# ComparisonSlider.Common.VDX.Components.AdEx

**Template/Demo URLs:** [Click Here](https://wiki.exponential.com/display/VDXConnect/SingleFramework.VDXConnect.Templates.AdEx)

## Updated for components-release - [HTML5 Components 3.11.0-5 Release](file:////display/VDXComponents/HTML5+Components+3.11.0-5)

## Functionality:

Component allows showing a comparison of 2 elements (image-image, video-image, video-video) by sliding or auto moving a border to show/hide the top element. There can be any amount of components in a unit.

## Muse setup:

Designer should drop a placeholder on a page, set size and position and add a unique title id.   
Also designer has to upload 2 files (image-image, video-video) to the **mainunit/assets** folder.

Configuration properties details

## ****Latest property updates:****

| **Property** | **Type** | **Default** | **Available in**  **Non-Muse Flow** | **Screenshot of property** | **Description** | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| isVertical | Boolean | false | No |  | experimental if true then separator would switch comparison into vertical mode) | | |
| leftVideo {      videoId     isMuted     hasAudio     loop } | Object     String     Boolean     Boolean     Boolean | Image Comparison by default | Yes (partial) |  | Parameters required for Left video: **videoId**: Define the video name which we want to show on the left side in video comparison.     **isMuted**: If set to true then sound would be initially muted.     **hasAudio**: If set to false then video would play muted permanently.     **loop**: If set to true then video would play in loop | | |
| rightVideo {      videoId     isMuted     hasAudio     loop } | Object     String     Boolean     Boolean     Boolean | Image Comparison by default | Yes |  | Parameters required for Right video:     **videoId**: Define the video name which we want to show on the right side in video comparison.     **isMuted**: If set to true then sound would be initially muted.     **hasAudio**: If set to false then video would play muted permanently.     **loop**: If set to true then video would play in loop | | |
| leftOverlay { **\*new in 3.10.2**     contentId     contentHtml     style } | Object     String     String     Css-Style |  |  |  | Defines left overlay:     **contentId**: [optional] id of the document block to overlay content.     **contentHtml**: [optional] html markup to overlay content.     **style**: additional inline-style for the overlay | | |
| rightOverlay { **\*new in 3.10.2**     contentId     contentHtml     style } | Object     String     String     Css-Style |  |  |  | | Defines right overlay:     **contentId**: [optional] id of the document block to overlay content.     **contentHtml**: [optional] html markup to overlay content.     **style**: additional inline-style for the overlay |
|  |  | Left video will play with the sound and right video will be muted. |  |  | This enables the sound icons for both videos in comparison slider and we can toggle the sound by clicking on the icon.  Once we enable this property, we can access all 4 possible states by clicking on the sound icons:   * Default – On initialisation | Property values: leftVideo.isMuted = false, rightVideo.isMuted = false, showMuteIcon = true. * Both Muted - Property values: leftVideo.isMuted = true, rightVideo.isMuted = true, showMuteIcon = false. * Left Video Muted - Property values: leftVideo.isMuted = true, rightVideo.isMuted = false, showMuteIcon = true. * Right Video Muted -  Property values: leftVideo.isMuted = false, rightVideo.isMuted = true, showMuteIcon = true. | | |
| videoFit | String | cover | Yes |  | “Object-fit" for videos inside the container. | | |
| playback | String | sync | No |  | Do not change the value. | | |
| syncGap | Number-time-s | 0.2 | No |  | Defines maximum playback progress difference between main and secondary videos for "sync" mode. Not supported yet ın VDX Studıo. | | |
| syncEndDiff | Number-time-s | 1.0 | No |  | Defines maximum duration difference between main and secondary videos for "sync" mode. Planned to be below 1.0 all the time. Not supported yet ın VDX Studıo. | | |
| slideIconWidth | Number |  | Yes |  | Set the size of slider icon through this property. | | |
| autoPlayEnable | Boolean | true | No |  | If set to false then auto play mode wouldn't start no event if autoPlay value is not 0 | | |
| autoPlay | Number-time-ms |  | Yes |  | Time to start auto-play after initialisation | | |
| autoPlayResume | Number-time-ms |  | Yes |  | The time after which the auto play will resume once the user interacts with the component. | | |
| autoPlayRepeat | Number | 0 | No |  | Amount of automatic swipe repetitions, 0 = infinite times | | |
| autoPlayTick | Number-time-ms |  | Yes |  | Travel duration in which the slider(with hand cursor) will travel from left-to-right and right-to-back in case of auto play. | | |
| autoPlayLeftRightPx | num Number | [0, 0] | Yes |  | Defines left and right offsets of both sides in AutoPlay mode.   If we set it as [10, 20] then during AutoPlay mode slider line will stop 10px away from the left side when going to the left and would stop 20px away from the right side when going to the right.  In this case, designer must know that maximum values for both sides can't be more than half of the comparison slider container. This parameter is optional and can be removed from the configuration, if not parameter at the configuration, default value will be set as [0, 0]. | | |
| autoPlayHintHtml | String-html | Optional in Muse flow | No |  | We can customize the [hand cursor](https://prnt.sc/9RRwwo8aWHFt) icon for “autoPlayTick” animation through css. | | |
| autoPlayHintImage | String-url | - | No |  | Alternative to autoPlayHintHtml property, image to be shown above side-icon during auto-play | | |
| autoPlayHandCursor | Boolean | true | No |  | Show hand hint image over slider icon to invite to interact with component | | |

## User defined parameters:

* "**placeholderId**" (String - placeholder title)
* **"isVertical"** (Boolean - experimental if true then separator would switch comparison into vertical mode)
* "**leftImageName**" (String - the name of the left image)
* "**rightImageName**" (String - the name of the right image)
* **"slideIcon"** (String - image name of an icon that will be showed to let user know how to use it)
* "**slideIconPosition**" (String - position of a slide icon, uses css property "background-position", default: "center", possible values - canter|top|bottom|0px 30% )
* "**animationOn**" (Boolean - enable/disable animation, default: true)
* "**startingPosition**" (Number - starting position of the slider in percents, default: 50, possible values - from 0 to 100)
* "**clickToMoveOn**" (Boolean - enable/disable sliding by click to an image, default: true)
* "**sliderWidth**" (Number - width of the slider line)
* "**sliderColor**" (String - color of the slider line, default: #000)
* "**sliderStyle**" (String, style of the slider line, default: solid, possible: none|dotted|dashed|solid|double)
* "**backgroundImageSize**" (String, css style - "background-size" for images inside the container, possible values - auto|contain|cover, default - cover)
* "**preloaderColor**" (String, - css style - 'background-color' of the preloader, default: rgb(0, 174, 239), possible values - rgb(x, x, x)|rgba(x, x, x, y)|#ffffff
* "**preloaderContainerHeight**" (String, - height of the preloader, default: 3px, possible values - Xpx|X% ),
* "**preloaderImage**" (String, - image/gif/svg for background of preloader, if set - height becames 100% and transparent background, you can set any height and background colors by setting preloaderHeight and preloaderColor, default: ""),
* "**preloaderImagePosition**" - (String, the position of the preload image inside the preload container, default: "center", possible values - top|bottom|center|left|right|top left|top center|top right|center left|center right|Xpx Ypx|X% Y% )
* "**preloaderImageSize**" (String, css style - "background-size" for images inside the container, possible values - auto|contain|cover, default - auto)
* "**preventRender**" (Boolean - can be used for any component, prevent calling of "render" function at the initialization, the "render" function can be called anytime by a designer, default: true)
* **"leftVideo"** (Object - contain data about leftVideo element)
  + **"videoId"** (String - string with videoId number)
  + **"isMuted"** (Boolean - controlled if the video is muted from component config, if it's muted from configuration user can't control mute/unmute state for this video, default: true)
  + **"hasAudio"** (Boolean - optional indicates if video has soundtrack or not)
  + **"loop"** (Boolean - if true then video would be playing in loop)
* **"rightVideo"** (Object - contain data about leftVideo element)
  + **"videoId"** (String - string with videoId number)
  + **"isMuted"** (Boolean - controlled if the video is muted from component config, if it's muted from configuration user can't control mute/unmute state for this video, default: true)
  + **"hasAudio"** (Boolean - optional indicates if video has soundtrack or not)
  + **"loop"** (Boolean - if true then video would be playing in loop)
* **"showMuteIcon"** (Boolean - show/hide control element with mute/unmute functionality)
* **"videoFit"** (String - css style - "object-fit" for videosinside the container, possible values - cover | contain | none, default - cover)
* **"playback"** (String - configures videoPllayback functionality, possible values - primary | sync | async, default - sync)
* **"syncGap"** (Number - time in second, means how much time gap can be before videos will be synchronized at playback "sync" mode, default - 0.2)
* **"slideIconWidth"** (Number - sets size of slide icon, default - 50)
* **"autoPlayEnable"** (Boolean - if false then auto play mode wouldn't start no event if autoPlay value is not 0)
* **"autoPlay"** (Number-time-ms - time to start auto-play after initialization, default - )
* **"autoPlayResume"** (Number-time-ms - time to resume auto-play after comparison click/tap interaction, default - 0)
* **"autoPlayTick"** (Number-time-ms - time between animation ticks, when sliding left-to-right and back, default - 0)
* **"autoPlayRepeat"** (Number - amount of automatic swipe repetitions, 0 = infinite times, default - 0)
* **"autoPlayLeftRightPx"** ([number, number] - left and right offsets for auto-play positions in px, default - [ 0, 0 ])
* **"autoPlayHintHtml"** (String-html - html overlay to be shown above side-icon during auto-play, default - optional)
* **"autoPlayHintImage"** (String-url - alternative to autoPlayHintHtml property, image to be shown above side-icon during auto-play, default - optional)
* **"autoPlayHandCursor"** (Boolean - show hand hint image over slider icon to invite to interact with component, default true)

## Video parameters description:

* The replay button will be shown when all videos will be finished
* The video controls section is not showing (progress bar, play/pause buttons)
* The comparison slider can show video and image at the same config (image/image, image/video, video/video)
* The comparison slider has its own mute/unmute button with 3 states support:
  1. enable left sound - disable right sound
  2. disable left sound - disable right sound
  3. disable left sound - enable right sound
* playback parameter description (component default value - 'sync')
  1. "primary" - left video will start, right video will be paused
  2. "sync" - video will be playing synchronous when video positions gap between 2 videos will be more than **syncGap** parameter value, right video position will be overwritten by left video position value. For the first phase, **syncGap** value will be set by component (default value - 0.2) second.
  3. "async" - videos will be played independently
* for VDXStudio flow value for 'showMuteIcon' parameter will have a dependency on the Video sound dropdown ([xd prototype from Harry](https://xd.adobe.com/view/cd984ff7-e7d9-4ffd-86be-c53c417eabdb-98cb/screen/dbd421e5-08c0-4d38-9b34-7e0da4bacc45?fullscreen)), Video sound dropdown values description:
  1. Default - User can control both of the videos mute/unmute states, left video start plaing with sound. Property values: leftVideo.isMuted = false, rightVideo.isMuted = false, showMuteIcon = true.
  2. Both Muted - Videos will be played without sound and user can't control mute/unmute states. Property values: leftVideo.isMuted = true, rightVideo.isMuted = true, showMuteIcon = false.
  3. Left Muted - Left video is muted, user can control mute/unmute state for right video.  Property values: leftVideo.isMuted = true, rightVideo.isMuted = false, showMuteIcon = true.
  4. Right Muted - Right video is muted, user can control mute/unmute state for left video.  Property values: leftVideo.isMuted = false, rightVideo.isMuted = true, showMuteIcon = true.

## Other parameters description and explanation:

* **"autoPlayLeftRightPx"** - defines left and right offsets of component sides for auto-mode. So if you set it as **[10, 20]** then during auto-mode separator line will stop 10px away from the left side when going to the left. And would stop 20px away from the right side when going to the right. In this case, designer needs to know that maximum values for both sides can't be more than half of the comparison slider container. This parameter is optional and can be removed from the configuration, if not parameter at the configuration, default value will be set as **[0, 0]**.

**Configuration parameters for video support (this parameter is added  to support video functionality, all other parameter are still in use):**

## Methods:

Expo.designerAPI.comparisonSlide   - get all copies of the component,  
Expo.designerAPI.comparisonSlide.get('placeholderId') - find a unique copy of the component

**reset**(animationOn) - set the slider to the default position, animationOn can be true or false, default - false.  
**init**(config) - reset default config, options is a object of componenta parameters  
**render**() - render a dom of the component with default config  
**update**(config) - destroy the dom and rerender it with a new options

**play**() - resumes playback of the component (video playback and autoplay)  
**resume**() - resumes playback of the component (video playback and autoplay)  
**pause**() - pauses playback of the component (video playback and autoplay)  
**stop**() - stop component (video playback and autoplay)

## Examples:

Expo.designerAPI.comparisonSlide.get('VDX\_comparisonSlide\_1').reset(); - reset only one copy of the component with a placeholder 'VDX\_comparisonSlide\_1'  
Expo.designerAPI.comparisonSlide.reset(); - reset all copies of the component.

## Component callbacks:

* "**onSliderClicked**" (fires when slider is clicked)
* "**onLeftImageClicked**" (fires when the left image is clicked)
* "**onRightImageClicked**" (fires when the right image is clicked)

## Designer-config code example:

{  
        name: "comparisonSlide",  
        props: {

            preventRender: true,  
            placeholderId:"comparisonSlide\_1",  
            leftImageName:"images/before1.jpg",  
            rightImageName:'images/after1.jpg',  
            slideIcon:'images/drag\_item.png',  
              
            startingPosition:30,  
            sliderWidth: 1,  
            sliderColor: 'rgba(0,0,0,.3)',  
        },  
        events: {  
            onSliderClicked: function(Expo, obj) {  
                Expo.designerAPI.firePixel("onSliderClicked\_1",[{"eventName":"intLive", "multi":true}]);  
            },  
            onLeftImageClicked: function(Expo, obj) {  
                Expo.designerAPI.firePixel("onLeftImageClicked\_1",[{"eventName":"intLive", "multi":true}]);  
            },  
            onRightImageClicked: function(Expo, obj) {  
                Expo.designerAPI.firePixel("onRightImageClicked\_1",[{"eventName":"intLive", "multi":true}]);  
            }  
        }  
    },              
    {  
        name: "comparisonSlide",  
        props: {  
            placeholderId:"comparisonSlide\_2",  
            preventRender: true,  
            leftVideo: {

                videoId: "video2", // videoId  
                isMuted: true,     // if video is muted, it can't be controlled mute/unmute by user   
            },  
            rightVideo: {  
                videoId: "video3",  
                isMuted: false,  
            },  
            showMuteIcon: true, // show/hide controls with mute/unmute functionality  
            videoFit: "cover",  // cover | contain | none  
            playback: "sync",   // "primary" | "sync" | "async"  
            syncGap: 0.2, // time gap between videos in seconds

            slideIcon:'',  
            animationOn:false,  
            startingPosition:70,  
            clickToMoveOn:false,  
            sliderWidth: 8,  
            sliderColor: 'rgba(0,0,0,.3)',  
            sliderStyle: 'dotted',  
        },  
        events: {  
            onSliderClicked: function(Expo, obj) {  
                Expo.designerAPI.firePixel("onSliderClicked\_2",[{"eventName":"intLive", "multi":true}]);  
            },  
            onLeftImageClicked: function(Expo, obj) {  
                Expo.designerAPI.firePixel("onLeftImageClicked\_2",[{"eventName":"intLive", "multi":true}]);  
            },  
            onRightImageClicked: function(Expo, obj) {  
                Expo.designerAPI.firePixel("onRightImageClicked\_2",[{"eventName":"intLive", "multi":true}]);  
            }  
        }  
    }

## Limitations:

* Both the videos sound cannot be unmuted at the same time.
* For the first release Phase, we allow videos to compare only with the same length of the video.
* For the first release Phase, we do not allow video-image compare configurations at both (Muse, nonMuse) flows.
* Supported video durations between min 15 seconds to 5 minutes maximum.

## Notes:

* The placeholder should be a container('button')

Code Snippet

* **Use this line of code on Tab to Initialise the Comparison Slider**  
  Expo.designerAPI.comparisonSlide.get("comparisonSlide").render();
* **Use this line of code on Tab to reset the Comparison Slider**  
  Expo.designerAPI.comparisonSlide.get("comparisonSlide").reset();

/\* Start ComparisonSlider Code from Here \*/

{

name: "comparisonSlide",

props: {

            preventRender: true,

placeholderId:"comparisonSlide1",

leftImageName:"images/before.jpg",

rightImageName:'images/after.jpg',

slideIcon:'images/drag\_icon.png',

startingPosition:50,

sliderWidth: 3,

sliderColor: 'rgba(0,0,0,.3)',

sliderStyle: 'dotted',

preloaderImage: 'images/clock\_preloader.gif',

preloaderImageSize: "auto",

preventRender: true

},

events: {

onSliderClicked: function(Expo, obj) {

Expo.designerAPI.firePixel("ComparisonSlide1\_onSliderClicked",[{"eventName":"intLive", "multi":true}]);

},

onLeftImageClicked: function(Expo, obj) {

Expo.designerAPI.firePixel("ComparisonSlide1\_onLeftImageClicked",[{"eventName":"intLive", "multi":true}]);

},

onRightImageClicked: function(Expo, obj) {

Expo.designerAPI.firePixel("ComparisonSlide1\_onRightImageClicked",[{"eventName":"intLive", "multi":true}]);

}

}

},

{

name: "comparisonSlide",

props: {

preventRender: true,

placeholderId: "comparisonSlide2",

// leftImageName: "images/comparison\_before.jpg",

// rightImageName: "images/comparison\_after.jpg",

leftVideo: {

videoId: "video2",

isMuted: false,

},

rightVideo: {

videoId: "video3",

isMuted: false,

},

showMuteIcon: true,

videoFit: "cover",

playback: "sync",

// syncGap: 0.2, // for current version is used default value from bundle

slideIcon: "images/comparison\_drag\_icon.png",

slideIconPosition: "center",

animationOn: true,

startingPosition: 50,

clickToMoveOn: true,

sliderWidth: 3,

sliderColor: "rgba(0, 0, 0, .3)",

sliderStyle: "dotted",

backgroundImageSize: "cover",

preloaderColor: "rgba(0, 179, 239, .9)",

autoPlay: 1000,

autoPlayResume: 3000,

autoPlayTick: 2000,

autoPlayLeftRightPx: [30, 50],

autoPlayHintHtml: "<img src=\"images/comparison\_handCursor.png\" style=\"width:40px; height:50px; transform:translate(-13px, 6px) rotate(-25deg) scale(.8)\">",

},

events: {

onSliderClicked: function(Expo, obj) {

Expo.designerAPI.firePixel("ComparisonSlide2\_onSliderClicked", [{"eventName":"intLive", "multi":true}]);

},

onLeftImageClicked: function(Expo, obj) {

Expo.designerAPI.firePixel("ComparisonSlide2\_onLeftImageClicked", [{"eventName":"intLive", "multi":true}]);

},

onRightImageClicked: function(Expo, obj) {

Expo.designerAPI.firePixel("ComparisonSlide2\_onRightImageClicked", [{"eventName":"intLive", "multi":true}]);

}

}

},

/\* End ComparisonSlider Code \*/

## Component Callbacks:

* "**onSliderClick**" (fires when slider is clicked)
* "**onLeftImageClick**" (fires when the left image is clicked)
* "**onRightImageClick**" (fires when the right image is clicked)

## Supported interaction Events and Detail

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **User Action** | **Event Name** | **Default Source**  **All in bound pixel with have a suffix TabTracker id which is defined by user** | **Default Sub Source** | **Parameter in insight pixel** | **Can i change source value in non muse BEFORE Advance Pixel Management Release?** | **Can i change source value in non muse AFTER Advance Pixel Management Release?** | **Can i change source value in Muse BEFORE Advance Pixel Management Release?** | **Can i change source value in Muse AFTER Advance Pixel Management Release?** | **Initiator(function/callback initiating the pixel)** |
| click on left Image | inbound | leftImage | {   componentName: "COMPONENT NAME",   tabName: "HOST TAB LABEL", // **Offers**   tabSequence: "tab{HOST\_TAB\_ID}", // **tab1, tab2, ...**   subSource: "leftImage", } | event=intLive and fr:{source) | NO | NO | YES | YES | onLeftImageClick |
| click on right Image | inbound | rightImage | {   componentName: "COMPONENT NAME",   tabName: "HOST TAB LABEL", // **Offers**   tabSequence: "tab{HOST\_TAB\_ID}", // **tab1, tab2, ...**   subSource: "rightImage", } | event=intLive and fr:{source) | NO | NO | YES | YES | onRightImageClick |
| click on slider | inbound | slider | {   componentName: "COMPONENT NAME",   tabName: "HOST TAB LABEL", // **Offers**   tabSequence: "tab{HOST\_TAB\_ID}", // **tab1, tab2, ...**   subSource: "slider", } | event=intLive and fr:{source) | NO | NO | YES | YES | onSliderClick |
| Video events | all supported video event | video{ID} | {   componentName: "COMPONENT NAME",   tabName: "HOST TAB LABEL", // **Offers**   tabSequence: "tab{HOST\_TAB\_ID}", // **tab1, tab2, ...**   subSource: "video{ID}", // **video1, video2, ...** } | all supported video event and fr:{videoID} | NO | NO | NO | NO | Create Video player Instance |
| replay button clicked | replayLive | replayLive | {   componentName: "COMPONENT NAME",   tabName: "HOST TAB LABEL", // **Offers**   tabSequence: "tab{HOST\_TAB\_ID}", // **tab1, tab2, ...**   subSource: "replayButton", } | event=intLive and fr:{source) | NO | NO | NO | YES | onReplayClick |
| Mute/Unmute button Clicked | muteLive unmuteLive | muteLive unmuteLive | {   componentName: "COMPONENT NAME",   tabName: "HOST TAB LABEL", // **Offers**   tabSequence: "tab{HOST\_TAB\_ID}", // **tab1, tab2, ...**   subSource: "muteButton" | "unmuteButton", } | event=intLive and fr:{videoID) | NO | NO | NO | YES | onMuteUnmuteClick |