# Altia Drawing Area Object / AltiaEx API Reference Manual

December 2015

# Table of Contents

# Chapter 1: The Drawing Area Object

The Drawing Area object allows you to have external graphics code draw within a rectangular area of an Altia Design. This code can be hand-written code that you wrote using the Drawing Area's AltiaEx API, or graphics code written in the machine's native window API (for example, Microsoft Windows, X11, etc.) that has been modified to work with the Drawing Area object. Your external graphics code is linked to Altia Design by placing the code within a shared library or DLL. This same code will also run inside DeepScreen generated code.

The Drawing Area object can be used when you have existing graphics code that you would like to incorporate into an Altia Design. This code can be ported to the Drawing Area's AltiaEx API if portability is required to run on different graphics systems, or the existing graphics code can be left in the native window API and modified to work with the Drawing Area object.

The Drawing Area object can also be used to allow custom graphics code to enhance an Altia Design. If the dynamic nature of your Altia Design makes it difficult for Altia objects to meet all the graphics needs then the Drawing Area object can be used with custom graphics code to meet that need.

To use the Drawing Area Object effectively, coding expertise is required. You must be able to use an API and build a shared library or DLL. If existing native graphics code is used, you must have access to the source of graphics code and modification is required.

**NOTE:** **The Drawing Area Object is included with Altia Design but requires a separate license for Altia library 760 (FLEXlm feature library_760) to activate it.**

**Please contact your Altia representative for more information.**

## 1.1 Where to Find Drawing Area Object Libraries, Sources, and Demonstrations

To add a Drawing Area Object from the Altia Model Library to your Altia model, navigate to the Insert ribbon's Model Library Gallery dropdown. Click on the Drawing Area library icon in the Purchased category within the dropdown Library Gallery. For more information on using the Model Library gallery, see Chapter 1 of the Altia User's Guide.

Drawing Area Object sources, libraries, and demonstrations are located under the Altia Design software's `drawingarea` directory. If Altia Design is installed in `c:\usr\altia`, this is `c:\usr\altia\drawingarea`.

## 1.2 How the Drawing Area Object Works

To open the Drawing Area Object from the Altia Model Library, navigate to the Insert ribbon's Model Library Gallery dropdown. Click on the Drawing Area library icon in the Purchased category within the dropdown Library Gallery.

For more information on using the Model Library gallery, see Chapter 1 of the *Altia User's Guide*.

The Drawing Area object works both in Altia Design and with DeepScreen generated code. The source code that you create to interface with the Drawing Area object in both environments is the same. For Altia Design and Runtime, the code must be linked into a shared library or DLL so it can run in the same address space. The shared library is linked with the Drawing Area's AltiaEx API library, **`altiaEx.lib`**. For DeepScreen, the code is linked directly into the DeepScreen generated code. In this case, there is no explicit AltiaEx API library such as **`altiaEx.lib`** for Altia Design and Runtime. Instead, the code for the DeepScreen version of the AltiaEx API library is simply a component of the DeepScreen generated source code. Although the implementation for the AltiaEx API is different between Design/Runtime (whereit is an explicit library, **`altiaEx.lib`**) and DeepScreen (where it is an implicit component of DeepScreen generated code), the interface is the same so user code does not change.

In Altia Design, the function **`altiaExInitialize`** is called when the shared library/DLL is loaded. This function initializes all the function pointers so your code can call functions in Altia Design. If you get a control editor error that says that this function has failed, it is usually caused by failing to include the altiaEx.lib into your shared library/DLL link or the **`altiaEx.lib`** file that you are linking with is out of date.

The Altia Design that uses a Drawing Area object must contain an **Extern Routine** Control statement. This statement indicates the external function in your DLL to call, any parameters to pass to that function, the name of the shared library/DLL to load, and what to do with the return value from that function. The Drawing Area object in the model library has 2 properties that allow you to specify the library name and the name of the function. It calls this function when the design is loaded by using the **`altiaInitDesign`** animation. You can customize this object to meet your particular needs. The external function that this statement calls should be your initialization inside of the shared library/DLL. The standard parameter to pass to this function is the object ID of the Drawing Area object calling this function.

This initialization function needs to record the object ID since this parameter is required in the AltiaEx API functions so Altia knows what Drawing Area object you are referencing. Since multiple Drawing Area objects that access the same shared library/DLL can be in a given Design file, your code should be written to handle this. The initialization function should call the function **`altiaExGetInfo`** to get information about its particular Drawing Area object. It also should register the callbacks it wants to be called when events inside of Altia occur. A list of the callbacks follows:

- **The Altia Redraw Callback** - Your code should only draw to the screen as a result of this callback. Altia needs to control when objects are drawn based upon their ordering. So when it is time to draw the contents of the Drawing Area object, your code should then draw. If events occur that change the state of your code and a redraw needs to reflect those changes, then your code should call **`altiaExUpdate`**. This tells the Drawing Area object that it needs to update that region of the screen. As a result of this call the Altia Redraw callback will be called and your code can then draw to the screen.

- **The Altia Resize Callback** - This is called when the Drawing Area object's size is changed based upon the built-in animations within the Drawing Area object. This allows your code to make any adjustments to handle the size change. This is not called if the Drawing Area object is scaled. To handle that case, the transform matrix passed into the Altia Redraw callback should be used. After each resize, Altia will call the Altia Redraw callback to redraw the contents of the Drawing Area.

- **The Altia Close Callback** - This is called when an Altia Drawing Area object is deleted. This occurs on exit and in the rare case that the Drawing Area object is part of a cloned object that has been deleted.

- **The Altia Show Callback** - This is called when the Drawing Area object is shown or hidden. This can occur as a result of show/hide animations, switching cards in a deck, or deleting the object via a cut/delete in the Altia Editor. In the last case the object is not deleted until the undo stack limit is reached. On exit from Altia, the Drawing Area object is hidden before the close callback is called.

- **The Altia Event Callback** - This is called when input events are available. Input events include timer events, key press events, mouse button, and mouse move events.

- **The Altia Move Callback** - When the Drawing Area object moves, this callback is called to report the change of position. Since the Redraw callback will report the position as well, this callback is only required if special code needs to be executed when the object is moved.
- **The Altia Runmode Callback** - This is only needed when working inside the Altia Design environment. It reports if the editor is in Run or Edit mode.

- **The Altia Activate Callback** - This reports when the window has been selected.

All coordinate locations passed to and from the AltiaEx API functions assume that the origin is the upper left corner.

The Drawing Area object has two built in functions that allow the width and the height of the object to be set (in pixels).

## 1.3 Demonstrations

The demonstrations described here are located in the `drawingarea\demos` directory of the Altia Design software installation. If Altia Design is installed in `c:\usr\altia`, this is `c:\usr\altia\drawingarea\demos`.

### Exapi Demonstration

This demo shows how to use the **`AltiaEx`** API drawing functions to draw vector graphics, text, and a bitmap into the Drawing Area object. Let's look at some of the functions defined in the file **`testdraw.c`**.

- **`initialize`** - This function is the initialization routine that gets called when an Altia Design file is loaded with a Drawing Area object. It is passed in the object ID of the Drawing Area object that is being initialized. The first thing that hap- pens is that the code adds a new object entry into its **`_objs`** array. This array keeps track of all the Drawing Area objects in the design that reference this shared library/DLL. Next the function calls the **`altiaExGetInfo`** routine to find out what window ID the drawing area object is in, the drawing area object's width and height, and the X and Y position of the upper left corner of the drawing area object relative to the upper left corner of the Altia window. Notice that it is possible for Altia to draw the Drawing Area object into more than one window when using multiple views. The window returned here is the main Altia Window. Next the various callbacks are registered. Then finally an Altia raster is created (a raster is a colored bitmap).

- **`setColor`** - This function is called by Control code using the **Extern Routine** statement. It is called when you want to change the background color the Drawing Area object. It finds the correct object, sets the object's new background color, and then requests that Altia redraw the portion of the Altia window that contains the Drawing Area object.

- **`altiaRedrawCB`** - This function is called when Altia wants the whole or a part of the Drawing Area object to be redrawn. For simplicity, this routine always draws the complete Drawing Area object. For a description of the parameters of this call see the AltiaEx API section below. The body of this function finds the correct object and sees if the show flag is set and if it is then it makes the AltiaEx API calls to render its drawing.

## 1.4 AltiaEx API

The AltiaEx API provides an interface between external code and Altia Design/ DeepScreen. Altia Design and DeepScreen have different versions of the altiaEx library to allow the ability to interface to either product. The different versions of the library have the same interface allowing your code to remain the same regardless of which product is being used. In the Altia Design case your code must be placed into a shared library/DLL while with DeepScreen the code can be directly linked in.

### 1.4.1 Initialization Functions

**altiaExInitialize(AltiaExInit_t *init, int size)**

This function is not called by your code. Instead, Altia will call this function to initialize the altiaEx library. Since data is passed between Altia and the library, a version number is included to verify that the library and Altia are compatible. If you get an error message that this function call has failed then the library and Altia are not compatible. You should make sure that the copy of the altiaEx library is up to date and you should then relink his program.

### 1.4.2 Callback Functions

**int altiaExRegCallback(int objId, int callbkType, AltiaDACallBkFunc func)**

This function lets you register one of your functions as a call back function. Altia will then call this function when it needs the client code to do certain things like redraw. You pass in the object ID of the Drawing Area object, the type of callback, and the function to call.

**Returns**
This function will return 1 if the object ID was found, otherwise it will return 0.

The types of callbacks are on the following pages.

## ALTIA_REDRAW_CALLBACK

This function will be called when Altia needs the drawing area to be redrawn.

**void (*AltiaDARedrawFunc) (int objId, void *win, void *dc,
int left, int top, int right, int bottom, int x, int y, int xoff, int yoff,
AltiaEx_Transformer_type *trans)**

**Parameters**

| | |
|---|---|
| **objId** | The object ID of the Drawing Area. |
| **win** | The window ID that Altia wants the code to draw into. Since Altia does “double buffering” the window ID can actually be a bitmap ID of the memory bitmap to draw into. If this is the case, Altia will later “blt” this bitmap to the screen. |
| **dc** | The device context is provided for convenience and it is not required to be used. This function just passes this device context to the drawing functions. |
| **Left, top, right, bottom** | The redraw area relative to the top left corner of the screen. This is the part of the screen that needs to be redrawn. Use the (**x**,**y**) parameters to convert this region to DAO coordinates. |
| **x, y** | Where the top left corner of the drawing area is located relative to the top left corner of the screen. |
| **xoff, yoff** | These will be zero if the window ID is a real window. If the window ID refers to a bitmap, then these values are the amount that the location parameters were changed to compensate for the size difference between the window and the bitmap. When Altia double buffers, the bitmap it draws into might not be the same size as the Altia window. Altia modifies the **x**, **y**, **left**, **bottom**, and **right** parameters to allow for this size change. If you are drawing to a window that is overlaying the drawing area, you can remove this adjustment by subtracting out **xoff** and **yoff** from these values. |
| **trans** | The transformation matrix only needs to be applied if you want to support drawing to a scaled or rotated drawing area. |

## ALTIA_RESIZE_CALLBACK

This function will be called when the width or height of the drawing area changes as a result of the built in animations in the Drawing Area having been changed.

**int (*AltiaDAResizeFunc) (int objId, int width, int height)**

**Parameters**

| | |
|---|---|
| **objId** | The object ID of the Drawing Area that was resized. |
| **width, height** | The new width and height of the object. |

**Returns**

This function should return 1 on success, 0 for failure.

## ALTIA_CLOSE_CALLBACK

This function will be called when a drawing area object is being deleted.

**int (*AltiaDACloseFunc) (int objId)**

**Parameters**

| | |
|---|---|
| **objId** | The object ID of the Drawing Area that is being deleted. |

**Returns**

This function should return 1 on success, 0 for failure.

## ALTIA_EVENT_CALLBACK

This function will be called when the Drawing Area object has gotten a mouse, keyboard, or timer event. If the Drawing Area object in the design has a stimulus area defined for keyboard and/or mouse then that object will be called when those events occur. Timer events will arrive if your external code has defined any system timers (not Altia's stimulus timers).

**void (*AltiaDAEventFunc) (int objId, void *win, AltiaEx_InputEvent_type *event)**

**Parameters**

| | |
|---|---|
| **objId** | The object ID of the Drawing Area. |
| **win** | Window ID of the window receiving the event. |
| **event** | Pointer to the event |

## ALTIA_SHOW_CALLBACK

This function will be called when the Drawing Area object has been hidden or shown or placed in the cut buffer via a cut edit operation.

**void (*AltiaDAShowFunc) (int objId, int showFlag)**

**Parameters**

| | |
|---|---|
| **objId** | The object ID of the Drawing Area being shown or hidden. |
| **showFlag** | 1 if the object is to be shown, 0 if it is to be hidden. |

## ALTIA_ RUNMODE_CALLBACK

This function will be called once this callback is registered and then if the run mode of the editor is changed.

**int (*AltiaDARunModeFunc) (int runmode)**

This function gets called when the Run mode of the editor changes. **runmode** will be 1 if the editor is in Run mode and a 0 if the editor is in Edit mode.

When running in the Altia Runtime or in DeepScreen this function will be called immediately after registering for this callback and never again and the **runmode** value will be a 1.

### 1.4.3 Additional Functions

**int altiaExGetInfo(int objId, void *win, int *x, int *y, int *width, int *height)**

This function allows you to query a Drawing Area object. Pass in the object ID of the Drawing Area you want to query and the function returns the other parameters. The window ID, X, and Y might differ from the values set to the Redraw Callback function and the redraw values.

**Parameters**

| | |
|---|---|
| **objId** | The object ID of the Drawing Area to be queried. |
| **win** | Returned window ID the object is in. If the Drawing Area object is being displayed in multiple windows, the window ID of the main Altia window will be returned. |
| **x, y** | Returned X, Y position. X and Y are in pixels and relative to the upper left corner of the Altia window to the upper left corner of the Drawing Area object. |
| **width, height** | Returned width and height. |

**Returns**
This function will return a value of 1 if the Drawing Area object was found and a 0 if not.

## int altiaExUpdate(int objId, int left, int top, int right, int bottom)

Call this function when a part of the Drawing Area object needs to be redrawn. Calling this function will lead to the Redraw callback being called to actually do the drawing. This approach allows for other Altia objects that overlap the Drawing Area to be redrawn.

**Parameters**

| | |
|---|---|
| **objId** | The object ID of the Drawing Area. |
| **Left, top, right, bottom** | Redraw area relative to the top left corner of the screen. The top value should usually be less than the bottom value as the top of the area will be closer to the upper left corner. If using DAO coordinates, the coordinates must first be converted to screen coordinates using the DAO position. This must be done prior to calling this function. |

**Returns**
This function will return a value of 1 if the Drawing Area object was found and a 0 if not.

## int altiaExRouteValue(char *eventName, double value)

Call this function when you want to change an animation value within Altia.

**Parameters**

| | |
|---|---|
| **eventName** | The animation name to change. |
| **value** | The value to send to the animation. |

**Returns**
This function will only return a 0 if in DeepScreen and the animation was not defined.   Otherwise, it returns a 1.

## int altiaExRouteText(char *eventName, const char *string)

Call this function when you want to change an animation string within Altia.

**Parameters**

| | |
|---|---|
| **eventName** | The animation name to change. |
| **string** | The string to send to the animation. |

**Returns**
This function will only return a 0 if in DeepScreen and the animation was not defined or not a string. Otherwise, it returns a 1.

## int   altiaExRouteConnectionValue(char   *connectionName, double value)

Call this function when you want to change animation values in other objects that are connected to this object's connection.

**Parameters**

| | |
|---|---|
| **connectionName** | The connection name to change. |
| **value** | The value to send to the connection. |

## int altiaExRouteConnectionText(char *connectionName, const  char *string)

Call this function when you want to change animation values in other objects that are connected to this object's connection and those animations are strings.

**Parameters**

| | |
|---|---|
| **connectionName** | The connection name to change. |
| **string** | The string to send to the connection. |

## int altiaExEllipseDraw(unsigned int objId, void* win, void*dc, AltiaEx_GraphicState_type *gs, AltiaEx_Ellipse_type *ellipse, ALTIA_BYTE fill, AltiaEx_Transform_type *trans)

Call this function when you want to draw an ellipse or a circle within the Drawing Area object.

**Parameters**

| | |
|---|---|
| **objId** | The object ID of the Drawing Area. |
| **win** | The window ID (should be the same that was passed in from the Redraw callback). |
| **dc** | Device context used to draw (should be the same that was passed in from the Redraw callback). |
| **gs** | The graphical state to use when drawing. It contains the foreground color, the background color, the pattern to fill with (if the **fill** parameter is not zero), and the brush used to paint the outline (if the **fill** parameter is zero). The other fields in the gs are ignored. |
| **ellipse** | Contains the location of the center point (x, y), and the radius in the horizontal direction (r1) and the radius in the vertical direction (r2). The filled entry is ignored. |
| **fill** | If the ellipse should be filled (set to 1) or not (set to 0). |
| **trans** | The transform matrix (should be the same that was passed in from the Redraw callback). |

**int altiaExRectDraw(unsigned int objId, void* win, void*dc, AltiaEx_GraphicState_type *gs, AltiaEx_Rect_type *rect, ALTIA_BYTE fill, AltiaEx_Transform_type *trans)**

Call this function when you want to draw a rectangle within the Drawing Area object including the background of the Drawing Area object.

**Parameters**

| | |
|---|---|
| **objId** | The object ID of the Drawing Area. |
| **win** | The window ID (should be the same that was passed in from the Redraw callback). |
| **dc** | Device context used to draw (should be the same that was passed in from the Redraw callback). |
| **gs** | The graphical state to use when drawing. It contains the foreground color, the background color, the pattern to fill with (if the **fill** parameter is not zero), and the brush used to paint the outline (if the **fill** parameter is zero). The other fields in the gs are ignored. |
| **rect** | Contains the location of the upper left corner (x0, y0), and the lower right corner (x1, y1). The filled entry is ignored. |
| **fill** | If the rectangle should be filled (set to 1) or not (set to 0). |
| **trans** | The transform matrix (should be the same that was passed in from the Redraw callback). |

**int altiaExLineDraw(unsigned int objId, void* win, void*dc, AltiaEx_GraphicState_type *gs, AltiaEx_Coord_type *coords, ALTIA_SHORT count, AltiaEx_Transform_type *trans)**

Call this function when you want to draw a line or a series of connected lines within the Drawing Area object.

**Parameters**

| | |
|---|---|
| **objId** | The object ID of the Drawing Area. |
| **win** | The window ID (should be the same that was passed in from the Redraw callback). |
| **dc** | Device context used to draw (should be the same that was passed in from the Redraw callback). |
| **gs** | The graphical state to use when drawing. It contains the foreground color, the background color, and the brush used to paint the line. The other fields in the gs are ignored. |
| **coords** | An array of line end points. |
| **count** | Number of end points in the **coords** array. |
| **trans** | The transform matrix (should be the same that was passed in from the Redraw callback). |

**int altiaExArcDraw(unsigned int objId, void* win, void*dc, AltiaEx_GraphicState_type *gs, AltiaEx_Coord_type *center, ALTIA_SHORT r1, ALTIA_SHORT r2, ALTIA_DOUBLE startAngle, ALTIA_DOUBLE endAngle, ALTIA_BYTE fill, AltiaEx_Transform_type *trans)**

Call this function when you want to draw an arc within the Drawing Area object.

**Parameters**

| | |
|---|---|
| **objId** | The object ID of the Drawing Area. |
| **win** | The window ID (should be the same that was passed in from the Redraw callback). |
| **dc** | Device context used to draw (should be the same that was passed in from the Redraw callback). |
| **gs** | The graphical state to use when drawing. It contains the foreground color, the background color, the pattern to fill with (if the **fill** parameter is not zero), and the brush used to paint the outline (if the **fill** parameter is zero). The other fields in the gs are ignored. |
| **center** | Coordinates for the center of the arc. |
| **r1, r2** | The **r1** parameter is the radius of the arc in the horizontal direction. The **r2** parameter is the radius in the vertical direction.. |
| **startAngle, endAngle** | The **startAngle** is the angle in degrees at which the arc should start drawing (where zero degrees is on a line at the same x value as the center and positive values go counter-clockwise). The **endAngle** is the angle at which to stop drawing the arc. |
| **fill** | If the arc should be filled (set to 1) or not (set to 0). |
| **trans** | The transform matrix (should be the same that was passed in from the Redraw callback). |

**int altiaExPolygonDraw(unsigned int objId, void* win, void *dc, AltiaEx_GraphicState_type *gs, AltiaEx_Coord_type *coords, ALTIA_SHORT count, ALTIA_BYTE fill, AltiaEx_Transform_type *trans)**

Call this function when you want to draw a polygon within the Drawing Area object.

**Parameters**

| | |
|---|---|
| **objId** | The object ID of the Drawing Area. |
| **win** | The window ID (should be the same that was passed in from the Redraw callback). |
| **dc** | Device context used to draw (should be the same that was passed in from the Redraw callback). |
| **gs** | The graphical state to use when drawing. It contains the foreground color, the background color, the pattern to fill with (if the **fill** parameter is not zero), the brush used to paint the outline (if the **fill** parameter is zero). The other fields in the gs are ignored. |
| **coords** | An array of each point of the polygon. |
| **count** | Number of points in the **coords** array. |
| **fill** | If the polygon should be filled (set to 1) or not (set to 0). |
| **trans** | The transform matrix (should be the same that was passed in from the Redraw callback). |

**int altiaExLabelDraw(unsigned int objId, void* win, void* dc, AltiaEx_GraphicState_type *gs, AltiaEx_Coord_type *loc, AltiaEx_Label_type *label, AltiaEx_Transform_type *trans)**

Call this function when you want to draw a text string within the Drawing Area object.

**Parameters**

| | |
|---|---|
| **objId** | The object ID of the Drawing Area. |
| **win** | The window ID (should be the same that was passed in from the Redraw callback). |
| **dc** | Device context used to draw (should be the same that was passed in from the Redraw callback). |
| **gs** | The graphical state to use when drawing. It contains the foreground color and the font to be used in drawing the text. The other fields in the gs are ignored. |
| **loc** | Contains the x,y location of the lower left corner of the text extent. |
| **label** | Contains the text to display and the font name. The font name appears in both the **gs** and the **label**. If the **gs** has a font name then that will override the **label**'s font, but if the **gs** does not have a font name then the **label**'s font name will be used. Font names are in X Logical Font Description format (XLDF). See the **altia.ini** file for information about this format. |
| **trans** | The transform matrix (should be the same that was passed in from the Redraw callback). |

**unsigned long altiaExCreateRaster(int width, int height, int bitsPerPixel, int bytesPerScan, unsigned long *ctable, int ctablesize, unsigned long rmask, unsigned long gmask, unsigned long bmask, unsigned char *data, unsigned char *transMask)**

This function creates a bitmap that later can be displayed within the Drawing Area object.

**Parameters**

| | |
|---|---|
| **width, height** | The width and height in pixels of the bitmap to create. |
| **bitsPerPixel** | The number of bits of color per each pixel. Currently this function supports only 8 or 24 for this value. |
| **bytesPerScan** | Number of bytes that make up a single scan line (row of pixels). This value should include any scan line padding required. |
| **ctable** | A pointer to a color table. This should be NULL if **bitsPerPixel** is set to 24 since the RGB data is in the bitmap data itself. It should point to an array of RGB values if **bitsPerPixel** is set to 8 since the bitmap data will contain indexes into this table of RGB data. |
| **ctablesize** | The number of color entries within the **ctable**. |
| **rmask, gmask, bmask** | The **rmask**, **gmask**, and **bmask** parameters tell the function how to interpret the color information. These mask values should set to one the bits that correspond to the RGB color positions in the RGB data. |
| **data** | The data parameter is the actual bitmap data. If **bitsPerPixel** is 8, this data should be a byte per pixel with each pixel being an index into the color table. If **bitsPerPixel** is 24, this data should be 3 bytes per pixel with each byte having red, green, or blue color information as specified by the masks. Unlike Microsoft Windows bitmap data the first byte of data represents the upper left corner of the bitmap. |
| **transMask** | Specifies which bits within the bitmap are to be transparent. If this parameter is NULL, then none of the bitmap is transparent. The format of the **transMask** is the Xbitmap format where there is no padding per scan line and a 1 indicates not transparent and 0 indicates transparent. |

**Returns**
This function returns an identifier that can be used in the **altiaExRasterDraw** function described below.

**int altiaExDeleteRaster(unsigned long raster)**

Call this function when you want to delete a raster that was created with **altiaExCreateRaster**. The parameter is the raster identifier to delete.

**Returns**
This function always returns 1.

**int altiaExRasterDraw(unsigned int objId, void* win, void* dc, AltiaEx_GraphicState_type *gs, int x, int y, unsigned long raster, AltiaEx_Transform_type *trans)**

Call this function when you want to draw a bitmap within the Drawing Area object.

**Parameters**

| | |
|---|---|
| **objId** | The object ID of the Drawing Area. |
| **win** | The window ID (should be the same that was passed in from the Redraw callback). |
| **dc** | Device context used to draw (should be the same that was passed in from the Redraw callback). |
| **gs** | The graphical state to use when drawing. It is used only for clip information but this is not currently implemented. |
| **x, y** | Where to place the lower left corner of the raster. |
| **raster** | The identifier from **altiaExCreateRaster**. |
| **trans** | The transform matrix (should be the same that was passed in from the Redraw callback). |

**int altiaExRepostEvent(unsigned int objId, void *win, AltiaEx_InputEvent_type *event)**

Call this function only if your external code places widget windows over the Altia window and you block mouse or keyboard events from getting to the Altia window while Altia is in Edit mode. Blocking these events can make it difficult to edit the Drawing Area object since the object can only be selected at the object's edges. Reposting these events to Altia while in edit mode allows Altia to receive these events and make selecting on the whole object possible.

<u>**Parameters**</u>

| | |
|---|---|
| **objId** | Not currently used. |
| **win** | The window ID to which the event is being sent. It must be the window ID of a valid Altia window. |
| **event** | The input event to send to Altia |