Map Panel Module

User Guide

1 800 470 2302
www.kymerasystems.com

Table of Contents

Introduction

The Kymera Map Panel module allows users to embed maps in their Ignition projects with the ability to add custom map markers with notes or embedded expressions, as well as zoom in and out and use a variety of map providers, or combine multiple information layers from different providers to create composite map views.

Usage

After installing the module, you will find the Kymera Map Panel component on the Display tab of the Component Palette. Simply drag the component on to a window to begin. Most desired functionality is handled in the properties and scripts defined below, but for more advanced users it is worth noting that the map panel extends Ignition's AbstractVisionPanel, and both the map points and the map shapes extend Java's JComponent.

Initially, the map panel has no defined map points. To add map points to the map panel, add them to the Data Points dataset. Map points will be drawn onto the map panel in the opposite order as they are defined in the dataset, the top row of the dataset being the point drawn on top of any others. The dataset has the following columns:

String Name	The id of the map point. Commonly the location name of the map point.
Double Latitude	The latitude of the map point location.
Double Longitude	The longitude of the map point location.
String Label Expression	An expression to display text on the label of the map point.
String Icon Expression	An expression to determine file path to the icon graphic.
String Hover Expression	An expression to determine file path to the icon graphic used when the point is hovered over.

Initially, the map panel has no defined map shapes. To add map shapes to the map panel, add them to the Data Shapes dataset. Map shapes will be drawn onto the map panel in the opposite order as they are defined in the dataset, the top row of the dataset being the shape drawn on top of any others. The dataset has the following columns:

String Name	The id of the map shape. Commonly the location name of the map shape.
String Latitude	A comma separated list of Latitudes for defining the shape.
String Longitude	A comma separated list of Longitudes for defining the shape.
String Stroke Color	Definition of color for the shape's border.
String Fill Color	Definition of color for the shape's interior.
Float Opacity	How opaque the shape should be drawn. Value clamped between 0…1
String Draw Mode	Parsed to the DrawMode enum. Defines how the shape is drawn. **PolyFilled** is a closed polygon with both border/fill. **PolyBorder** is just the border, using the stroke color. **Line** is similar, except it won't force a closed polygon.
String Label Expression	An expression to display text on the label of the map shape.

To zoom in on the map panel, scroll up on the mouse wheel or double click the left mouse button.

To zoom out on the map panel, scroll down on the mouse wheel or double click the right mouse button.

To move the map around, click and hold the mouse while moving around.

Properties

Data

Max Tile Cache Size

The maximum amount of cached tiles.
Scripting name `maxCacheSize`
Data type int

Repoll Delay

The milliseconds to wait before fetching a tile that failed to draw.
A value less than 1 indicates no repoll should be performed.
A repoll will only execute 5 times at most.
Scripting name `repollDelay`
Data type int

Data Points

The map points to draw on the map.
Scripting name `dataPoints`
Data type Dataset

Data Shapes

The map shapes to draw on the map.
Scripting name `dataShapes`
Data type Dataset

Appearance

Animated

Use animation to draw the maps when zooming?
Scripting name `animated`
Data type boolean

Bilinear Filtering

Whether or not to apply Bilinear Filtering to draw the map.
Scripting name `filtered`
Data type boolean

Zoom Level

The zoom level of the map panel, 1 being fully zoomed out.
Scripting name `zoom`
Data type int

Control Panel Visible

Whether or not the panel with pan and zoom buttons is visible.
Scripting name `controlVisible`
Data type boolean

Overlay Panel Visible

Whether or not the panel with debug information is visible.
Scripting name `overlayVisible`
Data type boolean

Debug

Debug

Draws borders around the individual map panel tiles.
Scripting name `debug`
Data type boolean

Hidden

GPS Location

The Latitude/Longitude values at the center of the map panel.
Scripting name `gpsLocation`
Data type Point2D.Double

Min Lat	Latitude on the left edge of the map screen. Scripting name `minLat` Data type double
Max Lat	Latitude on the right edge of the map screen Scripting name `maxLat` Data type double
Min Lon	Longitude on the bottom edge of the map screen Scripting name `minLon` Data type double
Max Lon	Longitude on the top edge of the map screen Scripting name `maxLon` Data type double
Map Position	Position of the map on the current zoom level in pixels Scripting name `mapPosition` Data type Point

Scripting : Map Panel

The following scripting functions are available on the map panel.

getTileServer().initializeTileServer()

Description

Clears all of the providers from the tile server. Rendering the map “blank” after it redraws.

Syntax

```
getTileServer().initializeTileServer()
```

Parameters

none

Returns

nothing

Scope

Client

Examples

The following snippet clears the existing layers on a map.

```
mapPanel = event.source.parent.getComponent('MapPanel')
mapPanel.getTileServer().initializeTileServer()
```

getTileServer().addLayer()

Description

Adds a new layer to the map panel, supplying map images from one of the supported providers.

Syntax

```
getTileServer().addLayer(opacity, url, type, accessToken)
```

Parameters

double `opacity` – Value between 0 and 1. Controls how transparent the layer is.
string `url` – Usage depends on provider, defines what information should be retrieved from provider.
string `type` – Which provider should this layer collect data from. Can be any of the following values: arcgis, google, osm, webdev
string `accessToken` – Optional. Access token used by some providers to grab data.

Returns

nothing

Scope

Client

Examples

The following snippet is an example of using Google

```
mapPanel = event.source.parent.getComponent('MapPanel')
mapPanel.getTileServer().initializeTileServer()

mapPanel.getTileServer().addLayer(1.0, "m", "google") #Roads
mapPanel.getTileServer().addLayer(1.0, "s", 'google") #Earth Mode
mapPanel.getTileServer().addLayer(1.0, "y", "google") #Earth Mode
mapPanel.getTileServer().addLayer(1.0, "p", "google") #Topography/Heightmap
mapPanel.getTileServer().addLayer(1.0, "t", "google") #Topography/Heightmap

mapPanel.forceRedraw() #must call or panel won't update until interaction
```

The following snippet is an example of using OSM

```
mapPanel = event.source.parent.getComponent('MapPanel')
mapPanel.getTileServer().initializeTileServer()

mapPanel.getTileServer().addLayer(1.0, "http://otile1.mqcdn.com/tiles/1.0.0/map/", "osm")

mapPanel.forceRedraw() #must call or panel won't update until interaction
```

The following snippet is an example of using ArcGis

```
mapPanel = event.source.parent.getComponent('MapPanel')
mapPanel.getTileServer().initializeTileServer()

mapPanel.getTileServer().addLayer(1.0, "", "arcgis")
#ArcGis defaults to the URI:
#http://server.arcgisonline.com/arcgis/rest/services/World_Imagery/MapServer/tile/

mapPanel.forceRedraw() #must call or panel won't update until interaction
```

The following snippet is an example of using Mapbox. Please note that it requires an access token, obtainable from the Mapbox website (www.mapbox.com).

```
mapPanel = event.source.parent.getComponent('MapPanel')
mapPanel.getTileServer().initializeTileServer()

mapPanel.getTileServer().addLayer(1.0, "mapbox.streets", "mapbox", "accessToken")
#The URI argument is the map-ID in mapbox, defining the style of the map. MapBox provides
#a number of common maps, and you can create custom styles associated with your account
#Your access token is available at https://www.mapbox.com/account/apps when logged in
```

For support with implementing the 'WebDev' provider, please contact Kymera Technical Support.

getRepollDelay()

Description

Gets the number of milliseconds to wait before trying to draw a failed map tile again. A value less than 1 indicates no repoll should be performed.

Syntax

```
getRepollDelay()
```

Parameters

none

Returns

int – The number of milliseconds to wait before trying to draw a failed map tile again.

Scope

Client

Examples

```
mapPanel = event.source.parent.getComponent('MapPanel')
delay = mapPanel.getRepollDelay()
```

setRepollDelay()

Description

Sets the number of milliseconds to wait before trying to draw a failed map tile again. A value less than 1 indicates no repoll should be performed.

Syntax

```
setRepollDelay(delay)
```

Parameters

int `delay`– The number of milliseconds to wait before trying to draw a failed map tile again.

Returns

nothing

Scope

Client

Examples

The following snippet increases the repoll delay by one second if the delay is less than a minute.

```
mapPanel = event.source.parent.getComponent('MapPanel')
delay = mapPanel.getRepollDelay()
if delay < 600000:
	mapPanel.setRepollDelay(delay + 1000)
```

getDataPoints()

Description

Gets the datapoints representing the map markers on the Map Panel from the Map Panel Data Points property.

Syntax

```
getDataPoints()
```

Parameters

none

Returns

Dataset – The data points representing the map markers.

Scope

Client

Examples

```
mapPanel = event.source.parent.getComponent('MapPanel')
mapPoints = mapPanel.getDataPoints()
```

setDataPoints()

Description

Sets the map points to display on the map. The dataset must have the following columns:

- Name
- Latitude
- Longitude
- Label Expression
- Icon Expression
- Hover Expression

Syntax

```
setDataPoints(dataPoints)
```

Parameters

Dataset `dataPoints` – A dataset of map points to display on the map.

Returns

nothing

Scope

Client

Examples

The following snippet adds a pink pin to the map for the City of Edmonton, Alberta. It shows a shadow when hovered over, and displays the name “Edmonton” when clicked.

```
mapPanel = event.source.parent.getComponent('MapPanel')
headers = ["Name", "Latitude", "Longitude", "Label Expression", "Icon Expression",
        "Hover Expression"]
data = []
data.append(["Edmonton", 53.55, -113.49, "Edmonton", "MapPanel/pink.png",
        "MapPanel/shadow-pink.png"])
mapPoints = system.dataset.toDataSet(headers, data)
mapPanel.setDataPoints(mapPoints)
```

getDataShapes()

Description

Gets the definitions of the shapes that will be drawn on the MapPanel.

Syntax

```
getDataShapes()
```

Parameters

none

Returns

Dataset – The data points representing the map shapes.

Scope

Client

Examples

```
mapPanel = event.source.parent.getComponent('MapPanel')
mapShapes = mapPanel.getDataShapes()
```

setDataShapes()

Description

Sets the map shapes to display on the map. The dataset must have the following columns:

- Name
- Latitude
- Longitude
- Stroke Color
- Fill Color
- Opacity
- Draw Mode
- Label Expression

Syntax

```
setDataShapes(dataShapes)
```

Parameters

Dataset `dataShapes` – A dataset of map shapes to display on the map.

Returns

nothing

Scope

Client

Examples

The following snippet adds a red square in central Alberta, containing Edmonton.

```
mapPanel = event.source.parent.getComponent('MapPanel')
headers = ["Name", "Latitude", "Longitude", "Stroke Color", "Fill Color",
        "Opacity", "Draw Mode", "Label Expression"]
data = []
data.append(["Edmonton Area", "52, 54, 54, 52", "-113, -113, -115, -115", "#000000",
        "#ff0000", 0.5, "PolyFilled", "Edmonton Area")
mapShapes = system.dataset.toDataSet(headers, data)
mapPanel.setDataShapes(mapShapes)
```

isAnimated()

Description

Returns the status of the Animated property of the Map Panel.

Syntax

```
isAnimated()
```

Parameters

none

Returns

boolean – True if the Animated property is checked.

Scope

Client

Examples

```
mapPanel = event.source.parent.getComponent('MapPanel')
animated = mapPanel.isAnimated()
```

setAnimated()

Description

Sets or clears the Animated property of the Map Panel.

Syntax

```
setAnimated(animated)
```

Parameters

boolean `animated` – The state of the Animated property to set.

Returns

nothing

Scope

Client

Examples

The following snippet enables animation when drawing the maps if it is turned off.

```
mapPanel = event.source.parent.getComponent('MapPanel')
animated = mapPanel.isAnimated()
if animated == 0:
	mapPanel.setAnimated(1)
```

isFiltered()

Description

Returns the status of the Bilinear Filtering property of the Map Panel.

Syntax

```
isFiltered()
```

Parameters

none

Returns

boolean – True if the Bilinear Filtering property is checked.

Scope

Client

Examples

```
mapPanel = event.source.parent.getComponent('MapPanel')
filtered = mapPanel.isFiltered()
```

setFiltered()

Description

Sets or clears the Bilinear Filtering property of the Map Panel.

Syntax

```
setFiltered(filtered)
```

Parameters

boolean `filtered` – The state of the Bilinear Filtering property to set.

Returns

nothing

Scope

Client

Examples

The following snippet enables Bilinear Filtering when drawing the maps if it is turned off.

```
mapPanel = event.source.parent.getComponent('MapPanel')
filtered = mapPanel.isFiltered()
if filtered == 0:
	mapPanel.setFiltered(1)
```

setGpsLocation()

Description

Centers the map around a set of latitude and longitude coordinates.

Syntax

setGpsLocation(lat, lon**)**

Parameters

double `lat` – The latitude of the map coordinate to center the map on.
double `lon` – The longitude of the map coordinate to center the map on.

Returns

nothing

Scope

Client

setGpsLocation(gpsLocation**)**

Parameters

Point2D.Double `gpsLocation` – A point containing the latitude and longitude of the map coordinate to center the map on.

Returns

nothing

Scope

Client

Examples

The following property change code executes on a table containing cities with their latitudes and longitudes to center the map on the city selected and zoom in.

```
if event.propertyName == "selectedRow" and event.newValue != -1:
	lat = event.source.data.getValueAt(event.newValue,"Latitude")
	lon = event.source.data.getValueAt(event.newValue,"Longitude")
	event.source.parent.getComponent('MapPanel').setZoom(9)
	event.source.parent.getComponent('MapPanel').setGpsLocation(lat,lon)
```

Using setGpsLocation(gpsLocation)

```
if event.propertyName == "selectedRow" and event.newValue != -1:
	from java.awt.geom import Point2D
	lat = event.source.data.getValueAt(event.newValue,"Latitude")
	lon = event.source.data.getValueAt(event.newValue,"Longitude")
	centerPoint = Point2D.Double(lat,lon)
	event.source.parent.getComponent('MapPanel').setZoom(9)
	event.source.parent.getComponent('MapPanel').setGpsLocation(centerPoint)
```

getGpsLocation()

Description

Returns the GPS coordinates at the center of the map panel.

Syntax

```
getGpsLocation()
```

Parameters

none

Returns

Point2D.Double – An x and y coordinate representing the center of the map panel.

Scope

Client

Examples

The following code snippet gets the x and y coordinate of the center of the map panel.

```
mapPanel = event.source.parent.getComponent('MapPanel')
centerPoint = mapPanel.getGpsLocation()
lat = centerPoint.x
lon = enterPoint.y
```

setAtomicGpsLocation()

Description

Centers the map around a set of latitude and longitude coordinates.

Syntax

```
setAtomicGpsLocation(atomicString)
```

Parameters

string `atomicString` – The latitude/longitude pair formatted as “lat, lon”.

Returns

nothing

Scope

Client

Examples

The code below will assign the location of the map panel to 40, -110.

```
mapPanel = event.source.parent.getComponent('MapPanel')
mapPanel.setAtomicGpsLocation(’40, -110’)
```

getAtomicGpsLocation()

Description

Returns the GPS coordinates at the center of the map panel as a single string.

Syntax

```
getAtomicGpsLocation()
```

Parameters

none

Returns

String – The latitude/longitude of the map panel formatted as “lat, lon”.

Scope

Client

Examples

The following code snippet prints the GPS location of the map panel.

```
mapPanel = event.source.parent.getComponent('MapPanel')
print mapPanel.getAtomicGpsLocation()
```

getZoom()

Description

Returns the current zoom level of the map.

Syntax

```
getZoom()
```

Parameters

none

Returns

int – The current zoom level.

Scope

Client

Examples

```
mapPanel = event.source.parent.getComponent('MapPanel')
print mapPanel.getZoom()
```

setZoom()

Description

Sets the current zoom level of the map.

Syntax

```
setZoom(zoom)
```

Parameters

int `zoom` – The zoom level to set.

Returns

nothing

Scope

Client

Examples

The following snippet zooms the map out as far as possible.

```
mapPanel = event.source.parent.getComponent('MapPanel')
mapPanel.setZoom(1)
```

getMapBounds()

Description

Returns the latitude and longitude of the top left corner, plus the width and height of the visible map.

Syntax

```
getMapBounds()
```

Parameters

none

Returns

Rectangle2D – A rectangle containing an x and y coordinate, plus the height (h) and width(w).

Scope

Client

Examples

The following snippet gets the location, height and width of the visible map.

```
mapPanel = event.source.parent.getComponent('MapPanel')
mapBounds = mapPanel.getMapBounds()
latitude = mapBounds.x
longitude = mapBounds.y
width = mapBounds.w
height = mapBounds.h
```

getMapPoints()

Description

Returns a collection of the map points.

Syntax

```
getMapPoints()
```

Parameters

none

Returns

Collection<MapPoint> – A collection of the map points.

Scope

Client

Examples

The following snippet prints the name of each map point.

```
mapPanel = event.source.parent.getComponent('MapPanel')
mapPoints = mapPanel.getMapPoints()
for point in mapPoints:
        print point.getName()
```

getMapShapes()

Description

Returns a collection of the map shapes.

Syntax

```
getMapShapes()
```

Parameters

none

Returns

Collection<MapShape> – A collection of the map shapes.

Scope

Client

Examples

The following snippet prints the name of each map shape.

```
mapPanel = event.source.parent.getComponent('MapPanel')
mapShapes = mapPanel.getMapShapes()
for shape in mapShapes:
        print shape.getName()
```

getMode()

Description

Returns the current mode of the map.

Syntax

```
getMode()
```

Parameters

none

Returns

int – The current mode. 0 corresponds to pan, 1 corresponds to draw.

Scope

Client

Examples

```
mapPanel = event.source.parent.getComponent('MapPanel')
print mapPanel.getMode()
```

setMode()

Description

Sets the current mode of the map panel.

Syntax

```
setMode(mode)
```

Parameters

int `mode` – The new mode. 0 corresponds to pan, and the map functions as one would expect. 1 corresponds with draw mode, and enables clicking to draw a polygon.

Returns

nothing

Scope

Client

Examples

The following snippet put's the map panel into draw mode.

```
mapPanel = event.source.parent.getComponent('MapPanel')
mapPanel.setMode(1)
```

getMouseEventConsumedDefault()

Description

Returns whether or not mouse events are currently being passed through to the map panel on unimplemented extension functions. When true, events are consumed, and not passed through, on false they are not consumed, and are passed through. Defaults to false.

Syntax

```
getMouseEventConsumedDefault()
```

Parameters

none

Returns

boolean – Whether or not events should be consumed.

Scope

Client

Examples

```
mapPanel = event.source.parent.getComponent('MapPanel')
print mapPanel.getMouseEventConsumedDefault()
```

setMouseEventConsumedDefault()

Description

Sets whether or not mouse events should be consumed on unimplemented extension functions.

Syntax

```
setMouseEventConsumedDefault(mouseEventConsumedDefualt)
```

Parameters

boolean `mouseEventConsumedDefault` – False disables consumption of events, True enables consumption of events.

Returns

nothing

Scope

Client

Examples

The following snippet enables default consumption of mouse events.

```
mapPanel = event.source.parent.getComponent('MapPanel')
mapPanel.setMouseEventConsumedDefault(True)
```

initDrawShape()

Description

Initializes and/or resets the current working shape. Good for “clearing” in progress work.

Syntax

```
initDrawShape()
```

Parameters

none

Returns

nothing

Scope

Client

Examples

The following snippet resets the shape currently being drawn.

```
mapPanel = event.source.parent.getComponent('MapPanel')
mapPanel.initDrawShape()
```

commitDrawShape()

Description

Takes the current working shape, adds it to the dataShapes dataset, and clears the working shape. If the working shape shares a name with an existing shape, the existing shape will be overwritten.

Syntax

```
commitDrawShape()
```

Parameters

nothing

Returns

nothing

Scope

Client

Examples

The following snippet will save the working shape with a name grabbed from a text field.

```
mapPanel = event.source.parent.getComponent('MapPanel')
textField = event.source.parent.getComponent('Shape Name')
mapPanel.getWorkingShape().setName(textField.text)
mapPanel.commitDrawShape()
```

getWorkingShape()

Description

Returns the shape that is currently being drawn. Available even when the map isn't in draw mode.

Syntax

```
getWorkingShape()
```

Parameters

none

Returns

MapShape – The icon path of the map point's hover icon.

Scope

Client

Examples

Sets the color of the working shape to red.

```
mapPanel = event.source.parent.getComponent('MapPanel')
shape = mapPanel.getWorkingShape()
shape.setFillColorString('#FF0000')
```

Scripting : Map Panel Extension Functions

The map panel has several extensible functions available to add some additional handling of functionality.

onPointAdded()

Description

Invoked when a new point is added to the map panel.

Syntax

```
onPointAdded(self, point)
```

Parameters

Component `self` – Refers to the component that is invoking this function.
MapPoint `point` – The point that was created.

Scope

Client

Examples

The following snippet in an onPointAdded() function will make the point's label visible.

```
ooint.setLabelVisible(True)
```

onPointRemoved()

Description

Invoked when a map point is removed from the panel.

Syntax

```
onPointRemoved(self, point)
```

Parameters

Component `self` – Refers to the component that is invoking this function.
MapPoint `point` – The point that was removed.

Scope

Client

Examples

The following snippet in an onPointRemoved() function will hide the point's label.

```
point.setLabelVisible(False)
```

onPointClicked()

Description

Invoked when a map point is clicked on.

Syntax

```
onPointClicked(self, event, point)
```

Parameters

Component `self` – Refers to the component that is invoking this function.
Event `event` – The mouse event relative to the point.
MapPoint `point` – The point that was clicked.

Returns

boolean – Whether or not the event was consumed. If it wasn't consumed the click will be passed through to the map panel.

Scope

Client

Examples

The following snippet in an onPointClicked() function will print the point's latitude and longitude when clicked.

```
print point.getLat()
print point.getLon()
return False
```

onPointPressed()

Description

Invoked when the mouse button is pressed down on the point.

Syntax

```
onPointPressed(self, event, point)
```

Parameters

Component `self` – Refers to the component that is invoking this function.
Event `event` – The mouse event relative to the point.
MapPoint `point` – The point that was pressed.

Returns

boolean – Whether or not the event was consumed. If it wasn't consumed the press will be passed through to the map panel.

Scope

Client

Examples

The following snippet in an onPointPressed() function will show the label when pressed

```
point.setLabelVisible(True)
return False
```

onPointReleased()

Description

Invoked when the mouse button is released over a point.

Syntax

```
onPointReleased(self, event, point)
```

Parameters

Component `self` – Refers to the component that is invoking this function.
Event `event` – The mouse event relative to the point.
MapPoint `point` – The point that was released.

Returns

boolean – Whether or not the event was consumed. If it wasn't consumed the release will be passed through to the map panel.

Scope

Client

Examples

The following snippet in an onPointReleased() function will hide the label when the mouse is released over the point.

```
point.setLabelVisible(False)
return False
```

onPointEntered()

Description

Invoked when the mouse cursor begins hovering over a point.

Syntax

```
onPointEntered(self, event, point)
```

Parameters

Component `self` – Refers to the component that is invoking this function.
Event `event` – The mouse event relative to the point.
MapPoint `point` – The point that was entered.

Scope

Client

Examples

The following snippet in an onPointEntered() function will change the point's icon every time it is hovered over. Meant to be used in tandem with onPointExited().

```
point.setIconExpression("MapPanel/green.png")
```

onPointExited()

Description

Invoked when the mouse cursor stops hovering over a point.

Syntax

```
onPointExited(self, event, point)
```

Parameters

Component `self` – Refers to the component that is invoking this function.
Event `event` – The mouse event relative to the point.
MapPoint `point` – The point that was exited.

Scope

Client

Examples

The following snippet in an onPointExited() function will change the point's icon every time it is no longer hovered over. Meant to be used in tandem with onPointEntered().

```
point.setIconExpression("MapPanel/blue.png")
```

onShapeAdded()

Description

Invoked when a new shape is added to the map panel.

Syntax

```
onShapeAdded(self, shape)
```

Parameters

Component `self` – Refers to the component that is invoking this function.
MapShape `shape` – The shape that was created.

Scope

Client

Examples

The following snippet in an onShapeAdded() function will make the shape's label visible.

```
shape.setLabelVisible(True)
```

onShapeRemoved()

Description

Invoked when a map shape is removed from the panel.

Syntax

```
onShapeRemoved(self, shape)
```

Parameters

Component `self` – Refers to the component that is invoking this function.
MapShape `shape` – The shape that was removed.

Scope

Client

Examples

The following snippet in an onShapeRemoved() function will hide the shape's label.

```
shape.setLabelVisible(False)
```

onShapeClicked()

Description

Invoked when a map shape is clicked on.

Syntax

```
onShapeClicked(self, event, shape)
```

Parameters

Component `self` – Refers to the component that is invoking this function.
Event `event` – The mouse event relative to the shape.
MapShape `shape` – The shape that was clicked.

Returns

boolean – Whether or not the event was consumed. If it wasn't consumed the click will be passed through to the map panel.

Scope

Client

Examples

The following snippet in an onShapeClicked() function will print the shape's name.

```
print shape.getName()
return False
```

onShapePressed()

Description

Invoked when the mouse button is pressed down on the shape.

Syntax

```
onShapePressed(self, event, shape)
```

Parameters

Component `self` – Refers to the component that is invoking this function.
Event `event` – The mouse event relative to the shape.
MapShape `shape` – The shape that was pressed.

Returns

boolean – Whether or not the event was consumed. If it wasn't consumed the press will be passed through to the map panel.

Scope

Client

Examples

The following snippet in an onShapePressed() function will show the label when pressed

```
shape.setLabelVisible(True)
return False
```

onShapeReleased()

Description

Invoked when the mouse button is released over a shape.

Syntax

```
onShapeReleased(self, event, shape)
```

Parameters

Component `self` – Refers to the component that is invoking this function.
Event `event` – The mouse event relative to the shape.
MapShape `shape` – The shape that was released.

Returns

boolean – Whether or not the event was consumed. If it wasn't consumed the release will be passed through to the map panel.

Scope

Client

Examples

The following snippet in an onShapeReleased() function will hide the label when the mouse is released over the point.

```
shape.setLabelVisible(False)
return False
```

onShapeEntered()

Description

Invoked when the mouse cursor begins hovering over a shape.

Syntax

```
onShapeEntered(self, event, shape)
```

Parameters

Component `self` – Refers to the component that is invoking this function.
Event `event` – The mouse event relative to the shape.
MapShape `shape` – The point that was entered.

Scope

Client

Examples

The following snippet in an onShapeEntered() function will show the label whenever the icon is hovered over.

```
shape.setLabelVisible(True)
```

onShapeExited()

Description

Invoked when the mouse cursor stops hovering over a shape.

Syntax

```
onShapeExited(self, event, shape)
```

Parameters

Component `self` – Refers to the component that is invoking this function.
Event `event` – The mouse event relative to the shape.
MapShape `shape` – The shape being exited.

Scope

Client

Examples

The following snippet in an onShapeExited() function will hide the shape's label.

```
shape.setLabelVisible(False)
```

onCommitShape()

Description

Invoked when a shape created via the draw mode is committed and added to the dataset.

Syntax

```
onCommitShape(self, shape)
```

Parameters

Component `self` – Refers to the component that is invoking this function.
MapShape `shape` – The shape that is being committed.

Scope

Client

Examples

The following snippet in an onCommitShape() function will set the shape's name.

```
shape.setName('New Shape Name')
```

Scripting : Map Points

The following scripting functions are available on the map panel to interact with map points.

getIconExpression()

Description

Returns the expression used to determine the icon path used by the current point.

Syntax

```
getIconExpression()
```

Parameters

none

Returns

String – The expression in a string format.

Scope

Client

Examples

The following snippet in an onPointClicked event on the map panel displays the icon expression of the point when clicked on.

```
print point.getIconExpresssion()
```

setIconExpression()

Description

Sets the expression used to determine the icon used by the current map point.

Syntax

```
setIconExpression(expression)
```

Parameters

String `expression` – The expression to determine the path to the map point icon.

Returns

nothing

Scope

Client

Examples

The following snippet changes the map point icon to a pink map marker.

```
point.setIconPath('MapPanel/pink.png')
```

getHoverExpression()

Description

Returns the expression used to determine the icon used by the current map point when the mouse hovers over it.

Syntax

```
getHoverExpression()
```

Parameters

none

Returns

String – The icon path of the map point's hover icon.

Scope

Client

Examples

The following snippet in an onPointClicked event on the map panel displays the hover icon expression of the point when clicked on.

```
print point.getHoverExpression()
```

setHoverExpression()

Description

Sets the expression used to determine the icon path used by the current map point when the mouse hovers over it.

Syntax

```
setHoverExpression(expression)
```

Parameters

String `expression` – The new expression to determine the path to the icon to set on the map point when the mouse hovers over it.

Returns

nothing

Scope

Client

Examples

The following snippet changes the map point hover icon to a pink map marker with a shadow.

```
point.setHoverExpression('MapPanel/shadow-pink.png')
```

getName()

Description

Returns the name of the map point.

Syntax

```
getName()
```

Parameters

none

Returns

String – The name of the map point.

Scope

Client

Examples

The following snippet in a mouseClicked event on the map panel displays the name of the map point when clicked on.

```
if event.mapPoint != None:
        print event.mapPoint.getName()
```

getLabelExpression()

Description

Returns the expression used to determine the text value of the map point's label.

Syntax

```
getLabelExpression()
```

Parameters

none

Returns

String – The map point's label's expression.

Scope

Client

Examples

The following snippet in an onPointClicked event on the map panel displays the label expression of the point when clicked on.

```
print point.getLabelExpression()
```

setLabelExpression()

Description

Sets an expression for the map point's label.

Syntax

```
setLabelExpression(expression)
```

Parameters

String `expression`– The expression to set for the map point's label.

Returns

nothing

Scope

Client

Examples

The following snippet sets the label's value to be the current time.

```
point.setLabelExpression('now(1000)')
```

isLabelVisible()

Description

Returns whether the label is showing on the map point or not.

Syntax

```
isLabelVisible()
```

Parameters

none

Returns

boolean – True if the label is showing on the map point.

Scope

Client

Examples

The following snippet in an onPointClicked event on the map panel checks if the label is showing on the map point before attempting to display it

```
if point.isLabelVisible() == 0:
        point.setLabelVisible(1)
```

setLabelVisible()

Description

Shows or hides the map point's label.

Syntax

```
setLabelVisible(visible)
```

Parameters

boolean `visible`– True to display the label, False to hide it.

Returns

nothing

Scope

Client

Examples

The following snippet hides the map points label if it is showing.

```
if point.isLabelVisible() == 1:
        point.setLabelVisible(0)
```

getEvaluatedLabelString()

Description

Evaluates the label expression and returns the value.

Syntax

```
getEvaluatedLabelString()
```

Parameters

none

Returns

String – The output of the map point expression.

Scope

Client

Examples

The following snippet in an onPointClicked() event, prints the point's label string.

```
print point.getEvaluatedLabelString()
```

Scripting : Map Shapes

The following scripting functions are available on the map panel to interact with map shapes.

getLabelExpression()

Description

Returns the expression used to determine the text value of the map shape's label.

Syntax

```
getLabelExpression()
```

Parameters

none

Returns

String – The map shape's label's expression.

Scope

Client

Examples

The following snippet in an onShapeClicked event on the map panel displays the label expression of the shape when clicked on.

```
print shape.getLabelExpression()
```

setLabelExpression()

Description

Sets an expression for the map shape's label.

Syntax

```
setLabelExpression(expression)
```

Parameters

String `expression` – The expression to set for the map point's label.

Returns

nothing

Scope

Client

Examples

The following snippet sets the label's value to be the current time.

```
shape.setLabelExpression('now(1000)')
```

isLabelVisible()

Description

Returns whether the label is showing on the map shape or not.

Syntax

```
isLabelVisible()
```

Parameters

none

Returns

boolean – True if the label is showing on the map shape.

Scope

Client

Examples

The following snippet in an onShapeClicked event on the map panel checks if the label is showing on the map shape before attempting to display it

```
if shape.isLabelVisible() == 0:
        shape.setLabelVisible(1)
```

setLabelVisible()

Description

Shows or hides the map shape's label.

Syntax

```
setLabelVisible(visible)
```

Parameters

boolean `visible` – True to display the label, False to hide it.

Returns

nothing

Scope

Client

Examples

The following snippet hides the map shape's label if it is showing.

```
if shape.isLabelVisible() == 1:
        shape.setLabelVisible(0)
```

getEvaluatedLabelString()

Description

Evaluates the label expression and returns the value.

Syntax

```
getEvaluatedLabelString()
```

Parameters

none

Returns

String – The output of the map shape expression.

Scope

Client

Examples

The following snippet in a onShapeClicked() event, prints the shape's label string.

```
print shape.getEvaluatedLabelString()
```

getStrokeWidth()

Description

Returns the width (in pixels) of the stroke used to draw borders/lines.

Syntax

```
getStrokeWidth()
```

Parameters

none

Returns

Float – The width of the stroke, clamped up to a minimum of 0.

Scope

Client

Examples

The following snippet in a onShapeClicked() event, doubles the stroke's width.

```
shape.setStrokeWidth(shape.getStrokeWidth() * 2)
```

setStrokeWidth()

Description

Sets the width of the stroke used for drawing the border/line.

Syntax

```
setStrokeWidth(strokeWidth)
```

Parameters

Float `strokeWidth` – The new stroke width. Minimum value of 0. Only affects the stroke of the border of polygons and of lines.

Returns

nothing

Scope

Client

Examples

The following snippet in a onShapeClicked() event, doubles the stroke's width.

```
shape.setStrokeWidth(shape.getStrokeWidth() * 2)
```

getDrawModeId()

Description

Returns the draw mode of the shape. 0 is PolyFilled. 1 is PolyBorder. 2 is Line. 3 is Point.

Syntax

```
getDrawModeId()
```

Parameters

none

Returns

Int – The Id of the current draw mode.

Scope

Client

Examples

The following snippet prints the shape's draw mode.

```
print shape.getDrawModeId()
```

setDrawModeId()

Description

Sets the draw mode of the shape. 0 is PolyFilled. 1 is PolyBorder. 2 is Line. 3 is Point.

Syntax

```
setDrawModeId(id)
```

Parameters

Int `id` – The new draw mode Id.

Returns

nothing

Scope

Client

Examples

The following snippet will change the shape's draw mode to Line.

```
shape.setDrawModeId(2)
```

setFillColorString()

Description

Parses the specified string value into a color, and applies it to the fill color.

Syntax

```
setFillColorString(fillColorString)
```

Parameters

String `fillColorString` – The color to be assigned. Hex is supported.

Returns

nothing

Scope

Client

Examples

The following snippet sets the shape's fill color to blue.

```
shape.setFillColorString("#0000FF")
```

setStrokeColorString()

Description

Parses the specified string value into a color, and applies it to the stroke color.

Syntax

```
setStrokeColorString(strokeColorString)
```

Parameters

String `strokeColorString` – The color to be assigned. Hex is supported.

Returns

nothing

Scope

Client

Examples

The following snippet sets the shape's stroke color to white.

```
shape.setStrokeColorString("#FFFFFF")
```

getOpacity()

Description

Returns the transparency value used when drawing the element.

Syntax

```
getOpacity()
```

Parameters

none

Returns

Float – Transparency value of the shape, between 0 and 1.

Scope

Client

Examples

The following snippet in a onShapeClicked() event, prints the shape's opacity.

```
print shape.getOpacity()
```

setOpacity()

Description

Sets the opacity for the shape.

Syntax

```
setOpacity(opacity)
```

Parameters

Float `opacity` – The new opacity. 1 is fully opaque, 0 is fully transparent. The value is automatically clamped between 0 and 1.

Returns

nothing

Scope

Client

Examples

Sets the shape to be 50% transparent.

```
shape.setOpacity(0.5)
```