uc6400 (view source)

Defined in: uf_view.h

Overview

Erases an object from a view.

Return

Flag
0 = OK
1 = View Does Not Exist
2 = Invalid Character In View Name
3 = Invalid Object Type

Environment

Internal and External

Required License(s)

gateway

```
int uc6400
(
    const char * cp1,
    tag_t np2
)
```

const char *	**cp1**	Input	Name Of View (UF_OBJ_NAME_NCHARS character max) ""= Work View
tag_t	**np2**	Input	Object identifier Of Object

---

uc6401 (view source)

Defined in: uf_view.h

Overview

Convert a view dependent object to model

NOTE: An individual ordinate dimension cannot be converted to a model object by itself. You may however convert an ordinate origin and all its associated ordinate dimensions to model objects by setting np2 = object identifier of the ordinate origin.

This user function will convert more object types to model dependent objects than UF_VIEW_convert_to_model().

Return

Flag
0 = OK
1 = View Does Not Exist
2 = Invalid Character In View Name
3 = Object Not View Dependent
4 = Ordinate Dimension cannot be converted
to a model object.

Environment

Internal and External

See Also

UF_VIEW_convert_to_model

Required License(s)

gateway

```
int uc6401
(
    const char * cp1,
    tag_t np2
)
```

const char *	**cp1**	Input	Name Of View (UF_OBJ_NAME_NCHARS character max) ""= Work View
tag_t	**np2**	Input	Object identifier Of Object To Convert To Model

---

uc6402 (view source)

Defined in: uf_view.h

Overview

Convert model object to view dependent

NOTE: An individual ordinate dimension cannot be converted to a view dependent object by itself. You may however convert an ordinate origin and all its associated ordinate dimensions to view dependent objects by setting np2 = object identifier of the ordinate origin.

This user function will convert more object types to view dependent objects than UF_VIEW_convert_to_view().

Return

Flag
0 = OK
1 = View Does Not Exist
2 = Invalid Character In View Name
3 = Object Already View Dependent
4 = Object Not Eligible For View Dependence
5 = Operation not available when work part
and display part are different
6 = Ordinate Dimension cannot be converted
to a view dependent object.

Environment

Internal and External

See Also

UF_VIEW_convert_to_view

Required License(s)

gateway

```
int uc6402
(
    const char * cp1,
    tag_t np2
)
```

const char *	**cp1**	Input	Name Of View (UF_OBJ_NAME_NCHARS character max) ""= Work View
tag_t	**np2**	Input	Object ID Of The Object To Convert To View Dependent

---

uc6403 (view source)

Defined in: uf_view.h

Overview

Removes the view dependent edits from an object.

Return

Flag
0 = OK
1 = View Does Not Exist
2 = Invalid Character In View Name
3 = Object Not View Modified

Environment

Internal and External

Required License(s)

gateway

```
int uc6403
(
    const char * cp1,
    tag_t np2
)
```

const char *	**cp1**	Input	Name of View (UF_OBJ_NAME_NCHARS character max) "" = Work View
tag_t	**np2**	Input	Object identifier of the object

---

uc6404 (view source)

Defined in: uf_view.h

Overview

Converts all view dependent objects to model.

Note: This user function uses the same object filter as that of the view dependent edit functionality in interactive NX.

Return

Flag
0 = OK
1 = View Does Not Exist
2 = Invalid Character In View Name
3 = Object Not View Modified

Environment

Internal and External

Required License(s)

gateway

```
int uc6404
(
    const char * cp1
)
```

const char *	**cp1**	Input	Name Of View (UF_OBJ_NAME_NCHARS character max) ""= Work View

uc6405 (view source)

Defined in: uf_view.h

Overview

Removes all view dependent edits from a view.

Return

Flag
0 = OK
1 = View Does Not Exist
2 = Invalid Character In View Name
3 = No View Modifications Exist

Environment

Internal and External

Required License(s)

gateway

```
int uc6405
(
    const char * cp1
)
```

const char *	**cp1**	Input	Name Of View (UF_OBJ_NAME_NCHARS character max) "" = Work View

uc6406 (view source)

Defined in: uf_view.h

Overview

Modify the COLOR/FONT/WIDTH of a curve in a view.

RP3 is the normalized start and stop parameters for the curve along the normalized curve from 0.0 (beginning) to 1.0 (end). The curve display is modified between the two specified limits.

Return

Flag
0 = OK
1 = View Does Not Exist
2 = Invalid Character In View Name
3 = Invalid Color Number
4 = Invalid Font Number
5 = Invalid Width Number
6 = Invalid Object Type
7 = Data Base Limit ForObject View Mods Full
8 = Attempt To Erase Curve On Full Parameter - Use uc6400

Environment

Internal and External

Required License(s)

gateway

```
int uc6406
(
    const char * cp1,
    tag_t np2,
    double * rp3,
    int ip4,
    int ip5,
    int ip6
)
```

const char *	**cp1**	Input	Name of the view (UF_OBJ_NAME_NCHARS character max) "" = Work View
tag_t	**np2**	Input	Object ID of The object to be modified
double *	**rp3**	Input	Start and end parameters of the modification 0.0 <= RP3[0 ..1] <= 1.0
int	**ip4**	Input	Modification color -1 = Do not change color 1-15 = Color number as defined in uf_obj.h
int	**ip5**	Input	Modification font -1 = Do not change font 0 = Invisible 1-4 = Font number as defined in uf_obj.h
int	**ip6**	Input	Modification width -1 = Do not change width 0-2 = Width Number as defined in uf_obj.h

uc6408 (view source)

Defined in: uf_view.h

Overview

Returns view dependent edit data for an object.
Please use UF_VIEW_ask_vde_data rather than uc6408

Environment

Internal and External

See Also

UF_VIEW_ask_vde_data

Required License(s)

gateway

```
int uc6408
(
    tag_t np1,
    int ip2,
    char cr3 [ UF_OBJ_NAME_BUFSIZE ] ,
    double * rr4,
    int * ir5,
    int * ir6,
    int * ir7
)
```

tag_t	**np1**	Input	Object ID of view dependent object
int	**ip2**	Input	Record number of modification
char	**cr3 [UF_OBJ_NAME_BUFSIZE]**	Output	View name
double *	**rr4**	Output	Parameter Range Of Modification - Range (0.0,1.0) returns two doubles rr4[2]
int *	**ir5**	Output	Color value, see uf_obj.h for values
int *	**ir6**	Output	Font value, see uf_obj.h for values
int *	**ir7**	Output	Width value, see uf_obj.h for values

uc6409 (view source)

Defined in: uf_view.h

Overview

Returns view dependent status of an object and the associated view name when the object is view dependent.

Drafting views will have a @0 appended to the end of the name (ViewName@0).
Modeling views will just conatin the name (ViewName).

Return

Error Code
0 = Success
1 = Invalid Object Type

Environment

Internal and External

Required License(s)

gateway

```
int uc6409
(
    tag_t np1,
    int * ir2,
    char cr3 [ UF_OBJ_NAME_BUFSIZE ]
)
```

tag_t	**np1**	Input	Object Identifier for view dependent status check
int *	**ir2**	Output	Status for the object 0 = Not view dependent 1 = View dependent
char	**cr3 [UF_OBJ_NAME_BUFSIZE]**	Output	View name. This must be a buffer allocated to contain at least UF_OBJ_NAME_BUFSIZE bytes.

---

uc6430 (view source)

Defined in: uf_view.h

Overview

Read View Center and Scale

If cp1 is blank, the work view is used.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name

Environment

Internal and External

Required License(s)

gateway

```
int uc6430
(
   const char * cp1,
   double * rr2,
   double * rr3
)
```

const char *	**cp1**	Input	View Name (UF_OBJ_NAME_NCHARS character max) "" = Work View
double *	**rr2**	Output	View Center (Absolute CSYS) x,y and z coordinates.
double *	**rr3**	Output	View Scale

uc6431 (view source)

Defined in: uf_view.h

Overview

Set View Center and Scale

If cp1 is blank, the work view is updated. The scale must be greater than 0.0. An invalid scale is an error and does not change the view center. uc6431 can edit a non-active view.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name
31 = Invalid Scale

Environment

Internal and External

Required License(s)

gateway

```
    int uc6431
    (
        const char * cp1,
        double * rp2,
        double rp3
    )
```

const char *	**cp1**	Input	View Name (UF_OBJ_NAME_NCHARS character max) "" = Work View
double *	**rp2**	Input	View Center (Absolute CSYS), in x,y, and z coordinates
double	**rp3**	Input	View Scale 0.0 = Use Existing Scale

uc6432 (view source)

Defined in: uf_view.h

Overview

Fit the View(s)

If cp1 is blank and ip2=1, the work view is updated. If ip2=2, the value in cp1 is ignored.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name
9 = View Not Active

Environment

Internal and External

Required License(s)

gateway

```
int uc6432
(
    const char * cp1,
    int ip2
)
```

const char *	**cp1**	Input	View name (UF_OBJ_NAME_NCHARS character max) "" = Work view
int	**ip2**	Input	Fit option 1 = View named in cp1 2 = All active views

---

uc6433 (view source)

Defined in: uf_view.h

Overview

Read View Matrix

If cp1 is blank, the work view is used.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name

Environment

Internal and External

Required License(s)

gateway

```
int uc6433
(
    const char * cp1,
    double * rr2
)
```

const char *	**cp1**	Input	View name (UF_OBJ_NAME_NCHARS character max) "" = Work view
double *	**rr2**	Output	View matrix (Nine element array)

uc6434 (view source)

Defined in: uf_view.h

Overview

Set View Matrix

If cp1 is blank, the work view is updated. The object Identifier in np3 must reference an object that has a coordinate system associated with it, such as an arc, conic, drafting object, CSYS, etc.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name
3 = View is a Drawing View or Drawing Member View
9 = View Not Active
31 = Invalid Type Object In np3
32 = Invalid Matrix In rp4
33 = Invalid Matrix Option

Environment

Internal and External

Required License(s)

gateway

```
int uc6434
(
    const char * cp1,
    int ip2,
    tag_t np3,
    double rp4 [ 6 ]
)
```

const char *	**cp1**	Input	View name (UF_OBJ_NAME_NCHARS character max) "" = Work view
int	**ip2**	Input	Matrix option 1 = Absolute csys 2 = WCS

			3 = Use CSYS of object in np3 4 = Use matrix in rp4
tag_t	**np3**	Input	Object identifier (If ip2=3)
double	**rp4 [6]**	Input	This argument is used when ip2 is 4. In this case, this will be the X-axis and Y-axis of the matrix (6 element array). rp4[0..2] will be the X-axis values and rp4[3..5] will be the Y-Axis values. The Z axis of the view matrix will be internally calculated by taking the cross product of the two input axis.

uc6435 (view source)

Defined in: uf_view.h

Overview

Rotate View Point around View Center

If cp1 is blank, the work view is updated. A compound rotation can be accomplished by setting several values of rp2 to non-zero. Negative angles are valid. The rotations are executed in the order of rp2[0], rp2[1], rp2[2].

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name
9 = View Not Active

Environment

Internal and External

Required License(s)

gateway

```
int uc6435
(
    const char * cp1,
    double * rp2
)
```

const char *	**cp1**	Input	View Name (UF_OBJ_NAME_NCHARS character max) "" = Work View
double *	**rp2**	Input	Rotation angles in degrees. [0] Down [1] Right [2] Counter-clockwise

uc6436 (view source)

Defined in: uf_view.h

Overview

Read View Clipping Planes

If cp1 is blank, the work view is used. The values of rr3 are not returned if the corresponding values of IR2 = 1.
NOTE: "Clipping Enabled by System" means NX
automatically detects that clipping planes are required and turns them on.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name

Environment

Internal and External

Required License(s)

gateway

```
int uc6436
(
    const char * cp1,
    int * ir2,
    double * rr3
)
```

const char *	**cp1**	Input	View Name (UF_OBJ_NAME_NCHARS character max) "" = Work View
int *	**ir2**	Output	Clipping Plane Status ir[2] [0] Front 1 = Clipping Disabled 2 = Clipping Enabled by User 3 = Clipping Enabled by System [1] Back 1 = Clipping Disabled 2 = Clipping Enabled by User 3 = Clipping Enabled by System
double *	**rr3**	Output	Clipping Plane Distance from the view origin [0] Front [1] Back

---

uc6437 (view source)

Defined in: uf_view.h

Overview

Set View Clipping Planes

If cp1 is blank, the work view is updated. The values of rp3 are ignored if the corresponding clipping plane is disabled (ip2) or auto-set. Using auto-set calculates the plane to be the minimum (for front plane) or maximum (for back plane) distance that displays all objects.

You cannot disable clipping planes in a view that has 3D vectors (see Uc6440 - Read view display type), and you cannot disable the front clipping plane in perspective views (see uc6439 - Set view perspective).

If the projection type is "PERSPECTIVE" and the front Z clipping plane is in front of the EYE POINT (see uc6438 - Read view perspective), the front Z clipping plane moves to the EYE POINT. Return Codes of 33 and 34 are warnings only.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name
3 = View Has 3D Vectors - Can Not Remove Clipping Planes
9 = View Not Active
31 = Front Plane Behind Back Plane
33 = Perspective View - Can Not Remove Front Plane
34 = Perspective View - Front Plane Moved To Eye Point

Environment

Internal and External

Required License(s)

gateway

```
int uc6437
(
    const char * cp1,
    int * ip2,
    double * rp3
)
```

const char *	**cp1**	Input	View name (UF_OBJ_NAME_NCHARS character max) "" = Work view
int *	**ip2**	Input	Clipping plane status [0] Front [1] Back 0 = Do not change status/distance 1 = Disable clipping 2 = Enable clipping 3 = Auto-set distance
double *	**rp3**	Input	Clipping plane distance (If ip2[x] = 2) (From view origin) [0] Front [1] Back

uc6438 (view source)

Defined in: uf_view.h

Overview

Read View Perspective

If cp1 is blank, the work view is used. rr3 is modified only if the projection type is "perspective" (ir2=2).

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name

Environment

Internal and External

Required License(s)

gateway

```
int uc6438
(
    const char * cp1,
    int * ir2,
    double * rr3
)
```

const char *	**cp1**	Input	View name (UF_OBJ_NAME_NCHARS character max) "" = Work View
int *	**ir2**	Output	Projection type 1 = Parallel 2 = Perspective
double *	**rr3**	Output	Perspective distance (If ir2 = 2)

---

uc6439 (view source)

Defined in: uf_view.h

Overview

Set View Perspective

If cp1 is blank, the work view is updated. The value of rp3 is used only if ip2=2. The value of rp4 is used only if ip2=3.
If projection type is "PERSPECTIVE" and the EYE POINT is set to be behind the current front Z clipping plane, the front Z clipping plane is moved equal to the EYE POINT.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name
3 = Drafting View Not Valid
9 = View Not Active
31 = Invalid Distance
32 = Invalid Projection Option

Environment

Internal and External

Required License(s)

gateway

```
int uc6439
(
    const char * cp1,
    int ip2,
    double rp3,
    double * rp4
)
```

const char *	**cp1**	Input	View name (UF_OBJ_NAME_NCHARS character max) "" = Work View
int	**ip2**	Input	Projection Option 1 = Make view parallel 2 = Make view perspective 3 = Change eye point
double	**rp3**	Input	Perspective distance (If ip2=2)
double *	**rp4**	Input	X, Y and Z coordinates of the Eye Point (Absolute CSYS) (If ip2=3)

uc6440 (view source)

Defined in: uf_view.h

Overview

Read view display type

If cp1 is blank, the work view is used.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name

Environment

Internal and External

Required License(s)

gateway

```
int uc6440
(
    const char * cp1,
    int * ir2
)
```

const char *	**cp1**	Input	View name (UF_OBJ_NAME_NCHARS character max) "" = Work view
int *	**ir2**	Output	Display type 2 = 2D display vectors 3 = 3D display vectors

uc6442 (view source)

Defined in: uf_view.h

Overview

Read View Drawing Parameters

If cp1 is blank, the work view is used.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name
3 = View Is A Drawing

Environment

Internal and External

Required License(s)

gateway

```
int uc6442
(
    const char * cp1,
    double rr2 [ 3 ] ,
    double * rr3
)
```

const char *	**cp1**	Input	View name (UF_OBJ_NAME_NCHARS character max) "" = Work View
double	**rr2 [3]**	Output	X, Y and Z of drawing reference point, (Absolute CSYS)
double *	**rr3**	Output	Drawing scale

uc6443 (view source)

Defined in: uf_view.h

Overview

Set view drawing parameters

If cp1 is blank, the work view is updated. The reference point and scale are used to place a view on a drawing. The scale must be greater than 0.0. UF6443 can edit a non-active view.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name
3 = View Is A Drawing
4 = Not allowed in "Work in member view" mode
31 = Invalid Scale

Environment

Internal and External

Required License(s)

gateway

```
int uc6443
(
    const char * cp1,
    double * rp2,
    double rp3
)
```

const char *	**cp1**	Input	View name (UF_OBJ_NAME_NCHARS character max) "" = Work View
double *	**rp2**	Input	X, Y and Z coordinates of the drawing reference point in absolute coordinates.
double	**rp3**	Input	Drawing scale 0.0 = No change

uc6444 (view source)

Defined in: uf_view.h

Overview

Read Surface/Solid Display Parameters

Please use
UF_VIEW_ask_surface_display_options
and
UF_VIEW_ask_fog_options
instead.

If cp1 is blank, the work view is used.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name

Environment

Internal and External

Required License(s)

gateway

```
int uc6444
(
    const char * cp1,
    int * ir2,
    int * ir3,
    int * ir4,
    int * ir5,
    int * ir6,
    int * ir7
)
```

const char *	**cp1**	Input	View Name (UF_OBJ_NAME_NCHARS character max) "" = Work View
int *	**ir2**	Output	Surface/Solid Face Display 1 = Grid Lines 2 = Solid Fill 3 = Flat Shading 4 = Gouraud Shading 5 = Phong Shading 6 = Hidden Surface
int *	**ir3**	Output	Depth Cueing 1 = Off 2 = On
int *	**ir4**	Output	Solid Unfixed Blends 1 = Visible 2 = Invisible
int *	**ir5**	Output	Solid Smooth Edges 1 = Visible 2 = Invisible
int *	**ir6**	Output	Solid Silhouettes 1 = Visible 2 = Invisible
int *	**ir7**	Output	Solid Hidden Edges 1 = Visible 2 = Invisible 3 = Dashed

uc6445 (view source)

Defined in: uf_view.h

Overview

Set Surface/Solid Display Parameters

Please use
UF_VIEW_set_surface_display_options

and
UF_VIEW_set_fog_options
instead.

If cp1 is blank, the work view is updated.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name
3 = Drafting View Not Valid
31 = Invalid Parameter

Environment

Internal and External

Required License(s)

gateway

```
int uc6445
(
    const char * cp1,
    int ip2,
    int ip3,
    int ip4,
    int ip5,
    int ip6,
    int ip7
)
```

const char *	**cp1**	Input	View Name (UF_OBJ_NAME_NCHARS character max) "" = Work View
int	**ip2**	Input	Surface/Solid Face Display 0 = Use Current Mode 1 = Grid Lines 2 = Solid Fill 3 = Flat Shading 4 = Gouraud Shading 5 = Phong Shading 6 = Hidden Surface
int	**ip3**	Input	Depth Cueing 0 = Use Current Mode 1 = Off 2 = On
int	**ip4**	Input	Solid Unfixed Blends 0 = Use Current Mode 1 = Visible 2 = Invisible
int	**ip5**	Input	Solid Smooth Edges 0 = Use Current Mode 1 = Visible 2 = Invisible
int	**ip6**	Input	Solid Silhouettes 0 = Use Current Mode 1 = Visible 2 = Invisible

int	**ip7**	Input	Solid Hidden Edges 0 = Use Current Mode 1 = Visible 2 = Invisible 3 = Dashed

uc6446 (view source)

Defined in: uf_view.h

Overview

Read View Layer Visibility Mask

If cp1 is blank, the work view is used. ir3 is returned only if ir2=2. For example, if an individual layer visibility mask exists for this view.
NOTE: This routine does not read the global layer mask.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name

Environment

Internal and External

Required License(s)

gateway

```
int uc6446
(
    const char * cp1,
    int * ir2,
    int * ir3
)
```

const char *	**cp1**	Input	View Name (UF_OBJ_NAME_NCHARS character max) "" = Work View
int *	**ir2**	Output	Mask Status 1 = Global 2 = Individual
int *	**ir3**	Output	256 word layer mask, indexed by layer number +1. So ir3[0] has the status of layer 1, ir3[1] has the status of layer 2 and so on. Each element of the array can have one of the following two values. 0 = Invisible In View 1 = Visible In View

uc6447 (view source)

Defined in: uf_view.h

Overview

Set View Layer Visibility Mask

If cp1 is blank, the work view is used. If ip2=1, any existing view layer visibility mask is deleted for this view. If ip2=2, a new layer visibility mask is created if none exists. ip3 is used only if ip2=2. For example, if an individual layer visibility mask is to be used for this view. UF6447 can edit a non-active view.
NOTE: This routine does not affect the global layer mask (see routines UF500x).

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name
31 = Invalid Mask Status

Environment

Internal and External

Required License(s)

gateway

```
int uc6447
(
    const char * cp1,
    int ip2,
    int * ip3
)
```

const char *	**cp1**	Input	View Name (UF_OBJ_NAME_NCHARS character max) "" = Work View
int	**ip2**	Input	Mask Status 1 = Global 2 = Individual
int *	**ip3**	Input	256 word layer mask, indexed by layer number +1. So ip3[0] has the status of layer 1, ip3[1] has the status of layer 2 and so on. Each element of the array can have one of the following two values. 0 = Invisible In View 1 = Visible In View

---

uc6448 (view source)

Defined in: uf_view.h

Overview

Read Work View Name
Please use UF_VIEW_ask_work_view rather than uc6448.

Return

work view name.

Environment

Internal and External

Required License(s)

gateway

```
void uc6448
(
    char cr1 [ UF_OBJ_NAME_BUFSIZE ]
)
```

char	**cr1 [UF_OBJ_NAME_BUFSIZE]**	Output	View Name

---

uc6449 (view source)

Defined in: uf_view.h

Overview

Change Work View

This function has no effect if a drawing is current.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name
9 = View Not Active

Environment

Internal and External

Required License(s)

gateway

```
int uc6449
(
    const char * cp1
)
```

const char *	**cp1**	Input	View Name (UF_OBJ_NAME_NCHARS character max)

---

uc6450 (view source)

Defined in: uf_view.h

Overview

Save View

cp1 is the name of the view to save. If cp1 is blank, the work view is saved.
cp2 is the name to save the view as. If cp2 is blank, the view is saved with the same name. If cp2 is not blank and a view of that name does not exist, a new view is created. If cp2 is not blank and a view of that name already exists, an error is returned.
ip3 and ip4 are used only if the view is saved with a new name (i.e., cp2 <>"" and cp2<> cp1).
NOTE: The view orientation is not changed for a "canned" view.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name
4 = View Already Exists
9 = View Not Active
10 = Operation Not Allowed in "Work in member View" mode
11 = Can't Save View because drawing is currently displayed
23 = Can't Save Drawing View With New Name

Environment

Internal and External

Required License(s)

gateway

```
int uc6450
(
    const char * cp1,
    const char * cp2,
    const int ip3,
    const int ip4
)
```

const char *	**cp1**	Input	View Name (UF_OBJ_NAME_NCHARS character max) "" = Work View
const char *	**cp2**	Input	Name To Save View As (UF_OBJ_NAME_NCHARS character max) "" = Current Name
const int	**ip3**	Input	Copy View Modifications 0 = No 1 = Yes
const int	**ip4**	Input	Move View Dependent Objects 0 = No 1 = Move

uc6454 (view source)

Defined in: uf_view.h

Overview

Delete View

Many views cannot be deleted. Among them are the "canned" views and work view. If a view contains a drawing, it cannot be deleted. In addition, any view that belongs to a layout cannot be deleted.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name
3 = Cannot Delete Canned View
4 = Cannot Delete Work View
5 = Cannot Delete Drawing View
6 = Cannot delete view. View belongs to at least one layout.
8 = Cannot delete view. View belongs to at least one layout.

Environment

Internal and External

Required License(s)

gateway

```
int uc6454
(
    const char * cp1
)
```

const char *	**cp1**	Input	View Name (UF_OBJ_NAME_NCHARS character max)

---

uc6455 (view source)

Defined in: uf_view.h

Overview

Rename View
Please use UF_VIEW_rename rather than uc6455.
If cp1 is blank, the work view is renamed. The "canned" views cannot be renamed. In addition, modeling views cannot be renamed as drawing view names and vice versa.

Return

Return Code
0 = OK
1 = Old View Does Not Exist
2 = Invalid View Name
3 = Can Not Rename Canned View
4 = New View Already Exists
5 = Can Not Rename Drawing View

Environment

Internal and External

Required License(s)

gateway

```
int uc6455
(
    const char * cp1,
    const char * cp2
)
```

const char *	**cp1**	Input	Old View Name (UF_OBJ_NAME_NCHARS character max) ""= Work View
const char *	**cp2**	Input	New View Name (UF_OBJ_NAME_NCHARS character max)

---

uc6456 (view source)

Defined in: uf_view.h

Overview

Cycle Views in Part
Please use UF_OBJ_cycle_objs_in_part with type UF_view_type rather than uc6456
This routine cycles the part and returns one view name per cycle. To
start the cycle, set CA1 to an empty string (""). The first
view name is returned in CA1. The next call returns the second view
name in CA1. When CA1 is returned with an empty string (ca1 returns ""),
all view names have been returned. View or drawing names that are
28 characters or less are appended with an "@n" string (where n is
any positive integer including zero). If your view name is more than 28
characters, the "@n" string is truncated.
NOTE: The cycle can be continued even if the view named in CA1 is
deleted.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name

Environment

Internal and External

Required License(s)

gateway

```
int uc6456
(
    char ca1 [ UF_OBJ_NAME_BUFSIZE ] ,
    int * ir2
)
```

char	**ca1 [UF_OBJ_NAME_BUFSIZE]**	Input / Output	On input this is the view name returned by the last call to this routine. Pass in an empty string, "", to start cycling.

			On output this is the next view name. When all view names have been cycled, and empty string, "", will be returned. The calling program must allocate a UF_OBJ_NAME_BUFSIZE buffer to hold the view name.
int *	**ir2**	Output	Active View Flag 0 = View Inactive 1 = View Active

uc6457 (view source)

Defined in: uf_view.h

Overview

Cycle Objects in View
Please use UF_VIEW_cycle_objects rather than uc6457
If cp1 is blank, the work view is used. ip2=1 returns all objects which are visible in the view. Objects which are out of the view bounds are not returned.

Different types of cycles can be intermingled. For instance, while cycling view dependent objects, you can also cycle objects erased in a view. You cannot delete or un-erase objects or view modifications during a cycle.

Warning: This function will return curves that are used to display a solid silhouette in a drawing member view. Use UF_DRAW_ask_group_of_curve on any curve returned to determine if the curve belongs to a UF_solid_silhouette_type group.

NOTE: return/IR4=9 is valid only if ip2=1.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Name
9 = View Not Active
31 = Invalid Selection Option

Environment

Internal and External

Required License(s)

gateway

```
int uc6457
(
    const char * cp1,
    int ip2,
    tag_t * na3
)
```

const char *	**cp1**	Input	View Name (UF_OBJ_NAME_NCHARS character max) "" Work View
int	**ip2**	Input	Object Selection 1 = Visible Objects 2 = View Dependent Objects 3 = Objects Erased In View 4 = Objects Modified In View
tag_t *	**na3**	Input / Output	On input, the last object found by this routine. Pass in a NULL_TAG to start the cycle. On output the next object. A NULL_TAG is returned when the cycle is complete.

UF_VIEW_add_to_view_set (view source)

Defined in: uf_view.h

Overview

Routine -
UF_VIEW_add_to_view_set

Description -
Add an orthographic view to view set

Return Codes -
ERROR_OK
UF_VIEW_INVALID_OBJECT
UF_VIEW_INVALID_VIEW_TAG
UF_VIEW_INVALID_ORIENTATION_TYPE

See Also -
UF_VIEW_create_view_set
UF_VIEW_remove_from_view_set
UF_VIEW_ask_view_set_by_name
UF_VIEW_ask_base_view_of_view_set
UF_VIEW_set_base_view_of_view_set
UF_VIEW_ask_views_of_view_set
UF_VIEW_delete_view_set

Environment

Internal and External

History

This routine is originally created in release NX3.0

Required License(s)

gateway

```
int UF_VIEW_add_to_view_set
(
    tag_t view_set_tag,
    UF_VIEW_standard_orientation_t type
)
```

tag_t	**view_set_tag**	Input	The view set to add

UF_VIEW_standard_orientation_t	**type**	Input	The orthographic view type to add

---

UF_VIEW_ask_base_view_of_view_set (view source)

Defined in: uf_view.h

Overview

Routine -
UF_VIEW_ask_base_view_of_view_set

Description -
Query the base view of the given a view set

Return Codes -
ERROR_OK
UF_VIEW_INVALID_OBJECT
UF_VIEW_INVALID_VIEW_TAG

See Also -
UF_VIEW_create_view_set
UF_VIEW_add_to_view_set
UF_VIEW_remove_from_view_set
UF_VIEW_ask_view_set_by_name
UF_VIEW_set_base_view_of_view_set
UF_VIEW_ask_views_of_view_set
UF_VIEW_delete_view_set

Environment

Internal and External

History

This routine is originally created in release NX3.0

Required License(s)

gateway

```
int UF_VIEW_ask_base_view_of_view_set
(
    tag_t view_set_tag,
    tag_t * base_view_tag
)
```

tag_t	**view_set_tag**	Input	The view set
tag_t *	**base_view_tag**	Output	The base view of the view set

---

UF_VIEW_ask_center (view source)

Defined in: uf_view.h

Overview

Ask View Center

If view is NULL_TAG, the work view is used.

The view center returned by this function is a point on the view plane at the center of the viewing volume. If the viewing volume is symmetric (as it usually is), then this point is also the view origin.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Tag - can not find view object from given view tag
3 = center pointer is NULL

Environment

Internal and External

See Also

UF_VIEW_set_center
uc6430

Required License(s)

gateway

```
int UF_VIEW_ask_center
(
    tag_t tag,
    double center [ 3 ]
)
```

tag_t	**tag**	Input	view tag
double	**center [3]**	Output	View Center (Absolute CSYS) x,y and z coordinates.

UF_VIEW_ask_current_xy_clip (view source)

Defined in: uf_view.h

Overview

Returns the current X and Y clip boundaries of the given view. The clip boundaries define a rectangle on the Z = 0 plane of View Space.

Please note the following:
The clip boundary returned is different to the one returned by UF_VIEW_ask_xy_clip. UF_VIEW_ask_xy_clip return the boundary saved in the data base for the view. UF_VIEW_ask_current_xy_clip return the current boundary.

If the view is in the current layout, has been changed in the layout (by, for example, a Zoom), and has not been saved since it was changed, then the clip boundaries returned by this function will reflect the changes.

This function is unusual in that, unlike most Open API functions, the coordinate space is View Space. View Space is parallel to the screen, with the center of the view at (0,0,0). The view center is mapped from Absolute Space to View Space by applying first the view rotation, then the view translation to it. The scale in View Space is the same as in Absolute Space.

Environment

Internal

See Also

UF_VIEW_ask_xy_clip

History

Original release was in NX5.0.

Required License(s)

gateway

```
int UF_VIEW_ask_current_xy_clip
(
    tag_t view_tag,
    double xy_clip_bounds [ 4 ]
)
```

tag_t	**view_tag**	Input	Tag of the view whose X-Y clip bounds are needed
double	**xy_clip_bounds [4]**	Output	Array of the bounds of the view, in view space (Minimum_X, Maximum_X, Minimum_Y, Maximum_Y)

UF_VIEW_ask_fog_options (view source)

Defined in: uf_view.h

Overview

Returns the fog options of the specified view. The view must have type UF_VIEW_MODEL_TYPE.

Environment

Internal and External

See Also

UF_VIEW_set_fog_options
UF_VIEW_fog_options_t

History

Originally released in NX4.0.

Required License(s)

gateway

```
int UF_VIEW_ask_fog_options
(
    tag_t view_tag,
    UF_VIEW_fog_options_p_t fog_options
)
```

tag_t	**view_tag**	Input	The view tag; if NULL_TAG, the work view is used.
UF_VIEW_fog_options_p_t	**fog_options**	Output	The fog options of the view.

UF_VIEW_ask_perspective (view source)

Defined in: uf_view.h

Overview

Ask View Perspective

If view is NULL_TAG, the work view is used. distance is modified only if the projection type is "perspective" (type = 2).

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Tag - can not find view object from given view tag
3 = Either type or distance pointer is NULL

Environment

Internal and External

See Also

UF_VIEW_set_perspective
uc6438

Required License(s)

gateway

```
int UF_VIEW_ask_perspective
(
    tag_t tag,
    int * type,
    double * distance
)
```

tag_t	**tag**	Input	view tag
int *	**type**	Output	Projection type 1 = Parallel 2 = Perspective
double *	**distance**	Output	Perspective distance (If type = 2)

UF_VIEW_ask_rotation (view source)

Defined in: uf_view.h

Overview

Ask View matrix

If view is NULL_TAG, the work view is used.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Tag - can not find view object from given view tag
3 = matrix pointer is NULL

Environment

Internal and External

See Also

UF_VIEW_set_rotation
uc6433

Required License(s)

gateway

```
int UF_VIEW_ask_rotation
(
    tag_t tag,
    double matrix [ 9 ]
)
```

tag_t	**tag**	Input	view tag
double	**matrix [9]**	Output	View rotation matrix (nine element array) matrix[0..2] will be the X-axis of the view matrix[3. 5] will be the Y-axis matrix[6..8] will be the Z-axis

---

UF_VIEW_ask_shaded_edge_options (view source)

Defined in: uf_view.h

Overview

Returns the shaded edge options of the specified view. The view must have type UF_VIEW_MODEL_TYPE.

Environment

Internal and External

See Also

UF_VIEW_set_shaded_edge_options
UF_VIEW_shaded_edge_options_t

History

Originally released in NX4.0.

Required License(s)

gateway

```
int UF_VIEW_ask_shaded_edge_options
(
    tag_t view_tag,
    UF_VIEW_shaded_edge_options_p_t shaded_edge_options
)
```

tag_t	**view_tag**	Input	The view tag; if NULL_TAG, the work view is used.
UF_VIEW_shaded_edge_options_p_t	**shaded_edge_options**	Output	The shaded edge options of the view.

UF_VIEW_ask_surface_display_options (view source)

Defined in: uf_view.h

Overview

Returns the rendering style and the Edge display options of
the specified view. The view must have type UF_VIEW_MODEL_TYPE.

Environment

Internal and External

See Also

UF_VIEW_set_surface_display_options
UF_VIEW_rendering_style_t
UF_VIEW_edge_display_options_t

History

Originally released in NX4.0.

Required License(s)

gateway

```
int UF_VIEW_ask_surface_display_options
(
    tag_t view_tag,
    UF_VIEW_rendering_style_p_t rendering_style,
    UF_VIEW_edge_display_options_p_t edge_display_options
)
```

tag_t	**view_tag**	Input	The view tag; if NULL_TAG, the work view is used.
UF_VIEW_rendering_style_p_t	**rendering_style**	Output	The rendering style of the view.
UF_VIEW_edge_display_options_p_t	**edge_display_options**	Output	The edge display options of the view.

UF_VIEW_ask_tag_of_view_name (view source)

Defined in: uf_view.h

Overview

Gets the tag of an existing view that you specify by name.

Environment

Internal and External

Required License(s)

gateway

```
int UF_VIEW_ask_tag_of_view_name
(
    const char * view_name,
    tag_t * view_tag
)
```

const char *	**view_name**	Input	The view name (UF_OBJ_NAME_NCHARS character max); if blank, work view is used.
tag_t *	**view_tag**	Output	The corresponding view tag

UF_VIEW_ask_type (view source)

Defined in: uf_view.h

Overview

This routine retrieves the type of the specified view.

Environment

Internal and External

Required License(s)

gateway

```
int UF_VIEW_ask_type
(
    tag_t view_tag,
    UF_VIEW_type_t * type,
    UF_VIEW_subtype_t * subtype
)
```

tag_t	**view_tag**	Input	The specified view tag.

UF_VIEW_type_t *	**type**	Output	The view type.
			UF_VIEW_MODEL_TYPE = 0,
			UF_VIEW_DRAWING_MEMBER_TYPE = 1,
			UF_VIEW_DRAWING_SHEET_TYPE = 2

UF_VIEW_subtype_t *	**subtype**	Output	The view subtype - drawing member views only.
			UF_VIEW_INVALID_SUBTYPE = -1,
			UF_VIEW_SECTION_SUBTYPE = 0,
			UF_VIEW_IMPORTED_SUBTYPE = 1,
			UF_VIEW_BASE_MEMBER_SUBTYPE = 2,

	UF_VIEW_ORTHOGONAL_SUBTYPE = 3,
	UF_VIEW_AUXILIARY_SUBTYPE = 4,
	UF_VIEW_DETAIL_SUBTYPE = 5,
	UF_VIEW_BREAK_SUBTYPE = 6

UF_VIEW_ask_vde_data (view source)

Defined in: uf_view.h

Overview

Retrieves the number of view dependent edits for an object, and the corresponding data for each modification.

Environment

Internal and External

See Also

For example please refer to the example

Required License(s)

gateway

```
int UF_VIEW_ask_vde_data
(
    tag_t object,
    int * number_edits,
    UF_VIEW_vde_data_p_t * vde_data
)
```

tag_t	**object**	Input	Tag of object
int *	**number_edits**	Output	Number of view dependent edits for the object.
UF_VIEW_vde_data_p_t *	**vde_data**	Output to UF_*free*	Array of vde structures containing the view, color, font, width, and range of each of the view dependent edits. Use UF_free() to free the memory.

UF_VIEW_ask_vde_data_with_type (view source)

Defined in: uf_view.h

Overview

Retrieve the number of view dependent edits for an object and the corresponding edit types

Environment

Internal and External

Required License(s)

gateway

```
int UF_VIEW_ask_vde_data_with_type
(
    tag_t object,
    int * number_edits,
    UF_VIEW_vde_data_and_type_p_t * vde_data
)
```

tag_t	**object**	Input	Tag of object
int *	**number_edits**	Output	Number of view dependent edits for object
UF_VIEW_vde_data_and_type_p_t *	**vde_data**	Output to UF_*free*	Array of vde type of each of the view dependent edits; use UF_free () to free memory

UF_VIEW_ask_view_light (view source)

Defined in: uf_view.h

Overview

This function returns the lighting attributes of a single specified view. Each view can have different lights assigned to it. The lights are now shared between the NX hardware shading and Photo shading.

If no lights were found with the part, the following 2 default lights are turned on in each view to ensure that shaded display is lit:
1. Light name: Default 1 Ambient
Light type: UF_VIEW_AMBIENT_LIGHT
2. Light name : Default 2 Up-Front Dist
Light type: UF_VIEW_DISTANT_LIGHT
Light mode: UF_VIEW_FIXED_TO_OBSERVER
Light Location: 0.8, 0.0, 0.4
Light to: 0.0, 0.0, 0.0

Environment

Internal and External

See Also

UF_VIEW_set_view_light

Required License(s)

gateway

```
int UF_VIEW_ask_view_light
(
    tag_t view,
    UF_VIEW_lighting_t * view_light
)
```

tag_t	**view**	Input	Tag of view
UF_VIEW_lighting_t *	**view_light**	Output	Data structure containing view lighting parameters

UF_VIEW_ask_view_set_by_name (view source)

Defined in: uf_view.h

Overview

Routine -
UF_VIEW_ask_view_set_by_name

Description -
Query the view set by the given name

Note -
The query is only against the current displayed part. If there is no view set with the given name, a NULL_TAG is returned.

Return Codes -
ERROR_OK
UF_VIEW_INVALID_NAME

See Also -
UF_VIEW_create_view_set
UF_VIEW_add_to_view_set
UF_VIEW_remove_from_view_set
UF_VIEW_ask_base_view_of_view_set
UF_VIEW_set_base_view_of_view_set
UF_VIEW_ask_views_of_view_set
UF_VIEW_delete_view_set

Environment

Internal and External

History

This routine is originally created in release NX3.0

Required License(s)

gateway

```
int UF_VIEW_ask_view_set_by_name
(
    const char * name,
    tag_t * view_set
)
```

const char *	**name**	Input	The view set name
tag_t *	**view_set**	Output	The view set match the name

UF_VIEW_ask_views_of_view_set (view source)

Defined in: uf_view.h

Overview

Routine -
UF_VIEW_ask_views_of_view_set

Description -
Query the list of views in a view set

Return Codes -
ERROR_OK
UF_VIEW_INVALID_OBJECT
UF_VIEW_INVALID_VIEW_TAG

See Also

UF_VIEW_create_view_set
UF_VIEW_add_to_view_set
UF_VIEW_remove_from_view_set
UF_VIEW_ask_view_set_by_name
UF_VIEW_ask_base_view_of_view_set
UF_VIEW_set_base_view_of_view_set
UF_VIEW_delete_view_set

Environment

Internal and External

History

This routine is originally created in release NX3.0

Required License(s)

gateway

```
int UF_VIEW_ask_views_of_view_set
(
    tag_t view_set_tag,
    int * num_views,
    tag_p_t * views_in_set
)
```

tag_t	**view_set_tag**	Input	The view set
int *	**num_views**	Output	The number of views in the set
tag_p_t *	**views_in_set**	Output to UF_*free*	The list of views in the set

UF_VIEW_ask_visible_objects (view source)

Defined in: uf_view.h

Overview

Returns visible objects in a view broken down by those which are entirely visible within the view and those which are visible but clipped by the view boundary (i.e those which cross the view boundary).

Note that this function is primarily designed to work with drafting member views. While it may be used with model views or drawing sheet views, the bounds of those latter views are not considered in the outcome. The bounds of drafting member views, including those with non-rectangular borders are considered.

If view is NULL_TAG, the work view is used.

Warning: When run in external mode, this function will return solids as visible which are occluded by other solids in the specified view. For a drafting member view, extracted edges may be used to force occlusion to be considered.

Environment

Internal and External

See Also

UF_VIEW_cycle_objects

History

Originally released in V19.0

Required License(s)

gateway

```
int UF_VIEW_ask_visible_objects
(
    tag_t view,
    int * n_visible,
    tag_t * * visible,
    int * n_clipped,
    tag_t * * clipped
)
```

tag_t	**view**	Input	The view to cycle in - if NULL_TAG, the work view is used
int *	**n_visible**	Output	The number of entirely visible objects
tag_t * *	**visible**	Output to UF_*free*	The list of entirely visible objects
int *	**n_clipped**	Output	The number of objects which cross the view boundary, yet some portion is visible
tag_t * *	**clipped**	Output to UF_*free*	The list of clipped objects

UF_VIEW_ask_visualization (view source)

Defined in: uf_view.h

Overview

Gets all of the visualization data. If you pass a NULL_TAG to the view argument, then the work view is used.

Please use
UF_VIEW_ask_surface_display_options
and
UF_VIEW_ask_fog_options

instead.

Environment

Internal and External

See Also

UF_VIEW_visualization_t

History

In the V15.0 release, new fields were added to the
UF_VIEW_visualization_t data structure.
In NX4.0, this function is superceded by
UF_VIEW_ask_surface_display_options and UF_VIEW_ask_fog_options.

Required License(s)

gateway

```
int UF_VIEW_ask_visualization
(
    tag_t view,
    UF_VIEW_visualization_t * view_data
)
```

tag_t	**view**	Input	View to retrieve data on
UF_VIEW_visualization_t *	**view_data**	Output	View visualization data

UF_VIEW_ask_work_view (view source)

Defined in: uf_view.h

Overview

This routine retrieves the work view tag.

Environment

Internal and External

See Also

uc6448

History

This function was originally released in V16.0.

Required License(s)

gateway

```
int UF_VIEW_ask_work_view
(
    tag_t * work_view
)
```

tag_t *	**work_view**	Output	Tag of work view

UF_VIEW_ask_xy_clip (view source)

Defined in: uf_view.h

Overview

Returns the X and Y clip boundaries of the given view. The clip boundaries define a rectangle on the Z = 0 plane of View Space.

This function is primarily intended for use by specialized applications such as Translators. Please note the following:
The clip boundaries returned are those saved in the data base for the view. If the view is in the current layout, has been changed in the layout (by, for example, a Zoom), and has not been saved since it was changed, then the clip boundaries returned by this function do NOT reflect the changes.

If this is not what you need, consider using uc6431 (Read View Center and Scale), which returns the current data for the view in the layout, if it is in the layout, or otherwise returns the data for the view from the data base.

This function is unusual in that, unlike most Open API functions, the coordinate space is View Space. View Space is parallel to the screen, with the center of the view at (0,0,0). The view center is mapped from Absolute Space to View Space by applying first the view rotation, then the view translation to it. The scale in View Space is the same as in Absolute Space.

Environment

Internal and External

See Also

UF_VIEW_set_xy_clip

Required License(s)

gateway

```
int UF_VIEW_ask_xy_clip
(
    tag_t view_tag,
    double xy_clip_bounds [ 4 ]
)
```

tag_t	**view_tag**	Input	Tag of the view whose X-Y clip bounds are needed
double	**xy_clip_bounds [4]**	Output	Array of the bounds of the view, in view space (Minimum_X, Maximum_X, Minimum_Y, Maximum_Y)

UF_VIEW_ask_Z_clip (view source)

Defined in: uf_view.h

Overview

Ask View Z-Clipping Planes

If view is NULL_TAG, the work view is used.
The values of distance are not returned if the corresponding values of status = 1.
NOTE: "Clipping Enabled by System" means NX automatically detects that clipping planes are required and turns them on.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Tag - can not find view object from given view tag
3 = Either status or distance pointer is NULL

Environment

Internal and External

See Also

UF_VIEW_set_Z_clip
uc6436

Required License(s)

gateway

```
int UF_VIEW_ask_Z_clip
(
    tag_t tag,
    int status [ 2 ] ,
    double distances [ 2 ]
)
```

tag_t	**tag**	Input	view tag
int	**status [2]**	Output	Clipping Plane Status status[2] [0] Front 1 = Clipping Disabled 2 = Clipping Enabled by User 3 = Clipping Enabled by System [1] Back 1 = Clipping Disabled 2 = Clipping Enabled by User 3 = Clipping Enabled by System
double	**distances [2]**	Output	Clipping Plane Distance from the view origin [0] Front [1] Back

UF_VIEW_ask_zoom_scale (view source)

Defined in: uf_view_ugopenint.h

Overview

Retrieves the zoom scale of the specified view. This is the value displayed on the Zoom View menu which is the current scale at which the view is displayed. If there are multiple occurrences of the view in the layout, only one arbitrary occurrence is retrieved.

The view must be on the current layout. On a drawing layout, the view must be a drawing view or a member work view.

Environment

Internal

Required License(s)

gateway

```
int UF_VIEW_ask_zoom_scale
(
    tag_t view_tag,
    double * scale
)
```

tag_t	**view_tag**	Input	The view tag; if NULL_TAG, work view is used
double *	**scale**	Output	The scale for the view

UF_VIEW_convert_to_model (view source)

Defined in: uf_view.h

Overview

This routine converts a view dependent object in the specified view to the model.

Environment

Internal and External

Note: This user function uses the same object filter as that of the view dependent edit functionality in interactive NX; you cannot change it.

See Also

For example please refer to the example

History

This function was originally released in V15.0.

Required License(s)

gateway

```
int UF_VIEW_convert_to_model
(
    tag_t view_tag,
    tag_t object_tag
)
```

tag_t	**view_tag**	Input	The tag of the view
tag_t	**object_tag**	Input	Tag of the view dependent object

UF_VIEW_convert_to_view (view source)

Defined in: uf_view.h

Overview

Converts a model object to the view dependent object in the specified view.

Note: This user function uses the same object filter as that of the view dependent edit functionality in interactive NX; you cannot change it.

Environment

Internal and External

See Also

For example please refer to the example

History

This function was originally released in V15.0.

Required License(s)

gateway

```
int UF_VIEW_convert_to_view
(
    tag_t view_tag,
    tag_t object_tag
)
```

tag_t	**view_tag**	Input	The specified view tag.
tag_t	**object_tag**	Input	The object tag.

UF_VIEW_copy_view (view source)

Defined in: uf_view.h

Overview

Makes a copy of the given view, creating a new view.
The view_to_copy may or may not be currently displayed. The new_view is NOT displayed by UF_VIEW_copy_view, but you can later replace the new view into the layout, using uc6464.

The data for the copy is taken from the "saved" parameters of the view, which are not necessarily the view parameters currently being used for the view. For example, if UF_VIEW_rotate_view has been used since the view was displayed, and the view has not been saved since the call to UF_VIEW_rotate_view, then the saved parameters do not include the effect of the rotation. If this is not what you want, you might call uc6450 first, but note that uc6450 does not save a change to the rotation of one of the eight "canned" views.

This function does not copy any view modifications made to objects in view_to_copy to new_view. Any view-dependent objects defined in view_to_copy are not moved to new_view but remain defined in view_to_copy.

Environment

Internal and External

See Also

uc6464
uc6450

Required License(s)

gateway

```
int UF_VIEW_copy_view
(
    tag_t view_to_copy,
    const char * name_of_new_view,
    tag_p_t new_view
)
```

tag_t	**view_to_copy**	Input	The tag of the view which you want to copy. This view may not be a drawing view, nor may it be a view on a drawing.
const char *	**name_of_new_view**	Input	The name of the new view. Must contain valid characters, must have no more than UF_OBJ_NAME_LEN characters. There must not already be a view of this name in the Displayed part.
tag_p_t	**new_view**	Output	The tag of the newly created view

UF_VIEW_create_view_set (view source)

Defined in: uf_view.h

Overview

Routine -
UF_VIEW_create_view_set

Description -
Create a view set from a base view and a list of orthographic view types.
The view set created will reside in the same part as the base view does.
The view set contains a list of orthographic user defined modeling views
with one of the view as base view. All the other view's orientation are
determined by the base view. When base view changes its orientation, all
the other view's orientation will update accordingly.

Note -
The base_view_tag must be a user defined modeling view, it can not
be any canned views. The base_view_type and other_view_types must
be any one within UF_VIEW_standard_orientation_t

Return Codes -
ERROR_OK
UF_VIEW_INVALID_VIEW_TAG
UF_VIEW_INVALID_OBJECT
UF_VIEW_INVALID_ORIENTATION_TYPE

UF_VIEW_CONFLICT_ORIENTATION_TYPE
UF_VIEW_INVALID_NAME
UF_VIEW_NOT_IN_DISPLAYED_PART

See Also -
UF_VIEW_add_to_view_set
UF_VIEW_remove_from_view_set
UF_VIEW_ask_view_set_by_name
UF_VIEW_ask_base_view_of_view_set
UF_VIEW_set_base_view_of_view_set
UF_VIEW_ask_views_of_view_set
UF_VIEW_delete_view_set

Environment

Internal and External

History

This routine is originally created in release NX3.0

Required License(s)

gateway

```
int UF_VIEW_create_view_set
(
    const char * name,
    tag_t base_view_tag,
    UF_VIEW_standard_orientation_t base_view_type,
    int num_other_views,
    UF_VIEW_standard_orientation_p_t other_view_types,
    tag_t * view_set_tag
)
```

const char *	**name**	Input	The view set name
tag_t	**base_view_tag**	Input	The Base View in the Set
UF_VIEW_standard_orientation_t	**base_view_type**	Input	The Base View orthographic type
int	**num_other_views**	Input	The number of other views to add to the set
UF_VIEW_standard_orientation_p_t	**other_view_types**	Input	A list of other view types to add in the set
tag_t *	**view_set_tag**	Output	the view set object created

UF_VIEW_cycle_objects (view source)

Defined in: uf_view.h

Overview

Cycle objects within a view.

If view is NULL_TAG, the work view is used.

A cycle type of UF_VIEW_VISIBLE_OBJECTS returns all objects which are visible in the view; when using a cycle type of UF_VIEW_VISIBLE_OBJECTS, objects which are outside of the view bounds are not returned. However, non-rectangular boundaries of drafting member views are not considered by UF_VIEW_cycle_objects. UF_VIEW_cycle_objects only works on the rectangular boundary of the view - as though the breakline/detail boundary was removed. If the desire is to obtain visible objects within a drafting member view with non-rectangular borders and with respect to the non-rectangular border itself, please refer to UF_VIEW_ask_visible_objects.

You must not delete, or un-erase objects, or add or remove view modifications during a cycle as the resulting changes will cause the outcome of the cycle to be unpredictable.

Do not attempt to delete objects when cycling the database in a loop. Problems can occur when trying to read the next object when the current object has been deleted. To delete objects, save an array with the objects in it, and then when you have completed cycling, use UF_OBJ_delete_array_of_objects to delete the saved array of objects.

Warning: This function will return curves that are used to display a solid silhouette in a drawing member view. Use UF_DRAW_ask_group_of_curve on any curve returned to determine if the curve belongs to a UF_solid_silhouette_type group.

Warning: This function can return objects which are visible within the specified view but which are on layers other than the user-accessible layers of 1 - 256 inclusive.

Warning: When run in external mode, this function will return solids as visible which are occluded by other solids in the specified view. For a drafting member view, extracted edges may be used to force occlusion to be considered.

Warning: For drafting member views with extracted edges on (Exact, Smart Lightweight, or Exact (Pre-NX 8.5)
with Extracted Edges toggle on) this function will not return solid faces and edges because they are not visible.

Environment

Internal and External

See Also

UF_VIEW_ask_visible_objects

History

Originally released in V18.0

Required License(s)

gateway

```
int UF_VIEW_cycle_objects
(
    tag_t view,
    UF_VIEW_cycle_objects_t type,
    tag_t * object
)
```

tag_t	**view**	Input	The view to cycle in - if NULL_TAG, the work view is used
UF_VIEW_cycle_objects_t	**type**	Input	Object Selection - one of: UF_VIEW_VISIBLE_OBJECTS UF_VIEW_DEPENDENT_OBJECTS

			UF_VIEW_ERASED_OBJECTS UF_VIEW_MODIFIED_OBJECTS
tag_t *	**object**	Input / Output	On input, the last object found by this routine. Pass in a NULL_TAG to start the cycle. On output, is set to the next object. A NULL_TAG is returned when the cycle is complete.

UF_VIEW_delete (view source)

Defined in: uf_view.h

Overview

Deletes a view or returns an error code stating why view could not be deleted.

Environment

Internal and External

Required License(s)

gateway

```
void UF_VIEW_delete
(
    tag_t view_obj_id,
    int * error_flag
)
```

tag_t	**view_obj_id**	Input	Object ID of view to delete
int *	**error_flag**	Output	Status code: 0 = Success 1 = Can not delete canned view 2 = Can not delete work view 3 = Can not delete drawing view 4 = Can not delete last view in canned lay. 5 = Can not delete sketch view 6 = Can not delete view, view belongs to at least one layout

UF_VIEW_delete_view_set (view source)

Defined in: uf_view.h

Overview

Routine -
UF_VIEW_delete_view_set

Description -
Delete the view set

Return Codes -
ERROR_OK

UF_VIEW_INVALID_OBJECT
UF_VIEW_INVALID_VIEW_TAG

See Also

UF_VIEW_create_view_set
UF_VIEW_add_to_view_set
UF_VIEW_remove_from_view_set
UF_VIEW_ask_view_set_by_name
UF_VIEW_ask_base_view_of_view_set
UF_VIEW_set_base_view_of_view_set
UF_VIEW_ask_views_of_view_set

Environment

Internal and External

History

This routine is originally created in release NX3.0

Required License(s)

gateway

```
int UF_VIEW_delete_view_set
(
    tag_t view_set_tag
)
```

tag_t	**view_set_tag**	Input	The view set

UF_VIEW_edit_view_light (view source)

Defined in: uf_view.h

Overview

Sets the lighting attributes of a single specified light.

You cannot change the light type, light mode, or direction of the light with this function.

Environment

Internal and External

See Also

UF_VIEW_set_view_light
UF_VIEW_ask_view_light

History

This routine is originally created in release NX3.0.3

Required License(s)

gateway

```
int UF_VIEW_edit_view_light
(
    const UF_VIEW_light_name_t light_name,
    UF_VIEW_light_attributes_p_t light_attrs
```

)

const UF_VIEW_light_name_t	**light_name**	Input	Name of light to edit
UF_VIEW_light_attributes_p_t	**light_attrs**	Input	Data structure containing light attributes

---

UF_VIEW_expand_view (view source)

Defined in: uf_view.h

Overview

Makes the specified view the work view then expands the view. If the view is already expanded or if the current layout is a single view, then the request is ignored. An active layout must exist. The view may be a member view in a drawing layout. Expanding a drawing member view is equivalent to entering the Work In Member View mode.

Environment

Internal

See Also

For example please refer to the example

Required License(s)

gateway

```
int UF_VIEW_expand_view
(
    tag_t view_tag
)
```

tag_t	**view_tag**	Input	The view tag; if NULL_TAG, work view is used

---

UF_VIEW_expand_work_view (view source)

Defined in: uf_view.h

Overview

Expands the current work view. If the view was already expanded or if the current layout is a single view, then the request is ignored. An active layout must already exist. The layout cannot be a layout with a drawing.

Environment

Internal

Required License(s)

gateway

```
int UF_VIEW_expand_work_view
(
    void
)
```

UF_VIEW_fit_view (view source)

Defined in: uf_view_ugopenint.h

Overview

Fits the geometry to the view by the specified fraction. A 1.00 fraction means all geometry is fitted to the whole view (There is a small fraction for padding). A 0.50 fraction is approximately a 1.00 fraction fit plus a half scale. This is equivalent to changing the percentage on the Display Preferences menu and doing a fit except the value is specified differently. If there are multiple occurrences of the view in the layout, only one arbitrary occurrence is changed.

The view must be on the current layout. On a drawing layout, the view must be a drawing view or a member work view.

Environment

Internal

Required License(s)

gateway

```
int UF_VIEW_fit_view
(
    tag_t view_tag,
    double fraction
)
```

tag_t	**view_tag**	Input	The view tag; if NULL_TAG, work view is used
double	**fraction**	Input	The fraction of the view from 0.0 to 1.0

UF_VIEW_is_expanded (view source)

Defined in: uf_view.h

Overview

Checks the expand state of the work view. If the current layout is a single view, the returned value for expanded is false. An active layout must currently exist. The expanded view may be a drawing member view. Checking the expanded state of a drawing member view is equivalent to checking the Work In Member View mode.

Environment

Internal

Required License(s)

gateway

```
int UF_VIEW_is_expanded
(
    logical * expanded
)
```

logical *	**expanded**	Output	True if the work view is expanded, else false

UF_VIEW_map_drawing_to_model (view source)

Defined in: uf_view.h

Overview

Maps a point in drawing space to absolute coordinates. Vectors cannot be mapped using this function. This function maps from drawing member views to model coordinates only. If the units of the part file and drawing are different and the input point is in part units then scale the point to the drawing units before calling this function.

Environment

Internal and External

See Also

UF_VIEW_map_model_to_drawing

History

Originally released in V16.0

Required License(s)

gateway

```
int UF_VIEW_map_drawing_to_model
(
    tag_t member_view,
    double drawing_pt [ 2 ] ,
    double model_pt [ 3 ]
)
```

tag_t	**member_view**	Input	A Member View on drawing
double	**drawing_pt [2]**	Input	2-D point on the drawing member view
double	**model_pt [3]**	Output	Point in absolute space

UF_VIEW_map_model_to_drawing (view source)

Defined in: uf_view.h

Overview

Maps a point in absolute space to drawing coordinates. Vectors cannot be mapped using this function. This function maps from drawing member views only. If the units of the part file and drawing are different then scale the mapped point to the units of the part file where required.

Environment

Internal and External

See Also

UF_VIEW_map_drawing_to_model

History

Originally released in V16.0

Required License(s)

gateway

```
int UF_VIEW_map_model_to_drawing
(
    tag_t member_view,
    double model_pt [ 3 ] ,
    double map_pt [ 2 ]
)
```

tag_t	**member_view**	Input	A Member View on drawing
double	**model_pt [3]**	Input	Point in absolute space
double	**map_pt [2]**	Output	2-D point on the drawing

UF_VIEW_pan_view (view source)

Defined in: uf_view_ugopenint.h

Overview

Translates the specified view to the specified center. This has the effect of changing the view origin with the View pull down menu and does not trigger a regen. If there are multiple occurrences of the view in the layout, only one arbitrary occurrence is translated.
The view must be on the current layout. On a drawing layout, the view must be a drawing view or a member work view.

Environment

Internal

Required License(s)

gateway

```
int UF_VIEW_pan_view
(
    tag_t view_tag,
    double center [ 3 ]
)
```

tag_t	**view_tag**	Input	The view tag; if NULL_TAG, work view is used
double	**center [3]**	Input	Coordinates in absolute space of the new center

---

UF_VIEW_remove_from_view_set (view source)

Defined in: uf_view.h

Overview

Routine -
UF_VIEW_remove_from_view_set

Description -
Remove an orthographic view from the view set

Return Codes -
ERROR_OK
UF_VIEW_INVALID_OBJECT
UF_VIEW_INVALID_VIEW_TAG
UF_VIEW_INVALID_ORIENTATION_TYPE

See Also -
UF_VIEW_create_view_set
UF_VIEW_add_to_view_set
UF_VIEW_ask_view_set_by_name
UF_VIEW_ask_base_view_of_view_set
UF_VIEW_set_base_view_of_view_set
UF_VIEW_ask_views_of_view_set
UF_VIEW_delete_view_set

Environment

Internal and External

History

This routine is originally created in release NX3.0

Required License(s)

gateway

```
int UF_VIEW_remove_from_view_set
(
    tag_t view_set_tag,
    UF_VIEW_standard_orientation_t type
)
```

tag_t	**view_set_tag**	Input	The view set to add
UF_VIEW_standard_orientation_t	**type**	Input	The view type to remove

---

UF_VIEW_rename (view source)

Defined in: uf_view.h

Overview

Renames a view.

Environment

Internal and External

History

Originally released in V17.0

Required License(s)

gateway

```
int UF_VIEW_rename
(
    tag_t view,
    const char * name
)
```

tag_t	**view**	Input	The view that is to be renamed.
const char *	**name**	Input	The new name for the view. (UF_OBJ_NAME_NCHARS character max)

UF_VIEW_restore_view (view source)

Defined in: uf_view_ugopenint.h

Overview

Restores the specified view to the parameters (original scale, translation, and rotation) that were in effect at either the last reset of restore data, the last view regen, the last fit, or view save operation. This is the same as the restore option on the View menu. If there is nothing to restore, the function is ignored. If there are multiple occurrences of the view in the layout, only one arbitrary occurrence is restored.

The view must be on the current layout. On a drawing layout, the view must be a drawing view or a member work view.

Environment

Internal

Required License(s)

gateway

```
int UF_VIEW_restore_view
(
    tag_t view_tag
)
```

tag_t	**view_tag**	Input	The view tag; if NULL_TAG, work view is used

UF_VIEW_rotate_view (view source)

Defined in: uf_view_ugopenint.h

Overview

Rotates the specified view using the view center as the origin of rotation and about the axis specified in the view space.

It does not trigger a regen. Negative angle means clockwise direction. If there are multiple occurrences of the view in the layout, only one arbitrary occurrence is rotated. The view must be on the current layout. The layout cannot be a layout with drawing.

Environment

Internal

Required License(s)

gateway

```
int UF_VIEW_rotate_view
(
    tag_t view_tag,
    double axis [ 3 ] ,
    double delta_angle,
    int count
)
```

tag_t	**view_tag**	Input	The view tag; if NULL_TAG, the work view is used
double	**axis [3]**	Input	X,Y,Z axis components in view space
double	**delta_angle**	Input	The angle in degrees
int	**count**	Input	The number of times to rotate

UF_VIEW_rotate_view_abs_csys (view source)

Defined in: uf_view_ugopenint.h

Overview

Rotate the specified view using the specified center as origin of rotation and about the specified axis. The center and the axis are specified in absolute coordinates. This is the similar to UF_VIEW_rotate_view but the user can specify the center in absolute coordinates and the axis is in absolute coordinates. It does not trigger a regen. A negative angle means rotation in the clockwise direction. If there are multiple occurrences of the view in the layout, only one arbitrary occurrence is rotated.

The view must be on the current layout. The layout cannot be a layout with drawing.

Environment

Internal

Required License(s)

gateway

```
int UF_VIEW_rotate_view_abs_csys
(
    tag_t view_tag,
    double center [ 3 ] ,
    double axis [ 3 ] ,
    double delta_angle,
    int count
)
```

tag_t	**view_tag**	Input	The view tag; if NULL_TAG, the work view is used.
double	**center [3]**	Input	point in abs space
double	**axis [3]**	Input	X,Y,Z axis components in abs space
double	**delta_angle**	Input	The angle in degrees
int	**count**	Input	The number of times to rotate

UF_VIEW_save_all_active_views (view source)

Defined in: uf_view.h

Overview

Saves all "active" views (those in the current layout) and retains their current names.

Environment

Internal and External

See Also

uc6450

History

Original release was in V14.0.

Required License(s)

gateway

```
int UF_VIEW_save_all_active_views
(
    void
)
```

UF_VIEW_set_base_view_of_view_set (view source)

Defined in: uf_view.h

Overview

Routine -
UF_VIEW_set_base_view_of_view_set

Description -
Set a new base view in the set.

Return Codes -
ERROR_OK
UF_VIEW_INVALID_OBJECT
UF_VIEW_INVALID_VIEW_TAG
UF_VIEW_INVALID_ORIENTATION_TYPE

See Also

UF_VIEW_create_view_set
UF_VIEW_add_to_view_set
UF_VIEW_remove_from_view_set
UF_VIEW_ask_view_set_by_name
UF_VIEW_ask_base_view_of_view_set
UF_VIEW_ask_views_of_view_set
UF_VIEW_delete_view_set

Environment

Internal and External

History

This routine is originally created in release NX3.0

Required License(s)

gateway

```
int UF_VIEW_set_base_view_of_view_set
(
    tag_t view_set_tag,
    UF_VIEW_standard_orientation_t type
)
```

tag_t	**view_set_tag**	Input	The view set
UF_VIEW_standard_orientation_t	**type**	Input	The new base view type

UF_VIEW_set_center (view source)

Defined in: uf_view.h

Overview

Set View Center

If view is NULL_TAG, the work view is used.

The view center returned by this function is a point on the view plane at the center of the viewing volume. If the viewing volume is symmetric (as it usually is), then this point is also the view origin.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Tag - can not find view object from given view tag
3 = center pointer is NULL

Environment

Internal and External

See Also

UF_VIEW_ask_center
uc6431

Required License(s)

gateway

```
int UF_VIEW_set_center
(
    tag_t tag,
    double* center
)
```

tag_t	**tag**	Input	view tag
double*	**center**	Input	View Center (Absolute CSYS) x,y and z coordinates.

UF_VIEW_set_fog_options (view source)

Defined in: uf_view.h

Overview

Sets the fog options of the specified view. The view must have type UF_VIEW_MODEL_TYPE.

Environment

Internal and External

See Also

UF_VIEW_ask_fog_options
UF_VIEW_fog_options_t

History

Originally released in NX4.0.

Required License(s)

gateway

```
int UF_VIEW_set_fog_options
(
    tag_t view_tag,
    UF_VIEW_fog_options_p_t fog_options
)
```

tag_t	**view_tag**	Input	The view tag; if NULL_TAG, the work view is used.
UF_VIEW_fog_options_p_t	**fog_options**	Output	New fog options for the view.

UF_VIEW_set_perspective (view source)

Defined in: uf_view.h

Overview

Set View Perspective

If view is NULL_TAG, the work view is used. The value of distance is used only
if option = 2. The value of eye_point is used only if option = 3.
If projection type is "PERSPECTIVE" and the EYE POINT is set to
be behind the current front Z clipping plane, the front Z clipping
plane is moved equal to the EYE POINT.

If the eye point is changed (option = 3), then the rotation matrix
of the view will also be changed.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Tag - can not find view object from given view tag
3 = Drafting View Not Valid
9 = View Not Active
31 = Invalid Distance
32 = Invalid Projection Option (not equal to 1, 2, or 3)

Environment

Internal and External

See Also

UF_VIEW_ask_perspective
uc6439

Required License(s)

gateway

```
int UF_VIEW_set_perspective
(
    tag_t tag,
    int option,
    double distance,
    double * eye
)
```

tag_t	**tag**	Input	View tag
int	**option**	Input	Projection Option 1 = Make view parallel 2 = Make view perspective 3 = Change eye point

double	**distance**	Input	Perspective distance (If option=2)
double *	**eye**	Input	X, Y and Z coordinates of the Eye Point (Absolute CSYS) (If option =3)

UF_VIEW_set_rotation (view source)

Defined in: uf_view.h

Overview

Set View matrix

If view is NULL_TAG, the work view is used.

The input axes do not have to be unit vectors. If the vectors are not perpendicular, they will be made so by adjusting the Y-axis. An error code = 32 will result if either of the input vectors is zero or if they are parallel.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Tag - can not find view object from given view tag
9 = View Not Active
32 = Cross product of input X and Y axes is zero

Environment

Internal and External

See Also

UF_VIEW_ask_rotation
uc6434

Required License(s)

gateway

```
int UF_VIEW_set_rotation
(
    tag_t tag,
    double axes [ 6 ]
)
```

tag_t	**tag**	Input	view tag
double	**axes [6]**	Input	View x and y axes (six element array) view_matrix[0..2] will be the X-axis values and view_matrix[3..5] will be the Y-Axis values. The Z axis of the view matrix will be internally calculated by taking the cross product of the two input axes.

UF_VIEW_set_scale (view source)

Defined in: uf_view.h

Overview

Set View scale

If view is NULL_TAG, the work view is used.

Scales the specified view by the specified scale factor. This is the same as changing the scale on the Zoom View menu and does not trigger a re-gen. If there are multiple occurrences of the view in the layout, then only one arbitrary occurrence is affected.

The view must be on the current layout. On a drawing layout, the view must be a drawing view or a member work view.

Return

Return Code
0 = OK
1 = View does not exist
2 = Invalid view tag -- cannot find view object from given view tag
31 = Input scale value is less than or equal to zero

Environment

Internal

Required License(s)

gateway

```
int UF_VIEW_set_scale
(
    tag_t tag,
    double scale
)
```

tag_t	**tag**	Input	The view tag; if NULL_TAG, work view is used
double	**scale**	Input	A positive absolute scale factor

UF_VIEW_set_shaded_edge_options (view source)

Defined in: uf_view.h

Overview

Sets the shaded edge options of the specified view. The view must have type UF_VIEW_MODEL_TYPE.

Environment

Internal and External

See Also

UF_VIEW_ask_shaded_edge_options
UF_VIEW_shaded_edge_options_t

History

Originally released in NX4.0.

Required License(s)

gateway

```
int UF_VIEW_set_shaded_edge_options
(
    tag_t view_tag,
    UF_VIEW_shaded_edge_options_p_t shaded_edge_options
)
```

tag_t	**view_tag**	Input	The view tag; if NULL_TAG, the work view is used.
UF_VIEW_shaded_edge_options_p_t	**shaded_edge_options**	Output	New shaded edge options for the view.

UF_VIEW_set_surface_display_options (view source)

Defined in: uf_view.h

Overview

Sets the rendering style and the edge display options of the specified view. The view must have type UF_VIEW_MODEL_TYPE.

Environment

Internal and External

See Also

UF_VIEW_ask_surface_display_options
UF_VIEW_rendering_style_t
UF_VIEW_edge_display_options_t

History

Originally released in NX4.0.

Required License(s)

gateway

```
int UF_VIEW_set_surface_display_options
(
    tag_t view_tag,
    UF_VIEW_rendering_style_t rendering_style,
    UF_VIEW_edge_display_options_p_t edge_display_options
)
```

tag_t	**view_tag**	Input	The view tag; if NULL_TAG, the work view is used.
UF_VIEW_rendering_style_t	**rendering_style**	Input	The new rendering style for the view.
UF_VIEW_edge_display_options_p_t	**edge_display_options**	Input	New edge display options for the view.

UF_VIEW_set_view_light (view source)

Defined in: uf_view.h

Overview

Sets the lighting attributes for a single specified view. The lights are now shared between the NX hardware shading and Photo shading.

If no lights were found with the part, the following 2 default lights are automatically turned on in each view to ensure shaded display becomes lit:
1. Light name: Default 1 Ambient
Light type: UF_VIEW_AMBIENT_LIGHT
2. Light name : Default 2 Up-Front Dist
Light type: UF_VIEW_DISTANT_LIGHT
Light mode: UF_VIEW_FIXED_TO_OBSERVER
Light Location: 0.8, 0.0, 0.4
Light to: 0.0, 0.0, 0.0

Environment

Internal and External

See Also

UF_VIEW_ask_view_light

Required License(s)

gateway

```
int UF_VIEW_set_view_light
(
    tag_t view_tag,
    UF_VIEW_lighting_t * view_light
)
```

tag_t	**view_tag**	Input	Tag of view. NULL_TAG = Work View
UF_VIEW_lighting_t *	**view_light**	Input	Data structure containing view lighting parameters

---

UF_VIEW_set_visualization (view source)

Defined in: uf_view.h

Overview

Sets all of the visualization data. If you pass a NULL_TAG to the view argument, then the work view is used. This is not intended to work for drawing member views. For drawing member views use UF_DRAW_set_view_display

Please use
UF_VIEW_set_surface_display_options
and
UF_VIEW_set_fog_options
instead.

Environment

Internal and External

See Also

UF_DRAW_set_view_display

For example please refer to the example

History

In the V15.0 release, new fields were added to the
UF_VIEW_visualization_s data structure.
In NX4.0, this function is superceded by
UF_VIEW_set_surface_display_options and UF_VIEW_set_fog_options.

Required License(s)

gateway

```
int UF_VIEW_set_visualization
(
    tag_t view,
    UF_VIEW_visualization_t * view_data
)
```

tag_t	**view**	Input	View to set data for
UF_VIEW_visualization_t *	**view_data**	Input	View visualization data

UF_VIEW_set_xy_clip (view source)

Defined in: uf_view.h

Overview

Sets the X and Y clip boundaries of the given view. The clip boundaries
define a rectangle in on the Z = 0 rectangle of View Space.

This function is primarily intended for use by specialized applications such
as Translators. It is a low-level access function which only modifies the
view's data base setting for the XY clip boundaries.

If the view is currently displayed, the display of the view is not modified by
this function, and if the view is later saved, the clip boundaries given to
this function are overwritten. If this is not what you need, consider using
uc6431 (Set View Center and Scale), which, for displayed views, modifies the
displayed parameters of the view (and not the data base parameters), and for
non-displayed views modifies the data base parameters of the view. Or,
consider using UF_VIEW_pan_view and/or UF_VIEW_zoom_view, which work only on
displayed views, without regenerating the display of the view.

This function is unusual in that, unlike most Open API functions, the
coordinate space is View Space. View Space is parallel to the screen, with
the center of the view at (0,0,0). The view center is mapped from Absolute
Space to View Space by applying first the view rotation, then the view
translation to it. The scale in View Space is the same as in Absolute Space.

Environment

Internal and External

See Also

UF_VIEW_ask_xy_clip

Required License(s)

gateway

```
int UF_VIEW_set_xy_clip
(
    tag_t view_tag,
    double xy_clip_bounds [ 4 ]
)
```

tag_t	**view_tag**	Input	Tag of the view whose X-Y clip bounds are to be set
double	**xy_clip_bounds [4]**	Input	Array of the bounds of the view, in view space (Minimum_X, Maximum_X, Minimum_Y, Maximum_Y)

UF_VIEW_set_Z_clip (view source)

Defined in: uf_view.h

Overview

Set View Z-Clipping Planes

If view is NULL_TAG, the work view is used. The values of distance are ignored if the corresponding clipping plane is disabled (status) or auto-set. Using auto-set calculates the plane to be the minimum (for front plane) or maximum (for back plane) distance that displays all objects.

You cannot disable clipping planes in a view that has 3D vectors (see Uc6440 - Read view display type), and you cannot disable the front clipping plane in perspective views (see uc6439 - Set view perspective).

If the projection type is "PERSPECTIVE" and the front Z clipping plane is in front of the EYE POINT (see uc6438 - Read view perspective), the front Z clipping plane moves to the EYE POINT. Return Codes of 33 and 34 are warnings only.

Return

Return Code
0 = OK
1 = View Does Not Exist
2 = Invalid View Tag - can not find view object from given view tag
3 = View Has 3D Vectors - Can Not
Remove Clipping Planes
4 = Either status or distance pointer is NULL
9 = View Not Active
31 = Front Plane Behind Back Plane
33 = Perspective View - Can Not Remove
Front Plane
34 = Perspective View - Front Plane Moved
To Eye Point

Environment

Internal and External

See Also

UF_VIEW_ask_Z_clip
uc6437

Required License(s)

gateway

```
int UF_VIEW_set_Z_clip
(
    tag_t tag,
    int status [ 2 ] ,
    double distances [ 2 ]
)
```

tag_t	**tag**	Input	view tag
int	**status [2]**	Input	Clipping plane status [0] Front [1] Back 0 = Do not change status/distance 1 = Disable clipping 2 = Enable clipping 3 = Auto-set distance
double	**distances [2]**	Input	Clipping plane distance (If status[i] = 2) (From view origin) [0] Front [1] Back

---

UF_VIEW_unexpand_work_view (view source)

Defined in: uf_view.h

Overview

Unexpands the current work view. If the view is already unexpanded or if the current layout is a single view then the request is ignored. An active layout must currently exist. The layout can contain a drawing. If the expanded view is a drawing member view, then the drawing is restored. Unexpanding a drawing member view is equivalent to exiting the Work In Member View mode.

Environment

Internal

Required License(s)

gateway

```
int UF_VIEW_unexpand_work_view
(
    void
)
```

UF_VIEW_update_view (view source)

Defined in: uf_view_ugopenint.h

Overview

Updates the display of the specified view when losses in resolution occur due to zooming or any incorrect operations that may affect the display. This is the same as the update option on the View menu. If there is nothing to update, the function is ignored. If there are multiple occurrences of the view in the layout, then only one arbitrary occurrence is updated. The view must be on the current layout.

Environment

Internal

Required License(s)

gateway

```
int UF_VIEW_update_view
(
    tag_t view_tag
)
```

tag_t	**view_tag**	Input	The view tag; if NULL_TAG, work view is used

UF_VIEW_zoom_view (view source)

Defined in: uf_view_ugopenint.h

Overview

Scales the specified view by the specified scale factor. This is the same as changing the scale on the Zoom View menu and does not trigger a regen. If there are multiple occurrences of the view in the layout, then only one arbitrary occurrence is affected.

The view must be on the current layout. On a drawing layout, the view must be a drawing view or a member work view.

Environment

Internal

Required License(s)

gateway

```
int UF_VIEW_zoom_view
(
    tag_t view_tag,
    double scale
)
```

tag_t	**view_tag**	Input	The view tag; if NULL_TAG, work view is used

double	**scale**	Input	A positive absolute scale factor