



ActiveX Control Interface Details

This document provides brief descriptions of the properties, methods, and events used to interface with the Brava! Desktop ActiveX control, version 2.0.

Contents:

File IO Options Section	7
Properties.....	7
RequestFileIOEvents:	7
File Access Section.....	7
Properties.....	7
Filename	7
MrkReviewFilename.....	7
MrkEditFilename	8
AllowDragAndDropFileOpen	8
XDLIsComplete	8
Methods	8
CloseFile	8
OpenMarkupReview	8
Close All MarkupReview	8
CloseMarkupReview	8
NewMarkup.....	9
OpenMarkupEdit	9
SaveMarkup.....	9
SaveAsMarkupEdit	9
CloseMarkupEdit.....	9
DownloadOriginalToLocalFile	9
Section Note:	9
Mouse Tool Section	10
Properties.....	10
MouseTool	10
Methods	11
CopySelection.....	11
View Control Section.....	11
Properties.....	11
Monochrome	11
BackgroundColor	11
ShowAbout.....	12
EnableAnimation.....	12
AnimationTime	12
EnableQuickDraw	12
DisplayLoadStatus	12
EnhancedRenderMode	13
EnhancedRenderModeAvailable.....	13
DisplayThumbnailPanel	13
ThumbnailSize	13
SearchText.....	13
DisplayName.....	13
UserName.....	13
DateFormat	13
TimeZone.....	14
EnableLineWeights	14
LockToZoomExtents	14

AllowPageChangeOnDocumentScroll	15
PagesRotatedByUser.....	15
Methods	15
ShowLayerInfo	15
Rotate90	15
Rotate90Counterclockwise	15
FitAll	15
FitWidth.....	15
MirrorRaster	15
GetCurrentViewportState	16
SetCurrentViewportState	16
CopyPageTextToClipboard.....	17
GetBXPageInfo	17
SetScaleToDevice	17
GetScaleToDevice	18
FindText	18
View State Section.....	19
Properties.....	19
NumberOfViewPins.....	19
Methods	19
GetViewPinsInfo	19
AddViewPin.....	19
NextViewPin.....	19
PreviousViewPin	19
GoToViewPin	19
ClearViewPins.....	20
Auto View State Section	20
Properties.....	21
AllowAutoSaveViewstates.....	21
Methods	21
BackView	21
ForwardView	21
GetAutoSavedViewStateInfo.....	21
Page Control Section	21
Properties.....	21
CurrentPageNumber	21
TotalPages.....	21
Methods	22
NextPage	22
PreviousPage.....	22
Printing Section.....	22
Properties.....	22
PaperSpaceWidth and PaperSpaceHeight	22
PaperViewSize.....	22
WatermarkLineSpacing.....	23
Methods	23
Print.....	23
ShowPrintCtrl	23
PrintCurrentView	23
PrintPageRange.....	23
UseCurrentWindowsDefaultPrinter	24
Banner and Watermark Section	24
Properties.....	24
Banner Strings	24
WatermarkBannerFontName	25
WatermarkBannerFontStyle.....	25
PersistBanners.....	26
SetCustomBannerValue.....	26
GetCustomBannerValue	26
ClearCustomBannerValues.....	26
Methods	26
EditIsoBanners.....	26

Measurement Section	26
Properties.....	26
CalibrationComplete	26
Methods	27
ShowMeasurementSettings	27
Markup Section	27
Properties.....	27
NumberReviewMarkups.....	27
NumberIntegrationMarkups.....	27
NumberEditableIntegrationMarkups.....	27
DisplayChangeMarkupReview	27
MarkupEditLoaded.....	27
MarkupEditDirty	27
AllowMarkupChangeOwner	27
TakeOwnershipOnMarkupConsolidation.....	28
EnableMarkupColorPalette	28
MarkupColor	28
WarnOnUnsavedMrkClose	28
RasterMrkFilename.....	28
RedactionsExist	28
MarkupStampTemplateLoaded.....	29
MarkupStampTemplateDirty	29
MrkStampEntityFilename	29
Methods	29
GotoMarkupPage	29
ShowFindAndRedactCtrl.....	29
RedactFromScript	29
ValidateRedactionScript.....	29
RedactionsHaveNotBeenBurnIn	30
MarkupBurnedIn	30
ConsolidateMarkups	30
ExecuteChangemark.....	30
GetChangemarkInfo.....	30
NewMarkupStampTemplate.....	31
CloseMarkupStampTemplate.....	31
SaveMarkupStampTemplate.....	31
SaveAsMarkupStampTemplate	31
OpenMarkupStampTemplate	32
User Interface Customization Section.....	32
Properties.....	32
DisplayToolbarTools	32
DisplayToolbarView	32
DisplayToolbarPage.....	32
DisplayFindCtrl.....	33
IntegrationDescription	33
IntegraionVersionInfo.....	33
SuppressInfoMessages.....	33
EnableRightMouseButtonMenu	33
UI and Functionality Allows	33
Methods	34
CtrlDisplayed.....	34
CtrlEnabled	34
DisplayCtrl.....	34
EnableCtrl	34
EnableAcceleratorKey.....	34
AccereratorKeyEnabled	34
InvokeCustomerIntegrationMethod	35
Visual Rights™ Section	35
Properties.....	35
Methods	35
ShowVisualRightsSettings	35
Help Section.....	35

Properties.....	36
RequestHelpDisplay	36
LaunchURLHelp.....	36
HelpUrl.....	36
Methods	36
DisplayHelpTopic	36
Error Message Section	36
Properties.....	36
ErrorMessage	37
Version/License Section	37
Properties.....	37
VersionString	37
LicenseString.....	37
SaveViewAs/ Burn-in/ Publish Section.....	37
Properties.....	37
UseStandardCSFOutput	37
Methods	37
SaveViewAs.....	37
BurnInMarkup	38
PublishToCSF	38
ExportCSF	38
ExportCSFSilent	38
ExportTiff.....	38
ExportTiffEx	39
ExportDWF	39
ExportPDF	40
ExportPDFEx	40
SaveViewAsEx.....	42
Compare Section	42
Properties.....	42
CompareFileName	42
CompareViewMode	42
CompareViewMixtureLevel	43
CompareViewHasBeenAligned.....	43
CompareViewAttached	43
CompareTotalPages	43
CompareCurrentPageNumber	43
Methods	43
CloseCompareFile	43
ClearCompareAlignment.....	43
Collaboration Interface Section	43
Properties.....	43
CollaborationMode.....	44
Methods	44
GetCurrentViewportState	44
SetCurrentViewportState	44
GetMarkupEntity	45
SetMarkupEntity.....	45
GetLayerData.....	45
SetLayerData.....	45
GetNumberOfLayers	45
GetLayerState.....	45
SetLayerState	45
GetWorldSpaceDimensions	46
External Reference File Section.....	46
Methods	46
GetXrefPath	46
SetXrefPath.....	46
GetXrefwarningLevel.....	46
SetXrefwarningLevel.....	47
Advanced Printing Section.....	47
Printer Specific Methods	47

BXEnumeratePrinters	47
BXGetPrinterIDs	47
BXGetPrinterNames.....	47
BXGetPrinterName	47
BXGetDefaultPrinterName.....	48
BXGetPaperSizesEnum.....	48
BXGetPaperSizesName.....	48
BXGetPaperSizesValue	49
BXGetPaperSourcesEnum	49
BXGetPaperSourcesName	49
BXGetResolutionsValue.....	50
BXGetDriverVersion.....	50
BXGetSupportsDuplex.....	50
BXGetSupportsCollate.....	51
BXGetIsColorPrinter.....	51
BXSetPaperSize	51
BXGetPaperSize	51
BXSetOrientation	52
BXGetOrientation.....	52
BXSetPrinter	52
Document Specific Methods	52
BXSetPrintScaleType.....	52
BXGetPrintScaleType	53
BXSetPrintScaleFactor	53
BXGetPrintScaleFactor.....	53
Blocks Section	54
GetNumberOfBlocks	54
GetBlockAttributes	54
SetBlockColor	54
ZoomToBlock.....	55
DisplayBlockAttributes	55
Events Section	55
Methods	55
RequestOpenFileGUI.....	55
FileLoaded	56
FileClosed.....	56
FileLoadFailure	56
RequestMarkupFileReview	56
MarkupReviewLoaded	56
MarkupReviewClosed	56
RequestMarkupFileEdit.....	56
MarkupEditCreated	56
MarkupEditLoaded.....	57
MarkupEditClosed.....	57
RequestMarkupSaveAs	57
RequestMarkupSaveAsDXF	57
RequestSaveViewAs.....	58
DisplayHelp.....	59
CDLEvent.....	59
CalibrationComplete	60
MarkupEntityModified.....	60
LayersChanged.....	60
ResolveSOBannerString	60
BDTXMouseDown	60
BDTXMouseUp	60
BDTXMouseMove	61
MarkupEditSaved.....	61
RequestMarkupSave.....	61
RequestMarkupNew	61
RequestCloseMarkupEdit	61
RequestCloseMarkupReview	61
RequestNewMarkupStampTemplate	61

RequestOpenMarkupStampTemplate.....	62
RequestSaveMarkupStampTemplate.....	62
RequestSaveAsMarkupStampTemplate.....	62
RequestCloseMarkupStampTemplate.....	62
MarkupStampTemplateLoadFailure.....	62
MarkupStampTemplateLoadSuccess.....	62
MarkupStampTemplateSaved.....	62
MarkupStampTemplateClosed.....	63
MarkupStampTemplateCreated.....	63
MarkupStampEntityLoadSuccess.....	63
MarkupStampTemplateLoadSuccess.....	63
RequestMarkupStampEntityFilename.....	63
RequestRedactionScript.....	63
RequestLocalFilenameForDownloadOriginal.....	63
RequestExportPDFFilename.....	63
RequestExportTiffFilename.....	64
RequestExportDWFFilename.....	64
RequestRasterMarkupFilename.....	64
DownloadOriginalFailure.....	64
DownloadOriginalSuccess.....	64
CompareFileLoadFailure.....	64
CompareFileLoadSuccess.....	64
CompareFileClosed.....	64
ExportPDFFailure.....	65
ExportPDFSuccess.....	65
ExportTiffFailure.....	65
ExportTiffSuccess.....	65
ExportDWFFailure.....	65
ExportDWFSuccess.....	65
RasterMarkupLoadFailure.....	65
RasterMarkupLoadSuccess.....	65
RequestInfoMessageDisplay.....	65
ControlInitialized.....	66
RequestExportCSFFilename.....	66
BeforeCopy.....	66
BeforePaste.....	66
AfterCopy().....	66
AfterPaste().....	66
XDLsComplete().....	66
PrintJobComplete().....	66
MetadataEntitySelected.....	67
MetadataEntityHovered.....	67
ChangemarkExecuted.....	67
MessageID Section.....	67
MessageID's.....	67
Control Identification Section.....	68
Toolbar controls:.....	69
Menu Item IDs.....	70
Help Topic IDs.....	72
Html Parameters.....	74
Control Key Combinations.....	75

File IO Options Section

The control will provide all file input and output user interface (e.g., File Open dialog, Save As dialog) unless the following property is set.

Properties

RequestFileIOEvents:

Boolean. Used to get/set the control's method of prompting the user for file IO. If set to TRUE, the control will fire an event to indicate that it is requesting a file to be opened, closed, or saved. See Events Section. The default value is FALSE.

Example:

```
RequestFileIOEvents=FALSE.
```

File Access Section

Properties

Filename

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

Example:

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

MrkReviewFilename

String. Used to get/set the markup file currently being read-only viewed. Because multiple read-only markups can be opened, this property is an asterisk delimited list of filenames.

Example:

```
viewObject.MrkReviewFilename = "C:/files/somefile.xml" // causes the control to  
// attempt to load the indicated  
//un-editable markup file.
```

MrkEditFilename

String. Used to get/set the markup file currently being viewed. Markup files opened for edit can only be opened one at a time. Setting this property while there is currently an editable markup file open will cause the current markup file to close.

Example:

```
viewObject.MrkEditFilename = "C:/files/somefile.xml" // causes the control to
// attempt to load the indicated
// markup file.
```

AllowDragAndDropFileOpen

Boolean. Used to get/set the control's ability to allow files to be dragged and dropped in to its client area for opening. Default is true.

XDLIsComplete

Boolean. This property will be FALSE if the currently viewed file has not yet completed publishing on the Brava server.

Methods

CloseFile

CloseFile()

This method closes the file currently being viewed. Also closes any currently open markup files.

OpenMarkupReview

OpenMarkupReview(BSTR filename)

This method causes the control to attempt to open a markup review file indicated by BSTR fileName.

Close All MarkupReview

CloseAllMarkupReview()

This method causes all markup review files currently open to be closed.

CloseMarkupReview

CloseMarkupReview(int index)

This method causes the control to close a particular markup review file indicated by the integer index. Markup review files are indexed in the order in which they were opened, with the first having an index of zero. If there are two or more review markup files

currently open, and an index of -1 is passed to *CloseMarkupReview*, the control will display a modal dialog from which the user can select the markup or markups to be closed.

NewMarkup

NewMarkup()

This method causes the control to create a new un-named editable markup.

OpenMarkupEdit

OpenMarkupEdit(BSTR fileName)

This method causes the control to open an existing editable markup file. If there is currently an editable markup open, this editable markup file will be closed.

SaveMarkup

SaveMarkup()

This method causes the control to save all changes to the currently open editable markup. If the currently open markup has not been previously saved, the control will prompt for a file name using one of the methods described in the File IO Option Section.

SaveAsMarkupEdit

SaveAsMarkupEdit(BSTR fileName)

This method causes the control to save the current state of the currently open editable markup. The markup will be saved in the file indicated by the file name.

CloseMarkupEdit

CloseMarkupEdit()

This method causes any editable markup file currently open to be closed.

DownloadOriginalToLocalFile

([in] BSTR LocalFileName)

If the currently viewed drawing or document is remote, this method copies that file to the path and filename defined in *LocalFileName*.

Section Note:

Any method or property that causes the control to create a file can not be accessed through HTML scripting languages. This includes the following:

- SaveAsMarkupEdit
- SaveAsMarkupStampTemplate
- SaveAsMarkupEditDXF
- SaveViewAs

SaveViewAsEx
DownloadOriginalToLocalFile
ExportPDF
ExportPDFEx
ExportTiff
ExportTiffEx
ExportDWF
ExportCSF
ExportCSFSilent

Mouse Tool Section

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

Properties

MouseTool

Short(integer). Used to get/set the currently active mouse tool. The following are acceptable values:

MT_SOURCEDOC_SELECT	= 1
MT_PAN	= 2
MT_ZOOM	= 3
MT_ROTATE	= 4
MT_MAGNIFIER	= 5
MT_CALIBRATE	= 6
MT_MEASURE_LINE	= 7
MT_MEASURE_POLYGON	= 8
MT_MEASURE_RECT	= 9
MT_REDLINE_EDIT	= 10
MT_REDLINE_SKETCH	= 11
MT_REDLINE_POLYGON	= 12
MT_REDLINE_TEXT	= 13
MT_REDLINE_ELLIPSE	= 14
MT_REDLINE_ARC	= 15
MT_REDLINE_RECTANGLE	= 16
MT_REDLINE_POLYLINE	= 17
MT_REDLINE_LINE	= 18
MT_REDLINE_ARROW	= 19
MT_REDLINE_SKETCHPOLYGON	= 20
MT_REDLINE_XOUT	= 21
MT_REDLINE_XOUT_TEXT	= 22
MT_REDLINE_UNDERLINE_TEXT	= 23
MT_REDLINE_HIGHLIGHT_TEXT	= 24
MT_REDLINE_CHANGEMARK	= 25

MT_REDLINE_POLYCLOUD	= 26
MT_REDLINE_RECTCLOUD	= 27
MT_REDLINE_RASTER	= 28
MT_PRINTREGION	= 29
MT_DYNAMIC_ZOOM	= 30
MT_REDLINE_SQUIGGLE_RECT	= 31
MT_MEASURE_COUNT	= 32
MT_REDLINE_BLOCKOUT	= 33
MT_ALIGN_COMPARE_DOCS	= 34
MT_MEASURE_POLYLINE	= 35
MT_COPYREGION	= 36
MT_MEASURE_CIRCLE	= 37
MT_REDLINE_STAMP	= 38

Methods

CopySelection

CopySelection()

If the currently active mouse tool is MT_SOURCEDOC_SELECT and a section of the document or drawing's text is selected, this method copies the selected text into the system clipboard.

View Control Section

Properties

Monochrome

Boolean. This property is used to get/set whether or not the current view is drawn as monochrome.

Example:

```
viewObject.Monochrome = 1 // causes the current view to be displayed
// only in black and white.
```

BackgroundColor

Short(integer). This property is used to get/set the background color of the current view. The following are acceptable values:

```
0 = BK_DEFAULT
1 = BK_BLACK
2 = BK_WHITE
3 = BK_GRAY
```

Example:

```
viewObject.BackgroundColor = 2 // causes the current view to be displayed with  
// a white background.
```

ShowAbout

Boolean. This property is used to get/set whether the control displays the about splash screen.

Example:

```
viewObject.ShowAbout = 1 // causes the “About” splash screen to be displayed.
```

EnableAnimation

Boolean. This property is used to get/set whether the control animates between view states.

Example:

```
viewObject.EnableAnimation = 0 // disable animation.
```

AnimationTime

Long(integer). Used to get/set the amount of time the control will allow for any animation between view states. This property is measured in milliseconds.

Example:

```
viewObject.AnimationTime = 0 // disable animation.  
viewObject.AnimationTime = 1000 // only allow animations that can be completed in  
// 1 or less seconds.
```

EnableQuickDraw

Boolean. This property is used to get/set whether the control is allowed to trade image quality for draw speed during panning or zooming. When the pan or zoom manipulation is complete, the control will always draw at full quality.

Example:

```
viewObject.EnableQuickDraw = 0 // always draw at full image quality.
```

DisplayLoadStatus

Boolean. This property is used to get/set whether the control displays file load and page change status. The status text appears in the lower left corner of the control.

Example:

```
viewObject.DisplayLoadStatus = 1 // display status during file loads and page changes.
```

EnhancedRenderMode

Boolean. This property is used to get/set whether the control uses anti-aliasing to draw vector lines.

Example:

```
viewObject.EnhancedRenderMode = 0 // make all vector lines draw anti-aliased.
```

EnhancedRenderModeAvailable

Boolean (Read-only). This property is used to determine if the control has the necessary system dlls to render in enhanced mode.

DisplayThumbnailPanel

Boolean. This property gets and sets the visibility state of the thumbnail panel.

ThumbnailSize

Integer. This property gets and sets the display size for individual thumbnails in the thumbnail panel. Valid range is 20 to 185.

SearchText

String. When this property is set, the control will search for the given text. Note that the currently open drawing or document must contain searchable text.

DisplayName

String. When this property is set, the control will display the given text in the tool information area. By default, this area displays the filename of the currently open document or drawing.

UserName

String. The control will use this property whenever it needs to determine the current user name. By default, the control queries the user name from the operating system. For example, if the *UserName* property is set to “NewUser1” any markup entities subsequently created will use “NewUser1” as the author name.

DateFormat

String. The control will use this property whenever it needs to determine the format of the date display. Date strings can be displayed in markup entity information, Changemark information, and banners/watermarks. The following are valid tokens for DateFormat:

%a	Abbreviated weekday name
%A	Full weekday name

%b	Abbreviated month name
%B	Full month name
%c	Date and time representation appropriate for locale (This is the default)
%d	Day of month as decimal number (01 - 31)
%H	Hour in 24-hour format (00 - 23)
%I	Hour in 12-hour format (01 - 12)
%j	Day of year as decimal number (001 - 366)
%m	Month as decimal number (01 - 12)
%M	Minute as decimal number (00 - 59)
%p	Current locale's A.M./P.M. indicator for 12-hour clock
%S	Second as decimal number (00 - 59)
%U	Week of year as decimal number, with Sunday as first day of week (00 - 53)
%w	Weekday as decimal number (0 - 6; Sunday is 0)
%W	Week of year as decimal number, with Monday as first day of week (00 - 53)
%x	Date representation for current locale
%X	Time representation for current locale
%y	Year without century, as decimal number (00 - 99)
%Y	Year with century, as decimal number
%z, %Z	Either the time-zone name or time zone abbreviation
%%	Percent sign

TimeZone

String. The control will use this property whenever it needs to determine the proper time zone for date display and adjusting the current system time. The *TimeZone* property can be specified as follows:

TimeZone = tzn[+|-]hh[:mm[:ss]] [dzn]

<i>tzn</i>	Three-letter time-zone name, such as PST. You must specify the correct offset from local time to UTC.
<i>hh</i>	Difference in hours between UTC and local time. Optionally signed.
<i>mm</i>	Minutes. Separated from hh by a colon (:).
<i>ss</i>	Seconds. Separated from mm by a colon (:).
<i>dzn</i>	Three-letter daylight-saving-time zone such as PDT. If daylight saving time is never in effect in the locality, set <i>TimeZone</i> without a value for dzn.

EnableLineWeights

Boolean. If set to FALSE, the control will draw all vector lines with a width of 1 pixel. Default is TRUE.

LockToZoomExtents

Boolean. If set to TRUE, causes the control to maintain the current view at extents during window resizing. Default is TRUE.

AllowPageChangeOnDocumentScroll

Boolean. If set to TRUE and a document is currently viewed, this property causes the vertical scroll bar and mouse wheel to change pages when the end or beginning of a page is reached. Default is TRUE. Note that this feature only applies to documents and not drawings.

PagesRotatedByUser

Boolean. (Read-only) If set to TRUE, this property indicates that one or more pages in the current document or drawing have been rotated from their original orientation.

Methods

ShowLayerInfo

ShowLayerInfo()

This method causes the control to display a modal dialog allowing the user to view and manipulate a drawing's layers.

Rotate90

Rotate90()

This method causes the control to rotate the current view 90 degrees clockwise.

Rotate90Counterclockwise

Rotate90Counterclockwise()

This method causes the control to rotate the current view 90 degrees counter-clockwise.

FitAll

FitAll()

This method causes the control to adjust the view so that all drawing elements fit into the visible area.

FitWidth

FitWidth()

This method causes the control to adjust the view so that the entire width of the document/drawing fits into the visible area.

MirrorRaster

MirrorRaster(BOOL Horizontal, int Page)

This method causes the control to mirror a raster document or drawing. Parameter “Page” is used to indicate which page of a multi-page document or drawing to mirror. If the parameter “Horizontal” is TRUE, the mirroring will be done about the Y (up and down) axis. If the parameter “Horizontal” is FALSE, the mirroring will be done about the X (left to right) axis.

GetCurrentViewportState

GetCurrentViewportState(double *EyeX, double *EyeY, double *Width, double *Height, double *Rotation, int CoordSys)

This method retrieves the current viewport values relative to the extents of the document/drawing. EyeX and EyeY are the x and y values of the center of the view. Width and Height indicate scale or zoom level. Rotation is measured in degrees. CoordSys governs the coordinate system used as follows.

BRAVAX_WORLD = 0

If CoordSys is set to 0, the real values returned by *GetCurrentViewportState* will be in the drawing/document’s native units.

BRAVAX_ABSOLUTE_SCALE = 1

If CoordSys is set to 1, the EyeX and EyeY returned by *GetCurrentViewportState* will be in the drawing/document’s native units. Width will be the scale value required to convert EyeX and EyeY into device space. Height is ignored.

BRAVAX_NDC = 2

If CoordSys is set to 2, the real values returned by *GetCurrentViewportState* will be in normalized device coordinates.

SetCurrentViewportState

SetCurrentViewportState(double EyeX, double EyeY, double Width, double Height, double Rotation, int CoordSys)

This method changes the current viewport state relative to the extents of the document/drawing. EyeX and EyeY are the x and y values of the center of the view. Width and Height indicate scale or zoom level. Rotation is measured in degrees. CoordSys governs the coordinate system used as follows:

BRAVAX_WORLD = 0

If CoordSys is set to 0, the real values used by *SetCurrentViewportState* will be in the drawing/document’s native units.

BRAVAX_ABSOLUTE_SCALE = 1

If *CoordSys* is set to 1, the *EyeX* and *EyeY* used by *SetCurrentViewportState* will be in the drawing/document's native units. *Width* will be the scale value required to convert *EyeX* and *EyeY* into device space. *Height* is ignored.

BRAVAX_NDC = 2

If *CoordSys* is set to 2, the real values used by *SetCurrentViewportState* will be in normalized device coordinates.

CopyPageTextToClipboard

CopyPageTextToClipboard()

This method extracts all of the text from the currently viewed page and inserts it into the Windows clipboard.

GetBXPageInfo

([in] int PageNumber,
*[out] double *ToInchesScale,*
*[out] double *Width,*
*[out] double *Heigh,*
*[out] BOOL *IsSingleRaster,*
*[out] int *SingleRasterDPI,*
*[out] BOOL *IsMonochromeRaster,*
*[out] BSTR *PageName,*
*[out] BSTR *PageMetaData)*

This method retrieves information about a particular page in a currently loaded document or drawing.

PageNumber = page to get information about

ToInchesScale = scale value used to convert native *Width* and *Height* unit to inches

Width = width in native units

Height = height in native units

IsSingleRaster = if returned TRUE, the page is composed entirely of one raster entity

SingleRasterDPI = if *IsSingleRaster* is TRUE, this value is the dots per inch of the raster entity.

IsMonochromeRaster = if *IsSingleRaster* is TRUE, this value indicates if raster is monochrome or color

PageName = page name

PageMetaData = not currently used

SetScaleToDevice

([in] double ScaleValue)

This method allows the current view to be scaled to inches. To scale the view to 100%, a ScaleValue of 1.0 would be used. A ScaleValue of 0.5 would cause the current view to be scaled to 50%.

GetScaleToDevice

([out] double *ScaleValue)

This method retrieves the current scale value.

FindText

([in] BSTR SearchString,
[in] BOOL SearchForward,
[in] BOOL MatchCase,
[in] short SearchPosition,
[in] BSTR SearchParams,
[out, retval] short *Result)

This method causes the control to search for and highlight text within the currently loaded document or drawing.

SearchString =- string to find

SearchForward = If set to TRUE, the control will search forward. If set to FALSE, the control will search backward.

MatchCase = if set to TRUE, the control will do a case-sensitive search

SearchPosition = indicates where (within the document or drawing) to do the search.

Possible values

SPT_FROMBEGINING = 1

SPT_FROM_CURRENT = 2

SPT_FROMEND = 3

SearchParams = not currently used.

Result = used to return the result of the search.

Possible values:

SRT_FULLFAIL = 1 - means SearchString was not found anywhere in the document or drawing.

SRT_PARTFAIL = 2 - means SearchString was not found between the current search position and end of the docuement if SearchForward = TRUE, or not found between the current search position and the beginning of the document if SearchForward = FALSE.

SRT_SUCCESS = 3 - means SearchString was found and highlighted between the current search position and end of the document if SearchForward = TRUE, or found and highlighted between the current

search position and the beginning of the document if SearchForward = FALSE.

View State Section

View Pins are a means of setting a marker at particular position/viewstate within a document/drawing.

The control can store a list of View Pin that exists for the duration of a single file load.

Properties

NumberOfViewPins

Integer. Read Only. This property indicates the number of View Pins currently stored.

Methods

GetViewPinsInfo

*([out] int *NumPins, [out] int *CurrentPinIndex)*

Retrieves the total number of View Pins in the current list and the index of the last viewed View Pin (current View Pin index).

AddViewPin

AddViewPin()

This method adds a new View Pin marker to the list.

NextViewPin

NextViewPin()

This method increments the current View Pin index and changes the view to that pin.

PreviousViewPin

PreviousViewPin()

This method decrements the current View Pin index and changes the view to that pin.

GoToViewPin

GoToViewPin([in] int PinIndex)

This method sets the current View Pin index to PinIndex and changes the view to that in.

ClearViewPins

ClearViewPins()

This method deletes the entire View Pin list.

Auto View State Section

The control has the ability to navigate "Back" and "Forward" during viewing session.

The current viewstate consists of:

- Zoom level
- Pan location
- Rotation
- Page number
- Compare view mode

A new viewstate is added to the viewstate stack when:

- Zoom level changes
- Rotation changes
- Page number changes
- Compare view mode changes

A new viewstate is NOT added when pan location changes. However, two new viewstates are added at the point of a normal viewstate change if the pan location has changed since the addition of the last saved viewstate. First the current state with the changed pan location is added then the new normal viewstate.

Example 1:

- Load multi-page (4 or more pages) document
- Hit page down 3 times
- Press Ctrl + Alt + Left Arrow (back) 3 times
- 1st back goes to top of page 3
- 2nd back goes to top of page 2
- 3rd back goes to top of page 1 (starting view)

Example 2:

- Load multi-page (4 or more pages) document
- Scroll the view until the fourth page is visible
- Press Ctrl + Alt + Left Arrow (back) 3 times
- 1st back goes to bottom of page 3
- 2nd back goes to top of page 3
- 3rd back goes to bottom of page 2

Properties

AllowAutoSaveViewstates

Boolean. If set to FALSE, the control will not save viewstates.
Defaults to TRUE.

Methods

BackView

BackView()
This method sends the viewstate back one state.

ForwardView

ForwardView()
This method sends the viewstate forward one state.

GetAutoSavedViewStateInfo

*([out] int *NumViewStates, [out] int *CurrentViewStateIndex)*
The method retrieves the number of viewstates that have been auto-saved and the current index in the saved viewstate stack.

Page Control Section

Properties

CurrentPageNumber

Short(integer). This property is used to get/set the page of a multi-page document currently being viewed. The first page of a multi-page document is indexed as 1.

Example:

```
viewObject.CurrentPageNumber = 106 // causes the control to display page 106.
```

TotalPages

Short(integer). Read Only. This property is used to get the total number of pages in the currently viewed multi-page document.

Example:

```
viewObject.CurrentPageNumber = viewObject.TotalPages - 1 // causes the control to
//display the second to the last page.
```

Methods

NextPage

NextPage()

This method causes the control to load and view the next page of a multi-page document or drawing. If the control is currently viewing the last page of a multi-page document or drawing, NextPage does nothing.

PreviousPage

PreviousPage()

This method causes the control to load and view the previous page of a multi-page document or drawing. If the control is currently viewing the first page of a multi-page document or drawing, PreviousPage does nothing.

Printing Section

Properties

PaperSpaceWidth and PaperSpaceHeight

Double(real). *PaperSpaceWidth* and *PaperSpaceHeight* provide the ability to establish paper space for format types that either do not have a paper space concept, or for display list files that are in non-modelspace: CGM, DWF, DGN, and the model page of DWG documents. Use this property to scale an image to a particular paper size. The renderer can then map the document's world space coordinates to real world coordinates. The values are measured in inches. The default is size D (22x36).

Example:

```
viewObject.PaperSpaceWidth = 22 // set paper size to D (22x36).
viewObject.PaperSpaceHeight = 36
```

PaperViewSize

String. This property provides the ability to establish paper space for format types that either do not have a paper space concept, or for display list files that are in non-modelspace: CGM, DWF, DGN, and the model page of DWG documents. Use this

optional property to scale an image to a particular paper size. The renderer can then map the document's world space coordinates to real world coordinates. This value affects the zoom percentage and size of print banners relative to the overall size of the printed image.

Enter the paper size value as “paper width, paper height, measurable units”. Valid values for measurable units (indicating inches or meters) are i, I, m, or M

Example:

PaperViewSize=8.5,11,i

The last value (i or m) must be present in order to convert the first two fields to numbers. If not set, or if this property does not exist, the default value is “22,36,i” indicating a paper size of 22 x 36 inches.

WatermarkLineSpacing

Integer. This property is used to get and set the density of the pattern used to draw the print watermark. A higher value causes the watermark to be more transparent, and a lower value causes the watermark to be more opaque. Valid values are:

0 = Opaque

1 = Dark (results in a large print spool size)

2 = Medium

3 = Light

Methods

Print

Print()

Causes the control to display the print/print preview modal dialog.

ShowPrintCtrl

This method will cause the control to display the print dialog. This method has the same effect as Method::Print.

PrintCurrentView

PrintCurrentView()

This method causes the control to do a “quick print” of the current view. The print output will be directed to the currently selected printer or to the default printer if no printer has been selected.

PrintPageRange

PrintPageRange(int Start, int End)

This method causes the control to print from “Start” to “End” pages of a document or drawing. The print output will be directed to the currently selected printer or to the default printer if no printer has been selected.

UseCurrentWindowsDefaultPrinter

UseCurrentWindowsDefaultPrinter()

This method forces the control to update its current printer to be the current Windows default printer. If the end user has changed the printer via the Brava Desktop Print dialog, this method will overrule those changes.

Banner and Watermark Section

Properties

Banner Strings

The following properties are string values used to get and set individual banner strings and watermarks. Positioned banners (e.g. TopLeft, RightCenter), can display up to ten separate lines. To indicate a line break use “<\n>”. Any strings set via these properties will not be editable via the *EditIsoBanners* method.

ScreenBanner
ScreenWatermark
Watermark
TopLeft
TopCenter
TopRight
BottomLeft
BottomCenter
BottomRight
LeftTop
LeftCenter
LeftBottom
RightTop
RightCenter
RightBottom

Tags (macros) can be included in the copyright banner string (such as %time, %date, %dbstring(db_token_name), etc). The following macros are supported:

%Date or **%Daydate** Inserts the date (or day and date) the print was spooled. If the tags are viewed on screen, the time at which the screen was last refreshed displays.

%SysDatePlusDays(x) Inserts a date the specified number of days past the system date. Replace "x" with the desired number of days. Negative numbers of days may be entered.

%DBString(x) This tag is used to resolve custom tags for products that integrate with Brava! Replace "x" with any printable character except a right parenthesis, ")". See notes below.

%Time Inserts the time the print was spooled based on a 12 hour clock (AM/PM). If the tags are viewed on screen, the date on which the screen was last refreshed displays.

%MilTime Inserts the time the print was spooled based on a 24 hour clock.

%Title Adds the name of the document.

%Page Adds the page number.

%TotalPages Inserts the total number of pages.

%Login (or %User) Specifies the user name of the person who issued the print.

%% Inserts a single % character.

Example:

The following string would put the current page number and current user name on the top center of each page printed through Brava! Desktop:

```
viewObject.TopCenter = "Page Number %Page User Name %User"
```

WatermarkBannerFontName

String. This property is used to get/set the font name used to render all banner strings.

Example:

```
viewObject.WatermarkBannerFontName = "Arial" // all banner strings use Arial.
```

WatermarkBannerFontStyle

Short(integer). This property is used to get/set the font style used to render all banner strings. The following are acceptable values:

IFS_NORMAL	= 0 (default)
IFS_BOLD	= 1
IFS_ITALIC	= 2
IFS_BOLDITALIC	= 3

Example:

```
viewObject.WatermarkBannerFontStyle = 3 // all banner strings are bold and italic.
```

PersistBanners

Boolean. If set to true, all banners and watermarks will be persisted from session to session. Defaults to false.

SetCustomBannerValue

([in] BSTR Key,[in] BSTR Value)

This method can be used to allow the control container to provide a string value for any unresolved token within a banner or watermark. The token value is indicated by Key and the replacement will be set in Value.

GetCustomBannerValue

([in] BSTR Key,[out,retval] BSTR Value)

This method is used to determine the replacement value for any custom token set by *SetCustomBannerValue*.

ClearCustomBannerValues

This method is will clear all custom banner values set by *SetCustomBannerValue*.

Methods

EditIsoBanners

EditIsoBanners()

This method causes the control to display the ISO Banners and Watermark modal dialog.

Measurement Section

Properties

CalibrationComplete

Boolean. (Read Only) This property is used to get a Boolean value that indicates if the scale for measurement operations has been set. Measurement scale must be set via the MT_CALIBRATE tool prior to using any of the other measurement tools (exception is MT_MEASURE_COUNT). Calibration must be re-done every time a new document or drawing is opened. Note that after a calibration has been completed via the MT_CALIBRATE tool, the currently active mouse tool will be automatically set to MT_MEASURE_LINE. If this is not the desired behavior, a container must adjust after receiving the *CalibrationComplete()* event. See Events Section.

Example:

```
If(viewObject.CalibrationComplete) //Only set the measure line tool if
                                //calibration is complete
{
    viewObject.MouseTool = MT_MEASURE_LINE
}
```

Methods

ShowMeasurementSettings

ShowMeasurementSettings()

Causes the control to display the modal measurement settings dialog.

Markup Section

Properties

NumberReviewMarkups

Boolean. (Read Only) This property is used to get the number of currently open markup review files.

NumberIntegrationMarkups

Integer. This property returns the total number of markups that are available through the integration dll.

NumberEditableIntegrationMarkups

Integer. This property returns the total number of editable markups that are available through the integration dll.

DisplayChangeMarkupReview

Boolean. Used to get/set whether the control displays the Changemarks review control.

MarkupEditLoaded

Boolean (Read Only) If set to TRUE, the control currently has an editable markup file open.

MarkupEditDirty

Boolean (Read Only) If set to TRUE, the control has made changes to the currently open editable markup files that have not been saved.

AllowMarkupChangeOwner

Boolean. If set to TRUE, the control allows entities in the currently open editable markup to be transferred to the current user name. Markup entity ownership is changed by clicking on the entity while depressing the <Shift> key while the markup selection tool is active. Default is FALSE.

TakeOwnershipOnMarkupConsolidation

Boolean. If set to TRUE, the control transfers ownership of all markup entities during a consolidation of open markups. See method ConsolidateMarkups for details. Default is FALSE.

EnableMarkupColorPalette

Boolean. If set to FALSE, the control will disable the user interface for changing the color of markup entities.

MarkupColor

String. Used to set the current color of markup entities. The following are valid strings and their corresponding RGB value:

white	=	RGB(255,255,255)
yellow	=	RGB(255,255,0)
magenta	=	RGB(255,0,255)
red	=	RGB(255,0,0)
cyan	=	RGB(0,255,255)
green	=	RGB(0,255,0)
blue	=	RGB(0,0,255)
darkGrey	=	RGB(128,128,128)
grey	=	RGB(192,192,192)
olive	=	RGB(128,128,0)
purple	=	RGB(128,0,128)
maroon	=	RGB(128,0,0)
teal	=	RGB(0,128,128)
pine	=	RGB(0,128,0)
darkBlue	=	RGB(0,0,128)
black	=	RGB(0,0,0)

WarnOnUnsavedMrkClose

Boolean. If set to false, the control will not display a warning dialog if it is closing an unsaved markup.

RasterMrkFilename

String. Used to get/set the raster file used with the raster markup tool. The raster markup tool can use JPEG or PNG raster files.

RedactionsExist

Boolean (Read Only) If set to 0, redactions exist.

Redactions can be burned in to a CSF file or exist inside a markup that is open for edit or review.

MarkupStampTemplateLoaded

Boolean (Read Only) If set to TRUE, the control currently has an editable markup stamp template file open.

MarkupStampTemplateDirty

Boolean (Read Only) If set to TRUE, the control has made changes to the currently open editable markup stamp template file that have not been saved.

MrkStampEntityFilename

String. Used to get/set the markup stamp template file to be used with the markup stamp tool.

Methods

GotoMarkupPage

GotoMarkupPage(int pgIterType)

This method loads a document or drawing page containing at least one markup entity.

The value of int pgIterType controls which page containing a markup entity to open.

The acceptable values for int pgIterType are:

IT_FIRST	= 1
IT_NEXT	= 2
IT_PREVIOUS	= 3
IT_LAST	= 4

ShowFindAndRedactCtrl

This method makes the control display the Find and Redact dialog.

RedactFromScript

([in] *BSTR RedactionScript*, [out, retval] int **NumRedactions*)

This method applies a redaction script file indicated by *RedactionScript*. The resulting number of redactions created is returned in *NumRedactions*.

ValidateRedactionScript

([in] *BSTR RedactionScript*, [out, retval] *BOOL *Valid*)

This method is used to determine if a given text file is a valid redaction script. The full path to the file is provided in *RedactionScript*. If valid, the Boolean parameter *Valid* will be returned as true.

RedactionsHaveNotBeenBurnIn

Boolean. (Read-only) If set to false, this method indicates that the current markup has redactions created but never burnt in to a new CSF file.

MarkupBurnedIn

Boolean. (Read-only) If set to true, this method indicates that the current file viewed has markup burned in. Note that this only applies to a CSF file.

ConsolidateMarkups

This method takes all of the entities in all of the markups that are open for review and places them in one editable markup. If there is not currently an editable markup open for review, this method creates a new editable markup.

ExecuteChangemark

([in] BSTR CMID, [in] long IDMode)

This method causes the control to display a particular Changemark. The Changemark displayed is determined by CMID and IDMode. Valid values for IDMode and their effect on the interpretation of CMID are:

BRAVAX_CMID_MODE_TITLE = 0

The control will attempt to execute the first Changemark within the list whose title is equal to CMID.

BRAVAX_CMID_MODE_INDEX = 1

The control will attempt to execute the Changemark within the list whose index is equal to CMID.

CMID_MODE_GUID = 2

The control will attempt to execute the Changemark within the list whose markup entity GUID is equal to CMID.

See *GetChangemarkInfo* for additional information.

GetChangemarkInfo

([in]long Index,
[out] BSTR *Guid,
[out] BSTR *Title,
[out] BSTR *Comment,
[out] BSTR *Author,
[out] BSTR *Link,
[out] BSTR *Metadata,
[out] long *Time,

[out] long *PageNum,
[out] BOOL *IsEditable)

This method retrieves information about a particular Changemark. Use property *NumberChangemarks* to determine the total number of Changemarks that currently exist.

Index =	The Changemark within the list whose information is to be retrieved.
Guid =	The Changemark's markup entity unique string identifier.
Title =	The Changemarks's title.
Comment =	The Changemarks's comment.
Author =	The Changemarks's author name.
Link =	The Changemarks's hyperlink if any exists.
Metadata =	Not currently used.
Time =	The time stamp indicating when the Changemark was created, measured in UTC time.
PageNum =	The page number on which the Changemark exists.
IsEditable =	0 if the Changemark is editable. 1 if the Changemark is not editable.

NewMarkupStampTemplate

This method causes the control to create a new un-named editable markup stamp template.

CloseMarkupStampTemplate

This method causes any editable markup stamp template file currently open to be closed.

SaveMarkupStampTemplate

This method causes the control to save all changes to the currently open editable markup stamp template.

If the currently open markup stamp template has not been previously saved, the control will prompt for a file name using one of the methods described in the [File IO Option Section](#).

SaveAsMarkupStampTemplate

(BSTR fileName)

This method causes the control to save the current state of the currently open editable markup stamp template file. The markup stamp template will be saved to the file indicated by the file name.

OpenMarkupStampTemplate

(BSTR fileName)

This method causes the control to open an existing editable markup stamp template file. If there is currently an editable markup stamp template open, this editable markup stamp template file will be closed.

User Interface Customization Section

Properties

DisplayToolbarTools

Boolean. Based on the value of DisplayToolbarTools, the top toolbar containing PAN, ZOOM WINDOW, ZOOM IN/OUT, and MAGNIFIER buttons will be displayed or removed.

Example:

```
viewObject.DisplayToolbarTools = FALSE // remove the mouse tools toolbar.
```

DisplayToolbarView

Boolean. Based on the value of DisplayToolbarView, the top toolbar containing MARKUP, PRINT, MEASURE, SELECT, FITALL, and FITWIDTH buttons will be displayed or removed.

Example:

```
viewObject.DisplayToolbarView = FALSE //remove the view tools toolbar.
```

DisplayToolbarPage

Boolean. Based on the value of DisplayToolbarPage, the bottom toolbar containing ROTATE90, SHOW LAYERS, TOGGLE COLOR/MONOCROME, CHANGE BACKGROUND COLOR, PAGE NEXT AND PREVIOUS, and PAGE COMBO buttons will be displayed or removed.

Example:

```
viewObject.DisplayToolbarPage = FALSE //remove the page tools toolbar.
```

DisplayFindCtrl

Boolean. Based on the value of DisplayFindCtrl, the bottom toolbar containing FIND and FIND TEXT COMBO buttons will be displayed or removed.

Example:

```
viewObject.DisplayFindCtrl = FALSE //remove the find tools toolbar.
```

IntegrationDescription

String. This property is for use with the standardized integration dll. The integration dll will be asked for a string representing its description.

IntegraionVersionInfo

String. This property is for use with the standardized integration dll. The integration dll will be asked for a string representing its version.

SuppressInfoMessages

Boolean. Setting this property to TRUE causes the control to not display error messages. By default, *SuppressInfoMessages* is set to FALSE.

EnableRightMouseButtonMenu

Boolean. If set to 0, the control will not display a menu when the right mouse button is clicked (while the cursor is within the drawing/document view area). The default is 1.

UI and Functionality Allows

Boolean. The following properties can be used to disable/enable particular control functions. If a given property is set to FALSE, the control will not allow the corresponding functionality. By default, all of these properties are set to TRUE.

- AllowFileOpen
- AllowPrinting
- AllowMarkup
- AllowMeasurement
- AllowLayers
- AllowFind
- AllowCopyText
- AllowMrkReview
- AllowMrkSaveAsDXF
- AllowMrkSaveAs
- AllowMrkSave
- AllowMrkOpen
- AllowMrkNew
- AllowPageControl
- AllowRotate

AllowBKColor
AllowMonochrome

Methods

CtrlDisplayed

*CtrlDisplayed(int ctrlID, BOOL *pVal)*

This method returns a boolean value to indicate if a control (button or other UI element) identified by int ctrlID has been removed. If the control is removed, BOOL pVal will be FALSE. See [Control Identification Section](#) for a list of acceptable control ID's.

CtrlEnabled

*CtrlEnabled(int ctrlID, BOOL *pVal)*

This method returns a boolean value to indicate if a control (button or other UI element) identified by int ctrlID has been disabled. If the control is disabled, BOOL pVal will be FALSE. See [Control Identification Section](#) for a list of acceptable control ID's.

DisplayCtrl

DisplayCtrl(int ctrlID, BOOL newVal)

This method allows a control (button or other UI element) to be displayed or removed based on the value of BOOL newVal. See [Control Identification Section](#) for a list of acceptable control ID's.

EnableCtrl

EnableCtrl(int ctrlID, BOOL newVal)

This method allows a control (button or other UI element) to be enabled or disabled based on the value of BOOL newVal. See [Control Identification Section](#) for a list of acceptable control ID's.

EnableAcceleratorKey

([in] int keyID, [in] BOOL shift, [in] BOOL newval)

This method allows a container to disable hot key (Ctrl + char) combinations. The key combination is specified by keyID, and shift indicates whether the shift key is pressed. If newval is set to false, the key command will be disabled. See [Control Key Combinations](#) for a list of valid values for keyID.

AccereratorKeyEnabled

*([in] int keyID, [in] BOOL shift, out, retval] BOOL*pval)*

This method gets the enabled state of a given key combination.

InvokeCustomerIntegrationMethod

*([in] BSTR Param1, [in] BSTR Param2, [out, retval] BSTR *pResponse)*

This method is for use with the standardized integration dll and can be used to pass custom information to the integration dll. Any response given by the integration dll will be returned in the *pResponse* string.

Visual Rights™ Section

Content Secure Format files (csf) can be published with individual Visual Rights enabled or disabled. The following list of properties are each Boolean values indicating the state of a particular Visual Right. These properties are valid only if the document or drawing currently being viewed is a Content Secure Format file (csf).

Properties

VRPrintingEnabled
VRCopyTextEnabled
VRMeasurementEnabled
VRLayersEnabled
VRSaveAsJPGEEnabled
VRMarkupReviewEnabled
VRMarkupEditEnabled
VRMarkupBurnInEnabled
VRRepublishingEnabled
VRDateExpiredEnabled
VRPasswordProtectEnabled

Methods

ShowVisualRightsSettings

ShowVisualRightsSettings()

This method causes the control to display a modal dialog that informs the user of the Visual Rights settings of the currently viewed Content Secure Format file. This method is available only if the document or drawing currently viewed is a Content Secure Format file (csf).

Help Section

Properties

RequestHelpDisplay

Boolean. This property is used to get/set a Boolean value indicating how to display the control Windows' help file. If this value is TRUE, the control will fire the DisplayHelp(contextID) and the container is responsible for handling the user's request for help. If set to FALSE, the control will attempt to launch the Window's help file itself. This property is set to FALSE by default.

LaunchURLHelp

Boolean. If set to true, the control launches the URL specified by the HelpURL property when any of the control's help buttons are pressed. Defaults to false.

See Also:

- HelpUrl
- Event DisplayHelp
- RequestHelpDisplay
- DisplayHelpTopic

HelpUrl

String. If property LaunchUrlHelp is set to true, the machine's default browser is invoked with this string whenever any help button in the control is pressed. Also, a parameter is appended to the HelpUrl string to indicate the help topic requested. The string passed to the browser is in the form:

"HelpUrl"?HContext="TopicID"

See section *HelpTopicIDs* for a listing of possible TopicIDs.

Methods

DisplayHelpTopic

DisplayHelpTopic(short TopicID)

This method causes the control to either launch the Windows help file with the given TopicID, or fire the DisplayHelp(ContextID) event with the given TopicID, based on the value of RequestHelpDisplay. See Help Topic ID's Section for a list of acceptable Topic ID's.

[Error Message Section](#)

Properties

ErrorMessage

String. Read Only. This property is used to get the current error message string.

Version/License Section

Properties

VersionString

String. Read Only. This property is used to get the version of control. The version string is formatted:

Major,Minor1,Minor2,BuildNo.

LicenseString

String. Read Only. This property is used to get the license string of control if applicable.

SaveViewAs/ Burn-in/ Publish Section

Properties

UseStandardCSFOutput

Boolean. If *RequestFileIOEvents* property is TRUE, setting *UseStandardCSFOutput* to FALSE will cause the control to fire the *RequestExportCSFFilename* event. This allows a container to provide file IO when creating CSF files. If *UseStandardCSFOutput* is set to TRUE (default), the control will provide the file IO when creating CSF files.

Methods

SaveViewAs

SaveViewAs(string filename, short type)

This method causes the control to save the current view as an alternate format (e.g. JPG). The resulting alternate format will be saved to a file named "filename".

Currently only JPG is supported, but in the future other formats may be added (e.g. PDF). The parameter type is specified as follows:

JPG - 0

BurnInMarkup

BurnInMarkup()

This method causes the control to create a new CSF file with the currently open markups burned-in. The control will prompt the user for a filename. If the currently viewed document is CSF, the newly created CSF will inherit all applicable Visual Rights security settings. If the currently viewed document or drawing is a native format, the user will be prompted to specify the desired Visual Rights setting to apply to the new CSF.

PublishToCSF

PublishToCSF(bool BurnInMarkups)

This method causes the control to create a new CSF file containing the content of currently viewed document or drawing. The user will be prompted for a filename. This method is only available if the currently viewed document or drawing is not already a CSF file. Because of Visual Rights settings, a CSF file can not be republished. If the parameter *BurnInMarkups* is TRUE, any currently open markups will be burned in to the resulting CSF file.

ExportCSF

([in] BSTR Filename,[in] BOOL BurnInMarkups)

This method creates a CSF rendition of the currently viewed document or drawing. If *BurnInMarkups* is set to TRUE, any open markups are burned into the resulting CSF file.

ExportCSFSilent

([in] BSTR Filename,[in] BOOL BurnInMarkups)

This method creates a CSF rendition of the currently viewed document or drawing. If *BurnInMarkups* is set to TRUE, any open markups are burned into the resulting CSF file. No dialogs for determining export options will be presented. If the currently viewed file is a CSF file, the Visual Rights effective in the new CSF file will be identical to the original. If the currently viewed file is of any other format, the resulting CSF file will have the most permissive Visual Rights possible.

ExportTiff

([in] BSTR Filename,[in] BOOL ForceMonochrome)

This method creates a TIFF rendition of the currently viewed document or drawing. This method has been superseded by ExportTiffEx.

Filename = output filename

ForceMonochrome = if set to True, the resulting TIFF file will be black and white only.

ExportTiffEx

(*[in]* BSTR *Filename*,
[in] BSTR *MetaDataString*,
[in] BOOL *ForceMonochrome*,
[in] BOOL *OutputChangemarks*,
[in] BOOL *OutputRedactionInfo*,
[in] int *Dpi*,
[in] int *ViewStateExports*,
[in] BOOL *UseCustomPageSize*,
[in] double *CustomPageWidth*,
[in] double *CustomPageHeight*)

This method creates a TIFF rendition of the currently viewed document or drawing.

Filename = output filename

MetaDataString = not currently used.

ForceMonochrome = if set to TRUE, the resulting TIFF file will be black and white only.

OutputChangemarks = not currently used.

OutputRedactionInfo = not currently used.

Dpi = controls pixel density of output TIFF.

ViewStateExports = a set of bit flags that indicate what view state elements of the currently open drawing or document will be reflected in the output file.

Current possible values:

EXPORT_VIEWSTATE_NONE = 0x000

The file will be exported in its default state. No changes to the appearance of the current drawing or document during the viewing session.

EXPORT_VIEWSTATE_ROTATIONS = 0x0001

All page rotations made during the viewing of the current drawing or document will be reflected in the output file.

UseCustomPageSize = controls whether the output TIFF uses the current document/drawing dimensions, or *CustomPageWidth* and *CustomPageHeight*.

CustomPageWidth = if *UseCustomPageSize* is TRUE, the width in inches of the output TIFF.

CustomPageHeight = if *UseCustomPageSize* is TRUE, the height in inches of the output TIFF.

ExportDWF

(*[in]* BSTR *Filename*,
[in] BSTR *MetaDataString*,
[in] BSTR *Password*,
[in] BOOL *PasswordProtected*,
[in] BOOL *OutputChangemarks*,
[in] BOOL *OutputRedactionInfo*,
[in] BOOL *FlattenLayers*,

[in] int ViewStateExports,
[in] BOOL UseCustomPageSize,
[in] double CustomPageWidth,
[in] double CustomPageHeight,
[in] BSTR Description);

This method creates a DWF rendition of the currently viewed document or drawing.

Filename = output filename

MetaDataString= not currently used.

Password= DWF password for viewing.

PasswordProtected= if set to TRUE, creates DWF file using password.

OutputChangemarks= not currently used.

OutputRedactionInfo= not currently used.

FlattenLayers= if set to TRUE, all entities on all layers in the current drawing or document will be placed in a single default layer on the output DWF.

ViewStateExports= a set of bit flags that indicate what view state elements of the currently open drawing or document will be reflected in the output file.

Current possible values:

EXPORT_VIEWSTATE_NONE = 0x000

The file will be exported in its default state. No changes to the appearance of the current drawing or document during the viewing session.

EXPORT_VIEWSTATE_ROTATIONS = 0x0001

All page rotations made during the viewing of the current drawing or document will be reflected in the output file.

UseCustomPageSize = controls whether the output DWF uses the current document/drawing dimensions, or *CustomPageWidth* and *CustomPageHeight*.

CustomPageWidth= if *UseCustomPageSize* is TRUE, the width in inches of the output DWF.

CustomPageHeight= if *UseCustomPageSize* is TRUE, the height in inches of the output DWF.

Description = not currently used.

ExportPDF

([in] BSTR Filename,[in] BOOL MrkAsPDFAnnotations)

This method creates a PDF rendition of the currently viewed document or drawing. If *MrkAsPDFAnnotations* is set to TRUE, any open markups are converted to PDF annotations in the resulting PDF file.

ExportPDFEx

([in] BSTR Filename,

[in] BSTR MetaDataString,

[in] BSTR OwnerPassword,

[in] *BSTR UserPassword*,
[in] *BOOL PasswordProtected*,
[in] *BOOL MrkAsPDFAnnotations*,
[in] *BOOL OutputChangemarks*,
[in] *BOOL OutputRedactionInfo*,
[in] *BOOL DoBlockAttributes*,
[in] *BOOL DoHyperlinks*,
[in] *BOOL DoBookmarks*,
[in] *BOOL DoLayers*,
[in] *BOOL PDFRights*,
[in] *int ViewStateExports*,
[in] *BOOL UseCustomPageSize*,
[in] *double CustomPageWidth*,
[in] *double CustomPageHeight*)

This method creates a PDF rendition of the currently viewed document or drawing.

Filename = output filename.

MetaDataString = not currently used.

OwnerPassword = PDF password for editing.

UserPassword = PDF password for viewing.

PasswordProtected = if set to TRUE creates PDF file using *OwnerPassword* and *UserPassword*.

MrkAsPDFAnnotations = if set to TRUE all markup currently open will be inserted in to the output PDF as PDF annotations. If set to FALSE all currently open markup will be "burned-in" as normal PDF entities.

OutputChangemarks = If set to "1" any Changemarks existing in the currently open drawing/document will be numbered and listed at the end of the PDF output. If not set to "1" Changemarks will not be numbered and listed at the end of the PDF output.

OutputRedactionInfo = not currently used.

DoBlockAttributes = if set to TRUE any blocks with attributes in the current drawing or document will be inserted into the output PDF as hot spot block attributes.

DoHyperlinks = if set to TRUE any hyperlink entities in the current drawing or document will be inserted into the output as PDF hyperlinks.

DoBookmarks = if set to TRUE any bookmark entities in the current drawing or document will be inserted into the output as PDF bookmarks.

DoLayers = if set to TRUE all layers in the current drawing or document will have a corresponding layer in the output PDF.

PDFRights = not currently used

ViewStateExports = a set of bit flags that indicate what view state elements of the currently open drawing or document will be reflected in the output file.

Current possible values:

EXPORT_VIEWSTATE_NONE = 0x000

The file will be exported in its default state. No changes to the appearance of the current drawing or document during the viewing session.

EXPORT_VIEWSTATE_ROTATIONS = 0x0001

All page rotations made during the viewing of the current drawing or document will be reflected in the output file.

UseCustomPageSize = controls whether the output PDF uses the current document/drawing dimensions, or CustomPageWidth and CustomPageHeight.

CustomPageWidth = if UseCustomPageSize is TRUE, the width in inches of the output PDF.

CustomPageHeight = if UseCustomPageSize is TRUE, the height in inches of the output PDF.

SaveViewAsEx

(*[in]*BSTR *Filename*, *[in]*short *Type*, *[in]*short *Quality*)

This method works the same as SaveViewAs except it allows the addition of a Quality parameter

Filename = output filename.

Type = output type (see SaveViewAs)

Quality = integer value to control output quality. Must be a value between 1 and 100. 1 is the lowest quality and smallest output, 100 is the highest quality and largest output.

Compare Section

Properties

CompareFileName

String. Used to get/set the name of the document or drawing to be compared against the currently open file.

CompareViewMode

Integer. Used to get/set the current compare mode. The following are valid values:

CVM_OVERLAY	= 1
CVM_OVERLAY_DIFF	= 2
CVM_SIDE_BY_SIDE	= 3
CVM_SIDE_BY_SIDE_DIFF	= 4
CVM_OLDER_DOC	= 5

CVM_NEWER_DOC	= 6
CVM_ADDITIONS	= 7
CVM_DELETIONS	= 8
CVM_UNCHANGED	= 9

CompareViewMixtureLevel

Integer. Used to get/set the percent value of the amount that different entities in the compare drawing are shown while in CVM_OVERLAY_DIFF mode or CVM_SIDE_BY_SIDE_DIFF mode. If CompareViewMixtureLevel is 50, the open file and compare file display equally. A value of 100 displays only the compare file while a value of 0 displays only the open file.

CompareViewHasBeenAligned

Boolean. (Read-only) This property is set to true if the compare alignment tool has been used.

CompareViewAttached

Boolean. (Read-only) This property is set to true if a compare document has been attached.

CompareTotalPages

Integer. (Read-only) This property is used to get the number of pages in the compare document.

CompareCurrentPageNumber

Integer. (Read-only) This property is set to get the page number currently displayed by the compare document.

Methods

CloseCompareFile

This method closes any drawings or documents that have been opened as compare files.

ClearCompareAlignment

If the compare alignment tool has been used to align a base and compare document, this method causes the alignment of the two documents to return to their original default values.

Collaboration Interface Section

Properties

CollaborationMode

Boolean. Used to get/set the control property that governs the firing of collaboration events. If set to true, the control will fire the following events:

CDLEvent with an eventide of NE_VIEW_CHANGED

MarkupEntityModified

Example:

```
Viewobject.CollaborationMode = TRUE. //Tell the control to inform the container on all viewstate changes
```

Methods

GetCurrentViewportState

*(double *EyeX, double *EyeY, double *Width, double *Height, double *Rotation, int CoordSys)*

This method retrieves the current viewport values relative to the extents of the document/drawing. EyeX and EyeY are the x and y values of the center of the view. Width and Height indicate scale or zoom level. Rotation is measured in degrees. CoordSys governs the coordinate system used as follows:

BRAVAX_WORLD = 0

If CoordSys is set to 0, the real values returned by *GetCurrentViewportState* will be in the drawing/document's native units.

BRAVAX_ABSOLUTE_SCALE = 1

If CoordSys is set to 1, the EyeX and EyeY returned by *GetCurrentViewportState* will be in the drawing/document's native units. Width will be the scale value required to convert EyeX and EyeY into device space. Height is ignored.

BRAVAX_NDC = 2

If CoordSys is set to 2, the real values returned by *GetCurrentViewportState* will be in normalized device coordinates.

SetCurrentViewportState

(double EyeX, double EyeY, double Width, double Height, double Rotation, int CoordSys)

This method changes the current viewport state relative to the extents of the document/drawing. EyeX and EyeY are the x and y values of the center of the view. Width and Height indicate scale or zoom level. Rotation is measured in degrees. CoordSys governs the coordinate system used as follows:

BRAVAX_WORLD = 0

If CoordSys is set to 0, the real values used by *SetCurrentViewportState* will be in the drawing/document's native units.

BRAVAX_ABSOLUTE_SCALE = 1

If CoordSys is set to 1, the EyeX and EyeY used by *SetCurrentViewportState* will be in the drawing/document's native units. Width will be the scale value required to convert EyeX and EyeY into device space. Height is ignored.

BRAVAX_NDC = 2

If CoordSys is set to 2, the real values used by *SetCurrentViewportState* will be in normalized device coordinates.

.

GetMarkupEntity

(*[in] BSTR EntityID, [out,retval] BSTR *EntityData*)

This method retrieves a string representation of a markup entity uniquely identified by EntityID. The string representation can then be used to modify a markup entity through SetMarkupEntity. See Event *MarkupEntityModified* for more information.

SetMarkupEntity

(*[in] BSTR EntityID, [in] BSTR *EntityData*)

This method modifies a markup entity uniquely identified by EntityID with the data in EntityData string. See Event *MarkupEntityModified* for more information.

GetLayerData

(*[out,retval] BSTR *LayerData*)

This method returns the current drawing layer visibility state map in string form.

SetLayerData

(*[in] BSTR *LayerData*)

This method modifies the current drawing layer visibility state map.

GetNumberOfLayers

(*[out,retval] int *NumLayers*)

This method returns the number of layers in the current drawing layer visibility state map.

GetLayerState

(*[in] int Index,[out] BSTR *LayerName,[out] BOOL *DefaultVisibility,[out,retval] BOOL *CurrentVisibility*)

This method retrieves information for an individual layer identified by Index.

SetLayerState

(*[in] int Index, [in] BOOL *CurrentVisibility*)

This method modifies the visibility for an individual layer identified by Index.

GetWorldSpaceDimensions

([out] double *Width, [out] double *Height)

This method gets Width and Height of the currently viewed document or drawing in the file's native units.

External Reference File Section

Methods

GetXrefPath

([in] XRefPathTypes PathType, [out,retval] BSTR *XRefPath)

This method returns a listing of the paths the control searches in order to find drawing external references each individual path is serrated by a semicolon. XRefPathTypes can be set to one of the following:

BRAVAX_DWG_XREF_PATH

Paths searched to find external reference files for AutoCad.

BRAVAX_DWG_SHXFONT_PATH

Paths searched to find AutoCad font and shape files.

BRAVAX_DGN_XREF_PATH

Future use only.

SetXrefPath

([in] XRefPathTypes PathType, [in] BSTR XRefPath)

This method modifies the paths the control searches to find various drawing external reference files. See *GetXRefPath* for XRefPathTypes possible values

GetXrefwarningLevel

([in] XRefPathTypes PathType, [out,retval] int *Level)

This method returns an integer to indicate the control's behavior if it is unable to find a drawing's externally referenced file. See *GetXRefPath* for XRefPathTypes description. Possible Levels are:

BRAVAX_XREF_WARN = 0

Display a warning message on failure to find an external reference.

BRAVAX_XREF_IGNORE = 1

Do not display a warning message on failure to find an external reference.

BRAVAX_XREF_ABORT = 2

Display a warning message on failure to find an external reference and abort the file load.

SetXrefwarningLevel

(*[in] XRefPathTypes PathType, [in] int Level*)

This method modifies the control's behavior if it is unable to find a drawing's externally referenced file. See *GetXrefWarningLevel* and *GetXrefPath*.

Advanced Printing Section

Printer Specific Methods

BXEnumeratePrinters

(*[out, retval] long *NumPrinters*)

long *NumbPrinters = Total number of printers found and enumerated on the system.

BXEnumeratePrinters creates a list of printers accessible on the current Windows system. *BXEnumeratePrinters* must be called prior to using any other API methods.

BXGetPrinterIDs

(*[out retval] BSTR *IDs*)

BSTR *IDs = List of unique string IDs.

BXGetPrinterIDs retrieves a list of unique string IDs for all of the printers enumerated by a call to *BXEnumeratePrinters*. A unique string ID is used to get/set information and settings pertaining to a particular printer in subsequent API methods. The list is delimited using the characters "<>":

[printer id 1]<>[printer id 2]<>[printer id 3]...

BXGetPrinterNames

(*[out retval] BSTR *Names*)

BSTR *Names = List of string printer names.

BXGetPrinterNames retrieves a list of printer names for all of the printers enumerated by a call to *BXEnumeratePrinters*. Note that this list should contain exactly one name for each printer ID retrieved using method *BXGetPrinterIDs*. The list is delimited using the characters "<>":

[printer name 1]<>[printer name 2]<>[printer name 3]...

BXGetPrinterName

(*[in] BSTR PrinterID, [out,retval] BSTR *Name*)

BSTR PrinterID = Used to identify the printer.

BSTR *Name = Name of the printer identified by *PrinterID*.

BXGetPrinterName retrieves the printer name of the printer identified by *PrinterID*.

BXGetDefaultPrinterName

([out, retval] BSTR *Name)

BSTR *Name = Name of the printer currently set as the Windows default printer.

BXGetDefaultPrinterName retrieves the printer name of the printer currently set as the Windows default printer.

BXGetPaperSizesEnum

([in] BSTR PrinterID,[out,retval] BSTR *SizesEnum)

BSTR PrinterID = Used to identify the printer.

BSTR *SizesEnum = List of enum values for each paper size available on the printer identified by *PrinterID*.

BXGetPaperSizesEnum retrieves a list of values for each paper size available on the printer identified by *PrinterID*. Each element in this list is a string that corresponds to an integer value that uniquely identifies an available paper size. See method:*BXSetPaperSize* for additional details. The list is delimited using the characters "<>":

[paper size 1]<>[paper size 2]<>[paper size 3]...

BXGetPaperSizesName

([in] BSTR PrinterID,[out,retval] BSTR *SizesName)

BSTR PrinterID = Used to identify the printer.

BSTR *SizeName = List of name values for each paper size available on the printer identified by *PrinterID*.

BXGetPaperSizeName retrieves a list of names (e.g. "Letter, 8.5x11") for each paper size available on the printer identified by *PrinterID*. Note that this list should contain exactly one paper size name for each paper size enum retrieved using *BXGetPaperSizesEnum*. The list is delimited using the characters "<>":

[paper size 1]<>[paper size 2]<>[paper size 3]...

BXGetPaperSizesValue

*([in] BSTR PrinterID,[out,retval] BSTR *SizesValue)*

BSTR PrinterID = Used to identify the printer.

BSTR *SizesValue = List of string values for each paper size available on the printer identified by *PrinterID*.

BXGetPaperSizesValue retrieves a list of strings that indicate the physical size of each paper size available on the printer identified by *PrinterID*. Each element in this list will be a string in the form: [Width'x'Height] where Width and Height are integer values measured in 0.1 mm. Note that this list should contain exactly one paper size value for each paper size enum retrieved using method *BXGetPaperSizesEnum*. The list is delimited using the characters "<>":

[paper size value 1]<>[paper size value 2]<>[paper size value 3]...

BXGetPaperSourcesEnum

*([in] BSTR PrinterID,[out,retval] BSTR *SourcesEnum)*

BSTR PrinterID = Used to identify the printer.

BSTR *SourcesEnum = List of enum values for each paper sources available on the printer identified by *PrinterID*.

BXGetPaperSourcesEnum retrieves a list of values for each paper source available on the printer identified by *PrinterID*. Each element in this list is a string that corresponds to an integer value that uniquely identifies an available paper source. See method:*BXSetPaperSource* for additional details. The list is delimited using the characters "<>":

[paper source 1]<>[paper source 2]<>[paper source 3]...

BXGetPaperSourcesName

*([in] BSTR PrinterID,[out,retval]BSTR *SourcesName)*

BSTR PrinterID = Used to identify the printer.

BSTR *SourcesName = List of name values for each paper sources available on the printer identified by *PrinterID*.

BXGetPaperSourcesName retrieves a list of names (e.g. "Upper Paper Tray") for each paper source available on the printer identified by *PrinterID*. Note that this list should contain exactly one paper source name for each paper source enum

retrieved using method *BXGetPaperSourcesEnum*. The list is delimited using the characters "<>":

[paper source name 1]<>[paper source name 2]...

BXGetResolutionsValue

([in] BSTR PrinterID,[out,retval] BSTR *ResolutionsValue)

BSTR PrinterID = Used to identify the printer.

BSTR *ResolutionsValue = List of values for each resolution available on the printer identified by PrinterID.

BXGetPaperSourcesEnum retrieves a list of strings that indicate the resolutions available on the printer identified by *PrinterID*. Each element in this list will be a string in the form: [ResX'x'ResY] where ResX and ResY are integer values measured in dots per inch. See method:*BXSetResolution* for additional details.

The list is delimited using the characters "<>":

[resolution 1]<>[resolution 2]<>[resolution 3]...

BXGetDriverVersion

([in] BSTR PrinterID,[out,retval] long *DriverVersion)

BSTR PrinterID = Used to identify the printer.

long *DriverVersion = Integer version value.

BXGetDriverVersion retrieves the version number of the printer driver for the printer identified by *PrinterID*.

BXGetSupportsDuplex

([in] BSTR PrinterID,[out,retval] BOOL*SupportsDuplex)

BSTR PrinterID = Used to identify the printer.

BOOL *SupportsDuplex = Boolean duplex value.

BXGetSupportsDuplex returns true if the printer identified by *PrinterID* supports Duplex printing mode.

BXGetSupportsCollate

*([in] BSTR PrinterID,[out,retval] BOOL *SupportsCollate)*

BSTR PrinterID = Used to identify the printer.
BOOL *SupportsCollate = Boolean collate value.

BXGetSupportsCollate returns true if the printer identified by *PrinterID* supports collation.

BXGetIsColorPrinter

*([in] BSTR PrinterID,[out,retval] BOOL *IsColorPrinter)*

BSTR PrinterID = Used to identify the printer.
BOOL *IsColorPrinter = Boolean color ability value.

BXGetIsColorPrinter returns true if the printer identified by *PrinterID* is able to print color output.

BXSetPaperSize

([in] BSTR PrinterID,[in] long PaperSize)

BSTR PrinterID = Used to identify the printer.
long PaperSize = Enum value to indicate which paper size to use.

BXSetPaperSize tells the printer identified by *PrinterID* to output using the paper size indicated by *PaperSize*. *PaperSize* must be one of the values retrieved by a call to *BXGetPaperSizesEnum*.

BXGetPaperSize

*([in] BSTR PrinterID,[out,retval] long *PaperSize)*

BSTR PrinterID = Used to identify the printer.
long PaperSize = Enum value to indicate which paper size is currently in use.

BXGetPaperSize gets the current paper size of the printer identified by *PrinterID*. *PaperSize* will be one of the values retrieved by a call to *BXGetPaperSizeEnum*.

BXSetOrientation

([in] BSTR PrinterID,[in] long Orientation)

BSTR PrinterID = Used to identify the printer.
long Orientation = Integer value used to indicate landscape or portrait print output.

BXSetOrientation allows the printer identified by *PrinterID* to be put in landscape or portrait mode using one of the following values:

ORIENT_PORTRAIT = 1
ORIENT_LANDSCAPE = 2

BXGetOrientation

*([in] BSTR PrinterID,[out,retval] long *Orientation)*

BSTR PrinterID = Used to identify the printer.
long *Orientation = Integer value used to indicate landscape or portrait print output.

BXGetOrientation gets the current orientation of the printer identified by *PrinterID*. Valid values are:

ORIENT_PORTRAIT = 1
ORIENT_LANDSCAPE = 2

BXSetPrinter

([in] BSTR PrinterID)

BSTR PrinterID = Used to identify the printer.

A call to *BXSetPrinter* will cause the Brava control to use the printer identified by *PrinterID* for all subsequent printing operations. Note that the printer used can be reset by an end-user interacting with the standard Brava Print dialog.

Document Specific Methods

BXSetPrintScaleType

([in] long PrintScaleType)

long PrintScaleType = Integer value indicating the scaling type used.

BXSetPrintScaleType allows the currently open document to be print scaled in one of the following ways:

PRT_SCALE_FIT = 0

Make the document fit the extents of the paper.

PRT_SCALE_INSIDE_BANNERS = 1

Make the document fit the extents of the paper, but scale down so that any banners will be printed outside the extents of the document.

PRT_SCALE_DOCUMENT = 2

Use the document's native scale.

BXGetPrintScaleType

([out,retval] long *PrintScaleType)

long *PrintScaleType = Integer value indicating the scaling type used.

BXGetPrintScaleType returns the print scale type used by the currently open document.

BXSetPrintScaleFactor

([in] double PrintScaleFactor)

double PrintScaleFactor = Real value indicating the scaling factor.

BXSetPrintScaleFactor allows a custom scale factor to be applied to each page printed in the currently open document. Note that the *PrintScaleFactor* is only applicable if the *PrintScaleType* is set to PRT_SCALE_DOCUMENT.

BXGetPrintScaleFactor

([out,retval] double *PrintScaleFactor)

double *PrintScaleFactor = Real value indicating the scaling factor.

BXGetPrintScaleFactor returns the custom scale factor to be applied to each page printed in the currently open document. Note that the *PrintScaleFactor* is only applicable if the *PrintScaleType* is set to PRT_SCALE_DOCUMENT

Blocks Section

GetNumberOfBlocks

(*[in] long PageNum,[out, retval] long *NumBlocks*)

GetNumberOfBlocks returns the total number of blocks found on the page specified by *PageNum*.

GetBlockAttributes

(*[in] long PageNum,[in] long BlockIndex,[out, retval] BSTR *Attributes*)

GetBlockAttributes returns the attributes associated with the block found at *BlockIndex* within the list of blocks found on the page specified by *PageNum*.

The Attributes will be in the form of each individual attribute delimited by the following character sequence “<>”

Each individual attribute will contain a tag-value combination in the following format:

“[tag string]: [value string]”

Example return value of *GetBlockAttributes* for a block containing three attributes:

“[tagA]: [valueA]<>[tagB]: [valueB]<>[tagC]: [valueC]”

SetBlockColor

(*[in] long PageNum,[in] long BlockIndex,[in] BSTR Color, [out, retval] BOOL *Success*)

SetBlockColor will set the rendering color of the block found at *BlockIndex* within the list of blocks found on the page specified by *PageNum*.

If the block can be found and its color set, the return value *Success* will be set to TRUE.

If the block can not be found or its color can not be set, the return value *Success* will be set to FALSE.

The parameter *Color* is interpreted as either a string representation of an RGB value or one of the following strings:

“white”	=	RGB(255,255,255)
“yellow”	=	RGB(255,255,0)
“magenta”	=	RGB(255,0,255)
“red”	=	RGB(255,0,0)
“cyan”	=	RGB(0,255,255)
“green”	=	RGB(0,255,0)
“blue”	=	RGB(0,0,255)
“darkGrey”	=	RGB(128,128,128)
“grey”	=	RGB(192,192,192)
“olive”	=	RGB(128,128,0)
“purple”	=	RGB(128,0,128)

“maroon”	=	RGB(128,0,0)
“teal”	=	RGB(0,128,128)
“pine”	=	RGB(0,128,0)
“darkBlue”	=	RGB(0,0,128)
“black”	=	RGB(0,0,0)

ZoomToBlock

*([in] long PageNum,[in] long BlockIndex,[out, retval] BOOL *Success)*

ZoomToBlock will cause the Brava viewer to zoom to the block found at *BlockIndex* within the list of blocks found on the page specified by *PageNum*.

If the block can be found and the view state change is successful, the return value *Success* will be set to TRUE.

If the block can not be found or the view state change is unsuccessful, the return value *Success* will be set to FALSE.

DisplayBlockAttributes

*([in] long PageNum,[in] long BlockIndex,[out, retval] BOOL *Success)*

DisplayBlockAttributes method will cause the control to display a dialog box containing the information about the block indicated by *BlockIndex* and found on the page specified by *PageNum*. *Success* will be set to 1 if the block is successfully displayed, 0 if not.

Events Section

Methods

RequestOpenFileGUI

RequestOpenFileGUI()

Fired when the user has attempted to open a document or drawing and

Property:*RequestFileIOEvents* property is TRUE. The container is then responsible for prompting the user for a document or drawing file name to open and then setting Properties:*Filename* to open the desired document or drawing file.

FileLoaded

FileLoaded(BSTR Filename)

Fired when a document or drawing is successfully loaded.

FileClosed

FileClosed()

Fired when a document or drawing is closed.

FileLoadFailure

FileLoadFailure(BSTR Filename)

Fired when an attempt to open a document or drawing has failed.

Property: *ErrorMessage* can be used for more information.

RequestMarkupFileReview

RequestMarkupFileReview()

Fired when a user has attempted to open a review markup file, and

Property: *RequestFileIOEvents* is TRUE. The container is then responsible for prompting the user for a review markup file name to open and then calling

method: *OpenMarkupReview*(BSTR fileName) to open the desired review markup file.

MarkupReviewLoaded

MarkupReviewLoaded(BSTR Filename)

Fired when a review markup file has been successfully loaded.

MarkupReviewClosed

MarkupReviewClosed(BSTR Filename)

Fired when a review markup file has been closed.

RequestMarkupFileEdit

RequestMarkupFileEdit()

Fired when the user has attempted to open an editable markup file, and

Property: *RequestFileIOEvents* property is TRUE. The container is then responsible for prompting the user for an editable markup file name to open and then calling

method: *OpenMarkupEdit*(BSTR fileName) to open the desired editable markup file.

MarkupEditCreated

MarkupEditCreated()

Fired when the control has created a new editable markup.

MarkupEditLoaded

MarkupEditLoaded(BSTR Filename)

Fired when an editable markup file has been successfully loaded.

MarkupEditClosed

MarkupEditClosed(BSTR Filename)

Fired when an editable markup file has been closed.

RequestMarkupSaveAs

RequestMarkupSaveAs()

Fired when the user has attempted to save an editable markup file, and *RequestFileIOEvents* property is TRUE. The container is then responsible for prompting the user for an editable markup file name to save and then calling the method: *SaveAsMarkupEdit*(BSTR fileName) to save the desired editable markup file.

RequestMarkupSaveAsDXF

RequestMarkupSaveAsDXF()

Fired when the user has attempted to save a markup file, and a DXF and *RequestFileIOEvents* property is TRUE. The container is then responsible for prompting the user for a DXF file name to save and then calling the method: *SaveAsMarkupEditDXF* (BSTR fileName) to save the desired editable markup file as DXF.

MarkupLoadFailure

MarkupLoadFailure(BSTR Filename)

Fired when an attempt to open a markup has failed. Property: *ErrorMessage* can be used for more information.

BurnInFailure

BurnInFailure(BSTR Filename, PublishOutputOptions option)

Fired when an attempt to burn in the currently open a markup has failed. Property: *ErrorMessage* can be used for more information.

PublishOutputOptions indicate the type of publishing output requested by the user. The following are valid values:

BRAVAX_PO_FILE_ONLY	=	0
BRAVAX_PO_EMAIL	=	1
BRAVAX_PO_WEB_PAGE	=	2

The parameter *Filename* returned by any of the publishing events is based on the *PublishOutputOptions* value and is as follows:

BRAVAX_PO_FILE_ONLY	=	Full path of published file.
---------------------	---	------------------------------

BRAVAX_PO_EMAIL	=	Title of email message (generally the published file name).
BRAVAX_PO_WEB_PAGE	=	Full path of the directory containing the published file, the html created, and any support files referenced by the html.

BurnInSuccess

BurnInSuccess(BSTR Filename, PublishOutputOptions option)

Fired when an attempt to burn in the currently open a markup has succeeded.

Property: *ErrorMessage* can be used for more information.

See *PublishOutputOptions* in “BurnInFailure” section for additional details.

RequestSaveViewAs

RequestSaveViewAs(long type)

Fired when the user has attempted to save the current view to and alternate format (e.g. JPG) and *RequestFileIOEvents* property is TRUE. The parameter “type” indicates the desired alternate format. Currently only JPG is supported. See [SaveViewAs](#) section for details. The container is then responsible for prompting the user for an appropriate file name to save and then calling the method: *SaveViewAs*(fileName, type).

SaveViewAsFailure

SaveViewAsFailure(BSTR Filename)

Fired when an attempt to save the current view to an alternate format (e.g. JPG) has failed. Property: *ErrorMessage* can be used for more information.

SaveViewAsSuccess

SaveViewAsSuccess(BSTR Filename)

Fired when an attempt to save the current view to an alternate format (e.g. JPG) has succeeded. Property: *ErrorMessage* can be used for more information.

PublishFailure

PublishFailure(BSTR Filename, PublishOutputOptions option)

Fired when an attempt to convert the currently viewed document or drawing to CSF has failed. Property: *ErrorMessage* can be used for more information.

See *PublishOutputOptions* in “BurnInFailure” section for additional details.

PublishSuccess

PublishSuccess(BSTR Filename, PublishOutputOptions option)

Fired when an attempt to convert the currently viewed document or drawing to CSF has succeeded. Property:*ErrorMessage* can be used for more information.
See *PublishOutputOptions* in “BurnInFailure” section for additional details.

DisplayHelp

DisplayHelp(long contextID)

Fired when the user has requested a display of help information, and Property:*RequestHelpDisplay* is TRUE. The container is then responsible for displaying the help information that pertains to the given contextID. See [Help Topic ID's](#) Section for a list of acceptable contextID's.

CDLEvent

CDLEvent(long eventID)

Fired with an event ID to indicate one of the following occurrences:

NE_PAGECHANGED = 2

Indicates that the page of a drawing or document has changed.

NE_SOMETHINGISCOPYABLE = 7

Indicates that the MT_SOURCEDOC_SELECT mouse tool has selected text that can be copied to the system clipboard OR
Indicates that a markup entity has been selected.

NE_NOTHINGISCOPYABLE = 8

Indicates that a piece of text is no longer selected OR
Indicates that a markup entity has become unselected.

NE_SOMETHINGISPASTEABLE = 9

Indicates that a markup entity has been copied to the clipboard.

NE_NOTHINGISPASTEABLE = 10

Indicates that a markup entity is no longer available from the clipboard.

NE_STARTEDEDITINGREDLINETEXT = 11

Indicates that a markup text entity has been created.

NE_STOPPEDEEDITINGREDLINETEXT = 12

Indicates that a markup text entity has been finalized.

NE_MOUSETOOLCHANGED = 15

Indicates that the mouse tool has changed. Property:*MouseTool* can be used to determine the currently active mouse tool.

NE_VIEW_CHANGED = 28

Fired when the view state of the document or drawing has changed. Note that this event is only fired if the Property:*CollaborationMode* has been set to TRUE.

CalibrationComplete

CalibrationComplete()

Fired when the mouse tool is MT_CALIBRATE and the user has completed the calibration that is required to begin using other measurement tools.

MarkupEntityModified

(BSTR MarkupID)

This event is fired when any modification to a markup entity is finalized. This includes entity creation and deletion. MarkupID is a string used to uniquely identify the markup entity that was modified. MarkupID can then be used in subsequent calls to *GetMarkupEntity* and *SetMarkupEntity*.

LayersChanged

This event is fired when the visibility of drawing layers are modified through the controls layer dialog.

ResolveISOBannerString

([in] BSTR token, [in]BSTR parameter, [in,out]BSTR result)*

This event is fired when any banner or watermark contains a replacement token that can not be resolved. The unresolved token is provided in BSTR token and any accompanying parameter is provided in BSTR parameter. Listeners to this event can provide the token replacement value in BSTR* result.

BDTXMouseDown

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

Fired when a mouse button is pressed while the cursor is over the control's view area. View coordinates are returned in X and Y. *MouseButtonID* indicates which mouse button was pressed:

LEFT_MOUSE_BUTTON_ID = 0
RIGHT_MOUSE_BUTTON_ID = 1

BDTXMouseUp

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

Fired when a mouse button is released while the cursor is over the control's view area. View coordinates are returned in X and Y. *MouseButtonID* indicates which mouse button was pressed:

LEFT_MOUSE_BUTTON_ID = 0
RIGHT_MOUSE_BUTTON_ID = 1

BDTXMouseMove

*(long *X,long *Y,BOOL LButtonDown,BOOL RButtonDown)*

Fired when the mouse is moved while the cursor is over the control's view area. View coordinates are returned in X and Y. Current mouse button states are returned in *LButtonDown* and *RButtonDown*.

MarkupEditSaved

(BSTR Filename)

Fired when a currently open editable markup is saved to local disk. Filename indicates the save location.

RequestMarkupSave

Fired when the user has attempted to save a previously saved editable markup file - and *RequestFileIOEvents* property is TRUE. The container is then responsible for calling the method:*SaveMarkup()* to save the desired editable markup file.

RequestMarkupNew

Fired when the user has attempted to create a new editable markup file - and *RequestFileIOEvents* property is TRUE. The container is then responsible for calling the method:*NewMarkup()* to complete the creation of the editable markup file.

RequestCloseMarkupEdit

Fired when the user has attempted to close an editable markup file - and *RequestFileIOEvents* property is TRUE. The container is then responsible for calling the method:*CloseMarkupEdit()* to complete the closure of the editable markup file.

RequestCloseMarkupReview

Fired when the user has attempted to close a review-only markup file - and *RequestFileIOEvents* property is TRUE. The container is then responsible for calling the method:*CloseMarkupReview()* to complete the closure of the review-only markup file.

RequestNewMarkupStampTemplate

Fired when the user has attempted to create a new editable stamp template file - and *RequestFileIOEvents* property is TRUE. The container is then responsible for calling the method: *NewMarkupStampTemplate()* to complete the creation of the editable markup file.

RequestOpenMarkupStampTemplate

Fired when the user has attempted to open an editable stamp template file - and *RequestFileIOEvents* property is TRUE. The container is then responsible for calling the method: *OpenMarkupStampTemplate (BSTR fileName)* to complete the opening of the editable stamp template file.

RequestSaveMarkupStampTemplate

Fired when the user has attempted to save an editable stamp template file - and *RequestFileIOEvents* property is TRUE. The container is then responsible for calling the method: *SaveMarkupStampTemplate ()* to complete the saving of the editable stamp template file.

RequestSaveAsMarkupStampTemplate

Fired when the user has attempted to save an editable stamp template file to a new file name - and *RequestFileIOEvents* property is TRUE. The container is then responsible for calling the method: *SaveAsMarkupStampTemplate (BSTR fileName)* to complete the saving of the editable stamp template file.

RequestCloseMarkupStampTemplate

Fired when the user has attempted to close an editable stamp template - and *RequestFileIOEvents* property is TRUE. The container is then responsible for calling the method: *CloseMarkupStampTemplate ()* to complete the closing of the editable stamp template file.

MarkupStampTemplateLoadFailure

(BSTR Filename)

Fired when an attempt to load a stamp template file has failed.

MarkupStampTemplateLoadSuccess

(BSTR Filename)

Fired when an attempt to load a stamp template file has succeeded.

MarkupStampTemplateSaved

(BSTR Filename)

Fired when a stamp template file has been saved.

MarkupStampTemplateClosed

(*BSTR Filename*)

Fired when a stamp template file has been closed.

MarkupStampTemplateCreated

Fired when a stamp template file has been created.

MarkupStampEntityLoadSuccess

(*BSTR Filename*)

Fired when an attempt to load a stamp entity file into the stamp markup tool has failed.

MarkupStampTemplateLoadSuccess

(*BSTR Filename*)

Fired when an attempt to load a stamp entity file into the stamp markup tool has succeeded.

RequestMarkupStampEntityFilename

Fired when the user has attempted to instantiate the stamp markup tool with a new stamp file - and *RequestFileIOEvents* property is TRUE. The container is then responsible for setting the property *MrkStampEntityFilename* to the full path of a valid stamp file to complete the instantiation of the stamp markup tool.

RequestRedactionScript

Fired when the user has attempted to run a redaction script - and *RequestFileIOEvents* property is TRUE. The container is then responsible for calling the method:*RedactFromScript* to provide a redaction script.

RequestLocalFilenameForDownloadOriginal

Fired when the user has attempted to copy a remote file to a local file - and *RequestFileIOEvents* property is TRUE. The container is then responsible for calling the method:*DownloadOriginalToLocalFile* to provide a file name to take the contents of the remote file.

RequestExportPDFFilename

Fired when the user has attempted to create a PDF rendition of the currently open drawing or document - and *RequestFileIOEvents* property is TRUE. The container is

then responsible for calling the method:*ExportPDF* to provide a file name to be used for the PDF rendition.

RequestExportTiffFilename

Fired when the user has attempted to create a TIFF rendition of the currently open drawing or document - and *RequestFileIOEvents* property is TRUE. The container is then responsible for calling the method:*ExportTiffEx* to provide a file name to be used for the TIFF rendition

RequestExportDWFFilename

Fired when the user has attempted to create a DWF rendition of the currently open drawing or document - and *RequestFileIOEvents* property is TRUE. The container is then responsible for calling the method:*ExportDWF* to provide a file name to be used for the DWF rendition

RequestRasterMarkupFilename

Fired when the user has attempted to use the raster markup tool and there is no raster file selected for the tool - and *RequestFileIOEvents* property is TRUE. The container is then responsible for calling the property:*RasterMrkFilename* to provide a raster file to be used in the raster markup tool.

DownloadOriginalFailure

(BSTR Filename)

Fired when the control was unable to copy the contents of remote file to local file.

DownloadOriginalSuccess

(BSTR Filename)

Fired when the control succeeded in copying the contents of a remote file to a local file.

CompareFileLoadFailure

(BSTR Filename)

Fired when the control was unable to open a document or drawing for compare.

CompareFileLoadSuccess

(BSTR Filename)

Fired when the control succeeded in opening a document or drawing for compare.

CompareFileClosed

(BSTR Filename)

Fired when the control has closed a document or drawing open for compare.

ExportPDFFailure

(BSTR Filename)

Fired when the control was unable to create a PDF rendition of the currently open drawing or document.

ExportPDFSuccess

(BSTR Filename)

Fired when the control succeeded in creating a PDF rendition of the currently open drawing or document.

ExportTiffFailure

(BSTR Filename)

Fired when the control was unable to create a TIFF rendition of the currently open drawing or document.

ExportTiffSuccess

(BSTR Filename)

Fired when the control succeeded in creating a TIFF rendition of the currently open drawing or document.

ExportDWFFailure

(BSTR Filename)

Fired when the control was unable to create a DWF rendition of the currently open drawing or document.

ExportDWFSuccess

(BSTR Filename)

Fired when the control succeeded in creating a DWF rendition of the currently open drawing or document.

RasterMarkupLoadFailure

(BSTR Filename)

Fired when the control was unable to open a raster file for use with the raster markup tool.

RasterMarkupLoadSuccess

(BSTR Filename)

Fired when the control succeeded in opening a raster file for use with the raster markup tool.

RequestInfoMessageDisplay

([in] long MessageID, [in]BSTR MessageText, [in,out]long Display)*

Fired just prior to the control displaying an error or information message. The message is identified by *MessageID*. See *MessageID* section for a list of possible values. *MessageText* will contain the text that will appear in the message dialog. If a container sets the value of *Display* to 0, the message will not be displayed.

ControlInitialized

(long *EventID*)

Fired when the control's windows are created. Parameter *EventID* is not currently used.

RequestExportCSFFilename

(*BOOL BurningMarkup*)

This event is only fired if the property *UseStandardCSFOutput* has been set to TRUE. Fired when the user has attempted to create a CSF rendition of the currently open drawing or document, *RequestFileIOEvents* property is TRUE. The container is then responsible for calling the method: *ExportCSF* to provide a file name to be used for the CSF rendition.

BeforeCopy

([in,out]long* *Continue*)

This event is fired immediately before the control copies something (e.g. text, markup entity) to the Windows Clipboard. A listener to this event can cause the copy to not be performed if the parameter *Continue* is set to 0.

BeforePaste

([in,out]long* *Continue*)

This event is fired immediately before the control pastes something (e.g. text, markup entity) from the Windows Clipboard. A listener to this event can cause the paste to not be performed if the parameter *Continue* is set to 0.

AfterCopy()

This event is fired immediately after the control has completed a copy to clipboard command.

AfterPaste()

This event is fired immediately after the control has completed a paste from clipboard command.

XDLIsComplete()

This event applies when the control is receiving drawings and documents from the Brava server. In some cases, files being published by the Brava Server will be sent to the Brava Viewer control before the server has completely published the file. In these cases, this event will be fired after the server has completed the entire publishing job.

PrintJobComplete()

This event is fired when the control has completed the spooling of a document to a printer.

MetadataEntitySelected

(BSTR Info, BSTR EntityData, long DisplayInfo)*

Fired when a metadata entity such as a AutoCad Block is clicked while the Selection or Pan tools are active. The Info parameter will be a string containing information about the entity clicked. The EntityData parameter is not currently used. If a listener to this event sets the value of DisplayInfo to 0, the control will not display the message box containing the string Info.

MetadataEntityHovered

(BSTR Info, BSTR EntityData, long DisplayInfo)*

Fired when a metadata entity such as an AutoCad Block is hovered over while the Selection or Pan tools are active. The Info parameter will be a string containing information about entity hovered. The EntityData parameter is not currently used. If a listener to this event sets the value of DisplayInfo to 0, the control will not display the pop-up message box containing the string Info.

ChangemarkExecuted

(BSTR CMID)

Fired when a Changemark entity is either selected from the Changemark review panel, clicked on with the selection tool active, or executed via Method ExecuteChangemark. CMID is a unique identifier for the Changemark executed.

MessageID Section

MessageID's

BRAVAX_UNSPECIFIED_ERROR	= 1,
BRAVAX_INVALID_DLFILE_EXT	= 2,
BRAVAX_INVALID_XDLFILE	= 3,
BRAVAX_INVALID_PAGENUMBER	= 4,
BRAVAX_CANT_READ_PAGE	= 5,
BRAVAX_CANT_LOCALIZE_FILE	= 6
BRAVAX_INVALID_ELEMENT_RECORD	= 7,
BRAVAX_CANT_CONSTRUCT_IO_CLASS	= 8,
BRAVAX_INVALID_FILE_ID_TAG	= 9,
BRAVAX_CANT_WRITE_DLPAGE	= 10,
BRAVAX_INVALID_MRKFILE_EXT	= 11,
BRAVAX_CANT_READ_MRKFILE	= 12,
BRAVAX_FUTURE_CDL_FILE_FORMAT	= 13,
BRAVAX_FUTURE_MRK_FILE_FORMAT	= 14,
BRAVAX_FORMAT_NOT_LICENSED	= 15,
BRAVAX_LOADER_NOT_FOUND	= 16,
BRAVAX_NO_PRINTER	= 17,

BRAVAX_CANT_FIND_SUPPORT_FILE	= 18,
BRAVAX_CANT_FIND_XREF_FILE	= 19,
BRAVAX_INTEGRATION_ERROR	= 20,
BRAVAX_FILELOAD_ERROR	= 21,
BRAVAX_MRKLOAD_ERROR	= 22,
BRAVAX_RASTERMRKLOAD_ERROR	= 23,
BRAVAX_INVALID_REDACTION_SCRIPT_ERROR	= 24,
BRAVAX_CANT_RESOLVE_MRKFILE	= 25,
BRAVAX_INTEGRATION_ERROR_INITIALIZATION	= 26,
BRAVAX_INTEGRATION_ERROR_CLIENTID	= 27,
BRAVAX_INTEGRATION_ERROR_FROM_SERVER	= 28,
BRAVAX_NATIVEFILE_INCORRECT_PASSWORD	= 29,
BRAVAX_MRKSTAMPENTITYLOAD_ERROR	= 30,
BRAVAX_INTERNALREFERENCEFILELOAD_ERROR	= 31,
BRAVAX_DOTNET3_NOT_INSTALLED_ERROR	= 32,
BRAVAX_SR_INCORRECT_PASSWORD	= 101,
BRAVAX_SR_FILE_EXPIRED	= 102,
BRAVAX_SR_INVALID_FILE	= 103,
BRAVAX_SR_3D_SCF	= 104,
BRAVAX_SR_START_DATE_LATER	= 105,
BRAVAX_SR_REDACTION_PASSWORD	= 106,
BRAVAX_SR_YOU_CHEATED	= 107,
BRAVAX_INVALID_LICENSE	= 200,
BRAVAX_LICENSE_EXPIRED	= 201,
BRAVAX_LM_REQUEST_RESULT_FAILED	= 202,
BRAVAX_LM_REQUEST_RESULT_REVOKED	= 203,
BRAVAX_LM_REQUEST_RESULT_EXPIRED	= 204,
BRAVAX_LM_REQUEST_RESULT_NO_SEATS_AVAILABLE	= 205,
BRAVAX_LM_REQUEST_RESULT_GENERAL_ERROR	= 206,
BRAVAX_LM_REQUEST_RESULT_COMMUNICATION_ERROR	= 207,
BRAVAX_REGISTER_LICENSE_NO_INTERNET_CONNECTION	= 208,
BRAVAX_REGISTER_LICENSE_UNABLE_TO_CONTACT_HOST	= 209,
BRAVAX_REGISTER_LICENSE_UNABLE_TO_GENERATE_REQUEST_INFO	= 210,
BRAVAX_REGISTER_LICENSE_COULD_NOT_RETRIEVE_RESPONSE	= 211,
BRAVAX_REGISTER_LICENSE_INVALID_RESPONSE_FROM_HOST	= 212,
BRAVAX_REGISTER_LICENSE_LICENSE_VERIFICATION_FAILED	= 213,
BRAVAX_REGISTER_LICENSE_GENERAL_ERROR	= 214,
BRAVAX_REGISTER_LICENSE_UNABLE_TO_UPDATE_LICENSE_LOCALLY	= 215,
BRAVAX_REGISTER_LICENSE_ALREADY_REGISTERED	= 216,
BRAVAX_REGISTER_LICENSE_HOST_UNABLE_TO_DECODE_LICENSE	= 217,
BRAVAX_REGISTER_LICENSE_HOST_DATABASE_UPDATE_FAILED	= 218,
BRAVAX_REGISTER_LICENSE_HOST_UNABLE_TO_REENCODE_LICENSE	= 219,
BRAVAX_REGISTER_LICENSE_HOST_UNABLE_TO_GENERATE_PIN	= 220,
BRAVAX_REGISTER_LICENSE_HOST_UNABLE_TO_GENERATE_DEMO_LICENSE	= 221,
BRAVAX_NO_DESIGNTIME_LICENSE	= 222,
BRAVAX_COULD_NOT_ACCESS_LICENSE_FILE	= 223,
BRAVAX_INVALID_LICENSE_FROM_BRAVA_SERVER	= 224,
BRAVAX_LICENSE_EXPIRATION_WARNING_PERIOD	= 225,

Control Identification Section

The following are the integer identifiers for all customizable controls:

Toolbar controls:

Page Toolbar Controls:

IDC_COMBO_PAGE	233
ID_BUTTON_ROTATION	32779
ID_BUTTON_LAYERS	32780
ID_BUTTON_MONOCHROME	32781
ID_BUTTON_BACKCOLOR	32782
ID_BUTTON_PREVPAGE	32783
ID_BUTTON_NEXTPAGE	32784

Mouse Tools Toolbar Controls:

ID_BUTTON_PAN	32768
ID_BUTTON_ZOOMWIN	32769
ID_BUTTON_ZOOM	32770
ID_BUTTON_MAGNIFIER	32771
ID_BUTTON_SHOWTHUMBNAILS	32875
ID_BUTTON_ABOUT	32772

View Tool Toolbar Controls:

IDC_COMBO_TB_SCALE	344
ID_BUTTON_MARKUP	32773
ID_BUTTON_MARKUPSAVE	32903
ID_BUTTON_PRINT	32774
ID_BUTTON_MEASURE	32775
ID_BUTTON_SELECT	32776
ID_BUTTON_EXTENTS	32777
ID_BUTTON_WIDTH	32778
ID_BUTTON_DOWNLOADORIGINAL	32836
ID_BUTTON_SAVEAS	32840

Markup Tool Controls:

ID_MRKTOOL_SELECT	503
ID_MRKTOOL_ARROW	505
ID_MRKTOOL_TEXT	507
ID_MRKTOOL_CHANGEMARK	509
ID_MRKTOOL_RASTERFILE	880
ID_MRKTOOL_STAMPFILE	1158
ID_MRKTOOL_CLOUD	511
ID_MRKTOOL_SKETCH	514
ID_MRKTOOL_LINE	517
ID_MRKTOOL_SHAPE	520
ID_MRKTOOL_REDACT	522
ID_MRKTOOL_TEXTMODIFY	523
ID_MRKTOOL_HYPERLINK	512

Menu Item IDs

Measurement menu items

ID_MENU_MEASURE_CALIBRATE	32785
ID_MENU_MEASURE_SETTINGS	32789
ID_MENU_MEASURE_LINE	32950
ID_MENU_MEASURE_POLYLINE	32951
ID_MENU_MEASURE_POLYGON	32952
ID_MENU_MEASURE_RECT	32953
ID_MENU_MEASURE_CIRCLE	32954
ID_MENU_MEASURE_COUNT	32955

Banner and watermark menu items

ID_MENU_ISOBANNER_DATE	32790
ID_MENU_ISOBANNER_SYSDATEPLUSDAYS	32791
ID_MENU_ISOBANNER_TIME	32792
ID_MENU_ISOBANNER_MILTIME	32793
ID_MENU_ISOBANNER_TITLE	32794
ID_MENU_ISOBANNER_PAGES	32795
ID_MENU_ISOBANNER_TOTALPAGES	32796
ID_MENU_ISOBANNER_BATESPGNO	32985
ID_MENU_ISOBANNER_LOGIN	32797
ID_MENU_ISOBANNER_USER	32798
ID_MENU_ISOBANNER_PERCENT	32799
ID_MENU_ISOBANNER_DBSTRING	32855

Find menu items

ID_MENU_FIND_DOWN	32801
ID_MENU_FIND_UP	32802
ID_MENU_FIND_MATCHCASE	32803

Right-click menu items

ID_MENU_RP_BACKVIEW	32890
ID_MENU_RP_FITALL	32809
ID_MENU_RP_FITWIDTH	32810
ID_MENU_RP_PANTOOL	32877
ID_MENU_RP_VIEWPINPOP	32885
ID_MENU_RP_ADDVIEWPIN	32886
ID_MENU_RP_NEXTVIEWPIN	32887
ID_MENU_RP_REMOVEALLVIEWPINS	32889
ID_MENU_RP_REVIEWCM	32811
ID_MENU_RP_PAGCTRLPOP	32892
ID_MENU_RP_NEXTPAGE	32819
ID_MENU_RP_PREVPAGE	32818
ID_MENU_RP_PAGCTRLFIRST	32894
ID_MENU_RP_PAGCTRLLAST	32893
ID_MENU_RP_PREVMRK	32823
ID_MENU_RP_NEXTMRK	32824
ID_MENU_RP_COPYREGION	32883
ID_MENU_RP_COPY	32813
ID_MENU_RP_PASTE	32814
ID_MENU_RP_DELETE	32815
ID_MENU_RP_UNDO	32870
ID_MENU_RP_REDO	32871

ID_MENU_RP_ROTATEMIRRORPOP	32882
ID_MENU_RP_ROTATECLOCKWISE	32880
ID_MENU_RP_ROTATECOUNTERCLOCKWISE	32881
ID_MENU_RP_MIRROR	32841
ID_MENU_RP_BKCOLORPOP	32865
ID_MENU_RP_BKCOLORPOP_BLACK	32851
ID_MENU_RP_BKCOLORPOP_WHITE	32852
ID_MENU_RP_BKCOLORPOP_GRAY	32853
ID_MENU_RP_BKCOLORPOP_DEFAULT	32854
ID_MENU_RP_OPTIONSPOP	32884
ID_MENU_RP_ANIMATE	32817
ID_MENU_RP_LINEWTS	32842
ID_MENU_RP_ANTIALIAS	32856
ID_MENU_RP_SHOWTHUMBNAILS	32869

Markup menu items

ID_MENU_MRK_OPENREVIEW	32820
ID_MENU_MRK_CLOSEREVIEW	32821
ID_MENU_MRK_REVIEWCM	32822
ID_MENU_MRK_PREVMRK	32825
ID_MENU_MRK_NEXTMRK	32826
ID_MENU_MRK_NEWEDIT	32827
ID_MENU_MRK_OPENEDIT	32828
ID_MENU_MRK_SAVEEDIT	32829
ID_MENU_MRK_SAVEDXFEDIT	32830
ID_MENU_MRK_CLOSEEDIT	32831
ID_MENU_MRK_SAVEASEEDIT	32832
ID_MENU_MRK_CONSOLIDATE	32833
ID_MENU_MRK_REVIEW_MRK_PAGES	32863
ID_MENU_MRK_STAMPTEMPLATEPOP	32897
ID_MENU_MRK_NEWSTAMPTEMPLATE	32898
ID_MENU_MRK_OPENSTAMPTEMPLATE	32899
ID_MENU_MRK_SAVESTAMPTEMPLATE	32900
ID_MENU_MRK_SAVEASSTAMPTEMPLATE	32901
ID_MENU_MRK_CLOSESTAMPTEMPLATE	32902

Markup Text Background Options menu items

ID_MENU_MRKTEXTBACKGROUND_SELECT	32965
ID_MENU_MRKTEXTBACKGROUND_TRANSPARENT	32966
ID_MENU_MRKTEXTBACKGROUND_OPAQUE	32967
ID_MENU_MRKTEXTBACKGROUND_CURRENT	32968

Help-About menu items

ID_MENU_HELPABOUT_CONTENTS	32837
ID_MENU_HELPABOUT_ABOUT	32838
ID_MENU_HELPABOUT_SUGGEST	32839

Print menu items

ID_MENU_PRINT_PRINTDLG	32843
ID_MENU_PRINT_PRINTRGN	32844
ID_MENU_PRINT_BANNEREDIT	32845

Background color menu items

ID_MENU_BKCOLORPOP_BLACK	32847
--------------------------	-------

ID_MENU_BKCOLORPOP_WHITE	32848
ID_MENU_BKCOLORPOP_GRAY	32849
ID_MENU_BKCOLORPOP_DEFAULT	32850

Publish-export menu items

ID_MENU_SAVEAS_CSFPUBLISH	32866
ID_MENU_SAVEAS_PDFPUBLISH	32867
ID_MENU_SAVEAS_TIFFPUBLISH	32876
ID_MENU_SAVEAS_DWFPUBLISH	32896
ID_MENU_SAVEAS_SAVEASJPG	32868

Thumbnail menu items

ID_MENU_THUMB_REDUCE	32872
ID_MENU_THUMB_ENLARGE	32873
ID_MENU_THUMB_DEFAULT_SIZE	32874

Help Topic IDs

#define ZOOM_WINDOW	76
#define ZOOM_AND_PAN_TOOLS	75
#define WELCOME_TO_BRAVA_	74
#define WATERMARK	73
#define VISIBLE_LAYERS	72
#define VIEW_TOOLS	71
#define TAGS	68
#define STATUS_BAR	67
#define STANDARD_TOOLBAR	66
#define SOLID_AND_HOLLOW_SHAPES	65
#define SKETCH	64
#define SELECT_TOOL	63
#define SELECT_ENTITY_TOOL	62
#define SEARCH	61
#define SCRATCH	60
#define SAVING_MARKUPS	59
#define ROTATE	58
#define REVIEWING_CHANGEMARKS	57
#define RECTANGLE	56
#define PRINT_TO_SCALE	55
#define PRINT_TIPS_AND_TROUBLESHOOTING	54
#define PRINT_REGION	53
#define PRINT_A_FILE	52
#define PRINT	51
#define POLYLINE	50
#define POLYGON	49
#define PAN	48
#define PAGE_CONTROL	47
#define OVERLAY_MARKUPS_FOR_REVIEW	46
#define OPEN_A_DOCUMENT	45
#define MONOCHROME	44
#define MEASUREMENT_SETTINGS	43
#define MEASURE_COUNT	42
#define MEASURE	41
#define MARKUP_TOOLBAR	40

#define MARKUP_PROPERTIES	39
#define MARKUP_PAGES	38
#define MARKUP_LINE_WIDTH	37
#define MARKUP_FONT	36
#define MARKUP_FILES	35
#define MARKUP_COLOR	34
#define MAGNIFY	33
#define LINETHICKNESS	32
#define LINE_ARROW	30
#define LINE	29
#define LINE1	31
#define ISO_BANNERS	28
#define IMAGE_BASED	27
#define HYPERLINK	26
#define HOTKEYS	25
#define HIGHLIGHTS	24
#define HIDES	23
#define FONT_CHOOSER	22
#define FIT_WIDTH	21
#define FIT_ALL	20
#define EDITING_MARKUPS	19
#define EDITABLE_MARKUPS	18
#define EDIT_TEXT	17
#define DOCUMENT_PAGE_CONTROL	15
#define DISPLAY_OPTIONS	14
#define CROSSOUT	12
#define CREATING_MARKUPS	11
#define CREATING_CHANGEMARKS	10
#define COPY_TEXT	9
#define CONTACT_INFORMATION	8
#define CLOUD	7
#define CLOSING_MARKUPS	6
#define CHANGEMARKS_FILTERS	5
#define CALIBRATE	4
#define BACKGROUND_COLOR	3
#define ARC	2
#define WATERMARK1	1
#define ALLTOOLBARS	13
#define REDACTION	16
#define DLTEXT	69
#define BURNED_IN_MARKUPS	70
#define RIGHTMOUSE	77
#define CREATE_CSF	78
#define PUBLISHING_WITH_BRAVA_DESKTOP	79
#define SECURITY_TYPE	80
#define STARTEXPIRE	81
#define WHAT_IS_CSF_	82
#define UPDATE_VERSION	83
#define BLOCK_OUT	84
#define MIRROR	85
#define SAVE_VIEW	86
#define BACKGROUND_FILLS	87
#define DOWNLOADING_TOOL	88
#define ALIGN_COMPARISON_TOOL	89
#define OPEN_FOR_COMPARISON	90
#define WHY_CSF_	91

#define MEASURE_POLYLINE	92
#define SET_PATHS	93
#define SEND_FILE	94
#define LAUNCH_AUTOCAD	95
#define FILE_TYPES	96
#define COPY_REGION	97
#define FIND_AND_REDACT	98
#define PUBLISHING_MARKUPS_TO_PDF_FILES	99
#define MEASURE_MAGNIFICATION_TOOL1	100
#define INSERT_RASTER_IMAGE	101
#define REDACT_BY_SCRIPT	102
#define REDACT_MOUSE_TOOL	103
#define PUBLISHING_FILES_TO_PDF	104
#define BURNING_IN_MYRIAD_MARKUPS	105
#define BRAVA_DESKTOP_SKU_S	106
#define THUMBNAIL_PAGES	107
#define PUBLISHING_FILES_TO_DWF	108
#define PUBLISHING_FILES_TO_TIFF	109
#define PUBLISHING_FILES	110
#define BOOKMARKS	111
#define CUSTOMIZATION	112
#define CIRCLE	113
#define EXTRACT_CHANGEMARKS	114
#define MARKUP_STAMP	115
#define STAMP_TEMPLATES	116
#define MARKUP_TEXT_BACKGROUND	117
#define VIEWXPS_IDW	118
#define USING_SEARCH_MACROS	119

Html Parameters

The following properties can be set via html parameters:

- Filename
- RequestFileIOEvents
- RequestFileIOIntegration
- MrkReviewFilename
- MrkEditFilename
- CurrentPageNumber
- SearchText
- DisplayName
- UserName
- DateFormat
- TimeZone
- AllowFileOpen
- AllowPrinting
- AllowMarkup

AllowMeasurement
 AllowLayers
 AllowFind
 AllowCopyText
 AllowMrkSaveAsDXF
 AllowMrkSaveAs
 AllowMrkSave
 AllowMrkOpen
 AllowMrkNew
 AllowPageControl
 AllowRotate
 AllowBKColor
 AllowMonochrome
 EnableMarkupColorPalette
 MarkupColor
 WatermarkBannerFontName
 WatermarkBannerFontStyle
 BannerFontSize
 ScreenBanner
 ScreenWatermark
 Watermark
 TopLeft
 TopCenter
 TopRight
 BottomLeft
 BottomCenter
 BottomRight
 LeftTop
 LeftCenter
 LeftBottom
 RightTop
 RightCenter
 RightBottom

Control Key Combinations

To be used with the *EnableAcceleratorKey* method.

Combination	Action	KeyID Value
CTRL + A	Show about	1
CTRL + SHIFT + A	Change to pan tool	1
CTRL + SHIFT + B	Burn in markup	2
CTRL + ALT + B	Add a bookmark	1114178
CTRL + B	Toggle through bookmarks	2
ALT + B	Toggle through background colors	65602

CTRL + C	Copy selected	3
CTRL + SHIFT + C	Close editable markup	3
CTRL + SHIFT + D	Export PDF	4
CTRL + E	Fit all	5
CTRL + F	Mirror raster	6
CTRL + SHIFT + F	Export DWF	6
CTRL + G	Set print region tool	7
CTRL + H	Show help	8
CTRL + SHIFT + J	Save View as JPG	10
CTRL + K	Export CSF	11
CTRL + L	Show Layer dialog	12
CTRL + M	Open editable markup	13
CTRL + N	New editable markup	14
CTRL + O	Open file	15
CTRL + P	Show print dialog	16
CTRL + R	Open review markup	18
CTRL + SHIFT + R	Close review markup	18
CTRL + S	Save editable markup	19
CTRL + SHIFT + S	SaveAs editable markup	19
CTRL + T	Toggle thumbnail display	20
CTRL + SHIFT + T	Export TIFF	20
CTRL + V	Paste	22
CTRL + W	Fit width	23
CTRL + X	Delete	24
CTRL + SHIFT + X	Change to zoom tool	24
CTRL + Y	Redo	25
CTRL + Z	Undo	26
CTRL + SHIFT + Z	Change to magnifier tool	26
CTRL + SPACE	Rotate 90 degrees clockwise	32
CTRL + SHIFT + SPACE	Rotate 90 degrees counter-clockwise	32
Page Up	Previous page	33
CTRL + Page Up	Previous markup page	1048609
Page Down	Next page	34
CTRL + Page Down	Next markup page	1048610
End	Last page	35
Home	First page	36
CTRL + ALT + LeftArrow	Back View	1114149
CTRL + ALT + RightArrow	Forward View	1114151
CTRL + LeftArrow	Nudge compare view left	1048613
CTRL + UpArrow	Nudge compare view up	1048614
CTRL + RightArrow	Nudge compare view right	1048615
CTRL + DownArrow	Nudge compare view down	1048616
CTRL + Tab	Toggle older and newer compare docs	1048585
F3	Find Next Search string instance	114