplodeArrows	16
Methods	16
DisassmbleModel();.....	16
AssembleModel().....	17
DisassmbleModelLevel.....	17
GetMinAndMaxExplodeLevels	17
DisassmbleModelPart	17
Digital Rights Information Interface	17
InformAboutDigitalRights	17
MeasureEnabled.....	17
CrossSectionEnabled.....	18
PartViewingEnabled	18
PrintEnabled.....	18
Model\Application Information Interface	18
Properties	18
ModelIsAssembly	18
LicenseString.....	18
ProductVersion.....	18
Methods	18
FormatEnabled	18
Events	19
EntityClicked.....	19
MeasurementCreated.....	19
MouseMove.....	19

EntityDoubleClicked.....	19
BoundingBoxHighlighted	20
MouseDown.....	20
MeasurementClicked.....	20
ThreePointPlaneSet()	21
AssembleAvailable().....	21
ModelLoaded	21
SystemInitialized().....	21
MPCtrlMouseDown ()	21
MPCtrlMouseUp ()	21
Cut/Cross-Section Example Code	22
Model Tree Example HTML and Visual Basic Code	23
HTML Sample Code.....	23
Visual Basic Sample Code.....	26

Rendering Interface

Properties

BackgroundColor:

OLE_COLOR. Used to get/set the control's background color.

Example:

```
viewObject.BackgroundColor = 0x00FF00; // change the background to green.
```

DrawViewType

Integer. A value corresponding to the following enumeration:

```
typedef enum {DRAW_VIEW_FULL = 0,  
              DRAW_VIEW_SAMPLED,  
              DRAW_VIEW_VERTEX,  
              DRAW_VIEW_FEATURE}
```

Used to set the dynamic display type used while orbiting a views model.

Example:

```
viewObject.DrawViewType = 2 // while the model is being manipulated,  
                             // it will appear as a point cloud to speed up draw time.
```

ProjectionType

Integer. A value corresponding to the following enumeration:

```
typedef enum { IMAGE_PERSPECTIVE= 0,  
              IMAGE_ORTHOGRAPHIC};
```

Used to get/set the current projection mode for the view.

Example:

```
viewObject.ProjectionType = 1; // causes the model to be drawn with an  
                               // orthographic projection.
```

RenderType

Integer. A value corresponding to the following enumeration:

```
typedef enum {IMAGE_WIRE_FRAME = 0,  
              IMAGE_HIDDEN_LINE,  
              IMAGE_SHADED,  
              IMAGE_COMPOSITE,  
              IMAGE_BREP,
```

```
IMAGE_BREP_HIDDEN,  
IMAGE_BREP_COMPOSITE,  
IMAGE_FEATURE,  
IMAGE_FEATURE_HIDDEN_LINE,  
IMAGE_FEATURE_COMPOSITE};
```

Used to get/set the current rendering method.

Example:

```
viewObject.RenderType = 1; // cause the model to be drawn in hidden line  
mode
```

ShowOrientationCue

Boolean(integer). If set to true, will cause an axis orientation cue to be displayed in the lower left corner of the view.

Mouse Tool Interface

All interaction between the 3D view and the user's mouse is controlled by changing the property *LeftMouseButtonTool*.

LeftMouseButtonTool

Integer. A value corresponding to the following enumeration:

```
#define TOOL_TYPE_ORBIT           (0)  
#define TOOL_TYPE_ZOOM           (1)  
#define TOOL_TYPE_PAN            (2)  
#define TOOL_TYPE_SELECTION      (4)  
#define TOOL_TYPE_CROSS_SECTION (5)  
#define TOOL_TYPE_MEASUREMENT    (6)  
#define TOOL_TYPE_BBOX           (7)  
#define TOOL_TYPE_SNAP_TO_FACE  (8)  
#define TOOL_TYPE_ZOOMRECT       (9)  
#define TOOL_TYPE_MEASURE_EDIT  (10)  
#define TOOL_TYPE_INTERACTIVE_EXPLODE (11)
```

Used to get/set the current tool associated with the left mouse button.

Example:

```
viewObject.LeftMouseButtonTool = 2 // when the mouse is dragged while the left  
// mouse button is down, the model will be  
// panned.
```

TOOL_TYPE_ORBIT

While dragging the mouse with the left button down, the camera orbits. While dragging the mouse with the right button down, the camera zooms. While

dragging the mouse with both buttons down, the camera pans. This is the default tool.

TOOL_TYPE_ZOOM

While dragging the mouse with the left button down, the camera zooms.

TOOL_TYPE_PAN

While dragging the mouse with the left button down, the camera pans.

TOOL_TYPE_SELECTION

Allow the user to select individual parts/entities on the model. Displays the selected part/entity name.

TOOL_TYPE_CROSS_SECTION

Used to allow the user to select three model points to create a cutting plane. See Cross-section Interface section.

TOOL_TYPE_MEASUREMENT

Used to allow the user to select model points to create measurements. See Measurement Interface section.

TOOL_TYPE_BBOX

If the mouse is clicked while hovering over an entity/model part bounding box, the control will fire a *BoundingBoxHighlighted* event. See *BoundingBoxHighlighted* in the events section as well as *SetEntityBoundingBox* in the Model Part/Entity Manipulation section.

TOOL_TYPE_SNAP_TO_FACE

If the mouse is clicked while hovering over the model, the camera will be oriented so that the view plane is parallel to the selected model face.

TOOL_TYPE_ZOOMRECT

While dragging the mouse with the left button down, a rectangle is created into which the camera zooms.

TOOL_TYPE_MEASURE_EDIT

Allows the user to select measurements. If the mouse is dragged with the left button down while the measurement is selected, the measurement will be moved. The control will fire a *MeasurementClicked* event when a measurement is selected. See *MeasurementClicked* in the events section as well as *RemoveMeasurement/RemoveSelectedMeasurement* in the Measurement Interface section.

TOOL_TYPE_INTERACTIVE_EXPLODE

Allows the user to select and move individual model parts.

Model\View Manipulation Interface

Properties

FileName

String. Used to get/set the file currently being viewed.

Example:

```
viewObject.FileName = "C:/files/somefile.isf" // causes the control to attempt to
// load the indicated file.
```

AnimateViewChanges

Boolean(integer). If set to true, will cause the view to animate changes between predefined view types.

CurrentViewState

String. This is used to get/set the current view state information. This information is returned as a binary large object (BLOB), and must not be modified in any way prior to giving the view state to an instance of the viewer control.

This property allows you to set the view state of one instance of a control to the state of another. Note that this state information is machine independent, so you can set the view state of a control running on one machine to the current state of an instance running on another machine.

All information necessary to restore the current view is contained in the returned BLOB. The intended use of this property is for setting the initial state when synchronizing multiple instances of the viewer control, all of which are viewing the same content. Due to the computational expense of setting the state, if only the camera position is required to be updated, the *CurrentCameraPosition* property should be used instead.

Example:

```
// Get the view state of object 1
viewState = viewObject1.CurrentViewState;

// Set the view state of object2
viewObject2.CurrentViewState = viewState;
```

CurrentCameraPosition

String. This is used to get/set the current camera position information. This information is returned as a binary large object (BLOB), and must not be modified in any way prior to giving the data to an instance of the control.

This property allows the user to set the camera position of one instance of the control to the camera position being used in another instance of the control. This information is

machine independent, and is intended to be used to synchronize the camera positions for multiple instances of the viewer control which are all viewing the same content.

Example:

```
// Get the camera position of object 1
cameraPosition = viewObject1.CurrentCameraPosition;

// Set the camera position of object2
viewObject2.CurrentCameraPosition = cameraPosition;
```

UpAxis

Integer. A value corresponding to the following enumeration:

```
typedef enum{X_UP,
             Y_UP,
             Z_UP}
```

Used to get/set the current up axis.

Example:

```
viewObject.UpAxis = 1; // the model will be oriented so that
                       // the Y axis points up.
```

ViewAtExtents():

ViewAtExtents causes a zoom out or in so that the visible model geometry fits into the current view area.

Example:

```
viewObject.ViewAtExtents(); // Make all visible geometry fit into the current view.
```

Methods

SelectPredefinedView

SelectPredefinedView(short ViewType):

ViewType is a value corresponding to the following enumeration:

```
typedef enum {TOP = 0,
             BOTTOM,
             LEFT,
             RIGHT,
             FRONT,
             BACK,
             ISOMETRIC,
             DIMETRIC,
             TRIMETRIC,
             REVERSE };
```

SelectPredefinedView will cause the model to be viewed from one of the above specified camera positions.

Example:

```
viewObject.SelectPredefinedView(5); // view the model from the BACK camera  
// position
```

Model Part Entity Manipulation Interface

Methods

SetFileDisplayName

SetFileDisplayName([in] BSTR displayName):
Boolean(integer). Sets the name to use for displaying the file (the root item) in the Model Part tree. TRUE is returned if successful, otherwise FALSE is returned.

GetRootEntityID

GetRootEntityID([out, retval] double *RootID):
Returns the entity id of the top-most entity.

GetChild

GetChild(double EntityID, short Index, [out,retval] double *ChildID):
Returns a ChildID of the given EntityID.

Example:

```
viewObject.GetChildID(306, 2, &id); //returns the id of the second child of entity  
// 306
```

GetEntityDisplayed

GetEntityDisplayed(double EntityID, [out,retval] short* Displayed):
Returns a value indicating if the entity with id=EntityID is currently displayed.

Example:

```
viewObject.GetEntityDisplayed(306, &isDisplayed); //determines if entity306 is  
// currently displayed
```

GetEntityName

GetEntityName(double EntityID, [out,retval] BSTR *EntityName):
Returns a string that is the name of the entity identified by EntityID.

GetNumberOfChildren

GetNumberOfChildren(double EntityID, [out,retval] short* NumberOfChildren):

Returns the number of entities that are descendant from the entity identified by EntityID.

SetEntityBoundingBox

SetEntityBoundingBox(double EntityID, short BoxDisplayed)

Causes the entity identified by EntityID to hide or display its bounding box geometry based on the value of BoxDisplayed.

Example:

```
viewObject.SetEntityBoundingBox (306,1); // display the bounding box geometry  
// for entity 306.
```

SetEntityDisplayed

SetEntityDisplayed(double EntityID, short Displayed, short ApplyToChildren):

Causes the entity identified by EntityID to be hidden or displayed based on the value of Displayed.

Example:

```
viewObject.SetEntityDisplayed (306,1); // display entity 306
```

GetEntityBoundingBoxMinValues

GetEntityBoundingBoxMinValues(double EntityID, [out] double* X, [out] double* Y, [out] double* Z);

Retrieves the minimum 3D point of the bounding box of the entity identified by EntityID.

GetEntityBoundingBoxMaxValues

GetEntityBoundingBoxMaxValues(double EntityID, [out] double* X, [out] double* Y, [out] double* Z);

Retrieves the maximum 3D point of the bounding box of the entity identified by EntityID.

GetEntitySurfaceArea

GetEntitySurfaceArea(double EntityID, [out, retval] double* A);

Retrieves the surface area of the entity identified by EntityID.

GetEntityVolume

GetEntityVolume(double EntityID, [out, retval] double* V);

Retrieves the volume of the entity identified by EntityID.

GetEntityCenterOfMass

GetEntityCenterOfMass(double EntityID, [out] double* X, [out] double* Y, [out] double* Z);

Retrieves the 3D point that is the center of mass of the entity identified by EntityID.

View Output Interface

Methods

ExportViewAsJPG

ExportViewAsJPG(int width, int height, BSTR JpgFilename):

Creates a jpg snapshot of the current view.

Width - The width in pixels of the resulting jpg image. Width min = 25 Width max = 1400

Height - The height in pixels of the resulting jpg image. Height min = 25 Height max = 1400

JpgFilename - The filename of the jpg to be created.

Note that *ExportViewAsJPG* will overwrite the contents of *JpgFilename* if the file already exists.

GenerateDIB

GenerateDIB(int width, int height, int *DIB):

GenerateDIB creates a HBITMAP of the current view.

Width - The width in pixels of the resulting bitmap handle. Width min = 25 Width max = 1000

Height - The height in pixels of the resulting bitmap handle. Height min = 25 Height max = 1000

DIB - The handle to the bitmap is returned in DIB in the form of a Windows GDI HBITMAP.

Note that the calling application is responsible for destruction of the GDI handle.

Measurement Interface

Properties

MeasurementUnits

Integer. A value corresponding to the following enumeration:

```
typedef enum {UNITS_UNKNOWN=0,
              UNITS_MICRONS,
              UNITS_MILLIMETERS,
              UNITS_CENTIMETERS,
              UNITS_METERS,
              UNITS_KILOMETERS,
```

```
UNITS_MICROINCHES,  
UNITS_INCHES,  
UNITS_FEET,  
UNITS_MILES,  
UNITS_MILS}
```

Used to get/set the units used in measurement for the view.

Example:

```
viewObject. MeasurementUnits = 1; // cause all measurements to be done  
// in micron units.
```

MeasurementType

Integer. A value corresponding to the following enumeration:

```
typedef enum {LINEAR,  
EDGE,  
ARC_RECOGNITION,  
EDGE_ANGLE,  
THREE_POINT_CIRCLE,  
THREE_POINT_ARC,  
THREE_POINT_ANGLE}
```

Used to get/set the current measurement type.

Example:

```
viewObject. MeasurementType = 1; // the next measurement created will  
// be an edge measurement.
```

NumberOfMeasurements

Integer. A value corresponding to the total number of measurements currently existing on the model (Read Only).

FirstPickType

Integer. A value corresponding to the following enumeration:

```
typedef enum {PICK_NONE,  
PICK_EDGE,  
PICK_VERTEX,  
PICK_FACE,  
PICK_PARALLEL_EDGE,  
PICK_PERPENDICULAR_EDGE,  
PICK_NP_PLANAR_EDGE,  
PICK_ARC_CENTER,  
PICK_HIGHLIGHT_EDGE,  
PICK_ARC_CENTER_PLANAR,
```

```
PICK_POINT_ON_EDGE,  
PICK_FEATURE,  
PICK_FEATURE_DISTANCE}
```

This property controls the type of pick done when the MeasurementType = LINEAR.

Example:

```
viewObject.FirstPickType = 1; // The first mouse click during the linear  
measurement will viewObject.SecondPickType = 3; // highlight and pick the nearest  
model edge. // The next mouse click will highlight and pick the nearest model face.
```

SecondPickType and ThirdPickType

See FirstPickType.

Methods

RemoveMeasurementCues

RemoveMeasurementCues();

Removes all of the measurement cues that are on the model.

RemoveMeasurement

RemoveMeasurement([in] double EntityID); Removes a particular measurement cue identified by EntityID from the model.

UnselectAllMeasurements();

Causes any currently selected measurements to become unselected.

See property LeftMouseButtonTool/TOOL_TYPE_MEASURE_EDIT.

RemoveSelectedMeasurement();

Removes the currently selected measurement cue from the model.

See property LeftMouseButtonTool/TOOL_TYPE_MEASURE_EDIT.

SuspendMeasurement();

If the current mouse tool is type TOOL_TYPE_MEASUREMENT, this method will change the mouse tool to TOOL_TYPE_ORBIT while saving any completed measurement picks. See property

LeftMouseButtonTool/TOOL_TYPE_MEASUREMENT.

RestartMeasurement();

After a call to SuspendMeasurement(), this method will change the mouse tool back to TOOL_TYPE_MEASUREMENT so that measurement picking can continue. See property LeftMouseButtonTool/TOOL_TYPE_MEASUREMENT.

GetMeasurementValue

GetMeasurementValue(double EntityID, [out,retval] BSTR *MeasurementValue);
Returns a string representation of the measurement cue identified by EntityID

Cut/Cross-section Interface

For example code for the model tree and cross section functions, refer to the next section [Cut/Cross-Section Example](#) and [Model Tree Example HTML and Visual Basic Code](#).

Properties

NumberOfCrossSections

Integer. A value corresponding to the total number of cuts and cross-sections currently existing on the model (Read Only).

RemoveCrossSections

RemoveCrossSections():
Removes all of the cross sections that are on the model in the current view.

Methods

InitializeCCSection

InitializeCCSection(short InitialPos):

Initializes the current view to perform a model cut or cross-section. A call to this method will place a visible cutting plane into the view.

InitialPos defines how the cutting plane is initially placed and must be one of the following values:

```
enum{ CUTTING_PLANE_THREE_VERTEX = 3,  
      CUTTING_PLANE_MODEL_WIDTH = 4,  
      CUTTING_PLANE_MODEL_HEIGHT = 5,  
      CUTTING_PLANE_MODEL_DEPTH = 6}
```

The following descriptions are termed relative to an isometric view of the model.

CUTTING_PLANE_MODEL_WIDTH:

Places the cutting plane perpendicular to the axis that is right of the current up-axis. If the up-axis is the y-axis, this would place the plane perpendicular to the x-axis.

CUTTING_PLANE_MODEL_HEIGHT:

Places the cutting plane perpendicular to the axis that is the current up- axis.

CUTTING_PLANE_MODEL_DEPTH:

Places the cutting plane perpendicular to the axis that is left of the current up-axis. If the up-axis is the y-axis, this would place the plane perpendicular to the z-axis.

CUTTING_PLANE_THREE_VERTEX:

Calling *InitializeCCSection* with a value of CUTTING_PLANE_THREE_VERTEX allows you to specify the cutting plane by choosing three points on the surface of the model. After calling *InitializeCCSection*(CUTTING_PLANE_THREE_VERTEX), the interfacing application must set the *LeftMouseButtonTool* to TOOL_TYPE_CROSS_SECTION (5) (see [LeftMouseButtonTool](#) description). After setting the *LeftMouseButtonTool*, the user will select the desired model points to define the cutting plane. After the third point is selected, the cutting plane appears.

GetCCSectionPlane

GetCCSectionPlanePosMin(short *PosMin)

and

GetCCSectionPlanePosMax(short *PosMax)

These methods are used to get an integer interval within which the cutting plane can be repositioned. They are to be used in conjunction with the following SetCCSectionPlanePos method.

SetCCSectionPlanePos(short NewPos)

This method is called to re-position the cutting plane. The value NewPos must be between the value returned by GetCCSectionPlanePosMin and the value returned by GetCCSectionPlanePosMax. SetCCSectionPlanePos will cause the cutting plane to be re-positioned along the vector perpendicular to the plane.

GetCCSectionPlanePos(short *CurrentPos)

This method is called to return an integer value within the range established by return values of GetCCSectionPlanePosMin and GetCCSectionPlanePosMax. The returned value indicates the current position of the cutting plane.

FinalizeCCSection

FinalizeCCSection()

After a cutting plane is created and positioned using InitializeCCSection and SetCCSectionPlanePos, The interfacing application must call FinalizeCCSection to complete the cut or cross-section.

CancelCCSection

CancelCCSection()

This method is called to remove a cutting plane after initialization and before finalization.

Get and Set CCParm

GetCCParm (short ParamType, [out,retval] short *CurrentValue)

And

SetCCParm(short ParamType, short NewValue):

These methods allow the interfacing application to set an attribute of a cut or cross-section. They must be called after initialization and before finalization. ParamType indicates which attribute is to be changed, and NewValue is set to 1 to enable and 0 to disable.

Possible values for ParamType:

```
enum {CUT = 0,  
      CROSS_SECTION = 1,  
      ADJUST_VIEW = 2,  
      FILL_CROSS_SECTION = 3,  
      REVERSE_CUT = 4}
```

Example:

```
M3DRObj.SetCCParm(0,1) // perform a cut
```

```
M3DRObj.SetCCParm(1,1) // perform a cross-section
```

Explode Interface

Properties

ShowExplodeArrows

Boolean(integer). If set to true, will cause arrows to be drawn from the exploded position to the original position.

Methods

DisassmbleModel();

Causes each sub-part in the top-level assembly to be moved outward from the center of the model.

AssembleModel()

Reassembles each sub-part in the top-level assembly after an explode has been performed.

DisassmbleModelLevel

DisassmbleModelLevel(short Level); Causes each sub-part in the top-level assembly to be moved outward from the center of the model. If the model contains sub-assemblies(sub-parts with parts), these will be kept assembled based on the value of Level.

GetMinAndMaxExplodeLevels

GetMinAndMaxExplodeLevels([out] short* Min, [out] short* Max);

Retrieves the number of sub-assembly levels contained within a model. A model with one simple sub-part will return Min=1 and Max=2, a model with one sub-assembly will return Min=1 and Max=3, a model with one sub-assembly that sub-assembly containing a sub-assemble will return Min=1,Max=4 etc.

DisassmbleModelPart

DisassmbleModelPart(double EntityID); Causes each sub-part in the top-level assembly to be moved outward from the center of the part identified by EntityID. See LeftMouseButtonTool/TOOL_TYPE_SELECTION and Model Part/Entity Manipulation section for information on selecting sub-parts and getting IDs.

Digital Rights Information Interface

When MYRIAD 3D Reader ISF files are published, certain viewing features can be disabled. This section describes the digital rights settings specific to the currently loaded model.

InformAboutDigitalRights

Boolean(integer). If set to true, will cause a dialog to display upon model load that lists the features disabled for the model.

MeasureEnabled

Boolean(integer). Indicates if the currently viewed model was published with measurement disabled. If MeasureEnabled=false, none of the measurement interfaces will work. (Read Only)

CrossSectionEnabled

Boolean(integer). Indicates if the currently viewed model was published with cuts and cross-sections disabled. If CrossSectionEnabled=false, none of the cross-section interfaces will work. (Read Only)

PartViewingEnabled

Boolean(integer). Indicates if the currently viewed model was published with sub-part viewing disabled. (Read Only)

PrintEnabled

Boolean(integer). Indicates if the currently viewed model was published with printing disabled. The current version has no interfaces implemented for printing. (Read Only)

Model\Application Information Interface

Properties

ModellsAssembly

Boolean(integer). Indicates if the currently viewed model has sub-parts.

LicenseString

String. A string representation of the control's license number found in M3DR.ini. (Read Only)

ProductVersion

String. A string representation of the control's dll version in the form: "MajorVersion, MinorVersion1, MinorVersion2, BuildNumber" (Read Only)

Methods

FormatEnabled

FormatEnabled ([in] int FormatID,[out, retval] BOOL *pVal). This method returns TRUE in pVal if the given FormatID is license enabled. The following are valid values for FormatID:

#define MYRIAD_ICS_ID	(14) IronCad models
#define MYRIAD_INVENTOR_ID	(15) Inventor models
#define MYRIAD_SOLIDEDGE_ID	(17) Solid Edge models
#define MYRIAD_SOLIDWORKS_ID	(18) Solidworks models
#define MYRIAD_PROE_ID	(21) Pro-E models
#define MYRIAD_DWG_ID	(2) AutoCad models

Events

EntityClicked

EntityClicked(double *EntityID, long *X, long *Y, BSTR *EntityName)

Occurs when a mouse tool = TOOL_TYPE_SELECTION (4) and the mouse is clicked while over a sub-part.

EntityID - Double value uniquely identifying the sub-part.

X ,Y - The x and y value of the mouse location given in logical coordinates and relative to the upper left corner of the control window.

EntityName - String containing the name of the sub-part.

MeasurementCreated

MeasurementCreated(double *CueID)

Occurs when a mouse tool = TOOL_TYPE_MEASUREMENT(6) and a measurement cue has been created.

CueID - Double value uniquely identifying the measurement created.

MouseMove

MouseMove([long *X, long *Y)

Occurs when the mouse is moved within the control window.

X ,Y- The x and y value of the mouse location given in logical (pixel) coordinates and relative to the upper left corner of the control window.

EntityDoubleClicked

EntityDoubleClicked(double *EntityID, long *X, long *Y, BSTR *EntityName)

Occurs when a mouse tool = TOOL_TYPE_SELECTION (4) and the mouse is double clicked while over a sub-part.

EntityID - Double value uniquely identifying the sub-part.

X ,Y – The x and y value of the mouse location given in logical coordinates and relative to the upper left corner of the control window.

EntityName – String containing the name of the sub-part.

BoundingBoxHighlighted

BoundingBoxHighlighted(double *EntityID, long *X, long *Y, BSTR *EntityName)

Occurs when a mouse tool = TOOL_TYPE_BBOX (7) and the mouse is clicked while over a sub-parts bounding box.

EntityID - Double value uniquely identifying the bounding box.

X ,Y – The x and y value of the mouse location given in logical coordinates and relative to the upper left corner of the control window.

EntityName – String containing the size in current units of the side of the bounding box selected.

MouseDown

MouseDown(long *X, long *Y)

Occurs when the mouse is moved while a mouse button is down within the control window.

X ,Y – The x and y value of the mouse location given in logical (pixel) coordinates and relative to the upper left corner of the control window.

MeasurementClicked

MeasurementClicked(double *EntityID, long *X, long *Y, BSTR *EntityName)

Occurs when the mouse tool = TOOL_TYPE_MEASURE_EDIT (10) and the mouse is clicked while over a measurement cue.

EntityID - Double value uniquely identifying the measurement cue.

X,Y - The x and y value of the mouse location given in logical coordinates and relative to the upper left corner of the control window.

EntityName - String containing the measurements displayed text.

ThreePointPlaneSet()

Occurs when the mouse tool = TOOL_TYPE_CROSS_SECTION (5) and the user has picked three model points that define a cutting plane.

AssembleAvailable()

Occurs when any sub-part explosion has completed.

ModelLoaded

ModelLoaded(BSTR *FileName);

Occurs when a model is successfully loaded via setting the FileName property

FileName - The full path of the model loaded.

SystemInitialized()

Occurs when the 3D drawing kernel has been completely initialized.

MPCtrlMouseDown ()

MPCtrlMouseDown([out] long *X, [out] long *Y,[out]long *MouseButtonID)

Occurs when a mouse button is pressed down while over the 3D view.

X,Y	- The x and y value of the mouse location given in logical coordinates and relative to the upper left corner of the control window.
MouseButtonID	- indicates which mouse button was pressed. 0 for left and 1 for right.

MPCtrlMouseUp ()

MPCtrlMouseUp([out] long *X, [out] long *Y,[out]long *MouseButtonID)

Occurs when a mouse button is released while over the 3D view.

X,Y	- The x and y value of the mouse location given in logical coordinates and relative to the upper left corner of the control window.
MouseButtonID	- indicate which mouse button was pressed. 0 for left, and 1 for right.

Cut/Cross-Section Example Code

```
define CUTTING_PLANE_MODEL_HEIGHT 0
define OPTION_CUT 0
define OPTION_CROSSSECTION 1
define OPTION_FILL_CROSSSECTION 3

//To initialize the cutting plane. Place the plane perpendicular to the up-axis.

M3DRObj.InitializeCCSection(CUTTING_PLANE_MODEL_HEIGHT);

//Get the min and max plane positions

integer minPos,maxPos;
minPos = M3DRObj.GetCCSectionPlanePosMin();
maxPos = M3DRObj.GetCCSectionPlanePosMax();

//Find the position one third of the way between min and max and set the //plane to that
position.

integer deltaPos = maxPos-minPos;
integer oneThird = deltaPos/3
integer newPos = minPos + oneThird;

M3DRObj.SetCCSectionPlanePos(newPos);

//Instruct the cutting plane to create a cut and filled cross-section.

M3DRObj.SetCCParam(OPTION_CUT,1);
M3DRObj.SetCCParam(OPTION_CROSSSECTION,1);
M3DRObj.SetCCParam(OPTION_FILL_CROSSSECTION,1);

//Finalize

M3DRObj.FinalizeCCSection();
```

Model Tree Example HTML and Visual Basic Code

HTML Sample Code

The following HTML code is an example of the MYRIAD 3D Reader control used in conjunction with a tree control. Methods and properties of the MYRIAD 3D Reader control are accessed through routines written in VBScript.

```
<html>
<head>
<meta http-equiv="Content-Language" content="en-us">

<title>MYRIAD 3D Reader Part Tree Example</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<base target="_self">
  <SCRIPT LANGUAGE="VBSCRIPT">
    Sub demo()

      ' The filename string must refer to a valid 3DF file.
      M3DRObj.FileName =
      "http://www.myriadviewer.com/samples/full\Allen_wr_assy_SLDASM.3df"
      BuildTree()

    End Sub

*****
*
' BuildTree routine asks the MYRIAD 3D Reader object for it's topmost
' part or sub-assembly and then recursively fills the part tree.
' Note that each part or sub-assembly (entity) has a unique integer
' value associated with it. The following routines show examples of:
' GetRootEntityID
' GetEntityName
' GetNumberOfChildren
*****
*

    Sub BuildTree()

      treeParts.Nodes.Clear
      dRootID = M3DRObj.GetRootEntityID()

      sRootKey = "ID " & dRootID

      Set nodX = treeParts.Nodes.Add (, 4, sRootKey,
M3DRObj.GetEntityName(dRootID))
```

```

BuildTreeRecursive (sRootKey)

    nodX.Expanded = True

End Sub

Sub BuildTreeRecursive(sParentKey)

    dParentID = Mid(sParentKey, InStr(sParentKey, " "))

    For i = 0 To M3DRObj.GetNumberOfChildren(dParentID) - 1
        dChildID = M3DRObj.GetChild(dParentID, i)
        sChildKey = "ID " & dChildID
        ' 4 = tvwChild
        treeParts.Nodes.Add sParentKey, 4, sChildKey,
M3DRObj.GetEntityName(dChildID)

        BuildTreeRecursive (sChildKey)
    Next

End Sub

Dim dLastBoxedEntity

'*****
'
' treeParts_Click event occurs when the user single clicks on a node
' in the part tree. The MYRIAD 3D Reader object is then asked to
' highlight the entity associated with the clicked node. This routine
' shows an example of:
' SetEntityBoundingBox
'*****
'

Sub treeParts_Click()

    If treeparts.Nodes.Count > 0 Then

        M3DRObj.SetEntityBoundingBox dLastBoxedEntity, false

        sKey = treeparts.SelectedItem.Key
        dLastBoxedEntity = Mid(sKey, InStr(sKey, " "))

        If treeparts.SelectedItem.Index > 1 Then
            ' Highlight the entity.
            M3DRObj.SetEntityBoundingBox dLastBoxedEntity, true
        End If
    End If
End Sub

```

End If
End If

End Sub

```
*****  
*  
' treeParts_DblClick event occurs when the user double clicks on a node  
' in the part tree. The MYRIAD 3D Reader object is asked to  
' display (alone) the entity (and any sub parts) associated with the  
' clicked node. This routine shows an example of:  
' SetEntityDisplayed  
*****  
*
```

Sub treeParts_DblClick()

If treeparts.Nodes.Count > 0 Then

If treeparts.SelectedItem.Index > 0 Then

sKey = treeparts.SelectedItem.Key

dEntityID = Mid(sKey, InStr(sKey, " "))

' Display the entity alone.

M3DRObj.SetEntityDisplayed dEntityID,

(M3DRObj.GetEntityDisplayed(dEntityID) = 0), true

End If

End If

End Sub

</SCRIPT>

<meta name="Microsoft Border" content="none">

</HEAD>

<BODY BGCOLOR="#FFFFFF" onLoad="demo()">

<object name="treeParts" width="200" height="400"

progid=MSComctlLib.TreeCtrl.2

classid="clsid:C74190B6-8589-11D1-B16A-00C0F0283628">

<param name="Style" value="6">

</object>

<OBJECT ID="M3DRObj"

CLASSID="CLSID:C7582045-2191-11D6-B705-0040051594CE"

```
width="400" height="400">
  <param name="_ExtentX" value="16325">
  <param name="_ExtentY" value="11245">
  <EMBED NAME="M3DRObj"></EMBED>
</OBJECT>
```

```
<p>
This page shows an example of the MYRIAD 3D Reader control
used in conjunction with a tree control. Please note that
the MYRIAD 3D Reader control and MSComctlLib.TreeCtrl.2 must
be properly registered on the machine for this page to display
correctly.
</p>
```

```
</body></html>
```

Visual Basic Sample Code

The following is a code snippet from a Visual Basic application that uses the MYRIAD 3D Reader interface. The intention of the snippet is to show an example of loading a model and filling a tree control with the names of the parts and sub-assemblies contained in the model.

ModelList is the object name of a VB TreeView control
ctrlM3DR is the object name of the MYRIAD 3D Reader control

```
Public Sub SetFileName(sFileName)
  Dim dRootID As long

  ModelList.Nodes.Clear

  ' Load the requested model file:
  ctrlM3DR.FileName = sFileName

  ' Ask MYRIAD 3D Reader control for the entity id of
  ' the topmost part or sub-assembly in the model.
  ' Note that each part or sub-assembly is guaranteed
  ' to have a unique number associated with it.
  dRootID = ctrlM3DR.GetRootEntityID()

  ' -1 is an invalid entity id.
  If dRootID > -1 Then
    sRootKey = "ID " & dRootID
    Dim nodX As Node ' Declare Node variable.
    Set nodX = ModelList.Nodes.Add(, 0, sRootKey,
ctrlM3DR.GetEntityName(dRootID))
```

```
        BuildTreeRecursive (sRootKey)
    End If

End Sub

Sub BuildTreeRecursive(sParentKey)

    dParentID = Mid(sParentKey, InStr(sParentKey, " "))

    For i = 0 To ctrlM3DR.GetNumberOfChildren(dParentID) - 1
        dChildID = ctrlM3DR.GetChild(dParentID, i)
        sChildKey = "ID " & dChildID
        ModelList.Nodes.Add(sParentKey, 4, sChildKey,
ctrlM3DR.GetEntityName(dChildID))
        BuildTreeRecursive (sChildKey)
    Next

End Sub
```