Embed Viewer

- Introduction
- Properties
- Functions

Introduction

The Embed Viewer is a React component that encapsulates a viewer to show 3D spatial data.

The Embed Viewer developer manual includes the following sections

Properties :

Definition of the Embed Viewer React Component' Props (Properties) and each prop values and description.

Functions :

Definition of the Embed Viewer React Component' functions that can be called from 4DV Application.

Properties

Name	JS Data Type	Description
projectId	string	The ID of the project
streamingUrl	string	The URL of the streaming server
restUrl	string	The URL of rest server
viewId	string	The ID of the view
viewJSON	JSON Object	A JSON Object containing description of layers
project	JSON Object	The project entities associated to this view
zoomToExtent	Boolean	Show/Hide zoom to extent button
fullscreen	Boolean	Show/Hide full screen button
home	Boolean	Show/Hide Home button
options	Boolean	Show/Hide options button
showControlsInView	Boolean	Show the controls inside/outside of the view
onRegionChange	function	Event Notification is triggered when a change in a region occurs. The event includes the following properties: - voxelfarmPoints: the position of the region in Voxel Farm coordinates. - voxelfarmMinY: min Y in Voxel Farm coordinates. - voxelfarmMaxY: max Y in Voxel Farm coordinates. - worldPoints: the position of the region in World coordinates. - worldMinZ: min Z in world coordinates. - worldMaxZ: max Z in world coordinates.

onLayerClick	function	Event Notification is triggered when clicking on a layer. The event includes the following properties: - eventType: mousedown/mouseup - canvasPosition: the position of the canvas. - layerId: the layer Id where the click occurred. - Position: the position of the click in the screen. - voxelfarmPosition: the position of the layer in Voxel Farm coordinates. - windowPosition: the position of the window. - worldPosition: the position of the layer in world coordinates.
onCameraChange	function	Event Notification is triggered when the camera of the viewer change.
onCameraRotate	function	Event Notification is triggered when the camera of the viewer rotates.
onCrossSectionChange	function	Event Notification is triggered when the cross section of the viewer change.
onBusy	function	Event Notification is triggered when the viewer is busy.
onNewScene	function	Event Notification is triggered when the new scene is loaded.
onError	function	Event Notification is triggered when the viewer is throwing an error.
requestAuthToken	function	Optional Function if authentication is required, it returns the authorization token

Functions

Name	Parameters	Description
fetchProgramInputs	- programId: string - callback: function => callback returning the inputs	Return the current PYTHON programs inputs
refreshView	- selectedViewContainerMeta: ViewMetaData Object - findEntityFn: Function - callback: Function	Update the current view with new MetaData
loadRegion	- regionId: string	Load specific region in the viewer

loadRegionData	- regionData: interface => {
	mode: number (0: None, 1: default type),
	points: [],
	pointsWorld: [],
	minY: number (Float),
	maxY: number (Float),
	minWorldZ: number (Float),
	maxWorldZ: number (Float)
	}
		Load specific region data in the viewer

hideRegion	None	Hide the current loaded region
raycastViewPrimitives	- clientX: number - clientY: number	Ray-cast across all layers primitives in the view
requestFeatureInfo	- featureInfoQueryId: number - cellId: number - layerId: number - worldPoint: [3] - attributes: [] - customAttributes: [] - onComplete: function => callback on complete	Requests feature info to the content server for a given a voxel layer
zoomToExtents	None	Zooms the scene to the camera's extents

zoomToRegion	- regionData: interface => {
	mode: number (0: None, 1: default type),
	points: [],
	pointsWorld: [],
	minY: number (Float),
	maxY: number (Float),
	minWorldZ: number (Float),
	maxWorldZ: number (Float)
	}
		Zooms the region to the camera's extents

zoomIn	None	Zoom in the focus of the camera
zoomOut	None	Zoom out the focus of the camera
captureFrame	None	Capture a frame
frameWorldBounds	- worldBounds: [3]	Zoom a given bounds to the camera's extents
getCameraWorldPosition	None	Return the camera position in world coordinates
getWorldSceneBounds	None	Return the scene bounds in world coordinates
getOrthographicScale	None	Return the orthographic scale of the viewer
getDefaultGradientImage	- gradientId: string	Return the default gradient image used in the view
getCameraViewerPosition	None	Return the current position of the camera
getCameraWorldPosition	None	Return the current world position of the camera
getCameraAzimuthPlunge	None	Return the current camera azimuth plunge
getFocusViewerPosition	None	Return the current focus position of the viewer
getViewerProjection	None	Return the current viewer projection as a string
setOrthographicProjection	- enable: Boolean	Enable/disable orthographic projection

setUserSettings	- UserSettings: interface => {
	fontSize: number
	}
		Sets user settings for the viewer, such as font size for all labels

setLabelVisibility	- showLabels: Boolean	Set visibility for all labels

setVisibility	- layerId: string - enable: Boolean	Enable or disable visibility for a given meta component
setTransparency	- layerId: string - opacity: number (Float)	Set opacity for a given meta component
setDiffuseTransparency	- layerId: string - diffuseOpacity: number (Float)	Set diffuse map opacity for a given meta component
setDiffuseBlendMode	- layerId: string - diffuseBlendMode: number (Float)	Set diffuse blend mode for a given meta component

setDrillholeSettings	- layerId: string
	- drillholesSettings: interface => {
	showRoots: Boolean,
	showRootLabels: Boolean,
	showLines: Boolean,
	showTubes: Boolean,
	showColorAttributeLabels: Boolean,
	attributesTextSize: number,
	leftColorAttributeIndices: number[],
	rightColorAttributeIndices: number[],
	colorAttributeRawData: string
	}
		Set drill hole settings for a given meta vector component, such as collar visibility, collar label id visibility, holes rendered as lines or tubes, holes attribute label filter

setGeoChemSettings	- layerId: string
	- geoCheomSettings: interface => {
	showPolygon: Boolean,
	showInside: Boolean,
	polygonPoints: interface => {
	vfPoints: number[3],
	wPoints: number[3]
	},
	showZ: Boolean,
	geoShowLabels: Boolean,
	geoLabelSize: number,
	geoShowElevation: Boolean,
	planeValue: number,
	attributes: interface => {
	top: {
	attribute: string,
	color: [3],
	index: number
	},
	bottom: {
	attribute: string,
	color: [3],
	index: number
	}
	left: {
	attribute: string,
	color: [3],
	index: number
	}
	right: {
	attribute: string,
	color: [3],
	index: number
	}
	},
	geoSize: number,
	geoShape: number,
	}
		Set Geo chem settings for a given meta vector component, such as attribute label visibility, attribute label color, size and shape

setAttributeQuery	- layerId: string
	- attributeQuery: interface => {
	type: number,
	attributeQueryBuilderValue: string,
	builderAttributeFilters: [],
	builderAttributeRawData: string
	}
		Sets attribute query for a given meta component

setLabelQuery	- layerId: string
	
	- labelQuery: interface => {
	type: number,
	builderLabelFilters: [],
	builderLabelRawData: string,
	showPolygon: Boolean,
	showInside: Boolean,
	polygonPoints: interface => {
	vfPoints: number[3],
	wPoints: number[3]
	}
	}
		Sets label query for a given meta component

setBlockModelLimits	- layerId: string
	- BlockModelLimits: interface => {
	indexFilter: number,
	regionType: 0,
	bounds: interface => {
	min: {
	x: number (Float),
	y: number (Float),
	z: number (Float),
	},
	max: {
	x: number (Float),
	y: number (Float),
	z: number (Float),
	}
	},
	enable: {
	x: Boolean,
	y: Boolean,
	z: Boolean
	},
	}
		Sets block model limits for a given meta block tree component

setColorLegend	- layerId: string
	- colorLegend: interface => {
	type: number (1: Static Color, 2: Data Driven),
	dataDrivenColorAttribute: string,
	dataDrivenColorAttributeIndex: number,
	dataDrivenColorMinValue: number (Float),
	dataDrivenColorMaxValue: number (Float),
	dataDrivenColorStepSize: number (Float),
	dataDrivenUseAttributeGradient: Boolean,
	dataDrivenAttributeGradient: [],
	staticColorDoubleSided: Boolean,
	staticColorFrontSideRGB: [3],
	staticColorBackSideRGB: [3],
	gradientId: string,
	dataDrivenColorDistribution: []
	}
		Sets color legend for a given meta component

setCollarLegend	- layerId: string
	- collarLegend: interface => {
	type: number, (1: default type)
	staticColor: [3],
	staticRadius: number,
	textSize: number,
	}
		Sets collar legend for a given meta vector component

setFatLineThickness	- layerId: string - FlatlineThickness: number	Sets fat line thickness for a given meta vector component

setTubeDiameterLegend	- layerId: string
	- tubeDiameterLegend: interface => {
	scaleMode: number,
	cylinderRadius: number (Float),
	attribute: string,
	attributeIndex: number,
	rangeMin: number (Float),
	rangeMax: number (Float),
	rangeStep: number (Float),
	radiusMin: number (Float),
	radiusMax: number (Float)
	}
		Sets tube diameter legend for a given meta vector component

setTraceSettings	- layerId: string
	- traceSettings: interface => {
	traceThickness: number (Float)
	}
		Sets fat line thickness for a given meta vector component using trace settings

setMeshPropertiesAt	- layerId: string - meshIndex: number - meshVisible: Boolean - meshOpacity: number (Float) - meshUseParentColorLegend: Boolean - meshStaticColor: [3],	Sets mesh properties at a specific mesh index for a given meta surface component

setMeshProperties	- layerId: string
	- meshProperties: interface => {
	parentLayer: string,
	meshIndex: number,
	meshVisible: Boolean,
	meshOpacity: number (Float),
	meshUseParentColorLegend: Boolean,
	meshStaticColor: [3],
	meshStaticBackColor: [3]
	}
		Sets mesh properties for all meshes for a given meta surface component

setImageProperties	- layerId: string
	- imageProperties: interface => {
	visible: Boolean,
	transparency: number (Float),
	contrast: number (Float),
	brightness: number (Float),
	gamma: number (Float),
	blendMode: number,
	blendMakeColorTransparent: Boolean,
	blendInvertSelection: Boolean,
	blendTransparentColor: [3],
	blendSelectionTolerance: number (Float),
	blendOutputTransparency: number (Float),
	blendOutputColor: [3],
	bands: [4]
	}
		Sets image properties for a given meta block tree/surface/vector component

setImagePropertiesAt	- layerId: string - imageId: string - imageVisible: Boolean - imageOpacity: number (Float) - imageContrast: number (Float) - imageBrightness: number (Float) - imageGamma: number (Float) - imageBlendMode: number - imageBlendMakeColorTransparent: Boolean - imageBlendTransparentColor: [3] - imageBlendSelectionTolerance: number (Float) - imageBlendOutputColor: [3], - imageBlendInvertSelection: Boolean - imageBands: [3]	Sets the image properties at a specific image index for a given meta block tree/surface/vector component
setNavigationMode	- mode: number	Set the navigation mode of the camera - mode = 0 => use rotation as primary mouse input mode = 1 => use pane as primary mouse input
setViewMetaData	- compositedMeta: ViewMetaData Object	
setExtendedMeta	- index: number - extendedMeta: JSON Object - viewId: number (optional)	Set extended meta to the current meta data Object Note: The ViewId is optional parameter by default it has a value of 0 to indicate first view
applyZoomDelta	- deltaDistance: number	Apply a zoom given a delta value
enableCrossSectionFading	- enableFading: Boolean	Enable or Disables Cross Section Fading
alignCameraToDirection	- direction: number	Align the camera forward to the given direction
startSegment	- segmentType: number - keepSegmentAlive: Boolean - onCompleteSegment: function => callback on complete of the segment - onCancelSegment: function => callback on canceling of the segment	Start a segment procedure

stopSegment	- segmentId: number	Stop a segment procedure
startOrthoPoly	- orthoPolyType: number - keepOrthoPolyAlive: Boolean - onCompleteOrthoPoly: function => callback on complete of the Ortho poly - onCancelOrthoPoly: function => callback on cancel of the Ortho poly	Start an orthographic segment list procedure
stopOrthoPoly	- orthoPolyId: number	Stop an orthographic segment list procedure
createDot	- dotId: number - worldPoint: [3]	Create a dot in the scene
destroyDot	- dotId: number	Destroy an existing dot
createViewMetaData	None	Create ViewMetaData Object
addViewer	- projectId: string - viewId: string - layers: Array - findEntityFn: function => callback to find the entity from within the caller of the function - isBaseMap: Boolean - currentProject: JSON Object	Add viewer to the multiCRS Viewer pool
viewerExists	- layer: JSON Object	Check if a layer has a viewer associated with it.
setViewerProjection	- projectId: string - entityId: string	Set the current projection of the view based on the selected entity passed to the function as the second parameter
updateCrossSectionEnableHighlight	- enableHighlight:Boolean	Update the cross section to enable/disable highlight

updateCrossSectionPresetType	- presetType: number 0 : No Clipping; Hide +X +Y +Z +W 1 : Clip Outside; Show +X +Y +Z; Hide +W 2 : Clip Outside; Show +Z; Hide +X +Y +W 3 : Clip Outside; Show +X; Hide +Y +Z +W 4 : Clip Outside; Show +Y; Hide +X +Z +W 5 : Clip Normal; Show +Z; Hide +X +Y +W 6 : Clip Normal; Show -X; Hide +Y +Z +W 7 : Clip Normal; Show -Y; Hide +X +Z +W 8 : Clip Normal; Show -Z; Hide +X +Y +W 9 : Clip Normal; Show +X; Hide +Y +Z +W 10 : Clip Normal; Show +Y; Hide +X +Z +W 11 : Clip Normal; Show +W; Hide +X +Y +Z 12 : Clip Normal; Show -W; Hide +X +Y +Z 13 : Clip Outside; Show +W; Hide +X +Y +Z	Control the Cross section of the view by presets