OPENTEXT™

OpenText™ Brava!™ Enterprise

ActiveX API Interface Details Guide

This guide is intended for Brava! Enterprise developers, programmers, and integrators. It provides information for configuring the method and event options for the Brava! Enterprise ActiveX client API.

Brava! ActiveX API Interface Details Guide

Rev.: January 19, 2017

This documentation has been created for Brava! Enterprise software version 16.2.
It is also valid for subsequent software versions as long as no new document version is shipped with the product or is published at https://knowledge.opentext.com.

Open Text Corporation
275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1
Tel: +1-519-888-7111
Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677
Support: https://support.opentext.com
For more information, visit https://www.opentext.com

Disclaimer

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, Open Text Corporation and its affiliates accept no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

Table of Contents

BravaX API Interface Details Guide

Purpose

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

The audience for this document includes developers, programmers, and integrators with a solid knowledge of COM API programming.

To view a list of newly added and deprecated properties, methods, and events for this release, please refer to the product release notes document contained in your product documentation folder.

For the latest documentation updates please be sure to visit our product page on OpenText MySupport:

https://knowledge.opentext.com/knowledge/llisapi.dll/open/58922802

You can also visit the Brava! web page: OpenText™ Brava!™

FIle IO Options Section

File IO Options: Properties

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.

RequestFileIOEvents	
Type	Default
Boolean	`FALSE`
Description	
This property is used to get/set if the container is listening for I/O events. If set to TRUE, the control will fire an event to indicate that it is requesting a file to be opened, closed, or saved. The application containing the control will then be expected to handle all file IO related operations. See Events Section.	

File Access Section

File Access: Properties

Filename	
Type	Example
String	`viewObject.Filename=”C:/files/somefile.csf"`
Description	
This property is used to get/set the file currently being viewed.	

MrkReviewFilename	
Type	Example
String	`viewObject.MrkReviewFilename = "C:/files/somefile.xrl"`
Description	
This property is used to get/set the markup file currently being viewed as read-only. Because multiple read-only markups can be opened, this property is an asterisk delimited list of filenames.	

MrkEditFilename	
Type	Example
String	`viewObject.MrkEditFilename = "C:/files/somefile.xrl"`
Description	
This property is 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. If there is currently an editable markup file open that is in an unsaved state, the control will prompt the user to the save the markup.	

AllowDragAndDropFileOpen	
Type	Default
Boolean	`TRUE`
Description	

This property is used to get/set the control's ability to allow files to be dragged and dropped into its client area for opening.

DownloadOriginal	
Type	Example
String	
Description	
This property is used to get/set the URL to launch when the control is running in a webpage and the file being viewed is remote. If this property is not set, clicking the Download Original button in the user interface will prompt the user to copy the currently viewed file to local disk.	

FileIsCDL	
Type	Default
Boolean	Read-Only
Description	
This property indicates if the currently open file is a CSF file.	

File Access: Methods

CloseFile
Syntax
`CloseFile()`
Description
This method closes the file currently being viewed and any markup files currently open. If there is currently an editable markup file open that is in an unsaved state, the control will prompt the user to the save the markup.

OpenMarkupReview
Syntax
`OpenMarkupReview( [in] BSTR filename)`
Description

This method causes the control to attempt to open a markup review file indicated by `filename`.			
Parameters	I/O	Type	Description
`filename`	In	String	Name of file to open.

CloseMarkupReview			
Syntax			
`CloseMarkupReview( [in] int index)`			
Description			
This method causes the control to close a particular markup review file indicated by `index` .			
Parameters	I/O	Type	Description
`index`	In	Integer	Markup review files are indexed in the order in which they were opened, with the first having an `index` of 0. 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

CloseAllMarkupReview
Syntax
`CloseAllMarkupReview()`
Description
This method causes all markup files currently open for review to be closed.

NewMarkup
Syntax
`NewMarkup()`
Description
This method causes the control to create a new, un-named, editable markup.

OpenMarkupEdit
Syntax
`OpenMarkupEdit( [in] BSTR filename)`
Description

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. If there is currently an editable markup file open that is in an unsaved state, the control will prompt the user to the save the markup before closing it.

Parameters	I/O	Type	Description
`filename`	In	String	Because only one markup edit file can be opened, `filename` is a single file name.

SaveMarkup
Syntax
`SaveMarkup()`
Description
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 Options Section.

SaveAsMarkupEdit
Syntax
`SaveAsMarkupEdit( [in] BSTR filename)`
Description
This method saves the current state of the currently open editable markup. The markup will be saved in the file indicated by `filename`.

Parameters	I/O	Type	Description
`filename`	In	String	Because only one markup edit file can be opened, `filename` is a single file name.

CloseMarkupEdit
Syntax
`CloseMarkupEdit( [in] BOOL promptSave, [out] BOOL * Canceled)`
Description
This method closes any currently open editable markup file.

Parameters	I/O	Type	Description
`promptSave`	In	Boolean	If set to TRUE, and there is currently an editable markup file open that is in an unsaved state, the control will prompt the user to save. This prompt allows the user to cancel the close of the markup.
`Canceled`	Out	Boolean	If the user chooses to cancel, `Canceled` will be set to TRUE upon

			return of the method.

CloseMarkupEditEx			
Syntax			
`CloseMarkupEditEx( [in] BOOL promptSave, [In] IBxboolAnswer * Canceled)`			
Description			
This method closes any currently open editable markup file.			
Parameters	I/O	Type	Description
`promptSave`	In	Boolean	If set to TRUE, and there is currently an editable markup file open that is in an unsaved state, the control will prompt the user to save. This prompt allows the user to cancel the close of the markup.
`Canceled`	Out	Boolean Answer	`Canceled.Answer` Is set to TRUE if the user chooses to cancel the markup close.

DownloadOriginalToLocateFile			
Syntax			
`DownloadOriginalToLocalFile( [in] BSTR LocalFileName)`			
Description			
If the currently viewed drawing or document is remote, this method copies that file to the path and filename defined in `LocalFileName`.			
Parameters	I/O	Type	Description
`LocalFileName`	In	String	A path and filename on the local machine.

AppendFile
Syntax
`AppendFile( [in] BSTR filename)`
Description
This method appends the contents of the document specified by `filename` to the currently open file. For this method to succeed, the following conditions must be met: 1. There must be a document open 2. There must not be a markup file open 3. The viewer must not be in Compare mode 4. The file specified by `filename` must be a licensed file type 5. Appending must be licensed 6. The open file may not be a CSF file

7. The file being appended may not be a CSF file

If successful, all pages of the appended document are added to the end of the open document. Appended documents cannot be removed from the document without closing. Reopening the original document causes the control to save the current state of the currently open editable markup. Note that if the appended file contains XREFs that cannot be located, a notification alert is not displayed to the user.

Parameters	I/O	Type	Description
`filename`	In	String	A path with file name.

GetDocumentExtension

Syntax

```
GetDocumentExtension( [in] BSTR Filepath, [out, retval] BSTR *Ext)
```

Description

This method will attempt to examine a given file and return the file's normal extension string.

Parameters	I/O	Type	Description
`Filepath`	In	String	Full path of the local file to examine.
`Ext`	Out/Retval	String	If the examination is successful, this string will contain the normal extension for the given file.

GetDocumentContent

Syntax

```
GetDocumentContent( [in] BSTR Filepath, [out, retval] int *Content)
```

Description

This method will examine a given file and return a value indicating that the file contains 2D data, 3D data, or both 2D and 3D data.

Parameters	I/O	Type	Description
`Filepath`	In	String	Full path of the local file to examine.
`Content`	Out/Retval	Integer	Integer value indicating the file content. Possible values are: **Value** / **Description** 0 — No data found 1 — Only 2D data found

			2 Only 3D data found 3 Both 2D and 3D data found

Section Note

Any File Access 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

SaveViewAs
SaveViewAsEx
DownloadOriginalToLocalFile
ExportPDF
ExportPDFEx
ExportTiff
ExportTiffEx
ExportCSF
ExportCSFSilent
ExportCSFEx

Mouse Tool Section

Mouse Tool: Properties

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

MouseTool	
Type	Example
Short (Integer)	`MouseTool=1`
Description	

This property is used to get/set the currently active mouse tool and cursor. Setting of `MouseTool` requires that the control currently has loaded a document or drawing. Mouse tools that create markup entities can only be set if the control has a markup currently open for edit.

The following are acceptable values:

Value	MouseTool	Value	MouseTool	Value	MouseTool
1	Select	21	Markup crossout	41	Markup byzantine arrow
2	Pan	22	Markup strikeout text	42	Markup takeoff line
3	Zoom	23	Markup underline text	43	Markup takeoff polyline
4	Rotate	24	Markup highlight text	44	Markup takeoff polygon
5	Magnifier	25	Markup Changemark entity	45	Markup takeoff rectangle
6	Calibrate	26	Markup polygonal cloud	46	Markup takeoff circle
7	Measure line	27	Markup cloud	47	Markup takeoff count
8	Measure polygon	28	Markup image	48	Markup Changemark cloud
9	Measure rectangle	29	Print region	49	Markup Changemark highlight text
10	Markup edit	30	Dynamic zoom	50	Markup highlight

11	Markup sketch	31	Markup squiggle rectangle	51	Markup Changemark highlight
12	Markup polygon	32	Measure count	52	Markup Changemark arrow
13	Markup text	33	Markup redact area	53	Markup strikethrough text
14	Markup ellipse	34	Align compare documents	54	Markup Blockout
15	Markup arc	35	Measure polyline	55	Markup Signature
16	Markup rectangle	36	Copy region	56	Markup Checkview
17	Markup polyline	37	Measure circle	57	Markup Changemark strikethrough
18	Markup line	38	Markup stamp		
19	Markup arrow	39	Markup rounded rectangle		
20	Markup sketch polygon	40	Markup allow area		

Mouse Tool: Methods

CopySelection
Syntax
`CopySelection()`
Description
If the currently active mouse tool is `Select` and a section of the document or drawing's text or markup text is selected, this method copies the selected text into the system clipboard. If a markup entity or group of entities is selected, this method copies them to the system clipboard.

DeleteSelection
Syntax
`DeleteSelection()`
Description

This method deletes any markup entity or group of entities currently selected.

Paste
Syntax
`Paste()`
Description
This method causes a markup entity or group of entities, previously copied into the system clipboard, to be pasted from the clipboard. If currently editing a text entity, the text from the clipboard is pasted at the caret position.

View Control Section

View Control: Properties

Monochrome	
Type	Default
Boolean	`FALSE`
Description	
This property is used to get/set if the current view is drawn as monochrome. Use true to cause the current view to display only in black and white.	

BackgroundColor	
Type	Example
Short (integer)	`BackgroundColor=0`
Description	

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

Value	**BackgroundColor**
0	Default, based on the document format
1	Black
2	White
3	Gray

ShowAbout	
Type	Default
Boolean	`TRUE`
Description	
This property is used to get/set if the control displays the About splash screen. True causes the "About" splash screen to display.	

EnableAnimation	
Type	Default
Boolean	`FALSE`
Description	
This property is used to get/set whether the control animates between view states. False disables animation.	

AnimationTime	
Type	Examples

Long (integer)	`viewObject.AnimationTime = 0`
	Disables animation.
	`viewObject.AnimationTime = 1000`
	Only allows animations that can be completed in 1 second or less

Description	
This property is used to get/set the amount of time the control will allow for any animation between view states. The value is measured in milliseconds.	

EnableQuickDraw	
Type	Default
Boolean	`FALSE`
Description	
This property is used to get/set if 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. False indicates to always draw at full image quality.	

DisplayLoadStatus	
Type	Default
Boolean	`TRUE`
Description	
This property is used to get/set whether the control displays file load and page change status animation. The status animation appears in the upper left corner of the control. True displays status during file load and page changes.	

EnhancedRenderMode	
Type	Default

Boolean	TRUE
Description	
This property is used to get/set if the control uses anti-aliasing to draw vector lines. If set to TRUE, anti-aliasing is used.	

DisplayThumbnailPanel	
Type	Default
Boolean	<Deprecated>
Description	
This property is used to get/set the visibility state of the thumbnail panel. `DisplayThumbnailPanel` has been deprecated. Please use the `SetDisplayPanel()` and `GetDisplayPanel()` methods.	

ThumbnailSize	
Type	Example
Integer	`ThumbnailSize=75`
Description	
This property is used to get/set the display size for individual thumbnails in the thumbnail panel. The size is measured in pixels. Valid range is 20 to 185 pixels.	

SearchText	
Type	Example
String	`SearchText=any text search string`
Description	
This property is used to get/set the text to search. Note that the currently open drawing or document must contain searchable text.	

Note the following API methods are available for text selection (search). Please see the *BravaExtensibleInterface_UserGuide.pdf* for detailed information on these methods.

`TextSel_GetPageText`

`TextSel_GetSelection`

`TextSel_SetSelection`

`TextSel_GetTermHits`

`TextSel_SetTermHits`

DisplayName	
Type	Example
String	`DisplayName=MyFile`
Description	
This property is used to get/set the text to display in the tool information area. By default, this area displays the filename of the currently open document or drawing.	

UserName	
Type	Example
String	`UserName=NewUser`
Description	
This property is used to get/set the current user name. By default the control queries the user name from the operating system. If the `UserName` property is set to "NewUser1" any markup entities subsequently created will use "NewUser1" as the author name	

DateFormat	
Type	Default
String	`DateFormat=%c`
Description	

This property is used to get/set the format of the date display. The control will call back to the viewer to resolve all date time strings and the viewer will apply the format. Note that `DateFormat` only applies to places where the date and time are shown together, such as the Changemark display, the appended redaction and Changemark notes pages, etc. `DateFormat` does not apply to the `%date` and `%time` tokens in banners and watermarks as `%date` shows only date and `%time` shows only time. Date strings can be displayed in markup entity information, Changemark information, and Watermark/Banners.

Value	Description	Range
%a	Abbreviated weekday name	
%A	Full weekday name	
%b	Abbreviated month name	
%B	Full month name	
%c	Date and time representation appropriate for locale	
%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 = 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	
%z, %Z	Either the time-zone name or time zone abbreviation	
%%	Percent sign	

TimeZone	
Type	Syntax
String	`TimeZone = tzn[+ | -]hh[:mm[:ss] ][dzn]`

Description

This property is used to get/set the proper time zone for date display and to adjust the current system time. The `TimeZone` property can be specified as follows:

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

EnableLineWeights	
Type	Default

Boolean	TRUE
Description	
This property is used to get/set whether vector lines are drawn with line weights. If set to FALSE, the control draws all vector lines with a width of 1 pixel.	

ShowLineWeights	
Type	Default
Boolean	TRUE
Description	
This property is used to get/set whether plot weights (if available in the file) are used in the viewer and for export. If set to TRUE and EnableLineWeights=TRUE they will display in the viewer and will also apply to export formats. This option can also be set in the viewer GUI with the option "Set line weights to plot weights (if available)" accessed from the *Options-->CAD Settings* menu.	

LockToZoomExtents	
Type	Default
Boolean	TRUE
Description	
This property is used to get/set if the current view state is at extents. If set to TRUE, the control maintains the current view at extents during window resizing.	

AllowPageChangeOnDocumentScroll	
Type	Default
Boolean	TRUE
Description	
This property is used to get/set the vertical scroll action for a document. 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. Note that this feature only applies to documents and not drawings.	

PagesRotatedByUser	
Type	Default
Boolean	<Read-Only>
Description	
This property is used to determine if one or more pages in the current document or drawing have been rotated from their original orientation.	

DisplayMrkExistIcon	
Type	Default
Boolean	`TRUE`
Description	
This property is used to get/set if the markup notification icon should be displayed in the top left corner of the control.	

DisplaySearchAndReviewPanel	
Type	Default
Boolean	`FALSE`
Description	
This property is used to get/set if the Verify panel is displayed.	

TabbedPanelWidth	
Type	Default
Integer	`212`
Description	
This property is used to get/set if the width of the right side panel.	

View Control: Methods

ShowLayerInfo
Syntax
`ShowLayerInfo()`
Description
This method causes the control to display a modal dialog allowing the user to view and manipulate a drawing's layers.

Rotate90
Syntax
`Rotate90()`
Description
This method causes the control to rotate the current view 90 degrees clockwise.

Rotate90Counterclockwise
Syntax
`Rotate90Counterclockwise()`
Description
This method causes the control to rotate the current view 90 degrees counter-clockwise.

FitAll
Syntax
`FitAll()`
Description
This method causes the control to adjust the view so that all drawing elements fit into the visible area.

FitWidth
Syntax
`FitWidth()`
Description
This method causes the control to adjust the view so that the entire width of the document/drawing fits into the visible area.

MirrorRaster			
Syntax			
`MirrorRaster( [in] BOOL Horizontal, [in] int Page)`			
Description			
This method causes the control to mirror a raster document. Note that this method has no effect on non-raster drawings or documents.			
Parameters	I/O	Type	Description

Horizontal	In	Boolean	If TRUE, the mirroring will be done about the Y (up and down) axis. If FALSE, the mirroring will be done about the X (left and right) axis.
Page	In	Integer	Indicates which page of a multi-page document or drawing to mirror.

SetCurrentViewportState

Syntax

```
SetCurrentViewportState( [in] double EyeX, [in] double EyeY, [in] double Width, [in] double Height, [in] double Rotation, [in] int CoordSys)
```

Description

This method changes the current viewport state relative to the extents of the document/drawing.

Parameters	I/O	Type	Description
EyeX	In	Double	Specifies the X value of the center of the view.
EyeY	In	Double	Specifies the Y value of the center of the view.
Width, Height	In	Double	Width and Height indicate scale or zoom level of the view.
Rotation	In	Double	Rotation of the view, measured in degrees.
CoordSys	In	Integer	CoordSys governs the coordinate system and is used as follows:

Value	Viewport	Description
0	World	The real values used by SetCurrentViewportState() will be in the drawing/document's native units.
1	Absolute Scale	The EyeX and EyeY set 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.
2	Normalized device coordinates	The real values used by SetCurrentViewportState() will be in normalized device coordinates.

GetCurrentViewportState

Syntax

```
GetCurrentViewportState( [out] double *EyeX, [out] double * EyeY, [out]
```

`double * Width, [out] double * Height, [out] double * Rotation, [in] int CoordSys)`			
Description			
This method retrieves the current viewport values relative to the extents of the document/drawing.			
Parameters	I/O	Type	Description
`EyeX`	Out	Double	Specifies the X value of the center of the view.
`EyeY`	Out	Double	Specifies the Y value of the center of the view.
`Width, Height`	Out	Double	`Width` and `Height` indicate scale or zoom level of the view.
`Rotation`	Out	Double	Rotation of the view, measured in degrees.
`CoordSys`	In	Integer	`CoordSys` governs the coordinate system as described in `SetCurrentViewportState()`.

GetCurrentViewportStateEx			
Syntax			
`GetCurrentViewportStateEx( [in] IBxRealAnswer* EyeX, [in] IBxRealAnswer* EyeY, [in] IBxRealAnswer* Width, [in] IBxRealAnswer* Height, [in] IBxRealAnswer* Rotation, [in] int CoordSys)`			
Description			
This method retrieves the current viewport values relative to the extents of the document/drawing.			
Parameters	I/O	Type	Description
`EyeX`	Out	Real Answer	`EyeX.Answer` Specifies the X value of the center of the view.
`EyeY`	Out	Real Answer	`EyeY.Answer` Specifies the Y value of the center of the view.
`Width, Height`	Out	Real Answer	`Width.Answer` and `Height.Answer` indicate scale or zoom level of the view.
`Rotation`	Out	Real Answer	`Rotation.Answer` specifies the rotation of the view, measured in degrees.
`CoordSys`	In	Integer	`CoordSys` governs the coordinate system as described in `SetCurrentViewportState()`.

CopyPageTextToClipboard
Syntax
`CopyPageTextToClipboard()`

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

GetBXPageInfo			
Syntax			
`GetBXPageInfo ( [in] int PageNumber, [out] double * ToInchesScale, [out] double * Width, [out] double * Height, [out] BOOL * IsSingleRaster, [out] int * SingleRasterDPI, [out] BOOL * IsMonochromeRaster, [out] BSTR *PageName, [out] BSTR *PageMetaData)`			
Description			
This method retrieves the current viewport values relative to the extents of the document/drawing.			
Parameters	I/O	Type	Description
`PageNumber`	In	Integer	Page to get information about
`ToInchesScale`	Out	Double	Scale value used to convert native width and height unit to inches
`Width, Height`	Out	Double	Width and Height in native units
`IsSingleRaster`	Out	Boolean	If returned TRUE, the page is composed entirely of one raster entity
`SingleRasterDPI`	Out	Integer	If `IsSingleRaster` is TRUE, this value is the dots per inch of the raster entity
`IsMonochromeRaster`	Out	Boolean	If `IsSingleRaster` is TRUE, this value indicates if raster is monochrome or color
`PageName`	Out	String	Page name
`PageMetaData`	Out	String	Not currently used

GetBXPageInfoEx			
Syntax			
`GetBXPageInfoEx ([in] int PageNumber, [out] IBxRealAnswer * ToInchesScale, [out] IBxRealAnswer * Width, [out] IBxRealAnswer * Height, [out] IBxBoolAnswer * IsSingleRaster, [out] IBxRealAnswer * SingleRasterDPI, [out] IBxBoolAnswer * IsMonochromeRaster, [out] IBxStringAnswer *PageName, [out] IBxStringAnswer *PageMetaData)`			
Description			
This method retrieves the current viewport values relative to the extents of the document/drawing.			
Parameters	I/O	Type	Description

PageNumber	In	Integer	Page to get information about
ToInchesScale	Out	Real Answer	ToInchesScale.Answer will contain the scale value used to convert native width and height unit to inches.
Width, Height	Out	Real Answer	Width.Answer and Height.Answer contain the width and height of the page in native units.
IsSingleRaster	Out	Boolean Answer	If returned IsSingleRaster.Answer TRUE, the page is composed entirely of one raster entity.
SingleRasterDPI	Out	Real Answer	If IsSingleRaster.Answer is TRUE, SingleRasterDPI.Answer is the dots per inch value of the raster entity.
IsMonochromeRaster	Out	Boolean Answer	If IsSingleRaster.Answer is TRUE, IsMonochromeRaster.Answer will be set to TRUE if the raster is monochrome, and FALSE if the raster is color.
PageName	Out	String Answer	PageName.Answer will contain the page name.
PageMetaData	Out	String Answer	Not currently used

GetWorldSpacePoint

Syntax

```
GetWorldSpacePoint( [in] long DevX, [in] long DevY, [out] double * WorldX, [out] double * WorldY)
```

Description

This method is used to determine a document/drawing specific point that corresponds to a device/screen point.
Example:
Document X is a D size drawing with dimensions of 36in x 24in.
The control's document view area is 300 pixels x 200 pixels.
The current view of Document X is centered within the control's viewport.
Call GetWorldSpacePoint() with DevX=150 and DevY=100.
Values for WorldX and WorldY will be returned as 18 and 22.

Parameters	I/O	Type	Description
DevX, DevY	In	Long	In screen/pixel coordinates relative to the top-left corner of the control document view area
WorldX	Out	Double	X coordinate returned for worldspace
WorldY	Out	Double	Y coordinate returned for worldspace

GetWorldSpacePointEx

Syntax

`GetWorldSpacePointEx( [in] long DevX, [in] long DevY, [in] IBxRealAnswer * WorldX, [in] IBxRealAnswer * WorldY)`

Description

This method is used to determine a document/drawing specific point that corresponds to a device/screen point.
Example:
Document X is a D size drawing with dimensions of 36in x 24in.
The control's document view area is 300 pixels x 200 pixels.
The current view of Document X is centered within the control's viewport.
Call `GetWorldSpacePointEx()` with `DevX=150` and `DevY=100`.
Values for `WorldX.Answer` and `WorldY.Answer` will be returned as 18 and 22.

Parameters	I/O	Type	Description
`DevX, DevY`	In	Long	In screen/pixel coordinates relative to the top-left corner of the control document view area
`WorldX`	Out	Real Answer	X coordinate returned for worldspace in `WorldX.Answer`
`WorldY`	Out	Real Answer	Y coordinate returned for worldspace in `WorldY.Answer`

GetWorldSpaceRectCoordinate

Syntax

`GetWorldSpaceRectCoordinate( [in] int PageNumber, [in] int TargetPos, [out, retval] double * Coordinate)`

Description

This method is used to get the `coordinate` specified by `TargetPos` for the extents rectangle of a given page in world space.

Parameters	I/O	Type	Description
`PageNumber`	In	Integer	Indicates the page from which to get the coordinate.
`TargetPos`	In	Integer	`TargetPos` can have one of the following values: Value: 0 — Y coordinate of top edge Value: 1 — X coordinate of left edge Value: 2 — X coordinate of right edge Value: 3 — Y coordinate of bottom edge
`Coordinate`	Out	Double	Returns the coordinate point specified by `PageNumber` and `TargetPos`.

SetScaleToDevice			
Syntax			
`SetScaleToDevice( [in] double * ScaleValue)`			
Description			
This method allows the current view to be scaled to inches.			
Parameters	I/O	Type	Description
`ScaleValue`	In	Double	To scale the view to 100%, a value of 1.0 would be used. A value of 0.5 would cause the current view to be scaled to 50%.

GetScaleToDevice			
Syntax			
`GetScaleToDevice( [out] double * ScaleValue)`			
Description			
This method retrieves the current scale value.			
Parameters	I/O	Type	Description
`ScaleValue`	Out	Double	Returns scale value set by `SetScaleToDevice`.

GetScaleToDeviceEx			
Syntax			
`GetScaleToDeviceEx( [out] retval] double* ScaleValue)`			
Description			
This method retrieves the current scale value.			
Parameters	I/O	Type	Description
`ScaleValue`	Out	Double	Returns scale value set by `SetScaleToDevice`.

SetPersistedValue
Syntax
`SetPersistedValue( [in] BSTR section, [in] BSTR name, [in] BSTR value)`
Description
This method stores value for the given section and name. Values are persisted in the *ViewerConfig.xml* file located in the user profile directory. UI state for markup properties, publishing options, print and

publish banners, etc. are some of the values that are persisted in this file. The `SetPersistedValue()` method can be used to specify Integration specific UI settings before closing the application. Note that `SetPersistedValue()` has no effect on the behavior of the control. It is provided to allow integrators a means of persisting information or state particular to their integration.
Example:
`viewObject.SetPersistedValue("FileOptions", "AutoOpenMarkup", "1")`
Saves the auto load markup option. When the application is loaded next, if a document has associated markups, the Brava! control loads those markups.

`viewObject.SetPersistedValue("UICust", "Skin", "Black")`
Saves the theme (skin) as Black. When the application is loaded next, the Brava! control uses this theme.

Parameters	I/O	Type	Description
`section`	In	String	Indicates section of *ViewerConfig.xml*
`name`	In	String	Parameter name specified by `section` that you would like to persist
`value`	In	String	Indicates the value to persist that are possible for the parameter specified by `name`

GetPersistedValue

Syntax

```
GetPersistedValue( [in] BSTR section, [in] BSTR name, [out, retval] BSTR * pValue)
```

Description

This method retrieves the value stored by the viewer for the given section and name. If no value is found, an empty string is returned. See `SetPersistedValue()` for more information on persisted values.

Parameters	I/O	Type	Description
`section`	In	String	Indicates section of *ViewerConfig.xml*
`name`	In	String	Parameter name specified by `section` that you would like to persist
`pValue`	Out	String	The returned value for the given `section` and `name`.

ClearPersistedValues

Syntax

```
ClearPersistedValues()
```

Description

This method clears values from the Container section that have been previously read from the *ViewerConfig.xml* file or set by `SetPersistedValue()`.

WritePersistedSettings

Syntax

```
WritePersistedSettings([in] BSTR Options]
```

Description

This method forces the viewer to write the current state of all persisted settings to *ViewerConfig.xml*.

Parameters	I/O	Type	Description
`Options`	In	String	Not currently used

ReadPersistedSettings

Syntax

```
ReadPersistedSettings([in] BSTR Options]
```

Description

This method forces the viewer to read and update the current state of all persisted settings found in *ViewerConfig.xml*.

Parameters	I/O	Type	Description
`Options`	In	String	Not currently used

FindText

Syntax

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

Description

This method causes the control to search for and highlight text within the currently loaded document or drawing. Calling the function with an empty string for the `SearchString` parameter will clear the previously searched and highlighted text.

Parameters	I/O	Type	Description
`SearchString`	In	String	String to find
`SearchForward`	In	Boolean	If set to TRUE, the control will search forward. If set to FALSE, the control will search backward
`MatchCase`	In	Boolean	If set to TRUE, the control will do a case-sensitive search
`SearchPosition`	In	Short	Indicates where (within the document or drawing) to do the search: Value | Description

			1 — Search from beginning of document 2 — Search from current position in document 3 — Search from end of document
SearchParams	In	String	Not currently used
Result	Out	Short	Used to return the result of the search. Possible values: **Value** / **Result** / **Description** 1 / Full failure / SearchString was not found anywhere in the document or drawing 2 / Partial failure / SearchString was not found between the current search position and end of the document if SearchForward = TRUE, or not found between the current search position and the beginning of the document if SearchForward= FALSE 3 / Success / 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

SetDisplayPanel			
Syntax			
SetDisplayPanel([in] LONG PanelID, [in] BOOL Display)			
Description			
This method uses the value of Display to show or hide a right-side panel defined by PanelID. Note that this method only controls whether a panel is currently displayed, but does not disable the panel. To disable a panel, you can set one or more of the following values for the SetBravaXParam() method to FALSE: enablethumbnailpanel, enablechangemarkpanel, enableredactverifypanel, enablebookmarkpanel, and enabletakeoffpanel.			
Parameters	I/O	Type	Description
PanelID	In	Long	Specifies one of the available right side panels. Valid values for

			PanelID are: **Value** / **Description** 0 – Thumbnail 1 – Changemarks 2 – Verify Redactions 3 – Bookmarks 4 – Takeoff
Display	In	Boolean	If set to TRUE, the panel identified by PanelID will be displayed. If set to FALSE, and if the panel identified by PanelID is currently displayed, that panel will be hidden.

GetDisplayPanel

Syntax

```
GetDisplayPanel( [in] LONG PanelID, [out, retval] BOOL * IsDisplayed)
```

Description

This method is used to determine if a right-side panel identified by PanelID is currently displayed.

Parameters	I/O	Type	Description
PanelID	In	Long	See SetDisplayPanel() for valid values of PanelID.
IsDisplayed	Out	Boolean	Returns whether panel indicated by PanelID is displayed.

UIStateActive

Syntax

```
UIStateActive( [in] int StateID, [out, retval] BOOL * pVal)
```

Description

This method checks whether a certain portion of the user interface is currently active.

Parameters	I/O	Type	Description
StateID	In	Integer	See UIStateEnabled for possible values of StateID.
pVal	Out	Boolean	Returns whether state indicated by StateID is active.

UIStateEnabled

Syntax

`UIStateEnabled( [in] int StateID, [out, retval] BOOL * pVal)`

Description

This method checks whether a certain portion of the user interface is currently enabled.

Parameters	I/O	Type	Description
`StateID`	In	Integer	Possible values of StateID are:

Value	State	Value	State
1	Zoom extents tool	34	Calibrate tool
2	Pan tool	35	Measure line tool
3	Zoom tool	36	Measure polygon tool
4	Rotate tool	37	Measure rectangle tool
5	Magnifier tool	38	Delete
6	Allow quick draw	39	Markup image tool
7	Markup edit tool	40	Dynamic zoom tool
8	Undo	41	Markup squiggle rectangle tool
9	Redo	42	Allow animation
10	Markup sketch tool	43	Measure count tool
11	Markup polygon tool	44	Markup redact area tool
12	Markup ellipse tool	45	Make redaction
13	Markup arc tool	46	Align compare documents
14	Markup text tool	47	Measure polyline tool
15	Markup rectangle tool	48	Measure circle tool
16	Rotate 0 degrees	49	Markup stamp tool
17	Rotate 90 degrees	50	Markup rounded rectangle tool
18	Rotate 180 degrees	51	Markup allow area tool
19	Rotate 270 degrees	52	Markup byzantine arrow tool
20	Markup editing	53	Markup takeoff line tool

			<table><tr><td>21</td><td>Markup polyline tool</td><td>54</td><td>Markup takeoff polyline tool</td></tr><tr><td>22</td><td>Markup line tool</td><td>55</td><td>Markup takeoff polygon tool</td></tr><tr><td>23</td><td>Markup arrow tool</td><td>56</td><td>Markup takeoff rectangle tool</td></tr><tr><td>24</td><td>Markup sketch polygon tool</td><td>57</td><td>Markup takeoff circle tool</td></tr><tr><td>25</td><td>Markup crossout tool</td><td>58</td><td>Markup takeoff count tool</td></tr><tr><td>26</td><td>Select tool</td><td>59</td><td>Markup Changemarks cloud tool</td></tr><tr><td>27</td><td>Markup strikeout text tool</td><td>60</td><td>Markup Changemarks highlight text tool</td></tr><tr><td>28</td><td>Markup underline text tool</td><td>61</td><td>Markup highlight tool</td></tr><tr><td>29</td><td>Markup highlight text tool</td><td>62</td><td>Markup Changemark hightlight tool</td></tr><tr><td>30</td><td>Markup Changemarks tool</td><td>63</td><td>Markup Changemarks arrow tool</td></tr><tr><td>32</td><td>Markup polygonal cloud tool</td><td>64</td><td>Markup strikethrough text tool</td></tr><tr><td>33</td><td>Markup cloud tool</td><td>65</td><td>Markup block out tool</td></tr><tr><td>66</td><td>Markup signature tool</td><td>67</td><td>Markup Checkview tool</td></tr><tr><td>68</td><td>Markup Changemarks strikethrough tool</td><td></td><td></td></tr></table> See Functionality Identification Section for additional valid values of `StateID`.
`pVal`	Out	Boolean	Returns whether state indicated by `StateID` is enabled.

ApplyUIAdjustment
Syntax
`ApplyUIAdjustment( [in] BSTR AdjustData, [in] BOOL IsFile)`
Description

This method takes XML data that may contain any or all of four major, non-root action elements: "Disabled", "Enabled", "Hidden", and "Shown". Each contains a list of Control IDs which will be respectively disabled, enabled, hidden, and shown. See Control Identification Section for a list of valid Control IDs.
Example:

```
<?xml version="1.0" encoding="utf-8"?>
<ApplyUIAdjustment>
<Disabled>
<!--Disable the navigation toolbar color menu black option-->
<ControlID>101500</ControlID>
</Disabled>
<Hidden>
<!--// Hide the navigation toolbar color menu white option-->
<ControlID>101501</ControlID>
</Hidden>
<Shown>
<!--// Show the navigation toolbar color menu gray option-->
<ControlID>101502</ControlID>
</Shown>
<Enabled>
<!--// Enable the markup menu consolidate command-->
<ControlID>101005</ControlID>
</Enabled>
</ApplyUIAdjustment>
```

Parameters	I/O	Type	Description
`AdjustData`	In	String	XML data source
`IsFile`	In	Boolean	Specifies if `AdjustData` refers to a file that contains the XML data

ApplyCustomButtons

Syntax

```
ApplyCustomButtons( [in] BSTR ButtonData, [in] BOOL IsFile)
```

Description

This method provides description about a button that the control adds to the viewer's user interface. The XML-encoding is expected to match the Extension schema documented in the *BravaExtensibleInterface_UserGuide.pdf* document. See the *BravaExtensibleInterface_UserGuide.pdf* document for more details on specifying custom buttons.

Parameters	I/O	Type	Description
`ButtonData`	In	String	XML button data
`IsFile`	In	Boolean	If `IsFile` is TRUE, `ButtonData` specifies a fully-qualified UNC path of the file containing the XML data for the button. If `IsFile` is FALSE, `ButtonData` is the XML encoded string description for the button.

ApplyColorCustomization

Syntax

```
ApplyColorCustomization( [in] BSTR ColorData, [in] int DataSource)
```

Description

This method specifies which skin (theme) to use.

Parameters	I/O	Type	Description
ColorData	In	String	XML color data determined by DataSource
DataSource	In	Integer	Defines what type of skin to apply. It may be one of the following values: **Value** / **DataSource** / **Description** 0 / File / ColorData is the path to the file containing the skin 1 / String / ColorData contains the XML defining the skin 2 / Resource / ColorData is the name of one of the default skins defined in Brava!. The skin names can be obtained by calling EnumerateSkinNames().

GetCtrlScreenCoordinate

Syntax

```
GetCtrlScreenCoordinate( [in] int CtrlID, [in] int TargetPos, [out, retval] int * Coordinate)
```

Description

This method is used to get the coordinate specified by TargetPos for the component specified by CtrlID.

Parameters	I/O	Type	Description
CtrlID	In	Integer	See Control Identification Section for valid values of CtrlID.
TargetPos	In	Integer	TargetPos can have one of the following values: Value / Description 0 / Y coordinate of top edge 1 / X coordinate of left edge 2 / X coordinate of right edge 3 / Y coordinate of bottom edge

Coordinate	Out	Integer	Returns the coordinate point specified by CtrlID and TargetPos.

EnumerateSkinNames			
Syntax			
EnumerateSkinNames([out, retval] BSTR * pVal)			
Description			
This method returns the names of all the skin (theme) resources in the Brava! control. Names are separated by "<|>". For example, the returned strings may be "Black<|>Blue<|>Silver", where "Black", "Blue", and "Silver" are the names of the default skins. See the *BravaExtensibleInterface _UserGuide.pdf* document for more details on specifying themes (skins).			
Parameters	I/O	Type	Description
pVal	Out	String	Returned theme resources.

SetMarkupEntity			
Syntax			
SetMarkupEntity([in] BSTR EntityID, [in] BSTR EntityData)			
Description			
This method modifies a markup entity uniquely identified by EntityID with the data in EntityData string. Note that calling this method will result in clearing the undo history (used for the redo/undo function).			
	I/O	Type	Description
EntityID	In	String	Specifies the markup entity to set. See the MarkupEntityModified event for more information.
EntityData	In	String	Markup data to modify.

GetMarkupEntity			
Syntax			
GetMarkupEntity([in] BSTR EntityID, [out, retval] BSTR * EntityData)			
Description			
This method returns the data for the markup entity uniquely identified by EntityID in the EntityData string.			
	I/O	Type	Description
EntityID	In	String	Specifies the markup entity to set. See the MarkupEntityModified event for more information.

`EntityData`	Out	String	Returned markup data.

GetMarkupEntityCount

Syntax

```
GetMarkupEntityCount( [in] int markupEntityType, [in] BOOL bCountEditableOnly, [in] BOOL bCountBurnedIntoCSFOnly, [out, retval] int * pVal)
```

Description

This method returns the number of markup entities.

Parameters	I/O	Type	Description
`MarkupEntityType`	In	Integer	If set to 0, the count for all entity types are returned, else the count of entities of the given type is returned. Valid values of `markupEntityType` are:

Value	Description	Value	Description
1	Line	17	Stamp
2	Polyline	18	Arrow
3	Sketch	19	Advanced arrow
4	Arc	20	Takeoff line
5	Ellipse	21	Takeoff polyline
6	Polygon	22	Takeoff polygon
7	Sketch polygon	23	Takeoff rectangle
8	Crossout, Strikethrough, Strikeout, Underline	24	Takeoff circle
10	Text	25	Takeoff count
11	Changemark entity	26	Changemark reply
12	Cloud	27	Highlight
13	Image	28	Signature
15	Scratchout	29	Checkview

			<table><tr><td>16</td><td>Redact Area</td><td>30</td><td>Checkview reply</td></tr><tr><td></td><td></td><td>31</td><td>Bookmark</td></tr></table>
`bCountEditableOnly`	In	Boolean	If TRUE, only entities in a markup open for edit, if any, are counted.
`bCountBurnedIntoCSFOnly`	In	Boolean	If TRUE, only entities in a markup burned into the currently open CSF, if any, are counted.
`pVal`	Out	Integer	Returned number of markup entities.

SetLayerData

Syntax

```
SetLayerData( [in] BSTR * LayerData)
```

Description

This method modifies the current drawing layer visibility state map.
Example:
`viewObject.SetLayerData("0110");`
Sets the first layer to be off, the second and third layers to be on, fourth layer off.

Parameters	I/O	Type	Description
`LayerData`	In	String	Specifies the state of each layer as 0 (off) or 1 (on). The length of the string must match the number of layers in the drawing.

GetLayerData

Syntax

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

Description

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

Parameters	I/O	Type	Description
`LayerData`	Out	String	See `SetLayerData()` for the `LayerData` format example.

GetNumberOfLayers

Syntax

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

Description

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

Parameters	I/O	Type	Description
NumLayers	Out	String	Number of layers returned.

SetLayerState			
Syntax			
SetLayerState([in] int Index, [in] BOOL CurrentVisibility)			
Description			
This method modifies the visibility for the layer identified by Index.			
Parameters	I/O	Type	Description
Index	In	Integer	Specifies the layer number to set visibility state for.
CurrentVisibility	In	Boolean	If TRUE, the layer will be visible, if FALSE, the layer will not be visible.

GetLayerState			
Syntax			
GetLayerState([in] int Index, [out] BSTR * LayerName, [out] BOOL * DefaultVisibility, [out, retval] BOOL * CurrentVisibility)			
Description			
This method retrieves information for the layer identified by Index.			
Parameters	I/O	Type	Description
Index	In	Integer	Specifies the layer number to get visibility state for.
LayerName	Out	String	Returns name of layer specified by Index.
DefaultVisibility	Out	Boolean	Returns default visibility state of Index layer.
CurrentVisibility	Out	Boolean	Returns current visibility state of Index layer.

GetWorldSpaceDimensions			
Syntax			
GetWorldSpaceDimensions([out] double * Width, [out] double * Height)			
Description			
This method gets the Width and Height of the currently viewed document or drawing in the file's native units.			
Parameters	I/O	Type	Description

`Width`	Out	Double	Document `Width` returned in native units.
`Height`	Out	Double	Document `Height` returned in native units.

View State Section

View State: Properties and Methods

View Pins (called Placekeepers in the viewer user interface) are a means of setting a marker at particular position/viewstate within a document/drawing. The control can store a list of View Pins that exist for the duration of a single file load.

Properties

NumberOfViewPins	
Type	Default
Integer	<Read-Only>
Description	
This property indicates the number of View Pins currently stored.	

Methods

GetViewPinsInfo			
Syntax			
`GetViewPinsInfo( [out] int * NumPins, [out] int * CurrentPinIndex)`			
Description			
Retrieves the total number of view pins in the current list (`NumPins`) and the index of the last viewed view pin (`CurrentPinIndex`).			
Parameters	I/O	Type	Description
`NumPins`	Out	Integer	Returns total number of view pins in the current list.
`CurrentPinIndex`	Out	Integer	Returns index number of the last viewed view pin.

AddViewPin
Syntax
`AddViewPin()`

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

NextViewPin
Syntax
`NextViewPin()`
Description
This method changes the view to the next View Pin in the View Pin list.

PreviousViewPin
Syntax
`PreviousViewPin()`
Description
This method changes the view to the previous View Pin in the View Pin list.

GoToViewPin			
Syntax			
`GoToViewPin( [in] int PinIndex)`			
Description			
This method sets the current View Pin index to `PinIndex` and changes the view to that pin.			
Parameters	I/O	Type	Description
`PinIndex`	In	Integer	Specifies the current View Pin.

ClearViewPins
Syntax
`ClearViewPins()`
Description
This method deletes the entire View Pin list.

ApplyExternalHotSpots
Syntax

`ApplyExternalHotSpots (BSTR HotSpotDef, BOOL ReadFromFile, int *NumSpotsAdded)`
Description
Used to insert a set of hot spots areas into the current Brava! view. Spots placement can be defined in one of two ways: 1. A list of 2D world-coordinate points. 2. The system can be asked to search the text within the current document. A hot spot will be created as a rectangle around each piece of document text that satisfies the search. Regex searching is supported.

Parameters	Type	Description
`HotSpotDef`	String	XML representation of one or more hotspots. SPECIAL STRING "clearall" causes the deletion of all hot spots currently in the system.
`ReadFromFile`	Boolean	If set TRUE (1), the `HotSpotDef` parameter is interpreted as an XML file name. If set FALSE, the `HotSpotDef` parameter is interpreted as a string of XML data.
`NumSpotsAdded`	Integer	Return value when called from JavaScript. Indicates the number of hot spots created.

GetExternalHotSpots
Syntax
`GetExternalHotSpots (BSTR *XML)`
Description
This method will return the XML for all hot spots including spots created by search or created by point list and applied to all pages.

Parameters	Type	Description
`XML`	String	XML representation of the set of currently active hot spots. S

VisitExternalHotSpots
Syntax
`VisitExternalHotSpots (BSTR HotSpotID, BOOL Zoom, double ZoomScale, int HotState)`
Description
Used to insert a set of hot spots areas into the current Brava! view. Spots placement can be defined in one of two ways:

1. A list of 2D world-coordinate points.
2. The system can be asked to search the text within the current document. A hot spot will be created as a rectangle around each piece of document text that satisfies the search. Regex searching is supported.

Parameters	Type	Description
HotSpotID	String	Indicates the spot to be visited.
Zoom	Boolean	If set TRUE (1), the system will adjust the view state to focus on the spot identified by HotSpotID.
ZoomScale	Double	Real value used to adjust the zoom level. a value of 1.0 causes the view state to zoom to the extents of the spot. A value greater than 1.0 scales out.
HotState	Integer	Integer value used to adjust the display of the spot: 0 - Display the spot using the normal "non-hot" colors. 1 - Display the spot using the "hot/hovered" colors. 2 - Do not affect the color state of the spot.

GetExternalHotSpot

Syntax

```
GetExternalHotSpot (BSTR HotSpotID, BSTR *XML)
```

Description

This method will return the current state of a given hot spot.

Parameters	Type	Description
HotSpotID	String	Indicates the ID of the spot.
XML	String	XML representation of the hot spot indicated by HotSpotID.

SetExternalHotSpot

Syntax

```
SetExternalHotSpot (BSTR HotSpotID, BSTR *XML)
```

Description

This method will modify the state of a given hot spot. If XML is set to "clear", the spot is deleted from the system. If the spot indicated by HotSpotID does not already exist in the current set of spots, SetExternalHotSpot creates a new spot. Additionally:

- If the XML specifies a <Search> element, and the spot identified by HotSpotID already exists, no new search is performed.
- If the XML specifies both a <Search> and a <Points> element, the search element is ignored and the spot appears at the location specified.

- The `id` attribute in the XML must match the `HotSpotID` parameter.

Parameters	Type	Description
`HotSpotID`	String	Indicates the ID of the spot.
`XML`	String	XML representation used to modify the state of the spot indicated by `HotSpotID`.

XML Representation

SpotList - Outer element. Contains one or more ExternalHotSpot elements.

Attributes:

`version` - Indicates the hot spot definition version. Currently this is "1".

`numspots` - Indicates the number of ExternalHotSpot elements contained within.

ExternalHotSpot - Defines a hot spot.

Attributes:

`id` - String that will be used to identify a particular hot spot.

`page` - Integer to indicate the document page on which the spot will appear. A value of "-1" indicates that all pages will be considered. In the case of spots created by text search, every page will be searched. In the case of spots defined by a list of points, each document page will display an identical spot and each spot's ID will be appended with the page number on which it appears.

`tip` - String that will appear inside the hover-over pop-up when the mouse pointer is positioned over the spot and the current mouse tool is the pan/zoom or document selection tool.

Child Elements:

`Display` - Defines the appearance of the hot spot.

Child Elements:

`SpotColor` - Interior color of the spot. Colors are a hexadecimal ARGB quad in the form "#AARRGGBB".

`HotColor` - Interior color of the spot while hovered\hot. Colors are hexadecimal ARGB quad in the form "#AARRGGBB".

`BorderColor` - Border color of the spot. Colors are a hexadecimal ARGB quad in the form "#AARRGGBB".

`HotBorderColor` - Border color of the spot while hovered\hot. Colors are a hexadecimal ARGB quad in the form "#AARRGGBB".

Points - Defines the shape and position of the spot using a list of 2D world-space points.

Attributes:

`PointCount` - Integer indicating the number of points in the list. Note that the parent element Points must contain 3-N (non-linear) Point elements.

Child Elements:

`Point` - A single 2D world-space point.

Child Elements:

`X` - Real number x coordinate of the point.

`Y` - Real number y coordinate of the point.

Search - Defines search criteria for creating hot spots via text pattern matching.

Child Elements:

`Term` - String value used to search the document text. Regex is supported.

`Match_Case` - If this element is found, the search will be done in a case-sensitive manner.

`Append_Index_To_Id` - If this element is found, each spot that satisfies the pattern match will have the document position index appended to its ID string. This is to allow unique IDs when multiple spots are created via a single search.

ContextMenu - Defines a menu that will appear when the right mouse button is clicked while the mouse cursor is hovered above the spot. `ExternalHotSpotMouseEvent` will be fired with a value of MOUSEEVENT_CONTEXT_MENU when menu items are selected.

Attributes:

`id` - String that will be used to identify the top level menu.

Child Elements:

`MenuCommand` – An item in the menu.

Attributes:

`id` - String that will be used to identify this menu item.

`text` - String to be displayed on the menu.

`ContextMenuSub` – A sub pop-out menu item

Attributes:

`id` - String that will be used to identify this pop-out menu item.

`text` - String to be displayed on the pop-out menu.

`MenuSeparator` – A menu separator item.

Example XML

The following XML will search and hot spot all document text strings using a regex search that attempts to find email addresses. This XML will also place a hot spot on each document page as defined by a world-space point list.

```
<?xml version="1.0" encoding="UTF-8"?>
<SpotList version = "1" numspots="2">
 <ExternalHotSpot id="Email" page="-1" tip="spot is Email&#10;Email
is
hovered">
 <Search>
 <term>\b[\w\.-]+@[\w\.-]+\.\w{2,4}\b</term>
 <append_index_to_id/>
 </Search>
 <Display>
 <spotcolor>#8000dede</spotcolor>
 <hotcolor>#8005acdc</hotcolor>
 <bordercolor>#b4ff0000</bordercolor>
 <hotbordercolor>#b4ea647b</hotbordercolor>
 </Display>
 <ContextMenu id="foo">
 <MenuCommand id="foo1" text="this is foo 1"/>
 <MenuCommand id="foo2" text="this is foo 2"/>
 <MenuSeparator/>
 <MenuCommand id="foo3" text="this is foo 3"/>
 <ContextMenuSub id="bar" text="bar is sub">
 <MenuCommand id="bar1" text="this is bar 1"/>
 <MenuCommand id="bar2" text="this is bar 2"/>
 <MenuSeparator/>
 <ContextMenuSub id="zam" text="zam is sub">
 <MenuCommand id="zam1" text="this is zam 1"/>
 <MenuCommand id="zam2" text="this is zam 2"/>
 </ContextMenuSub>
 <MenuCommand id="bar3" text="this is bar 3"/>
```

```
 </ContextMenuSub>
 <MenuCommand id="foo4" text="this is foo 4"/>
 </ContextMenu>
 </ExternalHotSpot>
 <ExternalHotSpot id="3 points" page="-1" tip="spot is 3
points&#10;3 points
is hovered">
 <Display>
 <spotcolor>#8000dede</spotcolor>
 <hotcolor>#8005acdc</hotcolor>
 <bordercolor>#b4ff0000</bordercolor>
 <hotbordercolor>#b4ea647b</hotbordercolor>
 </Display>
 <points pointcount="3">
 <Point>
 <x>2728.56</x>
 <y>8009.65</y>
 </Point>
 <Point>
 <x>2728.56</x>
 <y>8172.36</y>
 </Point>
 <Point>
 <x>3200.17</x>
 <y>8172.36</y>
 </Point>
 </points>
 </ExternalHotSpot>
</SpotList>
```

Auto View State Section

Auto View State: Properties and Methods

The control has the ability to navigate "Back" and "Forward" during a 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 a 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 a 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	
Type	Default
Boolean	TRUE
Description	
This property is used to get/set saving of view states. If set to FALSE, the control will not save viewstates.	

Methods

BackView
Syntax
BackView()
Description
This method changes the viewstate to the previous state in the saved view states.

ForwardView
Syntax
ForwardView()
Description
This method changes the viewstate to the next state in the saved view states.

GetAutoSavedViewStateInfo			
Syntax			
GetAutoSavedViewStateInfo([out] int * NumViewStates, [out] int * CurrentViewStateIndex)			
Description			
This method retrieves the number of viewstates that have been auto-saved (NumViewStates) and the current index in the saved viewstate stack (CurrentViewStateIndex).			
Parameters	I/O	Type	Description
NumViewStates	Out	Integer	Returns total number of saved view states.
CurrentViewStateIndex	Out	Integer	Returns current index in the saved viewstate stack.

Page Control Section

Page Control: Properties and Methods

Properties

CurrentPageNumber	
Type	Example
Integer	`viewObject.CurrentPageNumber=106`
Description	
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.	

TotalPages	
Type	Default
Integer	<Read-Only>
Description	
This property is used to get the total number of pages in the currently viewed multi-page document. *Example*: `viewObject.CurrentPageNumber = viewObject.TotalPages - 1` This example causes the control to display the second to the last page.	

Methods

NextPage
Syntax
`NextPage()`
Description
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
Syntax
`PreviousPage()`
Description
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.

Watermark/Banners Section

Watermark/Banners: Properties

BannerStrings	
Type	Example
String	`viewObject.TopCenter = "Page Number %Page User Name %User"` Puts the current page number and current user name on the top center of each page printed through Brava!
Description	

The following properties are String values used to get/set individual banners and watermarks for print and export. Positioned banners (Example: `TopLeft`, `RightCenter`), can display up to ten separate lines. To indicate a line break use "<\n>". These properties are deprecated and are superceded by methods `SetBanner(), GetBanner(), SetWatermark(), GetWatermark()`.

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

Value (tag)	Description
%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 is displayed
%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

%DBUpdateString(field label)	Used when creating Stamp Templates to resolve custom integration values during the review process.
%prompt(“Label_String”)	Used in a token expression when creating a stamp template. When set, a prompt dialog displays to the end-user prompting them to manually edit a markup stamp's text string during review process.
%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 is displayed
%MilTime	Inserts the time the print was spooled based on a 24 hour clock
%Title	Inserts the name of the document
%Page	Inserts the page number
%TotalPages	Inserts the total number of pages
%Login or %User	Inserts the user name of the person who issued the print
%%	Inserts a single % character
%Hostname	Inserts the hostname of the machine that issued the print
%IPAddress	Inserts the IP Address of the machine that issued the print
©	Inserts the copyright symbol
®	Inserts the registered tratemark symbol
%BatesPgNo(x)	Bates Number. This tag is used to indicate the starting page number and the number of digits to use. For example, %BatesPgNo(0002) would place 0002 on the first page, 0003 on the next page, etc.

WatermarkBannerFontName	
Type	Example
String	`viewObject.WatermarkBannerFontName = "Arial"` This example sets all watermark and banner strings to use Arial.
Description	
This property is used to get/set the font name used to render all watermark and banner strings.	

WatermarkBannerFontStyle	
Type	Example
Short (Integer)	`viewObject.WatermarkBannerFontStyle = 3` This example sets all banner strings to use bold and italic.
Description	

This property is used to get/set the font style used to render all banner strings. This property does not apply to the watermark string. The following are acceptable values:

Value	Description
0	(Default) Normal
1	Bold
2	Italic
3	Bold Italic

BannerFontSize	
Type	Example
Short	`BannerFontSize=10`
Description	
This property is used to get/set the font size to use for banner strings. Valid range is 4-72	

PersistBanners	
Type	Default
Boolean	`FALSE`
Description	
This property is used to get/set whether watermarks and banners (print, publish, and screen) are persisted from session to session.	

WatermarkTileSize	
Type	Example
String	`viewObject.WatermarkTileSize = "2.5,3,m"` This example sets the tile size for the watermark to be 2.5 mm wide and 3 mm tall.
Description	
This property is used to get/set the size of a watermark tile that will be used for tiling across a printed page. The string contains three values separated by commas: width,height,units. Units can be `m` (millimeters) or `i` (inches).	

Watermark/Banners: Methods

SetBanner

Syntax

```
SetBanner( [in] BSTR BannerText, [in] int Pos, [in] BOOL ApplyToPrint, [in] BOOL ApplyToPublish, [in] BOOL LockForPrint, [in] BOOL LockForPublish)
```

Description

This method sets the banner string `BannerText` at the position specified by `Pos`.

Parameters	I/O	Type	Description
`BannerText`	In	String	Specifies the banner string text.
`ApplyToPrint`	In	Boolean	Specifies if the banner is to be applied for printing.
`ApplyToPublish`	In	Boolean	Specifies if the banner is to be applied for publishing (export).
`LockForPrint`, `LockForPublish`	In	Boolean	These can be set to TRUE to prevent the set value from being edited by the user. If setting the screen banner (Pos = 12), if `LockForPrint` and `LockForPublish` are both set to TRUE, the screen banner is not editable.
`Pos`	In	Integer	Specifies the Banner position. The possible values for `Pos` are:

Value	Location
0	Top left
1	Top center
2	Top right
3	Bottom left
4	Bottom center
5	Bottom right
6	Left top
7	Left center
8	Left bottom
9	Right top
10	Right center
11	Right bottom
12	Screen

GetBanner

Syntax

```
GetBanner( [in] int Pos, [in] BOOL GetPrint, [in] BOOL GetResolved, [out, retval] BSTR * BannerText)
```

Description

This method returns the banner string for the position specified by `Pos`.

Parameters	I/O	Type	Description
`Pos`	In	Integer	Specifies the Banner position. See `SetBanner` for possible values.
`GetPrint`	In	Boolean	Specifies whether to return the print or publish (export) banner.
`GetResolved`	In	Boolean	If FALSE, token values are returned without being resolved.
`BannerText`	Out	String	The returned banner string text. If `GetResolved` is TRUE, the returned `BannerText` has token values resolved.

GetBannerEditable

Syntax

```
GetBannerEditable( [in] int Pos, [in] BOOL GetPrint, [out, retval] BOOL * IsEditable)
```

Description

This method returns if the banner string for the position specified by `Pos` is editable.

Parameters	I/O	Type	Description
`Pos`	In	Integer	Specifies the Banner position. See `SetBanner` for possible values.
`GetPrint`	In	Boolean	Specifies whether to return the print or publish (export) banner.
`IsEditable`	In	Boolean	Returns whether banner string is editable. If FALSE, banner is not editable.

SetWatermark

Syntax

```
SetWatermark( [in] BSTR MarkText, [in] BOOL ApplyToPrint, [in] BOOL ApplyToPublish, [in] BOOL ApplyToView, [in] BOOL LockForPrint, [in] BOOL LockForPublish, [in] BOOL LockForView)
```

Description

This method is used to apply a watermark string specified by `MarkText`.

Parameters	I/O	Type	Description
`MarkText`	In	String	Specifies the watermark text.

ApplyToPrint	In	Boolean	Specifies if the watermark is a Print watermark.
ApplyToPublish	In	Boolean	Specifies if the watermark is a Publish (export) watermark.
ApplyToView	In	Boolean	Specifies if the watermark is a Screen watermark.
LockForPrint, LockForPublish, LockForView	In	Boolean	These can be set to TRUE to prevent the set value from being edited by the user.

GetWatermark

Syntax

```
GetWatermark( [in] int Target, [in] BOOL GetResolved, [out, retval] BSTR * MarkText)
```

Description

This method returns the watermark string for the type specified by Target.

Parameters	I/O	Type	Description
Target	In	Integer	Type of watermark. The possible values are: **Value** / **Description** 0 – Print 1 – Publish 2 – Screen
GetResolved	In	Boolean	If TRUE, the returned MarkText has token values resolved. If FALSE, token values are returned without being resolved.
MarkText	Out	String	Returned watermark text.

GetWatermarkOpacity

Syntax

```
GetWatermarkOpacity( [in] int Target, [out, retval] int *Opacity)
```

Description

This method returns the opacity value of the print of publish watermark from 20-100 with 20 being transparent and 100 being opaque.

Parameters	I/O	Type	Description
Target	In	Integer	Specifies if the opacity value for print or publish watermark should be

			returned. The possible values are: **Value** / **Description** 0 – Print 1 – Publish
`Opacity`	Out	Integer	The returned value for print or publish watermark opacity.

SetWatermarkOpacity			
Syntax			
`SetWatermarkOpacity( [in] int Target, [in] int Opacity)`			
Description			
This method sets the opacity value of the print or publish watermark.			
Parameters	I/O	Type	Description
`Target`	In	Integer	Specifies if the opacity value for print or publish watermark should be returned. The possible values are: **Value** / **Description** 0 – Print 1 – Publish
`Opacity`	In	Integer	The value for print or publish watermark opacity. The value should be an integer from 20-100 with 20 being transparent and 100 being opaque.

GetWatermarkEditable			
Syntax			
`GetWatermarkEditable( [in] int Target, [out, retval] BOOL * IsEditable)`			
Description			
This method returns if the watermark string for the type specified by Target is editable.			
Parameters	I/O	Type	Description
`Target`	In	Integer	See `GetWatermark()` for possible values of Target.
`IsEditable`	Out	Boolean	Returns whether watermark string is editable. If FALSE, watermark is not editable.

SetCustomBannerValue

Syntax

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

Description

This method can be used to allow the control container to provide a string value for any unresolved token within a banner or watermark.

Parameters	I/O	Type	Description
`Key`	In	String	Indicates the token value.
`Value`	In	String	Indicates token replacement value.

GetCustomBannerValue

Syntax

```
GetCustomBannerValue( [in] BSTR * Key, [out, retval] BSTR Value)
```

Description

This method is used to determine the replacement value, `Value`, for any custom token, `Key`, set by `SetCustomBannerValue()`.

Parameters	I/O	Type	Description
`Key`	In	String	Indicates the token value.
`Value`	Out	String	Returned token replacement value.

ClearCustomBannerValues

Syntax

```
ClearCustomBannerValues()
```

Description

This method clears all custom banner values set by `SetCustomBannerValue()`.

EditIsoBanners

Syntax

```
EditIsoBanners( [in] BOOL EditPrintBanners)
```

Description

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

Parameters	I/O	Type	Description
EditPrintBanners	In	Boolean	If TRUE, the Print Watermark/Banners dialog is displayed. If FALSE the Publish Watermark/Banners dialog is displayed.

Markup Section

Markup: Properties

NumberReviewMarkups	
Type	Default
Boolean	<Read-Only>
Description	
This property is used to get the number of currently open markup review files.	

MarkupEditLoaded	
Type	Default
Boolean	<Read-Only>
Description	
The property returns TRUE if the control currently has an editable markup file open, FALSE otherwise.	

MarkupEditDirty	
Type	Default
Boolean	<Read-Only>
Description	
This property returns TRUE if the control has made changes to the currently open editable markup file that have not been saved.	

AllowMarkupChangeOwner	
Type	Default
Boolean	FALSE
Description	
This property is used to get/set if 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.	

TakeOwnershipOnMarkupConsolidation	
Type	Default
Boolean	`FALSE`
Description	
This property is used to get/set if the control transfers ownership of all markup entities during a consolidation of open markups. See method `ConsolidateMarkups()` for details.	

EnableMarkupColorPalette	
Type	Default
Boolean	`TRUE`
Description	
This property is used to get/set if the control should enable/disable the user interface for changing the color of markup entities.	

MarkupColor	
Type	Example

String	`MarkupColor=red`
	`MarkupColor=#FF0000`
	Either of these examples sets the markup entity color to red.

Description	

Used to get/set the current color of markup entities. The string can either be a string or hexadecimal value shown below. The following are valid strings and their corresponding RGB values. Note that you can use any hex value of the form #RRGGBB and are not limited to only the ones shown in this table:

String	**Hex**	**RGB (for information only)**
white	#FFFFFF	RGB (255, 255, 255)
yellow	#FFFF00	RGB (255, 255, 0)
magenta	#FF00FF	RGB (255, 0, 255)
red	#FF0000	RGB (255, 0, 0)
cyan	#00FFFF	RGB (0, 255, 255)
green	#00FF00	RGB (0, 255, 0)
blue	#0000FF	RGB (0, 0, 255)
darkgray	#666666	RGB (128, 128, 128)
gray	#999999	RGB (192, 192, 192)

olive	#666600	RGB (128, 128, 0)
purple	#660066	RGB (128, 0, 128)
maroon	#660000	RGB (128, 0, 0)
teal	#006666	RGB (0, 128, 128)
pine	#006600	RGB (0, 128, 0)
darkblue	#000066	RGB (0, 0, 128)
black	#000000	RGB (0, 0, 0)

WarnOnUnsavedMrkClose	
Type	Default
Boolean	`TRUE`
Description	
The property is used to get/set if the control displays a warning dialog when closing a markup that has not been saved.	

RasterMrkFilename	
Type	Example
String	`RasterMrkFilename=someraster.png`
Description	
This property is used to get/set the raster file used with the raster markup tool. The raster markup tool can use JPEG, PNG or BMP raster files.	

RedactionsExist	
Type	Default
Boolean	<Read-Only>
Description	
This property returns TRUE if redactions exist. Redactions can be burned into a CSF file or exist inside a markup that is open for edit or review.	

NumberChangemarks	
Type	Default
Short (integer)	<Read-Only>

Description
This property returns the number of Changemarks that exist.

MarkupStampTemplateLoaded	
Type	Default
Boolean	<Read-Only>
Description	
This property returns TRUE if the control currently has an editable markup stamp template file open, FALSE otherwise	

MarkupStampTemplateDirty	
Type	Default
Boolean	<Read-Only>
Description	
This property returns TRUE if the control has made changes to the currently open editable markup stamp template file that have not been saved.	

MrkStampEntityFilename	
Type	Example
String	`MrkStampEntityFilename=somestamptemplate.xsp`
Description	
This property is used to specify the markup stamp template file (XSP) to be used with the markup stamp entity tool. Use Brava!'s Stamp Template tool (Markup > Stamp Templates > New) to create a file that can be specified for use with the markup toobar's Add Stamp tool.	

PreFinalizedRedactionTransparency	
Type	Example
Double	`PreFinalizedRedactionTransparency=50`
Description	
Used to get/set the current transparency of redaction entities before they are finalized (published) as a redaction. The value of the transparency can range from 0 (solid) to 100 (transparent).	

RedactionOption	
Type	Examples

Long, String	<See below>

Description

This property is used to get/set the value of the redaction options identified by the given `PathType` in the table below. The property translates to methods:

```
get: RedactionOption(Long PathType, BSTR*pVal)
set: RedactionOption(Long PathType, BSTR newVal)
```

The `PathType` value would be a number from 0 to 11 (see table below).
So, if `PathType = 0` (Path to the redaction scripts folder), then the `pVal` should be a path like `"C:\MyScripts\"`
If `PathType = 1` (Maximum number of reasons allowed), then `pVal` would be a string with a number value like `"5"`.

Examples:

```
viewObject.set_RedactionOption(1, "3");
```
Sets the maximum allowed redaction reasons to 3

```
viewObject.set_RedactionOption(2, "Reason1<|>Reason2");
```
Initializes the redaction reasons list with Reason1 and Reason2

```
viewObject.set_RedactionOption(3, "5");
```
Sets the maximum allowed search strings to 5

```
viewObject.set_RedactionOption(5, "Account Number (in form).xrs<|>Passport Number (in form).xrs");
```
Sets the redaction script files to populate the *Redact Using Scripts* dialog

```
viewObject.set_RedactionOption(11, "false");
```
Disallows users from adding their own redaction reason

PathType	**Value**
0	Path to the redaction scripts folder
1	Maximum number of reasons allowed
2	Redaction reasons
3	Maximum number of search strings allowed
4	Search strings
5	Most recently used scripts
6	Most recently used reasons
7	Sort reasons
8	Active scripts
9	Search ‘from’ phrases

10	Search ‘to’ phrases
11	Allow user reasons

MarkupBurnedIn	
Type	Default
Boolean	<Read-Only>
Description	
This property returns TRUE if the currently file viewed has markup(s) burned in. Note that this property only applies to a CSF file.	

VersionMarkup	
Type	Default
Boolean	TRUE
Description	
This property is used to get/set if the markup functionality is enabled/disabled.	

Markup: Methods

GotoMarkupPage			
Syntax			
`GotoMarkupPage( [in] int pgIterType)`			
Description			
This method loads a document or drawing page containing at least one markup entity based on the value of `pgIterType` .			
Parameters	I/O	Type	Description
`pgIterType`	In	Integer	Controls which page containing a markup entity to open. The acceptable values are: **Value** | **Page** 1 | First

			2 Next 3 Previous 4 Last

ShowFindAndRedactCtrl
Syntax
`ShowFindAndRedactCtrl()`
Description
This method causes the control to display the *Find and Redact* dialog.

ShowFindAndRedactFromToCtrl
Syntax
`ShowFindAndRedactFromToCtrl()`
Description
This method causes the control display the *Find and Redact From/To* dialog.

RedactFromScript			
Syntax			
`RedactFromScript( [in] BSTR RedactionScript, [out, retval] int * NumRedactions)`			
Description			
This method applies a redaction script file and returns the resulting number of redactions created.			
Parameters	I/O	Type	Description
`RedactionScript`	In	String	Indicates the full path to the redaction script file to use.
`NumRedactions`	Out	Integer	Returns the number of redactions created from script file indicated by `RedactionScript`.

ValidateRedactionScript
Syntax
`ValidateRedactionScript( [in] BSTR RedactionScript, [out, retval] BOOL * Valid)`
Description

This method is used to determine if a specified text file is a valid redaction script.			
Parameters	I/O	Type	Description
`RedactionScript`	In	String	Indicates the full path to the redaction script file to validate.
`Valid`	Out	Boolean	If script is valid, returned as TRUE.

ConsolidateMarkups
Syntax
`ConsolidateMarkups()`
Description
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 an editable markup is not currently open for edit, this method creates a new editable markup.

ExecuteChangemark			
Syntax			
`ExecuteChangemark( [in] BSTR CMID, [in] long IDMode)`			
Description			
This method causes the control to display a particular Changemark or Checkview entity. The Changemark or Checkview entity displayed is determined by `CMID` and `IDMode`. See `GetChangemarkInfo()` for additional information.			
Parameters	I/O	Type	Description
`CMID`	In	String	Indicates the Changemark ID
`IDMode`	In	Long	Valid values for `IDMode` and their effect on the interpretation of `CMID` are:

IDMode	CMID	Description
1	Title	The control will attempt to execute the first Changemark entity within the list whose title is equal to CMID
2	Index	The control will attempt to execute the Changemark entity within the list whose index is equal to CMID
3	GUID	The control will attempt to execute the Changemark entity within the list whose markup entity GUID is equal to CMID
4	GUID	The control will attempt to execute the

					Checkview entity within the list whose markup entity GUID is equal to CMID. If Checkview entities exist but the CMID string is not a valid GUID, the first Checkview in the list will be activated

GetChangemarkInfo

Syntax

```
GetChangemarkInfo( [in] long Index, [out] BSTR * Guid, [out] BSTR * Title, [out] BSTR * Comment, [out] BSTR * Author, [out] BSTR * Link, [out] BSTR * Metadata, [out] BSTR * Type, [out] BSTR * State, [out] long * Time, [out] long * PageNum, [out] BOOL * IsEditable)
```

Description

This method retrieves information about the Changemark entity specified by `Index`. Use property `NumberChangemarks` to determine the total number of Changemark entities that currently exist.

Parameters	I/O	Type	Description
`Index`	In	Long	Specifies the index of the Changemark entity within the list of Changemark entities.
`Guid`	Out	String	This is the unique string identifier of the entity.
`Title`	Out	String	Title of the Changemark entity.
`Comment`	Out	String	Comment of the Changemark entity.
`Author`	Out	String	Author of the Changemark entity.
`Link`	Out	String	Hyperlink associated with the Changemark entity, if any exists.
`Metadata`	Out	String	Not currently used.
`Type`	Out	String	Type of the Changemark entity.
`State`	Out	String	State of the Changemark entity.
`Time`	Out	Long	Indicates when the Changemark entity was created, measured in UTC time.
`PageNum`	Out	Long	Indicates the page number on which the Changemark entity exists.
`IsEditable`	Out	Boolean	Indicates if the Changemark entity is editable or not.

SetChangemarkConfig

Syntax

```
SetChangemarkConfig( [in] BSTR ChangemarkConfigXML)
```

Description

This method reads the Changemark configuration, including type, state, and color values, from the specified XML string (from *ChangemarkConfig.xml)*.
Example:

```
<?xml version="1.0" encoding="utf-8"?>
 <ChangemarkConfiguration>
   <Types>
     <ChangemarkType name="Action">
       <State name="For Discussion" defaultState="true">
         <Color>
           <red>0</red>
           <green>0</green>
           <blue>255</blue>
         </Color>
       </State>
       <State name="Approved">
         <Color>
           <red>0</red>
           <green>255</green>
           <blue>0</blue>
         </Color>
       </State>
     </ChangemarkType>
   </Types>
</ChangemarkConfiguration>
```

Parameters	I/O	Type	Description
`ChangemarkConfigXML`	In	String	Indicates the Changemark configuration XML file.

GetSignatureStrings

Syntax

```
GetSignatureStrings( [in] BSTR UserID, [in] BSTR DocID, [in] BSTR DocVersion,
[out] IBxStringAnswer * SignatureFilename, [out] IBxStringAnswer *
InitialsFilename, [out] IBxStringAnswer * SealImageFilename, [out]
IBxStringAnswer * NameString, [out] IBxStringAnswer * TitleString)
```

Description

This method returns the signature item data for a given `UserID`.

Parameters	I/O	Type	Description
UserID	In	Long	Not currently used.
DocID	In	String	Not currently used.
DocVersion	In	String	Not currently used.
SignatureFilename	Out	String	The filename of the signature raster is returned in `SignatureFilename.Answer`
InitialsFilename	Out	String	The filename of the initials raster is returned in `InitialsFilename.Answer`

SealImageFilename	Out	String	The filename of the seal image is returned in `SealImageFilename.Answer`
NameString	Out	String	The name string is returned in `NameString.Answer`
TitleString	Out	String	The title string is returned in `TitleString.Answer`

NewMarkupStampTemplate
Syntax
`NewMarkupStampTemplate()`
Description
This method causes the control to create a new, un-named, editable markup stamp template.

CloseMarkupStampTemplate			
Syntax			
`CloseMarkupStampTemplate( [in] BOOL promptSave, [out] BOOL * Canceled)`			
Description			
This method causes any currently open editable markup stamp template file to be closed.			
Parameters	I/O	Type	Description
`promptSave`	In	Boolean	If set to TRUE, and there is currently a stamp template file open that is in an unsaved state, the control will prompt the user to save the file. This prompt allows the user to cancel the close of the stamp template.
`Canceled`	Out	Boolean	If the user chooses to cancel, this parameter will be set TRUE upon return of the method.

CloseMarkupTakeoffTemplate			
Syntax			
`CloseMarkupTakeoffTemplate( [in] BOOL promptSave, [out] BOOL * Canceled)`			
Description			
This method causes any currently open takeoff template file to be closed.			
Parameters	I/O	Type	Description
`promptSave`	In	Boolean	If set to TRUE, and there is currently a takeoff template file open that is in an unsaved state, the control will prompt the user to save the file. This prompt allows the user to cancel the close of the takeoff template.

Canceled	Out	Boolean	If the user chooses to cancel, this parameter will be set TRUE upon return of the method.

CloseMarkupStampTemplateEx			
Syntax			
CloseMarkupStampTemplateEx([in] BOOL promptSave, [in] IBxBoolAnswer * Canceled)			
Description			
This method causes any currently open stamp template file.			
Parameters	I/O	Type	Description
promptSave	In	Boolean	If set to TRUE, and there is currently an editable stamp template file open that is in an unsaved state, the control will prompt the user to save. This prompt allows the user to cancel the close of the stamp template file.
Canceled	Out	Boolean Answer	Canceled.Answer is set to TRUE if user chooses to cancel the stamp template close.

SaveMarkupStampTemplate
Syntax
SaveMarkupStampTemplate()
Description
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			
Syntax			
SaveAsMarkupStampTemplate([in] BSTR fileName)			
Description			
This method causes the control to save the current state of the currently open editable markup stamp template file to the file indicated by fileName.			
Parameters	I/O	Type	Description
fileName	In	String	Indicates current file name.

SaveMarkupTakeoffTemplate

Syntax

```
SaveMarkupTakeoffTemplate()
```

Description

This method causes the control to save all changes to the currently available takeoff categories to a template. If the currently available takeoff categories have not been previously saved, the control will prompt for a file name using one of the methods described in the File IO Option Section.

SaveAsMarkupTakeoffTemplate

Syntax

```
SaveAsMarkupTakeoffTemplate( [in] BSTR FileName)
```

Description

This method causes the control to save the current state of the available takeoff categories to the file indicated by `FileName`.

Parameters	I/O	Type	Description
`FileName`	In	String	Indicates file name to save to.

OpenMarkupStampTemplate

Syntax

```
OpenMarkupStampTemplate( [in] BSTR fileName)
```

Description

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

Parameters	I/O	Type	Description
`fileName`	In	String	Indicates stamp template file name to open.

OpenMarkupTakeoffTemplate

Syntax

```
OpenMarkupTakeoffTemplate( [in] BSTR fileName)
```

Description

This method causes the control to open the existing takeoff template file specified by `fileName`. If there is currently a takeoff template open or a manually created takeoff category exists, this takeoff template file's categories will be added to the existing categories.

Parameters	I/O	Type	Description
`fileName`	In	String	Indicates takeoff template file name to open.

ModifyMarkupEntity

Syntax

```
ModifyMarkupEntity( [in] BSTR ID, [in] long Action, [in] BSTR ActionInfo)
```

Description

This method causes the control to modify or affect a particular markup entity or group of markup entities. The markup entities affected are determined by `ID`.

Parameters	I/O	Type	Description
`ID`	In	String	The `ID` parameter must be a string list of markup entity GUIDs delimited by "<|>". See the `MarkupEntityModified` event for more information about markup GUIDs
`Action`	In	Long	Valid values for Action and their effect on the markup entity are: Value: 0 — The control will attempt to select the markup entities Value: 1 — The control will attempt to deselect the markup entities
`ActionInfo`	In	String	Not currently used.

VisitMarkupEntity

Syntax

```
VisitMarkupEntity([in] BSTR Action, [in] BSTR Filter, [out, retval] BSTR* Guid)
```

Description

This method allows an integration to step through and select markup entities within the markup file currently open for edit. Markup entities will be displayed and selected from first to last page and from top-left to bottom-right on a given page.

Parameters	I/O	Type	Description
`Action`	In	String	Action can be one of the following: **Action**: First — **Description**: The first editable markup entity is viewed and selected

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td>Next</td><td>The markup selection is moved to the next selectable entity</td></tr>
<tr><td>Prev</td><td>The markup selection is moved to the previous selectable entity</td></tr>
<tr><td>Last</td><td>The last editable markup entity is viewed and selected</td></tr>
</table>
</td></tr>
<tr><td>Filter</td><td>In</td><td>String</td><td>Filter allows the list of visitable markup entities to be limited to a particular entity type. Filter can be one of the following:
<table>
<tr><td>Filter</td><td>Description</td></tr>
<tr><td>All</td><td>Allow all markup entities to be visited</td></tr>
<tr><td>Geometric</td><td>Allow only geometric markup entites to be visited. This includes Lines, Polylines, Polygons, Arcs, Ellipses, Clouds, and Arrows</td></tr>
<tr><td>Text</td><td>Allow only text markup entities to be visited</td></tr>
<tr><td>Image</td><td>Allow only image markup entities to be visited</td></tr>
<tr><td>Changemarks</td><td>Allow only Changemarks notes markup entities to be visited</td></tr>
<tr><td>Redaction</td><td>Allow only redaction markup entities to be visited</td></tr>
<tr><td>Stamp</td><td>Allow only stamp markup entities to be visited</td></tr>
</table>
</td></tr>
<tr><td>Guid</td><td>Out</td><td>String</td><td>Returns the string value of the unique identifier associated with the currently selected (visited) markup entity.</td></tr>
</table>

ViewMarkupEntity

Syntax

```
ViewMarkupEntity ([in] BSTR EntityID, [out, retval] BOOL* Found)
```

Description

This method allows an integration to cause the viewer control to adjust the current viewstate so that the markup entity identified by `EntityID` is visible.

Parameters	I/O	Type	Description
`EntityID`	In	String	Unique identifier associated with the target markup entity. See `GetMarkupEntityList` for information about determining available markup entity identifiers..
`Found`	In	Boolean	Returns a Boolean value indicating if the markup entity was found.

QueryExists			
Syntax			
`QueryExists( [in] BSTR Query, [out, retval] BOOL * pVal)`			
Description			
This method returns TRUE if the markup query specified by `Query`, resolves to a non-empty list of markup elements, FALSE otherwise.			
Parameters	I/O	Type	Description
`Query`	In	String	The query used to create a markup filter.
`pVal`	Out	Boolean	`pVal` is set to TRUE if the markup query specified by `Query`, resolves to a non-empty list of markup elements, FALSE otherwise.

ClearMarkupAndHistory			
Syntax			
`ClearMarkupAndHistory( [in] BOOL ShowFilterDialog)`			
Description			
This method permanently deletes all active markups from memory and clears the undo history. If the markup was opened from a file or was previously saved to a file, the contents of that file are not modified. If session scripting is active, the session is cleared.			
Parameters	I/O	Type	Description
`ShowFilterDialog`	In	Boolean	If TRUE, the filter dialog is displayed, allowing the user to select what type of markup to delete.

ClearMrkUndoHistory
Syntax
`ClearMrkUndoHistory()`
Description
This method clears the markup undo history.

TakeOwnershipOfAllMarkups
Syntax
`TakeOwnershipOfAllMarkups()`
Description
This method sets the owner of all editable markup entities to the current user. To take ownership of only a specific entity type, use `TakeOwnershipOfMarkups`

SignAll()
Syntax
`SignAll()`
Description
This method causes all currently unsigned Signature entities to be signed.

TakeOwnershipOfMarkups			
Syntax			
`TakeOwnershipOfMarkups([in] int markupEntityType ))`			
Description			
This method sets the owner of all markup entities of the specified type in the editable markup to the current user. For example, to take ownership of only Redact Area entities (blockouts), set `markupEntityType=16`.			
Parameters	I/O	Type	Description
`markupEntityType`	In	Integer	If set to 0, this is the same as `TakeOwnershipOfAllMarkups`. Valid values of `markupEntityType` are:

Value	Description	Value	Description
1	Line	17	Stamp
2	Polyline	18	Arrow
3	Sketch	19	Advanced arrow
4	Arc	20	Takeoff line
5	Ellipse	21	Takeoff polyline
6	Polygon	22	Takeoff polygon
7	Sketch polygon	23	Takeoff rectangle
8	Crossout, Strikethrough, Strikeout, Underline	24	Takeoff circle
10	Text	25	Takeoff count
11	Changemark entity	26	Changemark reply
12	Cloud	27	Highlight

			13 Image 28 Signature 15 Scratchout 29 Checkview 16 Redact Area 30 Checkview reply 31 Bookmark

EditScript			
Syntax			
`EditScript( [in] BSTR ScriptFileName, [out, retval] BSTR * pVal)`			
Description			
This method shows the Script Editor dialog.			
Parameters	I/O	Type	Description
`ScriptFileName`	In	String	Indicates name of redaction script file to edit. If not blank and specifies a valid script file, the script editor is shown. If blank, the script editor is shown for a new script.
`pVal`	Out	String	Contains the name of the newly created script, if one was saved.

ValidateRedactionScriptEx			
Syntax			
`ValidateRedactionScriptEx( [in] BSTR RedactionScript, [in] BOOL ShowErrors, [out, retval] BSTR * pResponse)`			
Description			
This method validates the redaction script file specified by `RedactionScript`.			
Parameters	I/O	Type	Description
`RedactionScript`	In	String	Indicates the redaction script file to validate.
`ShowErrors`	In	Boolean	If TRUE, error messages, if any, are reported (Error codes 24 or 37, see Message ID Section).
`pResponse`	Out	String	Script specific errors are returned in `pResponse` in XML form, as a bitwise OR of the following errors: **Value** **Description** 0x00 No error 0x01 Cannot read script file 0x02 Cannot write script file

			0x04 Invalid script format 0x08 Invalid XML format 0x10 Invalid page range 0x20 Invalid parameter 0x40 Missing parameter 0x80 Overwrote match whole word (warning) 0x100 Future script format 0x200 Invalid markup

GetTextUnderMarkupEntity

Syntax

```
GetDocTextUnderMarkupEntity( [in] BSTR EntityID, [in] BOOL
AddNewlinesBetweenFragments, [out, retval] BSTR * DocText)
```

Description

This method returns the document text that lies under the bounding box of the markup element identified by the guid `EntityID`.

Parameters	I/O	Type	Description
`EntityID`	In	String	Indicates the markup entity to retrieve text from under.
`AddNewlinesBetweenFragments`	In	Boolean	Specifies whether newline characters should be added between separate fragments of text found under the markup element.
`DocText`	Out	String	Document text returned.

GetEditingMarkupText

Syntax

```
GetEditingMarkupText([out, retval] BOOL *Editing)
```

Description

This method returns TRUE (1) if there is markup text entity currently being edited.

Parameters	I/O	Type	Description
`Editing`	Out	Boolean	Returns TRUE if a markup text entity is currently being edited. Returns FALSE if no markup text entity is being edited.

GetMarkupEntityListEx			
Syntax			
`GetMarkupEntityListEx( [in] BSTR Filter, [out,retval] BSTR* GuidList)`			
Description			
This method returns a list of GUIDs of markups satisfying conditions described in `Filter`.			
Parameters	I/O	Type	Description
`Filter`	In	String	XML data specifying the conditions that should be satisfied by the markups.
`GuidList`	Out	String	A string of the GUIDs separated by "<|>".

GetMarkupEntityList			
Syntax			
`GetMarkupEntityList( [in] int MarkupEntityType, [in] BSTR MetadataString, [out, retval] BSTR *GuidList)`			
Description			
This method returns a list of GUIDs of markups of the given `MarkupEntityType` and satisfying conditions described in `MetaDataString`.			
Parameters	I/O	Type	Description
`MarkupEntityType`	In	Integer	Indicates the markup entity type. `MarkupEntityType = 0`, the GUID list for ALL markups will be returned. See `TakeOwnershipOfMarkups` for a list of valid values of `MarkupEntityType`.

`MetadataString`	In	String	`<?xml version="1.0" encoding="UTF-8"?>`
			`<MarkupEntityListOptions>`
			`<Pagelist></Pagelist>`
			`<CopyableOnly>false</CopyableOnly>`
			`<SelectedOnly>false</SelectedOnly>`
			`<SkipDeleted>true</SkipDeleted>`
			`</MarkupEntityListOptions>`
			`Pagelist` can be left out, or be an empty string, to denote all pages. `Pagelist` is expressed similar to the published pages list ie: 1-2,4,7-9. Values shown are the default values. If `MetaDataString` is empty, GUID list for the given markup type for all pages of the document with the shown default values will be used.

`GuidList`	Out	String	A string of the GUIDs separated by "<|>". .

UpdateStampEntityTokenSet			
Syntax			
`UpdateStampEntityTokenSet( [in] BSTR GUID, [in] BSTR Tokens)`			
Description			
This method takes a markup entity identifier, `GUID`, and an XML string, `Tokens`, of resolved tokens and applies them to the token set with the matching identifier.			
Parameters	I/O	Type	Description
`GUID`	In	String	Indicates the markup unique identifier.
`Tokens`	In	String	Indicates any token values. Only %dbupdatestring tokens are resolvable through this method. The schema referenced in the original Tokens XML string may be used to validate the changed XML. Invalid token XML should not be expected to be applied by the viewer.

GetDocumentLoaderInfo			
Syntax			
`GetDocumentLoaderInfo([out,retval] BSTR* LoaderInfo)`			
Description			
This method returns a string containing the name and version of the format loader used to create the view of the current document or drawing. In the case of CSF or XDL files, the `LoaderInfo` will describe the format loader used in the original creation of the file.			
Parameters	I/O	Type	Description
`LoaderInfo`	Out	String	A sting in the form <Format Loader Name>|<Format Loader Version>.

GetMarkupLoaderInfo			
Syntax			
`GetMarkupLoaderInfo([in] BSTR MrKName,[out,retval] BSTR* LoaderInfo)`			
Description			
This method returns a string containing the name and version of the format loader that was in use at the creation a given markup file.			
Parameters	I/O	Type	Description
`LoaderInfo`	Out	String	A sting in the form <Format Loader Name>|<Format Loader Version>.
`MrkName`	Out	String	Returns the name of the markup files.

GetMarkupExtentsMatchDocumentExtents			
Syntax			
`GetMarkupExtentsMatchDocumentExtents([in] BSTR MrKName,[out, retval] BOOL *Match)`			
Description			
This method returns TRUE if the first page of the markup document has extents approximately (.01) equal to the matching page of the currently open document or drawing.			
Parameters	I/O	Type	Description
`MrkName`	In	String	The filename of the markup file.
`Match`	Out	Boolean	Boolean indicating if the markup file's extents match the extents of the document.

GetRedactionScriptExtentsMatchDocumentExtents			
Syntax			
`GetRedactionScriptExtentsMatchDocumentExtents([in] BSTR ScriptName,[out, retval] BOOL *Match)`			
Description			
This method returns TRUE if all markup pages contained in a redaction script file have extents approximately (.01) equal to the first page of the currently open document or drawing.			
Parameters	I/O	Type	Description
`ScriptName`	In	String	The filename of the script file.
`Match`	Out	Boolean	Boolean indicating if the redaction script's extents match the extents of the document.

TestMarkup			
Syntax			
`TestMarkup([in] BSTR Filename,[out, retval] int *Result)`			
Description			
This method tests a markup for validity. The method should be called after a document has been opened.			
Parameters	I/O	Type	Description
`Filename`	In	String	The name of the markup file to test.
`Result`	Out	Int	Possible values are: 0 = Valid markup file 1 = Invalid markup file

			2 = Future file format 3 = Incompatible extents. The first page of the markup document has extents that are not equal (within 1%) to the matching page of the currently open document or drawing.

PasteImageFromClipboard
Syntax
`PasteImageFromClipboard()`
Description
If the Windows clipboard contains valid image (pixel) data, this method will paste the image into the currently open editable markup. if there is no editable markup open, a new editable markup will be created.

SelectMarkupOnCurrentPage			
Syntax			
`SelectMarkupOnCurrentPage([in] int Filter)`			
Description			
This method will select markup elements on the current page.			
Parameters	I/O	Type	Description
`Filter`	In	Integer	Used to specify the type of element to select. Use 0 to select all and -1 to deselect all. Possible values are -1 = DESELECT_ALL 0 = SELECT_ALL 1 = LINE 2 = POLYLINE 3 = NONEDIT_POLYLINE 4 = ARC 5 = ELLIPSE 6 = POLYGON 7 = NONEDIT_POLYGON 8 = GEOMETRYGROUP 9 = HYPERCHANGEMARK 10 = TEXT 11 = CHANGEMARK 12 = CLOUD 13 = RASTER 14 = SQUIGLE 15 = NONEDIT_SQUIGLE 16 = BLOCKOUT 17 = GROUP 18 = ARROWLINE 19 = BYZANTINE_ARROWLINE 20 = MEASURE_LINE

			21 = MEASURE_POLYLINE 22 = MEASURE_POLYGON 23 = MEASURE_RECTANGLE 24 = MEASURE_CIRCLE 25 = MEASURE_COUNT 26 = CHANGEMARK_REPLY 27 = HIGHLIGHT 28 = SIGNATURE 29 = CHECKVIEW 30 = CHECKVIEW_REPLY 31 = BOOKMARK

User Interface Customize Section

User Interface Customization Properties and Methods

Properties

`DisplayToolbarTools`
`DisplayToolbarView`
`DisplayToolbarPage`
`DisplayFindCtrl`

Note that the above properties are completely deprecated and perform no function. See the `DisplayCtrl()` method for displaying controls.

SupressInfoMessages	
Type	Default
Boolean	`FALSE`
Description	
This property is used to get/set if the control displays error message. Setting this property to TRUE causes the control to not display error messages.	

EnableRightMouseButtonMenu	
Type	Default
Boolean	`TRUE`
Description	
This property is used to get/set if the control displays a menu when the right mouse button is clicked (while the cursor is within the drawing/document view area).	

UI and Functionality Allows	
Type	Default
Boolean	`TRUE` <for all>
Description	
The following properties can be used to disable/enable particular control functions. If a given property is set to FALSE, the control removes the corresponding UI thus disallowing the corresponding functionality:	

```
AllowFileOpen
AllowPrinting
AllowMarkup
AllowMeasurement
AllowLayers
AllowFind
AllowCopyText
AllowMrkReview

AllowMrkSaveAs
AllowMrkSave
AllowMrkOpen
AllowMrkNew
AllowPageControl
AllowRotate
AllowBKColor
AllowMonochrome
```

Methods

CtrlDisplayed			
Syntax			
`CtrlDisplayed( [in] int ctrlID, [out, retval] BOOL * pVal)`			
Description			
This method returns if a specified control (button or other UI element) has been removed or not.			
Parameters	I/O	Type	Description
`ctrlID`	In	Integer	See Control Identification Section for a list of acceptable control IDs.
`pVal`	Out	Boolean	Returns FALSE if the control identified by `ctrlID` has been removed, TRUE otherwise.

CtrlEnabled			
Syntax			
`CtrlEnabled( [in] int ctrlID, [out, retval] BOOL * pVal)`			
Description			
This method returns if a specified control (button or other UI element) has been disabled or not.			
Parameters	I/O	Type	Description
`ctrlID`	In	Integer	See Control Identification Section for a list of acceptable control IDs.

pVal	Out	Boolean	Returns FALSE if the control identified by ctrlID has been disabled, TRUE otherwise.

DisplayCtrl			
Syntax			
DisplayCtrl([in] int ctrlID, [in] BOOL newVal)			
Description			
This method allows a control (button or other UI element), specified by ctrlID, to be displayed or removed based on the value of newVal.			
Parameters	I/O	Type	Description
ctrlID	In	Integer	See Control Identification Section for a list of acceptable control IDs.
newVal	In	Boolean	If TRUE, the control is displayed, if FALSE, the control is removed.

EnableCtrl			
Syntax			
EnableCtrl([in] int ctrlID, [in] BOOL newVal)			
Description			
This method allows a control (button or other UI element), specified by ctrlID, to be enabled or disabled based on the value of newVal.			
Parameters	I/O	Type	Description
ctrlID	In	Integer	See Control Identification Section for a list of acceptable control IDs.
newVal	In	Boolean	If TRUE, the control is enabled, if FALSE, the control is disabled.

EnableAcceleratorKey			
Syntax			
EnableAcceleratorKey([in] int KeyID, [in] BOOL Shift, [in] BOOL newVal)			
Description			
This method allows a container to enable/disable hot key (<Ctrl> + <char>) combinations.			
Parameters	I/O	Type	Description
keyID	In	Integer	Specifies key combination. See Control Key Combinations for a list of valid values for keyID.
Shift	In	Boolean	Indicates if the <Shift> key is pressed in specified key combination.
newVal	In	Boolean	If FALSE, the key command is disabled, if TRUE the key command

			is enabled.

AcceleratorKeyEnabled			
Syntax			
`AcceleratorKeyEnabled( [in] int KeyID, [in] BOOL Shift, [out, retval] BOOL * pVal)`			
Description			
This method returns the enabled/disabled state of a given key combination specified by `keyID`.			
Parameters	I/O	Type	Description
`keyID`	In	Integer	Specifies key combination. See Control Key Combinations for a list of valid values for `keyID`.
`Shift`	In	Boolean	Indicates if the <Shift> key is pressed in specified key combination.
`newVal`	Out	Boolean	Returns if the key command is disabled or enabled.

GetFocusedBXID			
Syntax			
`GetFocusedBXID ([out, retval] int *ctrlID)`			
Description			
This method returns the `ctrlID` of the control (button or other UI element) that currently has keyboard focus.			
Parameters	I/O	Type	Description
`ctrlID`	Out	Integer	See Control Identification Section for a list of acceptable control IDs.

SetFocusedBXID			
Syntax			
`GetFocusedBXID ([in] int *ctrlID)`			
Description			
This method sets keyboard focus to the control (button or other UI element) identified by `ctrlID`.			
Parameters	I/O	Type	Description
`ctrlID`	In	Integer	See Control Identification Section for a list of acceptable control IDs.

Measurement Section

Measurement Properties and Methods

Properties

CalibrationComplete	
Type	Default
Boolean	<Read-Only>
Description	
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 Calibrate tool prior to using any of the other measurement tools (exception is the Measurement Count tool). Calibration must be re-done every time a new document or drawing is opened. Note that after a calibration has been completed via the Calibrate tool, the currently active mouse tool will be automatically set to the Measure Line tool. If this is not the desired behavior, a container must adjust after receiving the `CalibrationComplete()` event. See the `CalibrationComplete` event in the Events Section. *Example*: `If(viewObject.CalibrationComplete)` `viewObject.MouseTool = 7` (Measure line tool) Specifies to only set the measure line tool if calibration is complete.	

Methods

ShowMeasurementSettings
Syntax
`ShowMeasurementSettings()`
Description
This method causes the control to display the modal Measurement Settings dialog.

SetDefaultMeasureScale
Syntax
`SetDefaultMeasureScale( [in] double Scale)`
Description

This method sets the scale factor that will be applied to all measurements on pages where a scale factor has not been explicitly set, either through the API or through the viewer GUI.

Parameters	I/O	Type	Description
`Scale`	In	Double	Specifies the scale factor to apply to all measurements.

GetDefaultMeasureScale

Syntax

```
GetDefaultMeasureScale( [out, retval] double * Scale)
```

Description

This method returns the scale factor that is applied to all measurements on pages where a scale factor has not been explicitly set.

Parameters	I/O	Type	Description
`Scale`	Out	Double	Scale factor applied to all measurements.

SetDefaultMeasureScaleForPage

Syntax

```
SetDefaultMeasureScaleForPage( [in] long PageNum, [in] double Scale)
```

Description

This method sets the scale factor that is applied to all measurements on the page specified by `PageNum`. The page number is zero-based.

Parameters	I/O	Type	Description
`PageNum`	In	Long	Specifies the page to apply measurement scale factor.
`Scale`	In	Double	Specifies the scale factor to apply to all measurements on page.

GetDefaultMeasureScaleForPage

Syntax

```
GetDefaultMeasureScaleForPage( [in] long PageNum, [out, retval] double *
Scale)
```

Description

This method returns the scale factor that is applied to all measurements on the page specified by `PageNum`. If no scale value has been explicitly set for this page, the overall default value will be returned. The page number is zero-based.

Parameters	I/O	Type	Description
`PageNum`	In	Long	Specifies the page to apply measurement scale factor.

Scale	Out	Double	Specifies the scale factor applied to all measurements on page.

SetMeasurementOption

Syntax

```
SetMeasurementOption( [in] LONG OptionID, [in] BSTR newVal)
```

Description

This method sets the measurement option for the given OptionID to the given newVal.

Parameters	I/O	Type	Description
OptionID	In	Long	Valid values for OptionID are:

Value	Option	Description
1	English system	The string will be a series of name/value pairs separated by "<|>". The name and value are separated by a '|' character. The name will be the displayed name of the scale. The value will be the scale value.
2	Metric system	The string will be a series of name/value pairs separated by "<|>". The name and value are separated by a '|' character. The name will be the displayed name of the scale. The value will be the scale value.
3	Default setting	A description of the default measurement scale. This will be a string made of the following four values separated by "<|>". The default measurement system

0	English with fractional values
1	English
2	Metric
3	No units

The default Units

0	Inches
1	Feet
2	Meters
3	Millimeters

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td></td><td></td><td>
<table>
<tr><td>4</td><td>Centimeters</td></tr>
<tr><td>5</td><td>Miles</td></tr>
<tr><td>6</td><td>Kilometers</td></tr>
<tr><td>7</td><td>No units</td></tr>
<tr><td>8</td><td>Yards</td></tr>
</table>

The number of decimal places to display in measurements
<table>
<tr><td>0</td><td>1 decimal place</td></tr>
<tr><td>1</td><td>2 decimal place</td></tr>
<tr><td>2</td><td>3 decimal place</td></tr>
<tr><td>3</td><td>4 decimal place</td></tr>
<tr><td>4</td><td>5 decimal place</td></tr>
<tr><td>5</td><td>6 decimal place</td></tr>
</table>

The accuracy of architectural measurements
<table>
<tr><td>0</td><td>Whole numbers</td></tr>
<tr><td>1</td><td>Half units</td></tr>
<tr><td>2</td><td>Quarter units</td></tr>
<tr><td>3</td><td>Eighth units</td></tr>
<tr><td>4</td><td>Sixteenth units</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>Default scale</td><td>The default scale to be used for pages on which no scale is set. This is a name/value pair, separated by a '|' character.
The name is the name of the scale. The value is the scale value to apply to measurements.</td></tr>
</table>
</td></tr>
<tr><td>newVal</td><td>In</td><td>String</td><td>Specifies the measurement option indicated by OptionID.</td></tr>
</table>

GetMeasurementOption
Syntax
`GetMeasurementOption( [in] long OptionID, [out, retval] BSTR * Option)`

Description			
This method returns the value of a measurement `Option`.			
Parameters	I/O	Type	Description
`OptionID`	In	Long	See `SetMeasurementOption()` for possible values for `OptionID`.
`Option`	Out	String	Returns the value of the measurement specified by `OptionID` .

Visual Rights Section

Visual Rights Properties and Methods

Properties

Visual Rights Properties	
Type	Default
Boolean	<Read-Only>
Description	

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 read-only and immutable if the file currently viewed is a CSF file. If the file is not CSF, these properties can be read and set.
 	
`VRPrintingEnabled`
`VRCopyTextEnabled`
`VRMeasurementEnabled`
`VRLayersEnabled`
`VRMarkupEditEnabled`
`VRRepublishingEnabled`
`VRDateExpiredEnabled` (read-Only)
`VRPasswordProtectEnabled` (read-Only)
`VRMarkupEnabled`
`VRBannerEditEnabled`
`VRDownloadOriginalEnabled`
`VRSaveAsEnabled`
`VRBurnInMrkEnabled`	

Methods

ShowMeasurementSettings
Syntax
`ShowVisualRightsSettings()`
Description
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

Help Properties and Methods

Properties

RequestHelpDisplay	
Type	Default
Boolean	`FALSE`
Description	
This property is used to get/set how to display the control window's help file. If set to TRUE, the control will fire the `DisplayHelp()` method 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.	

LaunchUrlHelp	
Type	Default
Boolean	`FALSE`
Description	
This property is used to get/set if the control launches the URL specified by the `HelpUrl` property when any of the control's help buttons are pressed. See the `RequestHelpDisplay` and `HelpUrl` properties, the `DisplayHelpTopic()` method and the `DisplayHelp` event for more details.	

HelpUrl	
Type	Example
String	`HelpURL=C:\Users\someuser.IGC\IGC\x7_1`
Description	
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 the Help Topic IDs Section for a listing of possible TopicIDs.	

Methods

DisplayHelpTopic			
Syntax			
`DisplayHelpTopic( [in] short TopicID)`			
Description			
This method causes the control to either launch the Windows help file with the given `TopicID`, or fire the `DisplayHelp` event with the given `TopicID`, based on the value of `RequestHelpDisplay`.			
Parameters	I/O	Type	Description
`TopicID`	In	Short	See Help Topic IDs Section for a list of acceptable `TopicID` values.

Version/License Section

Version/License Properties and Methods

Properties

VersionString	
Type	Default
String	<Read-Only>
Description	
This property returns the version of the control. The version string is formatted: `Major.Minor1.Minor2.BuildNo`	

LicenseString	
Type	Default
String	<Read-Only>
Description	
This property returns the license string of the control, if applicable.	

PinNumber	
Type	Default
Long	<Read-Only>
Description	
This property returns the customer number based on the license.	

DistributorNumber	
Type	Default
Long	<Read-Only>
Description	
This property retrieves the distributor number from the license.	

Methods

FormatEnabled			
Syntax			
`FormatEnabled( [in] int FormatID, [out, retval] BOOL * pVal)`			
Description			
This method checks whether a particular format is enabled by the license.			
Parameters	I/O	Type	Description
`FormatID`	In	Integer	See `FormatDescription` for possible values of `FormatID`.
`pVal`	Out	Boolean	Returns if the format specified by `FormatID` is enabled by license .

FormatExtensions			
Syntax			
`FormatExtensions( [in] int FormatID, [out, retval] BSTR * pExts)`			
Description			
This method gets a list of the file extensions associated with the format specified by `FormatID` .			
Parameters	I/O	Type	Description
`FormatID`	In	Integer	See `FormatDescription` for possible values of `FormatID`.
`pExts`	Out	String	Extensions are returned as '|' delimited list in `pExts`.

FormatDescription			
Syntax			
`FormatDescription( [in] int FormatID, [out, retval] BSTR * pDsc)`			
Description			
This method returns a description of the format specified by `FormatID`.			
Parameters	I/O	Type	Description
`FormatID`	In	Integer	Indicates format identification number. Possible values are:

ID	Format	ID	Format

			<table><tr><td>0</td><td>FORMAT_ID_CSF</td><td>21</td><td>FORMAT_ID_FTK</td></tr><tr><td>1</td><td>FORMAT_ID_AUTOCAD</td><td>22</td><td>FORMAT_ID_GTX</td></tr><tr><td>2</td><td>FORMAT_ID_DWF</td><td>23</td><td>FORMAT_ID_GIF</td></tr><tr><td>3</td><td>FORMAT_ID_MCS</td><td>24</td><td>FORMAT_ID_IGES</td></tr><tr><td>4</td><td>FORMAT_ID_HPGL</td><td>25</td><td>FORMAT_ID_IDW</td></tr><tr><td>5</td><td>FORMAT_ID_906</td><td>26</td><td>FORMAT_ID_JPEG</td></tr><tr><td>6</td><td>FORMAT_ID_BMP</td><td>27</td><td>FORMAT_ID_MOT</td></tr><tr><td>7</td><td>FORMAT_ID_CAL_ISP</td><td>28</td><td>FORMAT_ID_PDF</td></tr><tr><td>8</td><td>FORMAT_ID_CGM</td><td>29</td><td>FORMAT_ID_PNG</td></tr><tr><td>9</td><td>FORMAT_ID_CIT</td><td>30</td><td>FORMAT_ID_CADKEY</td></tr><tr><td>10</td><td>FORMAT_ID_ME10</td><td>31</td><td>FORMAT_ID_RLC</td></tr><tr><td>11</td><td>FORMAT_ID_INSO</td><td>32</td><td>FORMAT_ID_RLE</td></tr><tr><td>12</td><td>FORMAT_ID_PCX</td><td>33</td><td>FORMAT_ID_SOLIDWORKS</td></tr><tr><td>13</td><td>FORMAT_ID_DG</td><td>34</td><td>FORMAT_ID_TXT</td></tr><tr><td>14</td><td>FORMAT_ID_DGN</td><td>35</td><td>FORMAT_ID_EMF</td></tr><tr><td>15</td><td>FORMAT_ID_DIF</td><td>36</td><td>FORMAT_ID_ICD</td></tr><tr><td>17</td><td>FORMAT_ID_ORCAD</td><td>37</td><td>FORMAT_ID_XPS</td></tr><tr><td>18</td><td>FORMAT_ID_DX</td><td>38</td><td>FORMAT_ID_BI2DL</td></tr><tr><td>19</td><td>FORMAT_ID_SGI</td><td></td><td></td></tr><tr><td>20</td><td>FORMAT_ID_EDM</td><td></td><td></td></tr></table>
pDesc	Out	String	Returns format description for format specified by FormatID.

IsFunctionalityLicenseEnabled			
Syntax			
IsFunctionalityLicenseEnabled([in] long FunctionID, [out, retval] BOOL * Result)			
Description			
This method returns TRUE if the functionality specified by FunctionID is enabled by license, FALSE otherwise.			
Parameters	I/O	Type	Description

FunctionID	In	Long	See Functionality Identification section for valid values of FunctionID.
Result	Out	Boolean	Returns TRUE if the functionality specified by FunctionID is enabled by license, FALSE otherwise.

Publish/Export Section

Publish (Export) Properties and Methods

Methods

SaveViewAs			
Syntax			
`SaveViewAs( [in] BSTR fileName, [in] short Type)`			
Description			
This method causes the control to save the current view as an alternate format (example: JPG). The resulting alternate format will be saved to a file specified by `fileName`. .			
Parameters	I/O	Type	Description
`fileName`	In	String	File name to use to save the current view as an alternate extension.
`Type`	In	Short	Currently only JPG is supported, but in the future other formats may be added. The default compression quality for JPG is 75. The parameter `Type` is specified as follows: **Value** | **Type** 0 | JPG

SaveViewAsEx			
Syntax			
`SaveViewAsEx( [in] BSTR fileName, [in] short Type, [in] short Quality)`			
Description			
This method works the same as `SaveViewAs()` except it allows the addition of `Quality`, an integer value to control output quality.			
Parameters	I/O	Type	Description
`fileName`	In	String	File name to use to save the current view as an alternate extension.
`Type`	In	Short	Format to save current view. See `SaveViewAs()`
`Quality`	In	Short	Must be a value between 1 and 100. 1 is the lowest quality and

			smallest output, 100 is the highest quality and largest output.

SetExportParam			
Syntax			
`SetExportParam( [in] LONG lType, [in] BSTR Name, [in] BSTR Value)`			
Description			
This method sets the values of the option name specified by `Name` for the export type specified by `Type`. Note that with this method, Brava! supports transparent redactions by publishing as a Draft version (`bravadtx.SetExportParam(BX_PDF_EXPORT, "FinalizeRedactions", "false"`). Passing true for the final parameter results in opaque redaction entities, while passing false results in transparent redaction entities.			
Parameters	I/O	Type	Description
`Type`	In	Long	Valid values for the export type are: **Value** / **Export Type** 1 / PDF 2 / TIFF 4 / CSF
`Name`	In	String	Supported Option Names are: `Name` / `Value` WriteExtraPagesToText / true/false. (Not supported for CSF) ExtraPagesTextFileName / true/false. (Not supported for CSF) FinalizeRedactions / true/false
`Value`	In	String	Set value (see above) for option name and type.

GetExportParam			
Syntax			
`GetExportParam( [in] LONG Type, [in] BSTR Name, [out, retval] BSTR * Value)`			
Description			
This method returns the value of the named option `Name`, for the specified export `Type`.			
Parameters	I/O	Type	Description

Type	In	Long	Export type. See SetExportParam() for valid values.
Name	In	String	Option name. See SetExportParam() for valid values.
Value	Out	String	See SetExportParam() for valid values.

ExportCSFEx

Syntax

```
ExportCSFEx( [in] BSTR Filename, [in] BSTR MetaDataString, [in] BSTR Password,
[in] BOOL PasswordProtected, [in] BSTR RedactionPassword, [in] BOOL
RedactionPasswordProtected, [in] DWORD Permissions, [in] int LayerSetting,
[in] int ExpireDateChoice, [in] LONGLONG ExpirationDate, [in] int
DaysToExpiration, [in] BOOL UseCustomPageSize, [in] double CustomPageWidth,
[in] double CustomPageHeight)
```

Description

This method creates a CSF rendition of the currently viewed document or drawing. See Common Export Parameters for a description of the parameters common to all exported formats.

Parameters	I/O	Type	Description
Filename	In	String	Indicates file name of CSF rendition produced.
MetadataString	In	String	XML string that specifies export settings. See Common Export Parameters.
Password	In	String	CSF password required when viewing the exported file.
PasswordProtected	In	Boolean	If set to TRUE, the exported file is created using the specified Password. If set to FALSE, the exported CSF file has no password and is not password protected.
RedactionPassword	In	String	When viewing a CSF with redactions, RedactionPassword will be required in order to see the redacted data. (If RedactionPassword is not provided when viewing the CSF, the redactions are opaque and the redacted data cannot be accessed (viewed or copy-pasted)).
RedactionPasswordProtected	In	Boolean	If set to TRUE, the exported file is created using the specified RedactionPassword. If set to FALSE, any redactions in the exported CSF file are opaque and the redacted data cannot be accessed (viewed or copy-pasted)).
Permissions	In	DWORD	Specifies the Visual Rights permission to be applied to the CSF file. It is a bit-wise flag with the following bits:

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><th>Bit</th><th>Permission</th></tr>
<tr><td>0x0001</td><td>Print / Copy / Save</td></tr>
<tr><td>0x0002</td><td>Measurement</td></tr>
<tr><td>0x0004</td><td>Markup</td></tr>
</table>
</td></tr>
<tr><td>LayerSetting</td><td>In</td><td>Integer</td><td>Specifies how layer data is exported:

<table>
<tr><td>Value</td><td>Setting</td><td>Description</td></tr>
<tr><td>0</td><td>None</td><td>No layer data is exported</td></tr>
<tr><td>2</td><td>All</td><td>All layers are exported with current visibility settings</td></tr>
<tr><td>3</td><td>Visible</td><td>Only the visible layers are exported on corresponding layers.</td></tr>
</table>
</td></tr>
<tr><td>ExpireDateChoice</td><td>In</td><td>Integer</td><td>Specifies when the CSF file will expire:

<table>
<tr><th>Value</th><th>Setting</th><th>Description</th></tr>
<tr><td>0</td><td>Never</td><td>The CSF file will never expire</td></tr>
<tr><td>1</td><td>Absolute date</td><td>The CSF will expire on a given date, specified by ExpirationDate</td></tr>
<tr><td>2</td><td>Relative date</td><td>The CSF will expire a given number of days, specified by DaysToExpiration, from the date of CSF creation</td></tr>
</table>
</td></tr>
<tr><td>ExpirationDate</td><td>In</td><td>Long</td><td>Specifies the date as a time_t value (in UTC) when the CSF file will expire. This parameter requires ExpireDateChoice to be set to 1.</td></tr>
<tr><td>DaysToExpiration</td><td>In</td><td>Integer</td><td>Specifies the number of days from the date of CSF creation when the CSF file will expire. 0 and positive integers are allowed. This parameter requires ExpireDateChoice to be set to 2.</td></tr>
<tr><td>UseCustomPageSize</td><td>In</td><td>Boolean</td><td>See Common Export Parameters</td></tr>
<tr><td>CustomPageWidth</td><td>In</td><td>Double</td><td>See Common Export Parameters</td></tr>
<tr><td>CustomPageHeight</td><td>In</td><td>Double</td><td>See Common Export Parameters</td></tr>
</table>

ExportPDFEx

Syntax

```
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 RemoveHiddenText, [in]
int  ViewStateExports, [in] BOOL UseCustomPageSize, [in] double
CustomPageWidth, [in] double CustomPageHeight)
```

Description

This method creates a PDF rendition of the currently viewed document or drawing. See Common Export Parameters for a description of the parameters common to all exported formats. This method automatically uses FileIOIntegration (when enabled in the web browser) to export to PDF. This means that this method will always send the exported PDF to the integration DLL when it is loaded.

Parameters	I/O	Type	Description
`Filename`	In	String	Indicates file name of PDF rendition produced.
`MetadataString`	In	String	XML string that specifies export settings. See Common Export Parameters.
`OwnerPassword`	In	String	Specifies the password required for editing the PDF file.
`UserPassword`	In	String	Specifies the password required for viewing the PDF file.
`PasswordProtected`	In	Boolean	If set to TRUE, the PDF file is created using `OwnerPassword` and `UserPassword`.
`MrkAsPDFAnnotations`	In	Boolean	If set to TRUE all currently open markup entities, if any, will be inserted into the exported PDF as PDF annotations. If set to FALSE all currently open markup entities will be "burned-in" as normal PDF entities.
`OutputChangemarks`	In	Boolean	See Common Export Parameters
`OutputRedactionInfo`	In	Boolean	See Common Export Parameters
`DoBlockAttributes`	In	Boolean	If set to TRUE any blocks with attributes in the current drawing or document are inserted into the exported PDF as hot spot block attributes.
`DoHyperlinks`	In	Boolean	If set to TRUE any hyperlink entities in the current drawing or document are inserted into the exported PDF as hyperlinks.
`DoBookmarks`	In	Boolean	If set to TRUE any bookmark entities in the current drawing or document are inserted into the exported PDF as PDF bookmarks.
`DoLayers`	In	Boolean	If set to FALSE, layer data is not exported in the PDF file. If set to TRUE, layer data, if any is exported according to the `LayeringMode` set in the `MetaDataString`

			parameter.
RemoveHiddenText	In	Boolean	If set to TRUE, hidden text, if any, is not exported to the PDF file. If set to FALSE, hidden text is exported as hidden text in the PDF file.
ViewStateExports	In	Boolean	See Common Export Parameters
UseCustomPageSize	In	Boolean	See Common Export Parameters
CustomPageWidth	In	Double	See Common Export Parameters
CustomPageHeight	In	Double	See Common Export Parameters

ExportTiffEx

Syntax

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

Description

This method creates a TIFF rendition of the currently viewed document or drawing. See Common Export Parameters for a description of the parameters common to all exported formats. This method automatically uses FileIOIntegration (when enabled in the web browser) to export to TIFF. This means that this method will always send the exported Tiff to the integration DLL when it is loaded.

Parameters	I/O	Type	Description
Filename	In	String	Indicates file name of TIFF rendition produced.
MetadataString	In	String	XML string that specifies export settings. See Common Export Parameters.
OutputChangemarks	In	Boolean	See Common Export Parameters
OutputRedactionInfo	In	Boolean	See Common Export Parameters
Dpi	In	Integer	Specifies the DPI (dots per inch) used for the TIFF data. DPI should be an integer between 60 and 1200
ViewStateExports	In	Boolean	See Common Export Parameters
UseCustomPageSize	In	Boolean	See Common Export Parameters
CustomPageWidth	In	Double	See Common Export Parameters
CustomPageHeight	In	Double	See Common Export Parameters

ExtractEmbeddedFile

Syntax

`ExtractEmbeddedFile( [in] BSTR OutputName, [in] int FileType, [in] int Page, [in] int Index, [in] BOOL CreateTemp, [out, retval] BSTR *ActualOutputName)`			
Description			
This method is used to extract embedded files from Brava-supported base files. Currently, this method is only used for extraction of PRC or U3D data from PDF files.			
Parameters	I/O	Type	Description
`OutputName`	In	String	Full path of file to which the embedded file will be written.
`FileType`	In	Integer	Integer value indicating the type of embedded file. Possible values are: Value: 0 — Type: PRC or U3D data from PDF file
`Page`	In	Integer	Zero based index of the page from which to extract the embedded file
`Index`	In	Integer	Zero based index to identify the embedded data within a given document page. This parameter is used if there is more than one embedded file per page. Otherwise it must be 0.
`CreateTemp`	In	Boolean	If `CreateTemp` is set to TRUE, the system will ignore the `OutputName` and extract the embedded file to a temporary file. See `ActualOutputName` for additional information.
`ActualOutputName`	Out/Retval		If the extraction is successful, this string will contain the full path of the extracted file. If `CreateTemp` is set to TRUE, `ActualOutputName` will be a temporary file. The system will delete the temporary file When the base document is closed. If `CreateTemp` is set to FALSE, `ActualOutputName` will be the full path sent in `OutputName`.

Common Export Parameters

Common Parameters
Description

<table>
<tr><td colspan="4">The following parameters are common to all export types (CSF, PDF, TIFF). See ExportCSFEx, ExportPDFEx, and ExportTiffEx.</td></tr>
<tr><td>Parameters</td><td>I/O</td><td>Type</td><td>Description</td></tr>
<tr><td>Filename</td><td>In</td><td>String</td><td>Indicates file name of the rendition produced.</td></tr>
<tr><td>MetaDataString</td><td>In</td><td>String</td><td>XML string that specifies export settings. See table* below.</td></tr>
<tr><td>OutputChangemarks</td><td>In</td><td>Boolean</td><td>Specifies if a page / pages are appended at the end of the exported document with a listing of all the Changemarks currently applied to the document, if any. If no Changemarks entities are present, this parameter is ignored. If set to TRUE, and Changemarks exist, the entities are numbered on the page they exist on and are listed on a separate page(s) at the end of the document.</td></tr>
<tr><td>OutputRedactionInfo</td><td>In</td><td>Boolean</td><td>Specifies if a page / pages are appended at the end of the published document with a listing of all the redaction reasons currently applied to the document, if any. If no redactions are present, this parameter is ignored. If set to TRUE, and redactions are being finalized, any redactions existing in the currently open drawing/document will be indexed by the redaction reason. If set to TRUE and redactions are being exported in draft mode, neither index nor redaction reason are output on the redaction. In both cases, a list at the end of the document will list the number of instances of each redaction reason. If set to FALSE, no list is appended at the end of the document</td></tr>
<tr><td>ViewStateExports</td><td>In</td><td>Boolean</td><td>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. Valid values:

<table><tr><th>Value</th><th>Viewstate</th><th>Description</th></tr><tr><td>0x0000</td><td>None</td><td>The file is exported in its default state. No changes to the appearance of the current drawing or document during the viewing session are applied.</td></tr><tr><td>0x0001</td><td>Rotations</td><td>All page rotations made during the viewing of the current drawing or document are reflected in the output file.</td></tr></table></td></tr>
<tr><td>UseCustomPageSize</td><td>In</td><td>Boolean</td><td>If set to FALSE, the current document/drawing dimensions are used for the output. If set to TRUE, CustomPageWidth and CustomPageHeight are used for the output document dimensions</td></tr>
</table>

CustomPageWidth	In	Double	If UseCustomPageSize is TRUE, this value specifies the width in inches of the output document
CustomPageHeight	In	Double	If UseCustomPageSize is TRUE, this value specifies the height in inches of the output document

***MetaDataString**: This string can be used to specify the following export settings:*

Tag	Description	Values
OutputPages Attribute: PageList	A string representing the list of pages to export	*Example:* "1-5,7". If not specified, or empty string, all pages are exported
VectorColorMode	Color setting to use for exported vector data (including text)	*One of:* **FullColor** (original colors in document) (default) **Grayscale** (shades of black and white) **OneColor** (Black and white)
RasterColorMode	Color setting to use for exported raster data	*One of:* FullColor (original colors in document) (default) Grayscale (shades of black and white) OneColor (Black and white)
MarkupColorMode	Color setting to use for exported markup data (including watermark and banners)	*One of:* **FullColor** (original colors in document) (default) **Grayscale** (shades of black and white) **OneColor** (Black and white)
FitInsideBanners	Specifies if the document data should be resized to fit within banner data, if banners are applied	true / false (default)
TIFF Only: CompressionType	The compression format to use for exported TIFF data	*One of:* **CCITTFAX4** **Packbits** **JPG** **LZW** (default)
PDF Only: LayeringMode	If layer data is being exported to the PDF output, this parameter specifies which layers are exported	*One of:* Visible (Only the layers that are currently visible in the view are exported) All (default -All layer data is exported with the current visibility setting (On/Off))
PDF Only: PDFCompatability	Specifies the compatibility setting for the published PDF file.	0 - None(default) 1 - PDF/A1-b 2 - PDF/E

PDF Only: `PDFFastWebView`	Specifies if the published PDF file will have Fast Web View enabled.	true / false (default)

Example: (Tiff Export)

```
<?xml version="1.0" encoding="utf-8"?>
<BXExportMetadata
          version="1.0.2"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns="http://schemas.MySite.com/igc-brava-file-export.xsd"
>
     <OutputPages PageList="1-5,7"/>
     <VectorColorMode>FullColor</VectorColorMode>
     <RasterColorMode>FullColor</RasterColorMode>
     <MarkupColorMode>Grayscale</MarkupColorMode>
     <FitInsideBanners>true</FitInsideBanners>
     <CompressionType>LZW</CompressionType>
</BXExportMetadata>
```

Compare Section

Compare Properties and Methods

Properties

CompareFilename	
Type	Example
String	`CompareFilename="C:\Users\Public\Documents\MyFile1.pdf"`
Description	
This property is used to get/set the name of the document or drawing to be compared against the currently open file.	

CompareViewMode	
Type	Example
Short (Integer)	`CompareViewMode=2`
Description	

This property is used to get/set the current compare mode.
The following are valid values:

Value	Compare Mode
1	Overlay
2	Overlay differences
3	Side-by-side
4	Side-by-side differences
5	Older document
6	Newer document
7	Additions
8	Deletions
9	Unchanged
10	Text compare

CompareViewMixtureLevel	
Type	Example
Short (integer)	`CompareViewMixtureLevel=50`
Description	
This property is used to get/set the percent value of the amount that different entities in the compare drawing are shown while in Overlay differences mode or Side by side differences 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	
Type	Default
Boolean	<Read-Only>
Description	
This property returns TRUE if the compare alignment tool has been used.	

CompareViewAttached	
Type	Default
Boolean	<Read-Only>
Description	
This property returns TRUE if a compare document has been attached.	

CompareTotalPages	
Type	Default
Integer	<Read-Only>
Description	
This property returns the number of pages in the compare document.	

CompareCurrentPageNumber	
Type	Default
Integer	<Read-Only>
Description	
This property returns the page number currently displayed by the compare document. The page number	

is 1 indexed (first page has number 1).

Methods

CloseCompareFile
Syntax
`CloseCompareFile()`
Description
This method closes any drawing or document that has been opened as a compare file.

ClearCompareAlignment
Syntax
`ClearCompareAlignment()`
Description
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.

ExportTextCompareReport			
Syntax			
`ExportTextCompareReport( [in] BSTR Filename, [in] BSTR MetaDataString);`			
Description			
This method creates a merged report of the text comparison of the open document compare document. The file is exported to the file system as a PDF.			
Parameters	I/O	Type	Description
`Filename`	In	String	Indicates file name of the PDF compare report produced. Can contain display strings for the documents being compared. If empty, the local filename is used.

`MetadataString`	In	String	XML representation of the document settings. Unlimited document descriptors may be added, separated by \n as follows:
			`<Document1>\\mymachine\TextCompareFiles\Before.txt`**`\n`**`SomeMoreDataAboutDoc1</Document1>`
			`<Document2>\\mymachine\TextCompareFiles\After.txt`**`\n`**`DocID</Document2>`
			
			If `MetaDataString` is empty, the local filename is used.

External Reference File Section

External Reference File Properties and Methods

Properties

EmbeddedFileWarningLevel	
Type	Example
Integer	`EmbeddedFileWarningLevel=2`
Description	
This property specifies what the viewer does upon being unable to load a file type embedded in the main document. See `SetXRefWarningLevel()` for valid values of the warning level.	

Methods

SetXRefPath			
Syntax			
`SetXRefPath( [in] XRefPathTypes PathType, [in] BSTR XRefPath)`			
Description			
This method modifies the paths the control searches to find various drawing external reference files.			
Parameters	I/O	Type	Description
`PathType`	In	Integer	Path type. `XRefPathTypes` are defined as follows: **PathType** — **Description** 0 — Paths searched to find external reference files for AutoCAD format files 1 — Paths searched to find AutoCAD font and shape files
`XRefPath`	In	String	Path the control searches to find Xref file. Each individual path is separated by a semicolon.

GetXRefPath

Syntax

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

Description

This method returns a listing of the paths the control searches in order to find drawing external references.

Parameters	I/O	Type	Description
PathType	In	Integer	See SetXRefPath() for a description of XRefPathTypes.
XRefPath	Out	String	Returns a listing of the paths the control searches in order to find drawing Xref files.

SetXRefWarningLevel

Syntax

```
SetXRefWarningLevel( [in] XRefPathTypes PathType, [in] int Level)
```

Description

This method specifies the control's behavior if it is unable to find a drawing's externally referenced file.

Parameters	I/O	Type	Description
PathType	In	Integer	See SetXRefPath() for a description of XRefPathTypes.
Level	In	Integer	Valid values for Level are: **Level** / **Description** 0 – Display a warning message on failure to find an external reference 1 – Do not display a warning message on failure to find an external reference 2 – Display a warning message on failure to find an external reference and abort the file load

GetXRefWarningLevel

Syntax

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

Description

This method returns an integer to indicate the control's behavior if it is unable to find a drawing's externally referenced file.

Parameters	I/O	Type	Description
PathType	In	Integer	See SetXRefPath() for XRefPathTypes description.
Level	Out	Integer	See SetXRefWarningLevel() for possible values of Level.

Printing Section

Printing Properties and Methods

Properties

PaperSpaceWidth and PaperSpaceHeight	
Type	Example

Double (real)	`viewObject.PaperSpaceWidth=22`
	`viewObject.PaperSpaceHeight=36`
	Sets (default) paper size to D (22x36)

Description	
`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.	

Methods

Print
Syntax
`Print()`
Description
This method causes the control to display the *Print/Print Preview* modal dialogs.

ShowPrintCtrl
Syntax
`ShowPrintCtrl()`
Description
This method causes the control to display the *Print* modal dialog. This method has the same effect as the `Print()` method.

PrintCurrentView

Syntax

```
PrintCurrentView()
```

Description

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

Syntax

```
PrintPageRange( [in] int Start, [in] int End)
```

Description

This method causes the control to print from `Start` to `End` pages of a document or drawing. Both are inexed 1. The print output will be directed to the currently selected printer or to the default printer if no printer has been selected.

Parameters	I/O	Type	Description
`Start`	In	Integer	Indicates the page number to start printing.
`End`	In	Integer	Indicates the page number to end printing.

PrintDesignatedPages

Syntax

```
PrintDesignatedPages( [in] BSTR PageList)
```

Description

This method causes the control to print the specified list of pages. The print output will be directed to the currently selected printer or to the default printer if no printer has been selected.

Parameters	I/O	Type	Description
`PageList`	In	String	A string representing the list of pages to print, and can include page range. Example: 1,2-5,9

UseCurrentWindowsDefaultPrinter

Syntax

```
UseCurrentWindowsDefaultPrinter()
```

Description

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

Advanced Printing Methods: Printer Specific

BXEnumeratePrinters			
Syntax			
`BXEnumeratePrinters( [out, retval] long * NumPrinters)`			
Description			
The method creates a list of printers accessible on the current Windows system. `BXEnumeratePrinters()` must be called prior to using any other printer specific API methods.			
Parameters	I/O	Type	Description
`NumPrinters`	Out	Long	Returns a list of accessible printers.

BXGetPrinterIDs			
Syntax			
`BXGetPrinterIDs( [out, retval] BSTR * IDs)`			
Description			
This method 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.			
Parameters	I/O	Type	Description
`IDs`	Out	String	The returned list of unique string printer IDs is delimited using the characters "<|>", ie: [printer id 1]<|>[printer id 2]<|>[printer id 3]

BXGetPrinterNames			
Syntax			
`BXGetPrinterNames( [out, retval] BSTR * Names)`			
Description			
This method 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()`.			
Parameters	I/O	Type	Description

Names	Out	String	The returned list of unique printer names is delimited using the characters "<|>", ie: [printer name 1]<|>[printer name 2]<|>[printer name 3]

BXGetPrinterName			
Syntax			
BXGetPrinterName([in] BSTR PrinterID, [out, retval] BSTR * Name)			
Description			
This method retrieves the printer name of the printer identified by PrinterID.			
Parameters	I/O	Type	Description
PrinterID	In	String	Unique printer ID from the list returned from method BXGetPrinterIDs().
Name	Out	String	The returned printer name identified by PrinterID.

BXGetDefaultPrinterName			
Syntax			
BXGetDefaultPrinterName([out, retval] BSTR * Name)			
Description			
This method retrieves the printer name of the printer currently set as the Windows default printer.			
Parameters	I/O	Type	Description
Name	Out	String	The returned default printer name.

BXGetPaperSizesEnum			
Syntax			
BXGetPaperSizesEnum([in] BSTR PrinterID, [out, retval] BSTR * SizesEnum)			
Description			
This method retrieves a list of values for each paper size available on the printer identified by PrinterID.			
Parameters	I/O	Type	Description
PrinterID	In	String	Unique printer ID from the list returned from method BXGetPrinterIDs().
SizesEnum	Out	String	Each element in this returned list is a string that corresponds to an integer value that uniquely identifies an available paper size. See BXSetPaperSize() for additional details. The list is delimited using the characters "<|>", ie: [paper size 1]<|>[paper size 2]<|>[paper size 3].

BXGetPaperSizesName			
Syntax			
`BXGetPaperSizesName( [in] BSTR PrinterID, [out, retval] BSTR * SizesName)`			
Description			
This method retrieves a list of names (e.g. "Letter, 8.5x11") for each paper size available on the printer identified by `PrinterID`.			
Parameters	I/O	Type	Description
`PrinterID`	In	String	Unique printer ID from the list returned from method `BXGetPrinterIDs()`.
`SizesName`	Out	String	Each element in this returned list is an available paper size name. 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 "<|>", ie: [paper size 1]<|>[paper size 2]<|>[paper size 3].

BXGetPaperSizesValue			
Syntax			
`BXGetPaperSizesValue( [in] BSTR PrinterID, [out, retval] BSTR * SizesValue)`			
Description			
This method retrieves a list of strings that indicate the physical size of each paper size available on the printer identified by `PrinterID`.			
Parameters	I/O	Type	Description
`PrinterID`	In	String	Unique printer ID from the list returned from method `BXGetPrinterIDs()`.
`SizesValue`	Out	String	Each element in this returned list will be a string of 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 "<|>", ie: [paper size value 1]<|>[paper size value 2]<|>[paper size value 3].

BXSetPaperSource
Syntax
`BXSetPaperSource( [in] BSTR PrinterID, [in] long PaperSource)`
Description

This method tells the printer identified by `PrinterID` to output using the paper from the source indicated by `PaperSource`. `BXSetPrinter()` must be called before printing for the setting to take effect.

Parameters	I/O	Type	Description
`PrinterID`	In	String	Unique printer ID from the list returned from method `BXGetPrinterIDs()`.
`PaperSource`	In	Long	`PaperSource` must be one of the values retrieved by a call to `BXGetPaperSourcesEnum()`.

BXGetPaperSource

Syntax

```
BXGetPaperSource( [in] BSTR PrinterID, [out, retval] long * PaperSource)
```

Description

This method gets the current paper source of the printer identified by `PrinterID`.

Parameters	I/O	Type	Description
`PrinterID`	In	String	Unique printer ID from the list returned from method `BXGetPrinterIDs()`.
`PaperSource`	Out	Long	`PaperSource` must be one of the values retrieved by a call to `BXGetPaperSourcesEnum()`.

BXGetPaperSourcesEnum

Syntax

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

Description

This method retrieves a list of values for each paper source available on the printer identified by `PrinterID`.

Parameters	I/O	Type	Description
`PrinterID`	In	String	Unique printer ID from the list returned from method `BXGetPrinterIDs()`.
`SourcesEnum`	Out	String	Each element in this returned list is a string that corresponds to an integer value that uniquely identifies an available paper source. See `BXSetPaperSource()` for additional details. The list is delimited using the characters "<|>", ie: [paper source 1]<|>[paper source 2]<|>[paper source 3].

BXGetPaperSourcesName

Syntax

BXGetPaperSourcesName([in] BSTR PrinterID, [out, retval] BSTR * SourcesName)			
Description			
This method retrieves a list of names (e.g. "Upper Paper Tray") for each paper source available on the printer identified by PrinterID.			
Parameters	I/O	Type	Description
PrinterID	In	String	Unique printer ID from the list returned from method BXGetPrinterIDs().
SourcesName	Out	String	Each element in this returned list is a name that corresponds to an available paper source. Note that the returned list should contain exactly one paper source name for each paper source enum retrieved using BXGetPaperSourcesEnum().The list is delimited using the characters "<|>", ie: [paper source name 1]<|>[paper source name 2].

BXSetDoRasterBanding			
Syntax			
BXSetDoRasterBanding([in] BSTR PrinterID, [in] BOOL DoRasterBanding)			
Description			
This method sets raster banding on or off if the printer identified by PrinterID has raster banding enabled.			
Parameters	I/O	Type	Description
PrinterID	In	String	Unique printer ID from the list returned from method BXGetPrinterIDs().
DoRasterBanding	In	Boolean	If raster banding is enabled, determines if raster banding is turned on or off.

BXGetDoRasterBanding			
Syntax			
BXGetDoRasterBanding([in] BSTR PrinterID, [out,retval] BOOL *DoRasterBanding)			
Description			
This method sets DoRasterBanding to TRUE if the printer identified by PrinterID has raster banding enabled .			
Parameters	I/O	Type	Description
PrinterID	In	String	Unique printer ID from the list returned from method BXGetPrinterIDs().

DoRasterBanding	Out	Boolean	Returns TRUE if raster banding is enabled, otherwise FALSE.

BXSetPaperSize

Syntax

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

Description

This method sets the printer identified by PrinterID to output using the paper size indicated by PaperSize. BXSetPrinter() must be called before printing for the setting to take effect.

Parameters	I/O	Type	Description
PrinterID	In	String	Unique printer ID from the list returned from method BXGetPrinterIDs().
PaperSize	In	Long	This must be one of the values retrieved by a call to BXGetPaperSizesEnum().

BXGetPaperSize

Syntax

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

Description

This method gets the current paper size of the printer identified by PrinterID.

Parameters	I/O	Type	Description
PrinterID	In	String	Unique printer ID from the list returned from method BXGetPrinterIDs().
PaperSize	Out	Long	This will be one of the values retrieved by a call to BXGetPaperSizesEnum()

BXSetOrientation

Syntax

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

Description

This method sets the printer identified by PrinterID in landscape or portrait mode using one of the Orientation values.

Parameters	I/O	Type	Description
PrinterID	In	String	Unique printer ID from the list returned from method BXGetPrinterIDs().

Orientation	In	Long	Determines whether to use portrait or landscape mode: **Value** / **Orientation** 1 / Portrait 2 / Landscape

BXGetOrientation			
Syntax			
BXGetOrientation([in] BSTR PrinterID, [out, retval] long * Orientation)			
Description			
This method gets the current Orientation of the printer identified by PrinterID.			
Parameters	I/O	Type	Description
PrinterID	In	String	Unique printer ID from the list returned from method BXGetPrinterIDs().
Orientation	Out	Long	See BXSetOrientation() for description of returned value Orientation.

BXSetPrinter			
Syntax			
BXSetPrinter([in] BSTR PrinterID)			
Description			
This method causes the Brava! control to use the printer identified by PrinterID for all subsequent printing operations. Note that the printer used can be changed by an end-user interacting with the standard Brava! Print dialog.			
Parameters	I/O	Type	Description
PrinterID	In	String	Unique printer ID from the list returned from method BXGetPrinterIDs().

BXSetPrintPageRange			
Syntax			
BXSetPrintPageRange([in] long PageRangeType, [in] string PageRangeData)			
Description			
This method sets the values in the page range controls of the Brava! print dialog.			
Parameters	I/O	Type	Description

PageRangeData	In	Long	Not currently used.
PageRangeType	In	Long	Integer value to indicate the page range value. possible values are as follows. Note that Designated Pages cannot currently be set via this method. **Value** / **Type** 0 – Print all pages 1 – Print current page 3 – Print current view

BXGetPrintPageRange			
Syntax			
BXGetPrintPageRange([in] IBxStringAnswer *PageRangeData,[out,retval] long *PageRangeType)			
Description			
This method gets the current values for the page range controls of the Brava! print dialog.			
Parameters	I/O	Type	Description
PageRangeData	In	ibxStringAnswer	Not currently used.
PageRangeType	In	Long	Integer value to indicate the page range value. possible values are: **Value** / **Type** 0 – Print all pages 1 – Print current page 2 – Print designated pages 3 – Print current view

Advanced Printing Methods: Document Specific

BXSetPrintScaleType
Syntax

`BXSetPrintScaleType( [in] long PrintScaleType)`			
Description			
This method causes the currently open document to be print-scaled in one of the ways identified by `PrintScaleType`.			
Parameters	I/O	Type	Description
`PrintScaleType`	In	Long	Valid values for print scale type are:

Value	PrintScaleType	Description
0	Fit	Make the document fit the extents of the paper
1	Inside banners	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
2	Document	Use the document's native scale
3	Scale	Use the scale value set by `BXSetPrinterScaleFactor`
4	Calibrated Scale	Use the scale value set by calibration

BXGetPrintScaleType			
Syntax			
`BXGetPrintScaleType( [out, retval] long * PrintScaleType)`			
Description			
This method returns the print scale type used by the currently open document.			
Parameters	I/O	Type	Description
`PrintScaleType`	Out	Long	See `BXSetPrintScaleType()` for valid print scale type values.

BXSetPrintUnitsType			
Syntax			
`BXSetPrintUnitsType( [in] long PrintUnitsType)`			
Description			
This method sets the units to use for printing.			
Parameters	I/O	Type	Description

PrintUnitsType	In	Long	Valid values for print units type are: **Value** / **Units** 0 / English 1 / Metric

BXGetPrintUnitsType			
Syntax			
BXGetPrintUnitsType([out, retval] long * PrintUnitsType)			
Description			
This method gets the current units used for printing.			
Parameters	I/O	Type	Description
PrintUnitsType	Out	Long	Returns current unit type set by BXSetPrintUnitsType().

BXSetPrintScaleFactor			
Syntax			
BXSetPrintScaleFactor([in] double PrintScaleFactor)			
Description			
This method sets the custom scale factor applied to each page printed in the currently open document. Note that PrintScaleFactoris only applicable if the PrintScaleType is set to 2 (Document) via BXSetPrintScaleType().			
Parameters	I/O	Type	Description
PrintScaleFactor	In	Double	Sets the custom scale factor to be used for printing.

BXGetPrintScaleFactor			
Syntax			
BXGetPrintScaleFactor([out, retval] double * PrintScaleFactor)			
Description			
This method returns the custom scale factor applied to each page printed in the currently open document. Note that PrintScaleFactor is only applicable if the PrintScaleType is set to 2 (Document) via BXSetPrintScaleType().			
Parameters	I/O	Type	Description
PrintScaleFactor	Out	Double	Returns the current custom scale factor for printing.

BXSetPrintColorMode

Syntax

```
BXSetPrintColorMode ( [in] int ColorMode)
```

Description

This method causes the currently open document to be printed using the color mode specified by `ColorMode`.

Parameters	I/O	Type	Description
`ColorMode`	In	Integer	Valid values for `ColorMode` are:

Value	PrintScaleType	Description
0	Vectors black	All vector lines will be printed black
1	Grayscale	Document will be printed in shades of gray
2	Document	Use the document's native colors

BXGetPrintColorMode

Syntax

```
BXGetPrintColorMode ( [out, retval] int * ColorMode)
```

Description

This method returns color mode used by the currently open document.

Parameters	I/O	Type	Description
`ColorMode`	Out	Integer	See `BXSetPrintColorMode()` for valid color mode values.

BXSetOrientToBestFit

Syntax

```
BXSetOrientToBestFit ( [in] BOOL OrientToBestFit)
```

Description

This method causes the currently open document to be printed in an orientation that best fits the shape of the current printer’s paper.

Parameters	I/O	Type	Description
`OrientToBestFit`	In	Boolean	If set to TRUE (1) the currently open document to be printed in an orientation that best fits the shape of the current printer’s

			paper.

BXGetOrientToBestFit			
Syntax			
`BXGetOrientToBestFit ( [out, retval] BOOL * OrientToBestFit)`			
Description			
This method returns the orient to best fit flag used by the currently open document.			
Parameters	I/O	Type	Description
`OrientToBestFit`	In	Boolean	Returns the current value for the orient to best fit flag.

BXGetPrintScaleFactor			
Syntax			
`BXGetPrintScaleFactor( [out, retval] double * PrintScaleFactor)`			
Description			
This method returns the custom scale factor applied to each page printed in the currently open document. Note that `PrintScaleFactor` is only applicable if the `PrintScaleType` is set to 2 (Document) via `BXSetPrintScaleType()`.			
Parameters	I/O	Type	Description
`PrintScaleFactor`	In	Double	Returns the current custom scale factor for printing.

BXSetPrintScaleRatio			
Syntax			
`BXSetPrintScaleRatioValue( [in] long PrintUnitsType, [in] long RatioValue)`			
Description			
This method sets the scaling to use when printing in different units.			
Parameters	I/O	Type	Description
`PrintUnitsType`	In	Long	Valid values for print units type are: **Value** / **Units** 0 / English 1 / Metric
`RatioValue`	In	Long	If `PrintUnitsType` is set to 0 (English), `RatioValue` may be

			one of the following values: **Value** / **Ratio** 0 / 1 1 / 1/24 2 / 1/48 3 / 1/96 4 / 1/192 If `PrintUnitsType` is set to 1 (Metric), `RatioValue` may be one of the following values: **Value** / **Ratio** 5 / 1/10 6 / 1/50 7 / 1/100

BXGetPrintScaleRatioValue			
Syntax			
`BXGetPrintScaleRatioValue( [in] long PrintUnitsType, [out, retval] long * RatioValue)`			
Description			
This method retrieves the current print scale for the units type specified by `PrintUnitsType`.			
Parameters	I/O	Type	Description
`PrintUnitsType`	In	Long	Sets the unit type for which the scale ratio value is to be returned.
`RatioValue`	Out	Long	Returns the current ratio value set by `BXSetPrintScaleRatioValue()`

Blocks Section

Blocks Methods

GetNumberOfBlocks and GetNumberOfNVBlocks			
Syntax			

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

Description			
This method returns the total number of blocks (or Name/Value blocks) found on the page specified by `PageNum`.			
Parameters	I/O	Type	Description
`PageNum`	In	Long	The page number for which the number of blocks will be returned.
`NumBlocks`	Out	Long	Total number of blocks returned.

GetBlockAttributes and GetNVBlockAttributes			
Syntax			
`GetBlockAttributes( [in] long PageNum, [in] long BlockIndex, [out, retval] BSTR * Attributes)` `GetNVBlockAttributes( [in] long PageNum, [in] long BlockIndex, [out, retval] BSTR * Attributes)`			
Description			
This method returns the attributes associated with the block (or Name/Value block) found at `BlockIndex` within the list of blocks found on the page specified by `PageNum`.			
Parameters	I/O	Type	Description
`PageNum`	In	Long	The page number for which the number of blocks will be returned
`BlockIndex`	In	Long	Indicates the specific block.
`Attributes`	Out	String	`Attributes` will be in the form of each individual attribute delimited by the character sequence: "<|>". Each individual attribute will contain a tag-value combination of the form: "[tag string]:[value string]". For example, the return value of `GetBlockAttributes()` for a block containing three attributes will be of the type:"[tagA]:[valueA]<|>[tagB]:[valueB]<|>[tagC]:[valueC]"

SetBlockColor and SetNVBlockColor

Syntax

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

Description

This method sets the rendering color of the block (or Name/Value block) found at `BlockIndex` within the list of blocks found on the page specified by `PageNum`.

Parameters	I/O	Type	Description
`PageNum`	In	Long	The page number for which the number of blocks will be returned.
`BlockIndex`	In	Long	Indicates the specific block.
`Color`	In	String	This parameter is interpreted as either a string representation of an RGB value or one of the strings described in `MarkupColor()`.
`ForceRedraw`	In	Boolean	If TRUE, the display will be updated immediately.
`Success`	Out	Boolean	If the block can be found and its color set, the return value will be set to TRUE. If the block cannot be found or its color can not be set, the return value will be set to FALSE.

ClearBlockColor and ClearNVBlockColor

Syntax

```
ClearBlockColor( [in] long PageNum, [in] long BlockIndex, [in] BOOL ForceRedraw, [out, retval] BOOL * Success)
ClearNVBlockColor( [in] long PageNum, [in] long BlockIndex, [in] BOOL ForceRedraw, [out, retval] BOOL * Success)
```

Description

This method clears a color set by `SetBlockColor()` for the block (or Name/Value block) found at `BlockIndex` within the list of blocks found on the page specified by `PageNum`.

Parameters	I/O	Type	Description
`PageNum`	In	Long	The page number for which the number of blocks will be returned.
`BlockIndex`	In	Long	Indicates the specific block.
`ForceRedraw`	In	Boolean	If TRUE, the display will be updated immediately.
`Success`	Out	Boolean	If the block can be found and its color reset, the return value `Success` will be set to TRUE. If the block cannot be found or its color cannot be reset, the return value `Success` will be set to FALSE.

ZoomToBlock and ZoomToNVBlock

Syntax

```
ZoomToBlock( [in] long PageNum, [in] long BlockIndex, [in] double
ScaleFactor, [out, retval] BOOL * Success)
ZoomToNVBlock( [in] long PageNum, [in] long BlockIndex, [in] double
ScaleFactor, [out, retval] BOOL * Success)
```

Description

This method causes the Brava! viewer to zoom to the block (or Name/Value block) found at `BlockIndex` within the list of blocks found on the page specified by `PageNum`.

Parameters	I/O	Type	Description
`PageNum`	In	Long	The page number for which the number of blocks will be returned.
`BlockIndex`	In	Long	Indicates the specific block.
`ScaleFactor`	In	Double	Specifies the zoom level and should be a value between 0.001 and 1000.
`Success`	Out	Boolean	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 cannot be found or the view state change is unsuccessful, `Success` will be set to FALSE.

DisplayBlockAttributes and DisplayNVBlockAttributes

Syntax

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

Description

This method causes the control to display a dialog box containing the information about the block (or Name/Value block) indicated by `BlockIndex` and found on the page specified by `PageNum`.

Parameters	I/O	Type	Description
`PageNum`	In	Long	The page number for which the number of blocks will be returned.
`BlockIndex`	In	Long	Indicates the specific block.
`Success`	Out	Boolean	`Success` will be set to TRUE if the block is successfully displayed, and FALSE otherwise.

Logging Section

Logging Methods

Note that while these methods are available via the interface, they are only writable to the local disk - not over the web to a web service seeking to capture a log.

BeginLogging			
Syntax			
`BeginLogging( [in] BSTR LogFileName, [in] BSTR LogStyleSheet)`			
Description			
This method tells the control to begin logging events. Many viewer events will be stored in the log.			
Parameters	I/O	Type	Description
`LogFileName`	In	String	When `EndLogging()` is called, the log will be written to `LogFileName` in XML format.
`LogStyleSheet`	In	String	If `LogStyleSheet` is not blank, a stylesheet directive will be added to the logfile using the stylesheet specified by `LogStyleSheet`.

EndLogging			
Syntax			
`EndLogging( [in] LONG WriteLog)`			
Description			
This method terminates logging.			
Parameters	I/O	Type	Description
`WriteLog`	In	Long	If `WriteLog` is 1, the log is written to the file specified when `BeginLogging()` was called.

BeginLogEvent
Syntax
`BeginLogEvent( [in] BSTR EventName, [in] BSTR Data)`
Description
This method adds a new XML element named `EventName` to the XML log.

Parameters	I/O	Type	Description
EventName	In	String	XML element name. Any log events that occur between the time BeginLogEvent() is called and the time EndLogEvent() is called will be added as child elements of the EventName element.
Data	In	String	Data is the element data.

EndEventLog
Syntax
EndEventLog()
Description
This method ends the most recent log event begun by a call to BeginLogEvent().

WriteAuditLogFile
Syntax
WriteAuditLogFile()
Description
This method causes the log to be flushed to disk.

SetAuditLogFileName			
Syntax			
SetAuditLogFileName([in] BSTR LogFileName)			
Description			
This method overwrites the log file name specified in a BeginLogging() call with LogFileName.			
Parameters	I/O	Type	Description
LogFileName	In	String	When EndLogging() is called, the log will be written to LogFileName in XML format.

StartSessionScript
Syntax
StartSessionScript()
Description
This method will begin recording any events that are logged, to a separate file. When StopSessionScript() is called, any events that can be executed through a script will be written to

the script for replaying later. If this method is called when logging is not enabled, the session script will not be written.

StopSessionScript

Syntax

```
StopSessionScript([in] BSTR ScriptFileName);
```

Description

This method ends recording events for a script and writes all scriptable actions to the script file specified by `ScriptFileName`.

Parameters	I/O	Type	Description
`ScriptFileName`	In	String	Name of the script file the recorded actions are written to.

Brava! Enterprise Specific Section

Brava! Integration Properties, Methods, and Events

Properties

RequestFileIOIntegration	
Type	Default
Boolean	FALSE
Description	
This property is used to get/set if the integration will perform I/O events.	

NumberIntegrationMarkups	
Type	Default
Integer	<Read-Only>
Description	
This property returns the total number of markups that are available through the integration DLL.	

NumberEditableIntegrationMarkups	
Type	Default
Integer	<Read-Only>
Description	
This property returns the total number of editable markups that are available through the integration DLL.	

NumberIntegrationMarkupStampTemplates	
Type	Default
Integer	<Read-Only>
Description	
This property returns the total number of markup stamps that are available through the integration DLL.	

IntegrationDescriptionOld	
Type	Default
String	<Read-Only>
Description	
This property returns a string representing an integration DLL's description. This property is for use with the standardized integration DLL.	

IntegrationVersionInfoOld	
Type	Default
String	<Read-Only>
Description	
This property returns a string representing an integration DLL's version. This property is for use with the standardized integration DLL.	

IntegrationDescription	
Type	Default
String	
Description	
This property is used to get/set a string representing an integration DLL's description. This property is for use with the standardized integration DLL.	

IntegrationVersionInfo	
Type	Default
String	
Description	
This property is used to get/set a string representing an integration DLL's version. This property is for use with the standardized integration DLL.	

IntegrationDLLName	
Type	Example
String	`IntegrationDLLName="SomeIntegration"`
Description	
This property is used to get/set the name of the integration DLL.	

DocID	
Type	Example
String	`DocID="http://Brava70Demo.MySite.com/IGC/samplefiles/File1.xdl"`
Description	
This property is used to get/set the file currently being viewed.	

XDLIsComplete	
Type	Default
Boolean	<Read-Only>
Description	
This property indicates if the currently viewed file has completed publishing on the Brava! server. If it returns FALSE, the file has not yet completed publishing on the server.	

CollaborationMode	
Type	Default
Boolean	FALSE
Description	
Setting this property to TRUE enables the integration to be informed about viewstate changes.	

Methods

InvokeCustomIntegrationMethod			
Syntax			
`InvokeCustomIntegrationMethod( [in] BSTR Param1, [in] BSTR Param2, [out, retval] BSTR * pResponse)`			
Description			
This method is for use with the standardized integration DLL and can be used to pass custom information to the integration DLL.			
Parameters	I/O	Type	Description
`Param1, Param2`	In	String	Any custom parameters used to pass custom information.
`pResponse`	Out	String	Any response given by the integration DLL will be returned in the

			`pResponse` string. See the Programmer's Guide for more details on using this method.

RequestIntegrationFileIO			
Syntax			
`RequestIntegrationFileIO( [in] FileIORequestTypes IOType)`			
Description			
This method is used to request that Brava! ActiveX Client perform file I/O through an integration DLL. This method and the `RequestFileIOEvents` property are mutually exclusive.			
Parameters	I/O	Type	Description
`IOType`	In	FileIORequestTypes	Type of file IO request (open, close, or save) to send through the integration DLL. Valid values for FileIORequestTypes include: 0 = Document open 1 = Markup open for review 2 = Markup open for edit 3 = Save editable markup as

Events

XDLIsComplete
Syntax
`XDLIsComplete()`
Description
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.

Miscellaneous Section

Miscellaneous Properties and Methods

Properties

HasKeyboardFocus	
Type	Default
Boolean	<Read-Only>
Description	
This property indicates whether the Brava! ActiveX control currently has keyboard focus.	

NotificationListener	
Type	Example
String	`NotificationListener="http://localhost/<notificationservlet>"`
Description	
Used by Brava! Enterprise Integrations, this property is used to get/set the URL that will receive notification of integration application events.	

Methods

Undo
Syntax
`Undo()`
Description
This method forces an undo of the last undo-able action, if there is one.

Redo
Syntax
`Redo()`

Description
This method forces a redo of the last redo-able action, if there is one.

ShowFocusedReview
Syntax
`ShowFocusedReview()`
Description
This method causes the Verify Entire Document dialog to be shown. The dialog can be accessed in the user interface from the Redact ->Verify ->Entire Document menu.

SetFocusedReviewOption			
Syntax			
`SetFocusedReviewOption( [in] LONG OptionID, [in] BSTR newVal)`			
Description			
This method sets the value of the focus rectangle specified by `OptionID`.			
Parameters	I/O	Type	Description
`OptionID`	In	Long	Currently the only valid value of `OptionID` is 1 (Focus rectangle Height).
`newVal`	In	String	Specifies the value (integer) of the height of the rectangle in pixels.

GetFocusedReviewOption			
Syntax			
`GetFocusedReviewOption( [in] LONG OptionID, [out, retval] BSTR * pVal)`			
Description			
This method gets the value of focus rectangle specified by `OptionID`. See `SetFocusedReviewOption()` for more details			
Parameters	I/O	Type	Description
`OptionID`	In	Long	Currently the only valid value of `OptionID` is 1 (Focus rectangle Height).
`pVal`	Out	String	Returned value (integer) of the height of the rectangle in pixels.

SetBravaXParam
Syntax

`SetBravaXParam( ([in] BSTR Name, [in] BSTR Value, [in] long UpdateAll)`

Description

Brava! Desktop supports multiple parameters that can be set as name-value pairs.
Some values that can be set using `SetBravaXParam()` can also be set via a property or method. In such cases, both ways of setting the `Value` are functionally equivalent. Please see the *BravaDesktop_BravaXParamsGuide.pdf* document for detailed information on the relevant parameters that can be referenced through the `SetBravaXParam()` method.
Example:

```
viewObject.SetBravaXParam("MarkupTextBackgroundType", " transparent", 0)
```

Sets the markup text background type to transparent and updates the display

```
viewObject.SetBravaXParam("WatermarkBannerFontName", "Arial", 0)
```

Sets the watermarks and banners font to Arial
Also updates the display to the current setting of all BravaX.parameters

Parameters	I/O	Type	Description
`Name`	In	String	The name of the parameter that will be set to `Value`.
`Value`	In	String	Set value of the BravaX parameter set by `Name` .
`UpdateAll`	In	Long	If `UpdateAll` is 1, all controls that have a BravaX parameter are updated to their current parameter setting. If `UpdateAll` is is 0, some UI settings are updated immediately. These settings are: markup color, redaction color, markup text background color, markup text background type, markup text default font.

Events Section

Event Methods

FileLoaded			
Syntax			
`FileLoaded( [in] BSTR Filename()`			
Description			
This event is fired when a document or drawing is successfully loaded..			
Parameters	I/O	Type	Description
`Filename`	In	String	Name of loaded document.

FileLoadFailure			
Syntax			
`FileLoadFailure( [in] BSTR Filename)`			
Description			
This event is fired when an attempt to open the document or drawing specified by `Filename`has failed.			
Parameters	I/O	Type	Description
`Filename`	In	String	Name of failed document.

FileClosed
Syntax
`FileClosed()`
Description
This event is fired when a document or drawing is closed.

FileIsPasswordProtected
Syntax
`FileIsPasswordProtected( [in] long PasswordType, [in] BSTR Filename, [in,`

out] long * PromptForPassword)			
Description			
This event is fired when the control attempts to open a password protected file. FileIsPasswordProtected() is fired before the control displays a dialog requesting the user to enter a password. CSF files can be password protected both for general opening/viewing and may also have password protection that allows any redaction entities to appear opaque.			
Parameters	I/O	Type	Description
PasswordType	In	Long	Indicates the type of password that will be requested. Valid values for PasswordType are: **Value** / **Password Type** 0 – CSF – general 1 – CSF – redaction 2 – Native format (PDF)
Filename	In	String	Filename is the full path and filename of the password protected document/drawing.
PromptForPassword	In/Out	Long	A listener to this event can suppress the password dialog by setting the PromptForPassword parameter to 0. Note that if PromptForPassword is set to 0, and PasswordType is set to 0 or 2, the document will not be opened.

ReportFileContent			
Syntax			
ReportFileContent([in] BSTR Filename, [in] long ContentType)			
Description			
This event is fired after every file load to indicate the type of data found within the file.			
Parameters	I/O	Type	Description
ContentType	In	Long	Indicates the type of data found. Valid values for ContentType are: Value / Data Type 0 – No data found 1 – File contains only 2D data 2 – File contains only 3D data 3 – File contains both 2d and 3D data
Filename	In	string	

LinkLaunchRequested

Syntax

```
LinkLaunchRequested( [in] long LinkType, [in] BSTR ID, [in] BSTR Metadata,
[in, out] long * CtrlHandle, [in, out] BSTR * Target)
```

Description

This event is fired just prior to the control launching a hyperlink.

Parameters	I/O	Type	Description
`LinkType`	In	Long	Indicates the type of link being launched. Valid values are: **Value** / **Link Type** 0 – Markup hyperlink 1 – Document hyperlink 2 – Document file
`ID`	In	String	The `ID` parameter is a string list of markup entity GUIDs delimited by "<|>". If the hyperlink is a property of a markup entity (LinkType = 2), `ID` will be the GUID identifier of that entity.
`Metadata`	In	String	Not currently used
`CtrlHandle`	In/Out	Long	A listener to this event can suppress the launch of the link by setting `CtrlHandle` to 0.
`Target`	In/Out	String	A listener to this event can also edit the target of the link launched by modifying the string value of the `Target` parameter.

LinkLaunchRequestedEx

Syntax

```
LinkLaunchRequestedEx( long LinkType, BSTR ID, BSTR Metadata, IBxBoolAnswer *
Suppress, IBxStringAnswer * Target)
```

Description

This event is fired just prior to the control launching a hyperlink.

Parameters	I/O	Type	Description
`LinkType`	In	Long	Indicates the type of link being launched. Valid values are: **Value** / **Link Type** 0 – Markup hyperlink

			1 Document hyperlink 2 Document file
ID	In	String	The ID parameter is a string list of markup entity GUIDs delimited by "<|>". If the hyperlink is a property of a markup entity (LinkType = 2), ID will be the GUID identifier of that entity.
Metadata	In	String	Not currently used
Suppress	Out	Boolean Answer	A listener to this event can suppress the launch of the link by setting Suppress.Answer to 0.
Target	Out	String Answer	A listener to this event can also edit the target of the link launched by modifying the string value of the Target.Answer parameter.

DisplayHelp

Syntax

```
DisplayHelp( [in] long contextID)
```

Description

This event is fired when the user has requested a display of help information, and the RequestHelpDisplay property is TRUE. The container is then responsible for displaying the help information that pertains to the given contextID. See Help Topic IDs Section for a list of acceptable values for contextID.

Parameters	I/O	Type	Description
contextID	In	Long	Indicates the help topic ID to display.

RequestLegalNotices

Syntax

```
RequestLegalNotices( IBxStringAnswer * pContents, IBxBoolAnswer * bHandled)
```

Description

This event is fired when a user clicks on the Legal Notices menu item of the Help icon on the information bar.

Parameters	I/O	Type	Description
pContents	Out	String Answer	The contents of the legal notices for the container are returned in pContents.Answer.
bHandled	Out	Boolean Answer	If the container is able to provide the text for the legal notices, bHandled.Answer is set to TRUE. If it is set to FALSE, the control will display the default legal notices.

CDLEvent

Syntax

```
CDLEvent( [in] long eventID)
```

Description

This event is fired with an `eventID` to indicate one of the occurrences below.

Parameters	I/O	Type	Description
`eventID`	In	Long	Indicates the ID of one of the following occurrences:

Value	Link Type	Description
2	Page changed	Indicates that the page of a drawing or document has changed
7	Something is copyable	Indicates that the Select mouse tool has selected text that can be copied to the system clipboard OR Indicates that a markup entity has been selected
8	Nothing is copyable	Indicates that a piece of text is no longer selected OR Indicates that a markup entity has become unselected
11	Started editing markup text	Indicates that a markup text entity has been created
12	Stopped editing markup text	Indicates that a markup text entity has been finalized
15	Mouse tool changed	Indicates that the mouse tool has changed. The MouseTool property can be used to determine the currently active mouse tool
28	View changed	Indicates that the view has changed
30	Region copied	Indicates that a region of the document has been copied to the Clipboard
100	Mirror Raster	Indicates that a raster image has been mirrored

CalibrationComplete

Syntax

```
CalibrationComplete()
```

Description
This event is fired when the mouse tool is *Calibrate* and the user has completed the measure tools calibration dialog.

LayersChanged
Syntax
`LayersChanged()`
Description
This event is fired when the visibility of drawing layers are modified through the control's Layer dialog.

ControlInitialized
Syntax
`ControlInitialized (long EventID)`
Description
This event is fired when the control's windows are created. See `CDLEvent` for description of `EventID`.

RequestSettingsDirectory			
Syntax			
`RequestSettingsDirectory (IBxStringAnswer * SettingsDir)`			
Description			
This event is fired immediately after the `ControlInitialized` event. It allows a listener to return an alternate path for the viewer to store settings persistence files. Note that this event is not fired from the Brava! Enterprise ActiveX Viewer because the settings persistence directory can be set via BravaXParams for Brava! Enterprise ActiveX Viewer.			
Parameters	I/O	Type	Description
`SettingsDir`	In	String Answer	The alternate directory to store persistence files will be set in `SettingsDir.Answer`.

ResolveISOBannerString
Syntax
`ResolveISOBannerString( [in] BSTR token, [in] BSTR parameter, [in, out] BSTR * result)`
Description
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`.

Parameters	I/O	Type	Description
`token`	In	String	The unresolved token.
`parameter`	In	String	Any accompanying parameter in the unresolved `token`.
`result`	In/Out	String	A listener to this event can provide the `token` replacement value in `result`.

ResolveISOBannerStringEx

Syntax

```
ResolveISOBannerStringEx(BSTR token, BSTR parameter, IBxStringAnswer * BXResult)
```

Description

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`.

Parameters	I/O	Type	Description
`token`	In	String	The unresolved token.
`parameter`	In	String	Any accompanying parameter in the unresolved `token`.
`BxResult`	Out	String Answer	A listener to this event can provide the `token` replacement value in `BxResult.Answer`.

ViewMouseEvent

Syntax

```
ViewMouseEvent( [in] long EventID, [in] long X, [in] long Y, [in] BSTR Metadata)
```

Description

This event is fired whenever any mouse activity happens while the cursor is within the document view area. This event replaces the `BDTXMouseDown`, `BDTXMouseUp`, and `BDTXMouseMove` events.

Parameters	I/O	Type	Description
`EventID`	In	Long	Indicates the type of mouse activity that has occurred. Valid values for `EventID` are: **Value** / **Event** 0 / Mouse move

			<table><tr><td>1</td><td>Mouse enter</td></tr><tr><td>2</td><td>Mouse leave</td></tr><tr><td>3</td><td>Left mouse button down</td></tr><tr><td>4</td><td>Left mouse button up</td></tr><tr><td>5</td><td>Right mouse button down</td></tr><tr><td>6</td><td>Right mouse button up</td></tr><tr><td>7</td><td>Left mouse button click</td></tr><tr><td>8</td><td>Right mouse button click</td></tr><tr><td>9</td><td>Left mouse button double click</td></tr></table>
`X, Y`	In	Long	X and Y indicate the mouse position at the time of the event. X and Y are given in screen/pixel coordinates relative to the top-left corner of the control document view area.
`Metadata`	In	String	Not currently used.

ExternalHotSpotMouseEvent		
Syntax		
`ExternalHotSpotMouseEvent( BSTR ID, long EventID, long X, long Y, BSTR Metadata)`		
Description		
This event is fired whenever any mouse activity happens while the cursor is within an external hot spot.		
Parameters	Type	Description
`ID`	String	String identifier for the hot spot.
`EventID`	Long	Integer indicating the type of mouse activity that has occurred. Valid values for `EventID` are: <table><tr><th>Value</th><th>Event</th></tr><tr><td>0</td><td>Mouse move</td></tr><tr><td>1</td><td>Mouse enter</td></tr><tr><td>2</td><td>Mouse leave</td></tr><tr><td>3</td><td>Left mouse button down</td></tr><tr><td>4</td><td>Left mouse button up</td></tr><tr><td>5</td><td>Right mouse button down</td></tr></table>

		<table><tr><td>6</td><td>Right mouse button up</td></tr><tr><td>7</td><td>Left mouse button click</td></tr><tr><td>8</td><td>Right mouse button click</td></tr><tr><td>9</td><td>Left mouse button double click</td></tr><tr><td>10</td><td>Context menu</td></tr></table>
`X, Y`	Long	X and Y integers indicate the mouse pointer coordinates in pixel (screen) space.
`Metadata`	String	Used to deliver additional information. Currently this is only used to return the id of a menu item selected.

BXKeyEvent

Syntax

`BXKeyEvent( [in] long EventID, [in] long KeyID, [in]` long CtrlDown,[in] long AltDown,[in] long ShiftDown,[in] IBxBoolAnswer * Handled)

Description

This event is fired when any part of the Brava! Viewer has keyboard focus and a key or key-sequence occurs.
An integration can prevent normal keyboard processing by setting `IBxBoolAnswer * Handled.Answer` to TRUE.

Parameters	I/O	Type	Description
`EventID`	In	Long	Indicates the type of keyboard event that has occurred. Possible values for `EventID` are (integer value shown in Hexidecimal): <table><tr><th>Value</th><th>Event</th></tr><tr><td>0x0100</td><td>Key down</td></tr><tr><td>0x0101</td><td>Key up</td></tr><tr><td>0x0102</td><td>Character</td></tr><tr><td>0x0103</td><td>Dead character</td></tr><tr><td>0x0104</td><td>System key down</td></tr><tr><td>0x0105</td><td>System key up</td></tr><tr><td>0x0106</td><td>System character</td></tr><tr><td>0x0107</td><td>System dead character</td></tr><tr><td>0x0109</td><td>Unicode character</td></tr></table>

KeyID	In	Long	Indicates the specific key that was used. This value will be either an ascii representation of the character or a virtual code.See https://msdn.microsoft.com/en-us/library/windows/desktop/dd375731(v=vs.85).aspx for a listing of Virtual key codes.
CtrlDown	In	Long	1 indicates that the Ctrl key is currently pressed. 0 indicates the Ctrl key is not pressed.
AltDown	In	Long	1 indicates that the Alt key is currently pressed. 0 indicates the Alt key is not pressed.
ShiftDown	In	Long	1 indicates that the Shift key is currently pressed. 0 indicates the Shift key is not pressed.
Handled	In/Out		An integration can prevent normal keyboard processing by setting IBxBoolAnswer * Handled.Answer to TRUE.

InternalBookmarkSelected

Syntax

```
InternalBookmarkSelected( [in] long PageNumber, [in] BSTR MetaDataString)
```

Description

This event is fired when an internal bookmark is selected on the page specified by PageNumber.

Parameters	I/O	Type	Description
PageNumber	In	Long	Page number index starting at 1.
MetaDataString	In	String	Not currently used.

NotifyButtonPressed

Syntax

```
NotifyButtonPressed( [in] long ButtonID)
```

Description

This event is fired when a button specified by ButtonID is pressed.

Parameters	I/O	Type	Description
ButtonID	In	Long	Valid values are: **ID** — **Button** 6 — Exit

CustomButtonInvoked

Syntax

```
CustomButtonInvoked( long ButtonID)
```

Description

This event is fired when a custom button specified by `ButtonID` is clicked. It is only fired for buttons added via the `ApplyCustomButtons()` method. See the *BravaExtensibleInterface_UserGuide.pdf* document for more details on custom buttons.

Parameters	I/O	Type	Description
`ButtonID`	In	Long	Indicates the button ID of the custom button.

CustomButtonDropInvoked

Syntax

```
CustomButtonDropInvoked( long ButtonID)
```

Description

This event is fired when a custom drop-down button specified by `ButtonID` is clicked. It is only fired for buttons added via the `ApplyCustomButtons()` method. See the *BravaExtensibleInterface_UserGuide.pdf* document for more details on custom buttons.

Parameters	I/O	Type	Description
`ButtonID`	In	Long	Indicates the button ID of the custom drop-down button.

BeforeCopy

Syntax

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

Description

This event is fired immediately before the control copies something (e.g. text, markup entity) to the Windows Clipboard.

Parameters	I/O	Type	Description
`Continue`	In/Out	Long	A listener to this event can cause the copy to not be performed if `Continue` is set to 0.

BeforePaste

Syntax

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

Description

This event is fired immediately before the control pastes something (e.g. text, markup entity) to the Windows Clipboard.

Parameters	I/O	Type	Description
`Continue`	In/Out	Long	A listener to this event can cause the paste to not be performed if `Continue` is set to 0.

AfterCopy

Syntax

```
AfterCopy()
```

Description

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

AfterPaste

Syntax

```
AfterPaste()
```

Description

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

PrintJobComplete

Syntax

```
PrintJobComplete( [in] BSTR JobName)
```

Description

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

Parameters	I/O	Type	Description
`JobName`	In	String	Name of the job sent to the printer.

MetadataEntitySelected

Syntax

```
MetadataEntitySelected( [in] BSTR Info, [in] BSTR EntityData, [in, out] long * DisplayInfo)
```

Description

This event is fired when a metadata entity such as a AutoCAD Block is clicked while the *Selection* or *Pan* tools are active.

Parameters	I/O	Type	Description
`Info`	In	String	A string containing information about the entity clicked.
`EntityData`	In	String	Not currently used.
`DisplayInfo`	In/Out	Long	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`.

MetadataEntitySelectedEx

Syntax

```
MetadataEntitySelectedEx( BSTR Info, BSTR EntityData, IBxBoolAnswer * Suppress)
```

Description

This event is fired when a metadata entity such as a AutoCAD Block is clicked while the *Selection* or *Pan* tools are active.

Parameters	I/O	Type	Description
`Info`	In	String	A string containing information about the entity clicked.
`EntityData`	In	String	Not currently used.
`Supress`	Out	Boolean Answer	If a listener to this event sets the value of `Suppress.Answer` to FALSE, the control will not display the message box containing the string `Info`.

MetadataEntityHovered

Syntax

```
MetadataEntityHovered([in] BSTR Info, [in] BSTR EntityData, [in, out] long * DisplayInfo)
```

Description

This event is fired when a metadata entity such as a AutoCAD Block is hovered over while the *Selection* or *Pan* tools are active.

Parameters	I/O	Type	Description
`Info`	In	String	A string containing information about the entity hovered.
`EntityData`	In	String	Not currently used.
`DisplayInfo`	In/Out	Long	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`.

RequestOpenFileGUI

Syntax

RequestOpenFileGUI()
Description
This event is fired when the user has attempted to open a document or drawing and the RequestFileIOEvents property is TRUE. The container is then responsible for prompting the user for a file name to open and then setting the Filename property to open the desired file.

RequestVersionForUrl			
Syntax			
RequestVersionForURL(IBxStringAnswer *VersionStr)			
Description			
This event is fired when determining the value of the version attribute in the form data of various URLs launched from the ActiveX control's splash screen and Help icon. If this event is not handled, the version of the COM module is used.			
Parameters	I/O	Type	Description
VersionStr	In/Out	String Answer	Indicates the version of the module or application to be used in URL form data. The string should be in the form: “[major].[minor].[servicepack].[build]” (e.g. “7.3.0.99”).

RequestInfoMessageDisplay			
Syntax			
RequestInfoMessageDisplay([in] long MessageID, [in] BSTR MessageText, [in, out] long * Display)			
Description			
This event is fired just prior to the control displaying an error or information message.			
Parameters	I/O	Type	Description
MessageID	In	Long	Identifies the message ID. See MessageID Section for a list of possible values.
MessageText	In	String	Contains the text that will appear in the message dialog
Display	In/Out	Long	If a container sets the value of Display to 0, the message will not be displayed.

RequestInfoMessageDisplayEx
Syntax
RequestInfoMessageDisplayEx(long MessageID, BSTR MessageText, IBxBoolAnswer * Suppress)

Description			
This event is fired just prior to the control displaying an error or information message.			
Parameters	I/O	Type	Description
MessageID	In	Long	Identifies the message ID. See MessageID Section for a list of possible values.
MessageText	In	String	Contains the text that will appear in the message dialog
Suppress	Out	Boolean Answer	If a container sets the value of Suppress.Answer to TRUE, the message will not be displayed.

RequestPageLoadForTextSearch			
Syntax			
RequestPageLoadForTextSearch ([in] long PageNumber, [in] IBxBoolAnswer *SuppressProgressDialog, [in] IBxBoolAnswer * CancelSearch)			
Description			
If PageNumber (indexed 1) has not already been loaded, this event is fired and the listener of the event has the opportunity to cancel further searching by returning CancelSearch = true.			
Parameters	I/O	Type	Description
PageNumber	In	Long	Page number index starting at 1.
CancelSearch	In	Boolean Answer	If a container sets the value of CancelSearch to true, the search will be canceled.
SuppressProgressDialog	In	Boolean Answer	Not currently used

RequestSaveViewAs			
Syntax			
RequestSaveViewAs([in] long type)			
Description			
This event is fired when the user has attempted to save the current view to an alternate format (e.g. JPG) and the RequestFileIOEvents property is TRUE.			
Parameters	I/O	Type	Description
type	In	Long	Indicates the desired alternate format. Currently only JPG is supported. The container is then responsible for prompting the user for an appropriate file name to save and then calling the SaveViewAsEx() method. See the SaveViewAsEx() method for details.

SaveViewAsSuccess			
Syntax			
`SaveViewAsSuccess( [in] BSTR Filename)`			
Description			
This event is fired when an attempt to save the current view to an alternate format (e.g. JPG) has succeeded.			
Parameters	I/O	Type	Description
`Filename`	In	String	Indicates the location and filename of the current view that has been successfully saved as JPG.

SaveViewAsFailure			
Syntax			
`SaveViewAsFailure( [in] BSTR Filename)`			
Description			
This event is fired when an attempt to save the current view to an alternate format (e.g. JPG) has failed.			
Parameters	I/O	Type	Description
`Filename`	In	String	Indicates the location and filename of the current view that has failed being saved as JPG.

RequestLocalFilenameForDownloadOriginal
Syntax
`RequestLocalFilenameForDownloadOriginal()`
Description
This event is fired when the user has attempted to copy a remote file to a local file, and the `RequestFileIOEvents` property is TRUE. The container is then responsible for calling the `DownloadOriginalToLocalFile()` method to provide a file name to take the contents of the remote file.

FileAppended
Syntax
`FileAppended( BSTR Filename)`
Description
This event is fired when a file is successfully appended to the open document.

Parameters	I/O	Type	Description
Filename	In	String	Indicates the location and filename of appended file.

FileAppendFailure			
Syntax			
FileAppendFailure(BSTR Filename)			
Description			
This event is fired when a file could not be appended to the open document.			
Parameters	I/O	Type	Description
Filename	In	String	Indicates the location and filename of failed appended file.

DownloadOriginalSuccess			
Syntax			
DownloadOriginalSuccess(BSTR Filename)			
Description			
This event is fired when an attempt to download the file specified by Filename from the server to the local system has succeeded.			
Parameters	I/O	Type	Description
Filename	In	String	Indicates the location and filename of the original file that is downloaded.

DownloadOriginalSuccess			
Syntax			
DownloadOriginalFailure(BSTR Filename)			
Description			
This event is fired when an attempt to download the file specified by Filename from the server to the local system has failed.			
Parameters	I/O	Type	Description
Filename	In	String	Indicates the location and filename of the original file that failed to download.

RequestMetadataHoverText
Syntax

`RequestMetadataHoverText( [in] IBxStringAnswer * pStringAnswer, [in] IBxBoolAnswer * suppress)`			
Description			
This event is fired whenever any metadata entity (block, markup, hyperlink) is hovered over.			
Parameters	I/O	Type	Description
`pStringAnswer`	Out	String Answer	The container may set `pStringAnswer.Answer` to the desired text (populating the pop-up message), if desired.
`suppress`	Out	Boolean Answer	The container can set `suppress.Answer` to true to prevent hover text from being displayed. In this case the value of `pStringAnswer` is ignored. If set to false, the value of `pStringAnswer` is used.

RequestTextCompareReportFilename
Syntax
`RequestTextCompareReportFilename()`
Description
This event is fired when the Text Compare Report button is pressed on the Compare toolbar, requesting a PDF rendition of the merged text compare report for the currently open and compare documents.

TextCompareReportResult			
Syntax			
`TextCompareReportResult( BSTR Filename, BOOL IsSuccess)`			
Description			
This event is fired when the control succeeded in creating a PDF rendition of the merged text compare report for the currently open and compare documents.			
	I/O	Type	Description
`Filename`	In	String	Specifies the name of the PDF file created.
`IsSuccess`	In	Boolean	Is set to TRUE if the PDF generated successfully, FALSE if the report failed to generate.

CompareFileLoadSuccess
Syntax
`CompareFileLoadSuccess(BSTR Filename)`
Description

This event is fired when the control succeeded in opening a file for compare.			
	I/O	Type	Description
`Filename`	In	String	Indicates the location and filename of the compare file that has succeeded loading.

CompareFileLoadFailure			
Syntax			
`CompareFileLoadFailure(BSTR Filename)`			
Description			
This event is fired when an attempt to load the file specified by `Filename` for Compare has failed.			
	I/O	Type	Description
`Filename`	In	String	Indicates the location and filename of the compare file that has failed loading.

CompareFileClosed			
Syntax			
`CompareFileClosed(BSTR Filename)`			
Description			
This event is fired when the compare file specified by `Filename` has been closed.			
	I/O	Type	Description
`Filename`	In	String	Indicates the location and filename of the compare file that has been closed.

FileIOIntegrationEvent			
Syntax			
`FileIOIntegrationEvent( [in] long EventID, [in] BSTR EventData)`			
Description			
This event is fired after the completion of any of the notification events listed in the table below. This method is only used for Brava! Enterprise clients that receive their file IO through the Integration DLL.			
Parameters	I/O	Type	Description
`EventID`	In	Long	Indicates the type of notification that has occurred. Valid values for `EventID` are:

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td>
<table>
<tr><th>Value</th><th>Notification Event</th></tr>
<tr><td>0</td><td>Document closed</td></tr>
<tr><td>1</td><td>Document loaded</td></tr>
<tr><td>2</td><td>Close markup tool</td></tr>
<tr><td>3</td><td>Markup open complete</td></tr>
<tr><td>4</td><td>Markup open for review complete</td></tr>
<tr><td>5</td><td>Markup save complete</td></tr>
<tr><td>6</td><td>Markup save as complete</td></tr>
<tr><td>7</td><td>Viewer closed</td></tr>
<tr><td>8</td><td>Document sent to printer</td></tr>
<tr><td>9</td><td>Markup stamp template tool closed</td></tr>
<tr><td>10</td><td>Markup stamp template open complete</td></tr>
<tr><td>11</td><td>Markup stamp template save complete</td></tr>
<tr><td>12</td><td>Markup stamp template save as complete</td></tr>
</table>
</td>
</tr>
<tr>
<td>EventData</td>
<td>In</td>
<td>String</td>
<td>
Indicates the event in the form of a string of the following format:

<code>"&DocID=[current doc id]&MarkupInfo=[info to identify markup]&User=[current user name]&[current value of NotificationListenerCustomParams BravaXParam]</code>
<table>
<tr><td>DocID/User</td><td>Valid strings for current doc ID and current user name</td></tr>
<tr><td>MarkupInfo</td><td>Blank for non-markup related events. When used, the EventData string consists of three substrings delimited by a '|': [MarkupName]|[integration specific data]|[Timestamp]</td></tr>
</table>
</td>
</tr>
</table>

Markup Event Methods

MarkupEditCreated
Syntax
`MarkupEditCreated()`
Description
This event is fired when the control has created a new editable markup.

MarkupEditLoaded			
Syntax			
`MarkupEditLoaded( [in] BSTR Filename)`			
Description			
This event is fired when an editable markup file specified by `Filename` has been successfully loaded.			
Parameters	I/O	Type	Description
`Filename`	In	String	The full path and filename of the markup file.

MarkupEditClosed			
Syntax			
`MarkupEditClosed [in] BSTR Filename)`			
Description			
This event is fired when an editable markup file specified by `Filename` has been closed.			
Parameters	I/O	Type	Description
`Filename`	In	String	The full path and filename of the markup file.

MarkupReviewLoaded			
Syntax			
`MarkupReviewLoaded( [in] BSTR Filename)`			
Description			
This event is fired when a markup file specified by `Filename` has been successfully loaded for review.			
Parameters	I/O	Type	Description
`Filename`	In	String	The full path and filename of the markup file.

MarkupReviewClosed			
Syntax			
`MarkupReviewClosed( [in] BSTR Filename)`			
Description			
This event is fired when a markup file specified by `Filename` has been closed.			
Parameters	I/O	Type	Description
`Filename`	In	String	The full path and filename of the markup file.

MarkupLoadFailure			
Syntax			
`MarkupLoadFailure( [in] BSTR Filename)`			
Description			
This event is fired when an attempt to open a markup specified by `Filename` has failed.			
Parameters	I/O	Type	Description
`Filename`	In	String	The full path and filename of the failed markup file.

RedactionReasonRequested			
Syntax			
`RedactionReasonRequested( [in] BSTR ID, [in] BSTR Metadata, [in, out] long * DisplayDlg, [in, out] BSTR * Reason)`			
Description			
This event is fired prior to editing the redaction reason of any redaction entity or group of redaction entities. The redaction entities that are having their reasons edited are identified by `ID`.			
Parameters	I/O	Type	Description
`ID`	In	String	A string list of redaction entity GUIDs delimited by "<|>". See the `MarkupEntityModified` event for more information about markup GUIDs. Note that each entry in the `ID` list must have a corresponding entry in the Reasons list and the order of both lists is synchronized.
`Metadata`	In	String	Not currently used
`DisplayDlg`	In/Out	Long	A listener to this event may suppress the dialog that the control displays to allow a user to enter redaction reasons by setting the value of `DisplayDlg` to 0. If `DisplayDlg` is set to 0, a listener may also edit the reasons by changing the string values found in `Reason`.

Reason	In/Out	String	A string of the current reasons for each entity delimited by "<|>".

RedactionReasonRequestedEx			
Syntax			
RedactionReasonRequestedEx(BSTR ID, BSTR Metadata, IBxStringAnswer* Reason)			
Description			
This event is fired prior to editing the redaction reason of any redaction entity or group of redaction entities. The redaction entities that are having their reasons edited are identified by ID.			
Parameters	I/O	Type	Description
ID	In	String	A string list of redaction entity GUIDs delimited by "<|>". See the MarkupEntityModified event for more information about markup GUIDs. Note that each entry in the ID list must have a corresponding entry in the Reasons list and the order of both lists is synchronized.
Metadata	In	String	Not currently used
Reason	Out	String Answer	Reason.Answer is set to the string of current reasons for each entity delimited by "<|>".

MarkupPropertyRequested			
Syntax			
MarkupPropertyRequested([in] long PropertyType, [in] BSTR ID, [in] BSTR Metadata, [in, out] long * CtrlHandle, [in, out] BSTR * Property)			
Description			
This event is fired prior to the editing of certain properties on a markup entity or group of entities. The type of property being edited is indicated by PropertyType. The markup entities that are having their properties edited are identified by ID.			
Parameters	I/O	Type	Description
PropertyType	In	Long	Indicates the type of property being edited. Currently value 0, property type Hyperlink is the only valid type.
ID	In	String	The ID parameter is a string list of markup entity GUIDs delimited by "<|>". See the MarkupEntityModified event for more information about markup GUIDs .Note that each entry in the ID list must have a corresponding entry in the Property list and the order of both lists is synchronized.
Metadata	In	String	Not currently used
CtrlHandle	In/Out	Long	A listener to this event may suppress the editing that the control normally allows a user to do, by setting the value of CtrlHandle to 0. If CtrlHandle is set to 0, a listener may also edit the

			properties by changing the string values found in the `Property` parameter.
`Property`	In/Out	String	A string version of the current property for each entity delimited by "<|>".

MarkupPropertyRequestedEx

Syntax

```
MarkupPropertyRequestedEx( long PropertyType, BSTR ID, BSTR Metadata,
IBxBoolAnswer * Handled, IBxStringAnswer * Property)
```

Description

This event is fired prior to the editing of certain properties on a markup entity or group of entities. The type of property being edited is indicated by `PropertyType`. The markup entities that are having their properties edited are identified by `ID`.

Parameters	I/O	Type	Description
`PropertyType`	In	Long	Indicates the type of property being edited. Currently value 0, property type Hyperlink is the only valid type.
`ID`	In	String	The `ID` parameter is a string list of markup entity GUIDs delimited by "<|>". See the `MarkupEntityModified` event for more information about markup GUIDs .Note that each entry in the ID list must have a corresponding entry in the `Property` list and the order of both lists is synchronized.
`Metadata`	In	String	Not currently used
`Handled`	Out	Boolean Answer	A listener to this event may suppress the editing that the control normally allows a user to do, by setting the value of `Handled.Answer` to FALSE. If set to FALSE, a listener may also edit the properties by changing the string values found in the `Property` parameter.
`Property`	In/Out	String	`Property.Answer` is set to the string version of the current property for each entity delimited by "<|>".

MarkupEntityModified

Syntax

```
MarkupEntityModified( BSTR MarkupID)
```

Description

This event is fired when any modification to a markup entity is finalized. This includes entity creation and deletion.

Parameters	I/O	Type	Description
`MarkupID`	In	Long	Used to uniquely identify the markup entity that was modified.

			MarkupID can then be used in subsequent calls to GetMarkupEntity() and SetMarkupEntity().

MarkupEntityMouseEvent			
Syntax			
MarkupEntityMouseEvent([in] BSTR ID, [in] long EventID, [in] long X, [in] long Y, [in] BSTR Metadata)			
Description			
This event is fired whenever any mouse activity happens while the cursor is above a markup entity.			
Parameters	I/O	Type	Description
ID	In	String	Uniquely identifies the markup entity. See the MarkupEntityModified event for more information about markup GUIDs.
EventID	In	Long	EventID indicates the type of mouse activity that has occurred. See the ViewMouseEvent event for valid values for EventID.
X, Y	In	Long	Indicate the mouse position at the time of the event. X and Y are given in screen/pixel coordinates relative to the top-left corner of the control document view area.
Metadata	In	String	Not currently used.

MarkupExtentsMismatched			
Syntax			
MarkupExtentsMismatched([in, out] long* action, [in, out] IBxBoolAnswer* bHandled)			
Description			
This event method allows an integration to tell the viewer control what action to take when a markup with mismatched page extents is being loaded.			
Parameters	I/O	Type	Description
action	Out	Long	Action to perform for opening markup files action = 0: cancel opening markup file(s) action = 1: continue opening markup file(s)
bHandled	Out	Boolean Answer	Returns a Boolean value indicating if the event was handled. If bHandled.Answer is set to FALSE, the value of action is ignored

MarkupEditSaved			
Syntax			
`MarkupEditSaved( BSTR Filename)`			
Description			
This event is fired when a currently open editable markup is saved to local disk.			
Parameters	I/O	Type	Description
`Filename`	In	String	Indicates the location and filename where the markup is saved.

MarkupStampTemplateSaved			
Syntax			
`MarkupStampTemplateSaved( BSTR Filename)`			
Description			
This event is fired when the stamp template file specified by `Filename` has been saved.			
Parameters	I/O	Type	Description
`Filename`	In	String	Indicates the location and filename where the stamp template is saved.

MarkupStampTemplateClosed			
Syntax			
`MarkupStampTemplateClosed( BSTR Filename)`			
Description			
This event is fired when the stamp template file specified by `Filename` has been closed.			
Parameters	I/O	Type	Description
`Filename`	In	String	Indicates the location and filename of the stamp template that has been closed.

MarkupStampEntityLoadSuccess			
Syntax			
`MarkupStampEntityLoadSuccess( BSTR Filename)`			
Description			
This event is fired when an attempt to load a stamp entity file specified by `Filename` into the stamp markup tool has succeeded.			
Parameters	I/O	Type	Description

Filename	In	String	Indicates the location and filename of the stamp entity that has been successfully loaded.

MarkupStampEntityLoadFailure			
Syntax			
MarkupStampEntityLoadFailure(BSTR Filename)			
Description			
This event is fired when an attempt to load a stamp entity file specified by Filename into the stamp markup tool has failed.			
Parameters	I/O	Type	Description
Filename	In	String	Indicates the location and filename of the stamp entity that has failed loading.

MarkupStampTemplateLoadSuccess			
Syntax			
MarkupStampTemplateLoadSuccess(BSTR Filename)			
Description			
This event is fired when an attempt to load the markup stamp template file specified by Filename has succeeded.			
Parameters	I/O	Type	Description
Filename	In	String	Indicates the location and filename of the stamp template that has succeeded loading.

MarkupStampTemplateLoadFailure			
Syntax			
MarkupStampTemplateLoadFailure(BSTR Filename)			
Description			
This event is fired when an attempt to load the markup stamp template file specified by Filename has failed.			
Parameters	I/O	Type	Description
Filename	In	String	Indicates the location and filename of the stamp template that has failed loading.

MarkupStampTemplateCreated

Syntax

```
MarkupStampTemplateCreated()
```

Description

This event is fired when a new stamp template file has been created.

ChangemarkExecuted

Syntax

```
ChangemarkExecuted( BSTR CMID)
```

Description

This event is fired when a Changemark entity is either selected from the Changemark review panel, clicked on with the *Selection* tool active, or executed via the `ExecuteChangemark()` method.

Parameters	I/O	Type	Description
`CMID`	In	String	A unique identifier for the Changemarks entity executed.

ResolveStampEntityTokenSet

Syntax

```
ResolveStampEntityTokenSet( [in] BSTR GUID, [in, out] BSTR * Tokens)
```

Description

This event is fired to notify an integrator to resolve or re-resolve %dbupdatestring tokens present in a markup stamp entity. All tokens within a stamp entity are resolved at stamp placement, and any %dbupdatestring or %prompt tokens are re-resolved when a stamp entity is double-clicked while the markup entity *Selection* tool is active. This method will only be fired during token resolution when there are %dbupdatestring tokens to be resolved.
Example:

```
<Tokens>
<Token>
<Name>dbupdatestring</Name>
<Parameter>tokenParam1</Parameter>
<Value>tokenValue1</Value>
</Token>
<Token>
<Name>dbupdatestring</Name>
<Parameter>tokenParam2</Parameter>
<Value>tokenValue2</Value>
</Token>
...
</Tokens>
```

Parameters	I/O	Type	Description

GUID	In	String	Identifies the stamp entity that is requesting token resolution.
Tokens	In/Out	String	An XML string containing a list of Token elements. The tokens may be resolved synchronously in languages that support input/output parameters by changing the value of the Tokens parameter or asynchronously using the UpdateStampEntityTokenSet() method. The Tokens XML will contain a reference to a XML schema against which the changed token XML may be validated. Invalid token XML should not be expected to be applied by the viewer.

RequestMarkupNew
Syntax
RequestMarkupNew()
Description
This event is fired when the user has attempted to create a new editable markup file, and the RequestFileIOEvents property is TRUE. The container is then responsible for calling the NewMarkup() method to complete the creation of the editable markup file.

RequestMarkupSave
Syntax
RequestMarkupSave()
Description
This event is fired when the user has attempted to save a previously saved editable markup file, and the RequestFileIOEvents property is TRUE. The container is then responsible for calling the SaveMarkup() method to save the desired editable markup file.

RequestCloseMarkupEdit
Syntax
RequestCloseMarkupEdit()
Description
This event is fired when the user has attempted to close an editable markup file, and the RequestFileIOEvents property is TRUE. The container is then responsible for calling the CloseMarkupEdit() method to complete the closure of the editable markup file.

RequestMarkupFileEdit
Syntax
RequestMarkupFileEdit()

Description
This event is fired when a user has attempted to open a markup file for edit, and the `RequestFileIOEvents` property is TRUE. The container is then responsible for prompting the user for a markup file name to open and then calling the `OpenMarkupEdit(BSTR fileName)` method to open the desired markup file for edit.

RequestMarkupFileReview
Syntax
`RequestMarkupFileReview()`
Description
This event is fired when a user has attempted to open a review markup file, and the `RequestFileIOEvents` property is TRUE. The container is then responsible for prompting the user for a review markup file name to open and then calling the `OpenMarkupReview(BSTR fileName)` method to open the desired review markup file.

RequestCloseMarkupReview
Syntax
`RequestCloseMarkupReview()`
Description
This event is fired when the user has attempted to close a review-only markup file, and the `RequestFileIOEvents` property is TRUE. The container is then responsible for calling the `CloseMarkupReview()` method to complete the closure of the review-only markup file.

RequestMarkupSaveAs
Syntax
`RequestMarkupSaveAs()`
Description
This event is fired when the user has attempted to save an editable markup file, and the `RequestFileIOEvents` property is TRUE. The container is then responsible for prompting the user for a file name to save the editable markup and then calling the `SaveAsMarkupEdit(BSTR fileName)` method to save the desired editable markup file.

RequestMarkupConsolidation
Syntax
`RequestMarkupConsolidation()`

Description
This event is fired when a user has attempted to consolidate all markup entities currently existing in all markups open for review into a single editable markup, and the `RequestFileIOEvents` property is TRUE. The container is then responsible for calling the `ConsolidateMarkups()` method to complete the consolidation.

RequestNewMarkupStampTemplate
Syntax
`RequestNewMarkupStampTemplate()`
Description
This event is fired when the user has attempted to create a new editable stamp template file, and the `RequestFileIOEvents` property is TRUE. The container is then responsible for calling the `NewMarkupStampTemplate()` method to complete the creation of the editable markup file.

RequestOpenMarkupStampTemplate
Syntax
`RequestOpenMarkupStampTemplate()`
Description
This event is fired when the user has attempted to open an editable stamp template file, and the `RequestFileIOEvents` property is TRUE. The container is then responsible for calling the `OpenMarkupStampTemplate()` method to complete the opening of the editable stamp template file.

RequestSaveMarkupStampTemplate
Syntax
`RequestSaveMarkupStampTemplate()`
Description
This event is fired when the user has attempted to save an editable stamp template file, and the `RequestFileIOEvents` property is TRUE. The container is then responsible for calling the `SaveMarkupStampTemplate()` method to complete the saving of the editable stamp template file.

RequestSaveAsMarkupStampTemplate
Syntax
`RequestSaveAsMarkupStampTemplate()`
Description
This event is fired when the user has attempted to save an editable stamp template file to a new file, and

the RequestFileIOEvents property is TRUE. The container is then responsible for calling the SaveAsMarkupStampTemplate() method to complete the saving of the editable stamp template file.

RequestSaveAsTakeoffTemplate
Syntax
RequestSaveAsTakeoffTemplate()
Description
This event is fired when the user has attempted to save a takeoff template file to a new file, and the RequestFileIOEvents property is TRUE. The container is then responsible for calling the SaveAsMarkupTakeoffTemplate() method to complete the saving of the takeoff template file.

RequestCloseMarkupStampTemplate
Syntax
RequestCloseMarkupStampTemplate()
Description
This event is fired when the user has attempted to close an editable stamp template file, and the RequestFileIOEvents property is TRUE. The container is then responsible for calling the CloseMarkupStampTemplate() method to complete the closing of the editable stamp template file.

RequestMarkupStampEntityFilename
Syntax
RequestMarkupStampEntityFilename()
Description
This event is fired when the user has attempted to instantiate the stamp markup tool with a new stamp file, and the RequestFileIOEvents property is TRUE. The container is then responsible for setting the MrkStampEntityFilename property to the full path of a valid stamp file to complete the instantiation of the stamp markup tool.

RequestRedactionScript
Syntax
RequestRedactionScript()
Description
This event is fired when the user has attempted to run a redaction script and the RequestFileIOEvents property is TRUE. The container is then responsible for calling the RedactFromScript() method to provide a redaction script.

RequestRedactionReasonDescriptions

Syntax

```
RequestRedactionReasonDescriptions( BSTR reasonCode, IBxStringAnswer *
pStringShortDescriptionCode, IBxStringAnswer * pStringLongDescription,
IBxBoolAnswer * bFound, IBxBoolAnswer * bHandled )
```

Description

This event is fired when the appending a redaction summary page during publishing of files to PDF and TIFF.

Parameters	I/O	Type	Description
`reasonCode`	In	String	Indicates the reason code for which the descriptions are to be retrieved.
`pStringShortDescriptionCode`	Out	String Answer	`pStringShortDescriptionCode.Answer` is set to the short description associated with the `reasonCode`
`pStringLongDescription`	Out	String Answer	`pStringLongDescriptionCode.Answer` is set to the long description associated with the `reasonCode`
`bFound`	Out	Boolean Answer	`bFound` is set to TRUE if the short and long descriptions were found for the given `reasonCode`, FALSE otherwise.
`bHandled`	Out	Boolean Answer	`bHandled` is set to TRUE if the container handled the event, FALSE otherwise

RequestChangemarkText

Syntax

```
RequestChangemarkText( [in] IBxStringAnswer * pStringAnswer, [in]
IBxBoolAnswer * pHandled)
```

Description

This event is fired when the integrator presses a button (copy from integration event handle, or copy from clipboard) in the Changemark Edit text box, allowing them to insert text into the Changemark entity or paste text from the clipboard. This event and the Changemark Edit Dialog button are only available if the viewer control is running under special licensing. Contact us for licensing details.

Parameters	I/O	Type	Description
`pStringAnswer`	In	String Answer	The container sets `pStringAnswer.Answer` to the desired text and sets `pHandled.Answer` to TRUE.
`pHandled`	In	Boolean Answer	If `pHandled.Answer` is set to FALSE, Brava! ignores the result passed in `pStringAnswer.Answer` and pastes text from the clipboa rd.

RequestRasterMarkupFilename
Syntax
`RequestRasterMarkupFilename()`
Description
This event is fired when a user has attempted to open a raster file for use with the raster markup tool, and the `RequestFileIOEvents` property is TRUE. The container is then responsible for prompting the user for a raster file name to open and then calling the `RasterMrkFilename` property to set the desired raster file for use with the raster markup tool.

RasterMarkupLoadSuccess			
Syntax			
`RasterMarkupLoadSuccess( BSTR Filename)`			
Description			
This event is fired when the control succeeded in opening a raster file for use with the raster markup tool.			
Parameters	I/O	Type	Description
`Filename`	In	String	The full path and filename of the raster file.

RasterMarkupLoadFailure			
Syntax			
`RasterMarkupLoadFailure( BSTR Filename)`			
Description			
This event is fired when the control is unable to open a raster file for use with the raster markup tool.			
Parameters	I/O	Type	Description
`Filename`	In	String	The full path and filename of the raster file.

RequestRasterFileForSignatureMarkup			
Syntax			
`RequestRasterFileForSignatureMarkup( BSTR UserID, BSTR DocVersion, long SignatureType, IBxStringAnswer * Filename, IBxBoolAnswer * Handled)`			
Description			
This event is fired when a user clicks the browse button on the “Set Signature Marks” dialog.			
Parameters	I/O	Type	Description

UserID	In	String	See Property UserName.
DocVersion	In	String	DocVersion is a parameter used by Brava! Enterprise Integrations.
SignatureType	In	Long	Valid values for SignatureType are: 0 = Sign 1 = Initial 4 = Seal
Filename	Out	String Answer	The raster filename requested in SignatureTypeis returned in Filename.Answer.
Handled	Out	Boolean Answer	Handled.Answer is set to TRUE if the request for a raster image is handled, FALSE otherwise.

RequestSignatureStrings

Syntax

```
RequestSignatureStrings( BSTR UserID, BSTR DocVersion, IBxStringAnswer *
SignatureFilename, IBxStringAnswer * InitialsFilename, IBxStringAnswer  *
SealImageFilename, IBxStringAnswer * NameString, IBxStringAnswer  *
TitleString, IBxBoolAnswer * Handled, IBxBoolAnswer * SuppressDlg)
```

Description

This event is fired anytime a Signature element requires data. It will be fired directly prior to the display of the “Set Signature Marks” dialog.

Parameters	I/O	Type	Description
UserID	In	String	See Property UserName.
DocVersion	In	String	DocVersion is a parameter used by Brava! Enterprise Integrations.
SignatureFilename	Out	String Answer	The name of the file containing the raster image of the signature is returned in SignatureFilename.Answer
InitialsFilename	Out	String Answer	The name of the file containing the raster image of the initials is returned in InitialsFilename.Answer
SealImageFilename	Out	String Answer	The name of the file containing the raster image of the seal is returned in SealImageFilename.Answer
NameString	Out	String Answer	The name associated with UserID is returned in NameString.Answer
TitleString	Out	String Answer	The title associated with UserID is returned in TitleString.Answer
Handled	Out	Boolean Answer	Handled.Answer is set to TRUE if the request for signature strings is handled, FALSE otherwise.
SupressDlg	Out	Boolean	A container can set SuppressDlg.Answer is set to TRUE

		Answer	to prevent the Set Signature Marks dialog from being displayed.

RequestSignatureStringsUpdate			
Syntax			
`ReportSignatureStringsUpdate ( BSTR UserID, BSTR DocVersion, IBxStringAnswer * SignatureFilename, IBxStringAnswer * InitialsFilename, IBxStringAnswer * SealImageFilename, IBxStringAnswer * NameString, IBxStringAnswer * TitleString)`			
Description			
This event is fired after the user clicks OK on the “Set Signature Marks” dialog. It notifies integrators when the signature item data may have been changed by the end-user.			
Parameters	I/O	Type	Description
`UserID`	In	String	See Property `UserName`.
`DocVersion`	In	String	`DocVersion` is a parameter used by Brava! Enterprise Integrations.
`SignatureFilename`	Out	String Answer	The name of the file containing the raster image of the signature is returned in `SignatureFilename.Answer`
`InitialsFilename`	Out	String Answer	The name of the file containing the raster image of the initials is returned in `InitialsFilename.Answer`
`SealImageFilename`	Out	String Answer	The name of the file containing the raster image of the seal is returned in `SealImageFilename.Answer`
`NameString`	Out	String Answer	The name associated with `UserID` is returned in `NameString.Answer`
`TitleString`	Out	String Answer	The title associated with `UserID` is returned in `TitleString.Answer`

AllSigned			
Syntax			
`AllSigned( BSTR UserID, long NumSignTemplates, long NumSigsSigned, BSTR EventData, IBxBoolAnswer * DisplayWarning)`			
Description			
This event is fired after the control attempts to sign all signature templates in the current document.			
Parameters	I/O	Type	Description
`UserID`	In	String	See Property `UserName`.
`NumSignTemplates`	In	Long	The number of signature templates that were available for signing.

NumSigsSigned	In	Long	The number of signature templates that were successfully signed.
EventData	In	String	Not currently used.
DisplayWarning	Out	Boolean Answer	The container can set DisplayWarning.Answer to TRUE to indicate that control should display a warning dialog if all signatures templates could not be signed successfully.

PreSignatureItemsSign			
Syntax			
PreSignatureItemSign(BSTR UserID, BSTR ItemID, BSTR EventData, long SigningAll, IBxBoolAnswer * AllowSign)			
Description			
This event is fired before one or more signature entities are signed.			
Parameters	I/O	Type	Description
UserID	In	String	See Property UserName.
ItemID	In	Long	The number of signature templates that were available for signing.
EventData	In	String	Not currently used.
SigningAll	In	Long	If all signatures entities are going to be signed SigningAll is set to 1. It is set to 0 otherwise
AllowSign	Out	Boolean Answer	A container can set AllowSign.Answer to FALSE to disallow signing.

BatchMrkProcess			
Syntax			
BatchMrkProcess(long Started, BSTR EventData)			
Description			
This event is fired when a batch process of markups, such as signing all signature templates, has started or ended.			
Parameters	I/O	Type	Description
Started	In	Long	Started is set to 1 to indicate that the batch process has started, 0 to indicate that the batch process has ended.
EventData	In	String	Currently not used.

Export Event Methods

RequestExportPDFFilename
Syntax
`RequestExportPDFFilename()`
Description
This event is fired when the user has attempted to create a PDF rendition of the currently open drawing or document - and the `RequestFileIOEvents` property is TRUE. The container is then responsible for calling the `ExportPDFEx()` method to provide a file name to be used for the PDF file.

RequestExportTiffFilename
Syntax
`RequestExportTiffFilename()`
Description
This event is fired when the user has attempted to create a TIFF rendition of the currently open drawing or document - and the `RequestFileIOEvents` property is TRUE. The container is then responsible for calling the `ExportTiffEx()` method to provide a file name to be used for the TIFF file.

RequestExportCSFExFilename
Syntax
`RequestExportCSFExFilename()`
Description
This event is fired when the user has attempted to create a CSF rendition of the currently open drawing or document - and the `RequestFileIOEvents` property is TRUE. The container is then responsible for calling the `ExportCSFEx()` method to provide a file name to be used for the CSF file.

PageOutput			
Syntax			
`PageOutput( long OutputType, long PageNumber, long TotalOutputPages, BSTR PageData)`			
Description			
This event is fired before a page is published or printed.			
Parameters	I/O	Type	Description

OutputType	In	Long	OutputType can have the following values: 0 = Publish to CSF 1 = Publish to PDF 2 = Publish to TIFF 100 = Print
PageNumber	In	Long	Page number (indexed 1) that is published or printed.
TotalOutputPages	In	Long	The total number of pages that are being published or printed.
PageData	In	String	Not currently used.

PageOutputComplete			
Syntax			
PageOutput(long OutputType, long TotalOutputPages, BSTR PageData)			
Description			
This event is fired after a page has completed publishing or printing.			
Parameters	I/O	Type	Description
OutputType	In	Long	OutputType can have the following values: 0 = Publish to CSF 1 = Publish to PDF 2 = Publish to TIFF 100 = Print
TotalOutputPages	In	Long	The total number of pages that are being published or printed.
PageData	In	String	Not currently used.

ExportCanceled			
Syntax			
ExportCanceled([in] long OutputType, [in] BSTR ExportData)			
Description			
This event is fired when the end-user clicks the Cancel button on any export status dialog.			
Parameters	I/O	Type	Description
OutputType	In	Long	Indicates the type of export that was canceled. OutputType can have the following values: 1 = Export format CSF 2 = Export format PDF 3 = Export format TIFF
ExportData	In	String	Not currently used.

PublishSuccess			
Syntax			
`PublishSuccess( [in] BSTR Filename, [in] PublishOutputOptions option)`			
Description			
This event is fired when an attempt to convert the currently viewed document or drawing to CSF has succeeded.			
Parameters	I/O	Type	Description
`Filename`	In	String	The parameter `Filename` returned by any of the publishing events is based on the `PublishOutputOptions.`
`option`	In	Publish Output Options	See `PublishOutputOptions` in the `BurnInFailure` event for additional publish option details.

PublishFailure			
Syntax			
`PublishFailure( [in] BSTR Filename, [in] PublishOutputOptions option)`			
Description			
This event is fired when an attempt to convert the currently viewed document or drawing to CSF has failed.			
Parameters	I/O	Type	Description
`Filename`	In	String	The parameter `Filename` returned by any of the publishing events is based on the `PublishOutputOptions.`
`option`	In	Publish Output Options	See `PublishOutputOptions` in the `BurnInFailure` event for additional publish option details.

BurnInSuccess			
Syntax			
`BurnInSuccess( [in] BSTR Filename, [in] PublishOutputOptions option)`			
Description			
This event is fired when an attempt to burn-in the currently open markup has succeeded.			
Parameters	I/O	Type	Description
`Filename`	In	String	The parameter `Filename` returned by any of the publishing events is based on the `PublishOutputOptions.`

option	In	Publish Output Options	See PublishOutputOptions in the BurnInFailure event for additional publish option details.

BurnInFailure

Syntax

```
BurnInFailure( [in] BSTR Filename, [in] PublishOutputOptions option)
```

Description

This event is fired when an attempt to burn-in the currently open markup has failed.

Parameters	I/O	Type	Description
Filename	In	String	The parameter Filename returned by any of the publishing events is based on the PublishOutputOptions.
option	In	Publish Output Options	PublishOutputOptions indicates the type of publishing output requested by the user. The following are valid values: Option: 0 — Full path of published file Option: 1 — Title of email message (generally the published file name)

ExportPDFSuccess

Syntax

```
ExportPDFSuccess( BSTR Filename)
```

Description

This event is fired when the control succeeded in creating a PDF rendition of the currently open drawing or document.

Parameters	I/O	Type	Description
Filename	In	String	Specifies the name of the PDF file created.

ExportPDFFailure

Syntax

```
ExportPDFFailure( BSTR Filename)
```

Description

This event is fired when the control was unable to create a PDF rendition of the currently open drawing or

document.			
Parameters	I/O	Type	Description
`Filename`	In	String	Specifies the name of file that failed to be created.

ExportTiffSuccess			
Syntax			
`ExportTIFFSuccess( BSTR Filename)`			
Description			
This event is fired when the control succeeded in creating a TIFF rendition of the currently open drawing or document.			
Parameters	I/O	Type	Description
`Filename`	In	String	Specifies the name of the TIFF file created.

ExportTiffFailure			
Syntax			
`ExportTIFFFailure( BSTR Filename)`			
Description			
This event is fired when the control was unable to create a TIFF rendition of the currently open drawing or document.			
Parameters	I/O	Type	Description
`Filename`	In	String	Specifies the name of file that failed to be created.

Message ID Section

Message Identification

ID	Description
1	Unspecified error
2	Invalid file extension
3	Invalid XDL file
4	Invalid page number
5	Cannot read page
6	Cannot localize file
7	Invalid element
8	Cannot construct I/O class
9	Invalid file ID tag
10	Cannot write page
11	Invalid markup file extension
12	Cannot read markup file
13	Future file format
14	Future markup file format
15	Format not licensed
16	Loader not found
17	No printer
18	Cannot find support file
19	Cannot find XRef file
20	Integration error
21	File load error
22	Markup file load error
23	Raster markup load error
24	Invalid redaction script
25	Cannot resolve markup file

26	Integration initialization error
27	Integration client ID error
28	Integration error from server
29	Incorrect native file password
30	Markup stamp entity load error
31	Internal reference file load error
32	.NET 3.0 not installed error
33	Future stamp file format error
34	Cannot read stamp file
35	Native file is protected
36	Cannot write file
37	Future redaction script error
38	Format not licensed continue
39	Markup file contains duplicate entities
40	Failed to allocate memory
41	Failed to export file
42	.NET 4.0 not installed
101	Security rights – Incorrect password
102	Security rights – File has expired
103	Security rights – Invalid file
104	Security rights – 3D format
105	Security rights – Later start date
106	Security rights – Redaction password error
107	Security rights – System time was altered
200	Invalid license
201	License has expired
202	License Manager – Request failed
203	License Manager – Request revoked
204	License Manager – Request expired
205	License Manager – No seats available
206	License Manager – General error
207	License Manager – Communication error

208	Register License – No Internet connection
209	Register License – Unable to contact host
210	Register License – Unable to generate request information
211	Register License – Unable to retrieve response
212	Register License – Invalid response from host
213	Register License – License verification failed
214	Register License – General error
215	Register License – Unable to update license locally
216	Register License – License already registered
217	Register License – Host unable to decode license
218	Register License – Host database update failed
219	Register License – Unable to re-encode license
220	Register License – Host unable to generate pin
221	Register License – Host unable to generate demo license
222	No design time license
223	Could not access license file
224	Invalid license from Brava! Server
225	License expiration warning period
226	Illegal new stamp template attempt

Control Identification Section

Control Identification

This section describes the Control Identifiers that allow integrators to customize the Brava! Interface. The various regions of the client interface can be modified to use custom color schemes and custom buttons.

Taskbar Controls

ID	Description
100	The top bar containing the main UI controls (**TaskRegion**)
101	The task bar's markup button
102	The task bar's save button
103	The task bar's print button
104	The task bar's select button
105	The task bar's annotate button
106	The task bar's review button
107	The task bar's redact button
108	The task bar's measure button
109	The task bar's takeoff button
110	The task bar's publish button
111	The task bar's download original button
112	The task bar's marketing message button
113	The task bar's markup exists icon
114	The task bar's custom bookmark button
115	The task bar's exit button used with integrations 
116	The taskbar’s Sign All button

General Panel Controls

ID	Description
200	The vertical bar that hides or displays the panel controls (

	PanelOpenCloseRegion, contains the panel toggle )
201	The window that contains the right-side panels (**PanelRegion**)
202	The tab buttons on the panel container 

Thumbnail Panel Controls

ID	Description
300	The thumbnail panel
301	The thumbnail panel first page button
302	The thumbnail panel previous page button
303	The thumbnail panel next page button
304	The thumbnail panel last page button
305	The thumbnail panel previous markup page button
306	The thumbnail panel next markup page button
307	The thumbnail panel list box full of thumbnails

Verify Panel Controls

ID	Description
400	The verify panel
401	The verify panel button to go to the first redaction
402	The verify panel button to go to the previous redaction
403	The verify panel button to go to the next redaction
404	The verify panel button to go to the last redaction
405	The verify panel help button
406	The verify panel no filter button
407	The verify panel reason filter button
408	The verify panel reason filter combo
409	The verify panel color filter button
410	The verify panel color selection button
411	The verify panel dropdown specifying a search target

412	The verify panel text "Verify"
413	The verify panel text showing the current position
414	The verify panel text indicating that document text is hidden
415	The verify panel top horizontal separator
416	The verify panel horizontal separator

Bookmark Panel Controls

ID	Description
500	The bookmark panel
501	The bookmark panel previous bookmark button
502	The bookmark panel next bookmark button
503	The bookmark panel help button
504	The bookmark panel search button
505	The bookmark panel search combo
506	The bookmark panel search separator
507	The bookmark panel bookmark display

Changemark Panel Display

ID	Description
600	The Changemark panel
601	The Changemark panel next Changemark button
602	The Changemark panel previous Changemark button
603	The Changemark panel find options button
604	The Changemark panel find reset button
605	The Changemark panel copy Changemark button
606	The Changemark panel help button
607	The Changemark panel help separator
608	The Changemark panel reply button
609	The Changemark panel sort button
610	The Changemark panel title list control

611	The Changemark panel text control
612	The Changemark panel find combo
613	The Changemark panel Changemark type control
614	The Changemark panel Changemark state control

Takeoff Panel Controls

ID	Description
700	The measure take off panel
701	The measure take off new category button
702	The measure take off edit category button
703	The measure take off delete category button
704	The measure take off export button
705	The measure take off category list
706	The measure take off total label
707	The measure take off type label
708	The measure take off total edit control
709	The measure take off total message
710	The measure take off scale label
711	The measure take off scale value

Checkview Panel Controls

ID	Description
2400	The Checkview panel
2401	The Checkview panel check list
2402	The Checkview panel ok button
2403	The Checkview panel not ok button
2404	The Checkview panel instruction display edit box
2405	The Checkview panel status text

2406	The Checkview panel help button
2407	The Checkview panel static text "instructions"

CAD Attribute Panel Controls

ID	Description
2500	The CADTAG panel
2501	The CADTAG panel color selection button
2502	The CADTAG panel zoom check box
2503	The CADTAG panel help button
2504	The CADTAG panel search button
2505	The CADTAG panel search combo
2506	The CADTAG panel tag display
2507	The CADTAG panel name/value list

Compare Controls

ID	Description
800	The compare toolbar
801	The compare bar align button
802	The compare bar clear alignment button
803	The compare bar help button
804	The compare bar overlay button
805	The compare bar overlay differences button
806	The compare bar side by side button
807	The compare bar text compare button
808	The compare bar open file button
809	The compare bar compare file button
810	The compare bar additions button
811	The compare bar deletions button
812	The compare bar unchanged button

813	The compare bar nudge button
814	The compare bar align instructions
815	The compare bar overlay transparency slider
816	The compare bar text compare report button 
817	The compare bar settings button

Text Compare Controls

ID	Description
900	The area containing all the controls for text compare
901	The text compare text indicating the open document
902	The text compare text indicating the compare document
903	The text compare text indicating the current compare position
904	The text compare next difference button
905	The text compare previous difference button
906	The text compare legend area for open document
907	The text compare vertical scrollbar for open document
908	The text compare open document area
909	The text compare alignment area
910	The text compare compare document area
911	The text compare vertical scrollbar for compare document
912	The text compare legend area for compare document
913	The text compare horizontal scrollbar
914	The text compare active document arrow indicators
915	The text compare top separator
916	The text compare bottom separator
917	The text compare horizontal scrollbar
918	The text compare resize handle

Navigation Bar Controls

ID	Description

1000	The bottom navigation bar (**NavigationRegion**)
1001	The navigation bar previous page button
1002	The navigation bar next page button
1003	The navigation bar page combo
1004	The navigation bar rotate button
1005	The navigation bar fit all button
1006	The navigation bar fit width button
1007	The navigation bar zoom slider button
1008	The navigation bar pan button
1009	The navigation bar zoom rectangle button
1010	The navigation bar magnifier button
1011	The navigation bar background color button
1012	The navigation bar layers button

Scroll Bar Controls

(**ScrollbarRegion**)

ID	Description
1100	The horizontal scroll bar
1200	The vertical scroll bar

Text Search Controls

ID	Description
1300	The text search control
1301	The text search find combo
1302	The text search find button
1303	The text search clear find button
3825	The text search results cancel button
3826	The text search results list
3827	The text search results column page
3828	The text search results column line

3829	The text search results found
3830	The text search results searched
3839	The text search results button image
3840	The text search results button, inversed image
3841	The text search results button
3842	The text search results tool tip

Focus View Controls

Verify Entire Document Window

ID	Description
1400	The focus pane information bar
1401	The focus view close button
1402	The focus view help button
1403	The focus view viewed text label
1404	The focus view viewed progress label
1405	The focus view page label
1406	The focus view restart button
1407	The focus view size button
1408	The scroll controls in the focus view
1409	The focus view page up button
1410	The focus view up button
1411	The focus view down button
1412	The focus view page down button

Information Bar Controls

ID	Description
1500	The toolbar containing the markup properties, measure results, and text find controls (**InformationRegion**)
1501	The help button on the info bar
1502	The file name displayed on the info bar

3D Information Bar Controls

ID	Description
5000	The 3D info bar
5001	The 3D info bar help menu
5002	The 3D info bar menu contents option
5003	The 3D info bar legal notices option

Measure Results Controls

ID	Description
1600	The measure/takeoff results bar
1601	The measure/takeoff distance label
1602	The measure/takeoff radius label
1603	The measure/takeoff count label
1604	The measure/takeoff first results text
1605	The measure/takeoff second results text
1606	The measure/takeoff third results text
1607	The measure/takeoff fourth results text
1608	The measure/takeoff perimeter label
1609	The measure/takeoff total distance
1610	The measure/takeoff height label
1611	The measure/takeoff circumference label
1612	The measure/takeoff angle label
1613	The measure/takeoff rect area label
1614	The measure/takeoff polygonal label
1615	The measure/takeoff circle area label
1616	The measure/takeoff remove last count label
1617	The measure/takeoff remove all counts label
1618	The measure take off cursor position label
1619	The measure take off snap check box

1620	The measure take off negative area check box
1621	The measure take off show leader check box

Markup Property Controls

ID	Description
1700	The bar containing information on markup properties
1701	The markup properties color button
1702	The markup properties fill combo
1703	The markup properties line weight combo
1704	The markup properties line style combo
1705	The markup properties arrow head combo
1706	The markup properties arrow tail combo
1707	The markup properties font combo
1708	The markup properties font size combo
1709	The markup properties stamp combo
1710	The markup properties browse for stamp button
1711	The markup properties image combo
1712	The markup properties browse for image button
1713	The markup properties text background button
1714	The markup properties text underline button
1715	The markup properties text italics button
1716	The markup properties text bold button
1717	The markup properties variable text button
1718	The markup properties hyperlink button
1719	The markup properties copy to pages button
1720	The redaction reason combo
1721	The redaction reason label
1722	The signature type combo
1723	The markup properties edit signature items button
1724	The markup properties create template signature check button

Measure Tools Controls

ID	Description
1800	The measure toolbar
1801	The measure line tool button
1802	The measure polyline tool button
1803	The measure polygon tool button
1804	The measure rectangle tool button
1805	The measure circle tool button
1806	The measure count tool button

Markup Tools Controls

ID	Description
1900	The markup toolbar
1901	The pop-out menu that appears from the measure toolbar (**PaletteSubMenuRegion**)
1902	The markup select tool button
1903	The markup arrow line tool button
1904	The markup text tool button
1905	The markup image tool button
1906	The markup stamp tool button
1907	The markup cloud tool button
1908	The markup polycloud tool button
1909	The markup highlight tool button
1910	The markup Changemarks tool button
1911	The markup highlight text Changemarks tool button
1912	The markup highlight Changemarks tool button
1913	The markup cloud Changemarks tool button
1914	The markup arrow Changemarks tool button
1915	The markup sketch tool button
1916	The markup sketch tool button

1917	The markup line tool button
1918	The markup polyline tool button
1919	The markup arc tool button
1920	The markup cross out tool button
1921	The markup scratch out tool button
1922	The markup arrow pointer tool button
1923	The markup rectangle tool button
1924	The markup rounded rectangle tool button
1925	The markup polygon tool button
1926	The markup ellipse tool button
1927	The markup strikeout tool button
1928	The markup strikethrough tool button
1929	The markup highlight text tool button
1930	The markup underline text tool button
1931	The markup check view tool button
1932	The markup signature tool button
1933	The markup strikethrough Changemark tool button

Redact Tools Controls

ID	Description
2000	The redaction toolbar
2001	The redaction select tool button
2002	The redact area tool button
2003	The redaction allow area tool button
2004	The redact privacy button
2005	The find and redact button
2006	The redact from - to button
2007	The redaction script button
2008	The redact page button
2010	The redact text tool button
2011	The redact polygon button

Takeoff Tools Controls

ID	Description
2100	The takeoff toolbar
2101	The takeoff select button
2102	The takeoff line button
2103	The takeoff polyline button
2104	The takeoff polygon button
2105	The takeoff rectangle button
2106	The takeoff circle button
2107	The takeoff count button

Splash Controls

ID	Description
2205	The splash picture
2206	The close pictures
2207	The version string
2208	The SKU string
2209	The expiration string
2210	The integration specific string (Brava! Enterprise)

Window Controls

ID	Description
2300	The main window for rendering documents and drawings.

Changemark Find Controls

ID	Description
101100	The Changemark panel find menu 'cumulative' option

101101	The Changemark panel find menu 'whole word' option
101102	The Changemark panel find menu 'match case' option
101103	The Changemark panel find menu 'term hit' option

Compare Toolbar Nudge Controls

ID	Description
101300	The compare toolbar nudge left menu option
101301	The compare toolbar nudge right menu option
101302	The compare toolbar nudge up menu option
101303	The compare toolbar nudge down menu option

Mouse Control Identification

Mouse Right-Click Markup Edit Controls

ID	Description
100000	The markup right-click cut command
100001	The markup right-click copy command
100002	The markup right-click paste command
100003	The markup right-click delete command
100004	The markup right-click select all command
100005	The markup right-click +/- symbol
100006	The markup right-click diameter symbol
100007	The markup right-click degree symbol
100008	The markup right-click copyright symbol
100009	The markup right-click registered symbol

Mouse Right-Click Markup Text Controls

ID	Description
100100	The markup properties right-click text background transparent option
100101	The markup properties right-click text background opaque option
100102	The markup properties right-click text background current color option
100103	The markup properties right-click text background select color option

Mouse Right-Click Controls

ID	Description
100200	The right-click previous view command
100201	The right-click fit all command
100202	The right-click fit width command
100203	The right-click pan command
100204	The right-click copy command
100205	The right-click paste command
100206	The right-click delete command
100207	The right-click undo command
100208	The right-click redo command
100209	The right-click copy region command
100210	The right-click redact selection command

Mouse Right-Click Page Controls

ID	Description
100300	The right-click page submenu
100301	The right-click page submenu next page command
100302	The right-click page submenu previous page command
100303	The right-click page submenu first page command
100304	The right-click page submenu last page command
100305	The right-click page submenu previous markup page command
100306	The right-click page submenu next markup page command

Mouse Right-Click Marks Controls

ID	Description
100400	The right-click marks submenu
100401	The right-click marks submenu add placekeeper command
100402	The right-click marks submenu next placekeeper command
100403	The right-click marks submenu previous placekeeper command
100404	The right-click marks submenu remove placekeeper command
100405	The right-click marks submenu next bookmark command
100406	The right-click marks submenu previous bookmark command
100407	The right-click marks submenu next Changemark entity command
100408	The right-click marks submenu previous Changemark entity command

Mouse Right-Click Rotate Controls

ID	Description
100500	The right-click rotate/mirror submenu
100501	The right-click rotate/mirror submenu rotate clockwise command
100502	The right-click rotate/mirror submenu rotate counter clockwise command
100503	The right-click rotate/mirror submenu mirror command

Mouse Right-Click Background Color Controls

ID	Description
100600	The right-click background color submenu
100601	The right-click background color submenu black option
100602	The right-click background color submenu white option
100603	The right-click background color submenu gray option
100604	The right-click background color submenu default option

Mouse Right-Click Options Controls

ID	Description
100700	The right-click options submenu
100701	The right-click options submenu animate option
100702	The right-click options submenu line weights option
100704	The right-click options submenu smooth edges option

Mouse Right-Click Show Pane Controls

ID	Description
100800	The right-click show pane submenu
100801	The right-click show pane submenu thumbnails command
100802	The right-click show pane submenu bookmarks command
100803	The right-click show pane submenu verify command
100804	The right-click show pane submenu Changemark command
100805	The right-click show pane submenu takeoff command
100806	The right-click show pane submenu Checkview command
100807	The right-click select all markup command
10808	The right-click create new markup group command
100809	The right-click dis-group the selected markup group command
100810	The right-click show pane submenu cadtag command

Mouse Right-Click Changemark Options Controls

ID	Description
101200	The Changemark panel sort menu title option
101201	The Changemark panel sort menu author option
101202	The Changemark panel sort menu date option
101203	The Changemark panel sort menu type option
101204	The Changemark panel sort menu state option
101205	The Changemark panel sort menu position option
101206	The Changemark panel sort menu replies option

Menu Control Identification

Markup Menu Controls

ID	Description
101000	The markup menu new command
101001	The markup menu open command
101002	The markup menu save command
101003	The markup menu save as command
101004	The markup menu close command
101005	The markup menu consolidate command
101006	The markup menu new stamp template command
101007	The markup menu open stamp template command
101008	The markup menu save stamp template command
101009	The markup menu save as stamp template command
101010	The markup menu close stamp template command

Variable Text Menu Controls

ID	Description
100900	The variable text menu date option
100901	The variable text menu system date plus days option
100902	The variable text menu time option
100903	The variable text menu military time option
100904	The variable text menu title option
100905	The variable text menu pages option
100906	The variable text menu total pages option
100907	The variable text menu Bates number option
100908	The variable text menu login option
100909	The variable text menu user option
100910	The variable text menu prompt option
100911	The variable text menu DBstring option

100912	The variable text menu DBUpdateString option
100913	The variable text menu % option
100914	The variable text menu hostname option
100915	The variable text menu IP address option
100916	The variable text menu registered symbol option
100917	The variable text menu copyright symbol option

Find Menu Controls

ID	Description
101400	The find menu down option
101401	The find menu up option
101402	The find menu match case option
101403	The find menu whole word only option
101404	The find menu regular expressions option
101405	The find menu term hit option

Color Menu Controls

ID	Description
101500	The navigation toolbar color menu black option
101501	The navigation toolbar color menu white option
101502	The navigation toolbar color menu gray option
101503	The navigation toolbar color menu default option
101504	The navigation toolbar color menu monochrome option
101505	The navigation toolbar color menu raster background option
101506	The navigation toolbar color menu raster background transparent option
101507	The navigation toolbar color menu raster background current color option
101508	The navigation toolbar color menu raster background change color option

Print Menu Controls

ID	Description

101600	The print menu print option
101601	The print menu print region option
101602	The print menu watermark/banners option

Review Menu Controls

ID	Description
101700	The review menu close review option
101701	The review menu Changemark review option
101702	The review menu next Changemark option
101703	The review menu previous Changemark option

Redact Menu Controls

ID	Description
101800	The redact menu publish submenu
101801	The redact menu publish to PDF option
101802	The redact menu publish to TIFF option
101803	The redact menu publish to draft submenu
101804	The redact menu publish to draft PDF option
101805	The redact menu publish to draft TIFF option
101806	The redact menu open redaction template option
101807	The redact menu run script option
101808	The redact menu verify submenu
101809	The redact menu verify submenu redaction option
101810	The redact menu verify submenu image areas option
101811	The redact menu verify submenu entire document option
101812	The redact menu publish to CSF option

Measure Menu Controls

ID	Description

101900	The measure menu measure/takeoff option
101901	The measure menu calibrate option
101902	The measure menu settings option

Publish Menu Controls

ID	Description
102000	The publish menu publish to CSF option
102002	The publish menu save as JPG option
102003	The publish menu publish to PDF option
102004	The publish menu publish to TIFF option

Help Menu Controls

ID	Description
102100	The help menu contents option
102101	The help menu about option
102103	The help menu for legal notices option

Thumbnail Menu Controls

ID	Description
102200	The thumbnail menu reduce option
102201	The thumbnail menu enlarge option
102202	The thumbnail menu default option

External Bookmarks Menu Controls

ID	Description
102300	The external bookmark menu open option
102301	The external bookmark menu add option
102302	The external bookmark menu delete option

Sign All Menu Controls

ID	Description
103300	The Taskbar edit signature items menu item

CAD Attribute Search Menu Controls

ID	Description
102400	The CADTAG panel search option filter menu item
102401	The CADTAG panel search option clear filter menu item
102402	The CADTAG panel search option include categories menu item
102403	The CADTAG panel search option include Name/Value pairs menu item

Functionality Identification Section

Functionality Identifiers

ID	Description	ID	Description
100	Copy	208	Auto save view states
101	Paste	209	Measure circle
102	File open	210	Markup text background color choice
103	Save view as	211	Markup stamp template
104	Markup all	212	Markup new stamp template
105	Markup review	213	Markup save stamp template
106	Markup edit	214	Markup save as stamp template
107	Markup open	215	Markup open stamp template
108	Markup edit new	216	Markup close stamp template
109	Markup edit save	217	Markup stamp entity tool
110	Markup edit save as	218	Browse for stamp entity
		219	Markup save task bar
112	Markup burn-in	220	Copy Changemarks
113	Markup edit close	221	All marks (right mouse pop-up)
114	Markup review open	222	Right mouse button menu
115	Markup review close	223	Markup allow area tool
116	Review Changemarks	224	Undo
117	Review markup pages	225	Redo
118	Print	226	Redact by script
126	Layers	227	Redact by search macro
127	Edit watermark/banners	228	Redact by search string
128	Next page	229	Next file
129	Previous page	230	Previous file
130	Rotate 90 degrees	231	Finalize redactions

131	Background color	232	Verify redactions
132	Monochrome	233	Open redaction
133	Page control	234	Search images
134	Mirror raster	235	Display bookmarks
135	Publish to CSF	236	Redaction script management
136	Markup integration available	237	Custom exit button
		238	Markup redaction help tool
138	Save view as JPG	239	External bookmarks
139	File close	240	Open external bookmarks
141	Select	241	Add external bookmarks
142	Markup select	242	Delete external bookmark
143	Compare documents	243	Redaction tools
144	Copy region	244	Markup byzantine arrow tool
		245	Next/Previous bookmark
145	Copy markups to pages	246	Next/Previous Changemarks
146	Browse for raster markup	247	Markup rounded rectangle tool
147	Publish to PDF	248	Cut
148	Pan tool	249	Delete
149	Zoom window	250	Select all markup text
150	Zoom	251	Verify panel
151	Magnifier	252	Show panel bar
152	Select tool	253	Redact page
153	Download original	254	Start redaction script
154	View at extents	255	Stop redaction script
155	View at width	256	Delete all markups
156	Help	257	OCR
157	Percent scale	258	Redact From/To phrase
158	Find text	259	Append files
159	Close compare		
160	Rotate tool	261	Visual Rights display
161	Markup sketch tool	262	View colors

162	Markup polygon tool	263	Display Measure results
163	Markup text tool	264	Display measure panel
164	Markup ellipse tool	266	Markup highlight
165	Markup arc tool	267	Markup Changemarks highlight
166	Markup rectangle tool	268	Markup Changemarks text highlight
167	Markup polyline tool	269	Markup Changemarks cloud
168	Markup line tool	270	Markup Changemarks arrow
169	Markup arrow tool	271	Markup strikethrough text
170	Markup polygon sketch tool	272	New takeoff
171	Markup cross out tool	273	Start compare alignment
172	Markup strike out text tool	274	Clear compare alignment
173	Markup underline text tool	275	Takeoff line
174	Markup highlight text tool	276	Takeoff polyline
175	Markup Changemarks tool	277	Takeoff polygon
176	Markup polygonal cloud tool	278	Takeoff rectangle
177	Markup cloud tool	279	Takeoff circle
178	Markup raster tool	280	Takeoff count
179	Markup redact tool	281	Takeoff edit
180	Markup rectangle squiggle tool	282	Takeoff delete
181	Markup hyperlink tool	283	Text compare
182	Compare alignment tool	284	External Changemarks text
183	Thumbnails	285	Publish to Draft PDF
184	Thumbnails resize	286	Publish to Draft TIFF
185	Thumbnails resize default	287	Finalize Publish
186	Thumbnails resize reduce	288	Compare Report
187	Thumbnails resize enlarge	289	Redaction Selected
188	Animation draw	290	Markup Redact Text
189	Show line weights	291	Raster Background Color
190	Compare nudge	292	Changemark Edit Type
191	Last page	293	Changemark Edit State
192	First page	294	Demo Licence
193	Markup take ownership	295	Compare Settings

194	Markup consolidate	296	Markup Checkview tool
195	Publish to TIFF	297	Display Checkview panel
196	Placekeepers	298	Markup Signature tool
197	Add placekeeper	299	Markup edit signature items
198	Next placekeeper	300	Markup edit sign all
199	Previous placekeeper	301	Clear find text
200	Clear placekeepers	302	Search results
201	View back	303	Paste image from clipboard
202	View forward	304	Strikethrough Text tool
203	Page changing	305	Markup Open takeoff template
205	Buy now	306	Markup Save takeoff template
206	Copy page text to clipboard	307	Markup SaveAs takeoff template
207	Go to placekeeper	308	Select all markups on page
		309	Group selected markup
		310	Explode selected markup
		311	Markup tool callout rectangle
		312	Markup tool callout circle
		313	Markup tool callout triangle
		314	Display CAD Attribute panel

Help Topic IDs Section

Help Topic Identifiers

ID	Description	ID	Description
1	BACKGROUND_FILLS	105	BURNING_IN_MYRIAD_MARKUPS
2	CREATING_CUSTOM_MACROS	106	BRAVA_DESKTOP_SKU_S
3	SCRIPT_EDITING_FOR_ADVANCED_USERS	107	THUMBNAIL_PAGES
5	BURNED_IN_MARKUPS	109	PUBLISHING_FILES_TO_TIFF
6	CLOSING_MARKUPS	110	PUBLISHING_FILES
7	ALLTOOLBARS	111	BOOKMARKS
8	WATERMARK1	112	CUSTOMIZATION
9	REDACT_TEXT_TOOL	113	CIRCLE
10	ARC	114	EXTRACT_CHANGEMARKS
11	BACKGROUND_COLOR	115	MARKUP_STAMP
12	BLOCK_OUT	116	STAMP_TEMPLATES
13	STARTEXPIRE	117	MARKUP_TEXT_BACKGROUND
14	DISPLAY_OPTIONS	118	VIEWXPS_IDW
15	DOCUMENT_PAGE_CONTROL	119	USING_SEARCH_MACROS
16	CHANGEMARKS_FILTERS	120	CALIBRATE
17	EDIT_TEXT	160	REDACTION
18	SET_BATES_NUMBER	170	VERIFYREDACTIONS
19	CONTACT_INFORMATION	190	EDITING_MARKUPS
20	FIT_ALL	200	AUTOMATION_FEATURES
21	CREATING_CHANGEMARKS	210	FIT_WIDTH
22	CREATING_MARKUPS	220	FONT_CHOOSER
23	CROSSOUT	222	WILDCARD_QUICK_TIPS
24	CSF_PUBLISHING	230	HIDES
25	HOTKEYS	290	LINE
26	HYPERLINK	350	HIGHLIGHTS

28	ISO_BANNERS	450	OPEN_A_DOCUMENT1
29	EDITABLE_MARKUPS	550	NEWFEATURES
30	LINE_ARROW	600	DLTEXT
31	LINE1	670	RECTANGLE
32	LINETHICKNESS	700	CLOUD
33	MAGNIFY	777	PATTERN_SEARCH
34	MARKUP_COLOR	833	VIEW_CSF
35	MARKUP_FILES	900	COPY_TEXT
36	MARKUP_FONT	950	LAUNCH_AUTOCAD
37	MARKUP_LINE_WIDTH	970	COPY_REGION
38	MARKUP_PAGES	980	EXPIRATION_DATES
39	MARKUP_PROPERTIES	1000	REDACT_ALL_PAGES
40	MARKUP_TOOLBAR	1001	FIND_AND_REDACT_FROMTO
41	MEASURE	1002	MEASUREMENT_TAKEOFF
42	MEASURE_COUNT	1003	TAKEOFF_CATEGORIES
43	MEASUREMENT_SETTINGS	1004	ACCUMULATION_RESULTS
44	MONOCHROME	1005	CHANGEMARK_CLOUD
46	OVERLAY_MARKUPS_FOR_REVIEW	1006	CHANGEMARK_ARROW
47	PAGE_CONTROL	1007	HIGHLIGHT
48	PAN	1008	TEXTHIGHLIGHT_CHANGEMARK
49	POLYGON	1009	HIGHLIGHT_CHANGEMARK
50	POLYLINE	1010	MEASURE_POLYLINE1
51	PRINT	1011	CHANGEMARK_DISCUSSION
52	PRINT_A_FILE	1012	REDACT_TOOLS
53	PRINT_REGION	1013	TEXT_COMPARE
54	PRINT_TIPS_AND_TROUBLESHOOTING	1014	TAKEOFF_REVIEW
55	PRINT_TO_SCALE	1015	CHANGEMARK_TYPES_AND_STATES
57	REVIEWING_CHANGEMARKS	1016	CONDITIONAL_MACROS
58	ROTATE	1017	CUSTOM_SKINS
59	SAVING_MARKUPS	1018	GLOSSARY_TERMS
60	SCRATCH	1019	OPEN_REDACTION_TEMPLATE
61	SEARCH	1020	OPENING_CLOSING_FILES

62	SELECT_ENTITY_TOOL	1021	PUBLISHING_REVIEW_DRAFTS
63	SELECT_TOOL	1022	REDACTION_REASONS
64	SKETCH	1023	TEXT_SEARCH_RULES
65	SOLID_AND_HOLLOW_SHAPES	1024	TEXT_SEARCH_STRINGS
67	STATUS_BAR	1025	VERIFY_ENTIRE_DOCUMENT
68	TAGS	1026	CAD_SETTINGS
72	VISIBLE_LAYERS	1027	DELETING_REDACTIONS
73	WATERMARK	1028	TEXT_COMPARE_REPORT
74	WELCOME_TO_BRAVA_	1030	PERMISSIONS
76	ZOOM_WINDOW	1070	CONSOLIDATING_MARKUPS
77	RIGHTMOUSE	1080	CHANGING_MARKUP_OWNERSHIP
79	PUBLISHING_WITH_BRAVA_DESKTOP	1090	COPYING_MARKUPS
82	WHAT_IS_CSF_	1120	USER_PERMISSIONS
83	UPDATE_VERSION	1130	REDACT_PRIVACY_INFORMATION
85	MIRROR	1131	BRAVA_TASK_PANE
86	SAVE_VIEW	1132	PLACEKEEPERS
88	DOWNLOADING_TOOL	1133	STAMP_TOKENS
90	OPEN_FOR_COMPARISON	1190	FEATURE_AVAILABILITY
91	WHY_CSF_	1200	REDACTION_CARVING_TOOL
93	SET_PATHS	2003	COMMAND_TOOLBAR
94	SEND_FILE	2010	ACKNOWLEDGEMENTS
95	MEASURE_POLYLINE	2018	UNDO_REDO
96	FILE_TYPES	2019	VERIFY_PANEL
97	ALIGN_COMPARISON_TOOL	2020	VIEW_LOG
98	FIND_AND_REDACT	2021	WATERMARKCSF
99	PUBLISHING_MARKUPS_TO_PDF_FILES	2022	KEYBOARD_508
100	MEASURE_MAGNIFICATION_TOOL	2023	PRIVILEGE_LOGGING
101	INSERT_RASTER_IMAGE	2024	REASON_CUSTOMIZATION
102	REDACT_BY_SCRIPT	2025	STRIKETHROUGH_CHANGEMARK
103	REDACT_MOUSE_TOOL		
104	PUBLISHING_FILES_TO_PDF		

HTML Parameters

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 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 CompareFilename

Control Key Combinations

Control Keys

Combination	Action	Key ID Value
CTRL + A	Show about	1048641
CTRL + SHIFT + A	Change to pan tool	1052737
CTRL + SHIFT + B	Burn in markup	1052738
CTRL + ALT + B	Add a bookmark	1114178
CTRL + B	Toggle through bookmarks	1048642
ALT + B	Toggle through background colors	65602
CTRL + C	Copy selected	1048643
CTRL + SHIFT + C	Close editable markup	1052739
CTRL + SHIFT + D	Export PDF	1052740
CTRL + E	Fit all	1048645
CTRL + F	Mirror raster	1048646
CTRL + G	Set print region tool	1048647
CTRL + H	Show help	1048648
CTRL + SHIFT + J	Save view as JPG	1052746
CTRL + SHIFT + K	Export CSF	1052747
CTRL + L	Show layer dialog	1048652

CTRL + M	Open editable markup	1048653
CTRL + N	New editable markup	1048654
CTRL + O	Open file	1048655
CTRL + P	Show print dialog	1048656
CTRL + R	Open review markup	1048658
CTRL + SHIFT + R	Close review markup	1052754
CTRL + S	Save editable markup	1048659
CTRL + SHIFT + S	SaveAs editable markup	1052755
CTRL + T	Toggle thumbnail display	1048660
CTRL + SHIFT + T	Export TIFF	1052756
CTRL + V	Paste	1048662
CTRL + W	Fit width	1048663
CTRL + X	Delete	1048664
CTRL + SHIFT + X	Change to zoom tool	1052760
CTRL + Y	Redo	1048665
CTRL + Z	Undo	1048666
CTRL + SHIFT + Z	Change to magnifier tool	1052762
CTRL + SPACE	Rotate 90 degrees clockwise	1048608
CTRL + SHIFT + SPACE	Rotate 90 degrees counter-clockwise	1052704
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
CTRL + SHIFT + V	Show verify panel	1052758
CTRL + SHIFT + K	Publish to CSF	1052747
CTRL + SHIFT + M	Show Changemarks panel	1052749
CTRL + ALT + A	Show CAD Attribute panel	1114177
CTRL + ALT + E	Show Checkview panel	1114181
SPACEBAR	Open Next file in folder	NA
BACKSPACE	Open Previous file in folder	NA