WDP API Introduction

Powered by 51WORLD

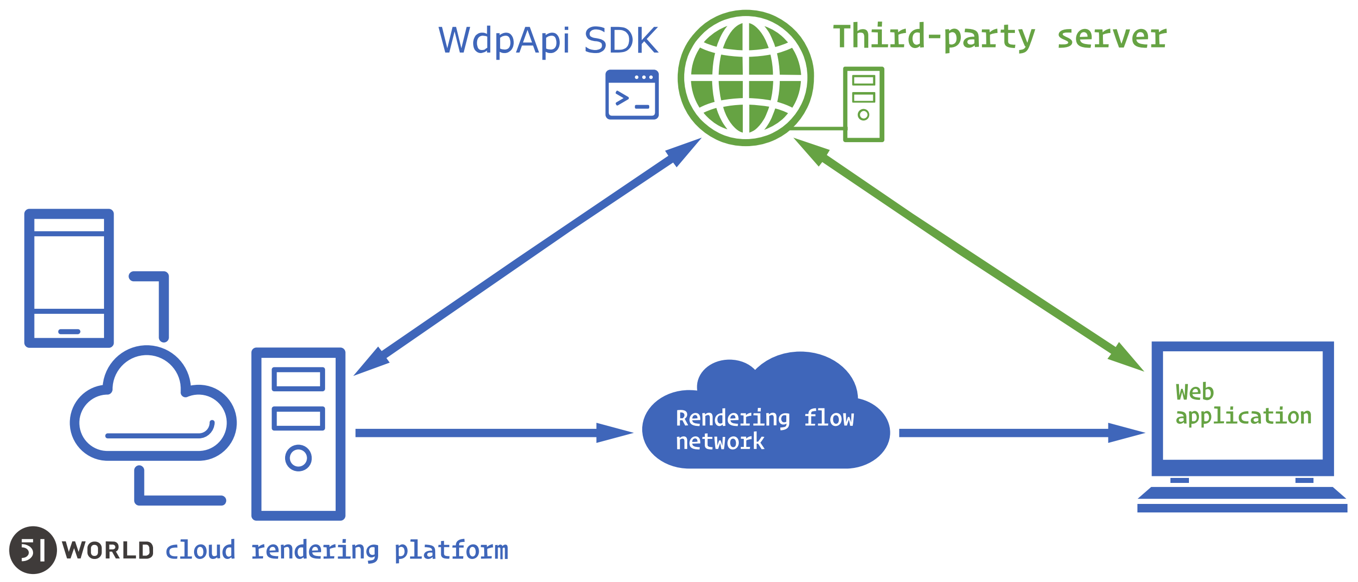
- 51WORLD WdpApi is a JavaScript programming interface designed to connect web pages with the 51WDP online rendering platform.
- With this API, developers can create HTML5 UI elements on web pages and seamlessly interact with 3D scenes rendered in the cloud bidirectionally.
- Compatible with popular frontend frameworks such as React and Vue, it streamlines the development process, allowing for the integration of high-quality 3D graphics in web applications.
- By leveraging 51WORLD WdpApi and 51WDP, developers can build web applications featuring immersive 3D graphics without the need for powerful local hardware.
- Shifting rendering computations to the cloud enables real-time, high-quality 3D graphics on the web without utilizing local device resources.
- 51WORLD WdpApi simplifies the complexity of developing web applications that integrate cloud-rendered 3D graphics, facilitating the creation of interactive 3D experiences on web pages.
- It eliminates barriers for developers looking to seamlessly integrate high-fidelity 3D graphics on the web.

Technology Architecture

- The 3D scenes are rendered via the 51WORLD cloud and streamed in real-time to the web, enabling synchronized interaction between the frontend and backend.

- Pixel streaming is integrated as a plugin within the engine. The plugin encodes the graphical stream data from the host server and sends it via instant communication to browsers and devices on the receiving end. By running the rendering engine on high-performance host systems, users can experience the same visual quality and full functionality across all devices as they would on the host machine. The pixel streaming plugin communicates between the host server and client, and can run on a single server or dynamically scale across a GPU cloud environment with sufficient hardware.

- WdpApi provides methods that can be invoked on the web frontend, allowing users to directly send commands to the 51WORLD cloud rendering platform from the webpage. Additionally, by registering functions to listen to events sent by the 51WORLD cloud rendering program, users can respond to these events on the web based on their types.



WdpApi API Reference

Init Scene

```
npm install wdpapi
```

```
import WdpApi from "wdpapi";
```

```
const App = new WdpApi({
    "id": "player", //the DOM ID of scene video streaming
    "url": "https://dtp-api.51aes.com", //[optional] rendering service address
    "order": "8099702a64dbb8ef4a0a2f7b5b1c42b0", //[optional] rendering order
    "resolution": [1920,1080], //[optional] scene video streaming resolution;
    "debugMode": "normal", //[optional] none: no function log; normal: normal function log
    "keyboard": { //[optional]
        "normal": false, //[optional]  whether open keyboard move event (WASDQE)
        "func": false //[optional]  whether open keyboard function event (F1~F12)
    }
});
```

Parameter description:

parameter	value	type	remark
id		string	Render the Dom node of the 3D scene window
url		string	The address of the cloud rendering service
order		string	The rendering order
resolution	[100~7680, 100~4320]	[Integer]	Set resolution Note: Chrome browser supports up to 4K: 4096*2160; With 51Browser, you can reach 8K support 7680*4320
debugMode	none, norml, high	string	none: do not print logs; normal: normal logs (data transfer results); high: Advanced logs, including low-level transmission logs such as mouse clicks;
keyboard	normal	boolean	normal: true/false keyboard events
	func	boolean	func: true/false Browser F1~F12 function keys

Reset Scene Parameter

```
await App.System.SetOption({
    "url": "https://dtp-api.51aes.com",
    "order": "2399702a64dbb8ef4a0a2f7b5b1c41a0",
    "resolution": [3840,2160]
})
```

Rendering Mode

```
await App.Renderer.SetRendererMode('type', [3840, 2160]);
```

Parameter description:

Parameter	Value	Type	Notes
type	full, fixed	String	Scene video resolution mode; full: adapt to container automatically; fixed: fixed
resolution	[100~7680, 100~4320]	Integer	Scene video resolution, valid while "fixed" Note: Chrome browser supports 4K(4096 * 2160); 51Browser supports 8K(7680 * 4320)

Set the timeout period for API requests

Used for big data API interface call

```
await App.System.SetTimeoutTime(30000); //30s; default: 10s
```

Start Scene

```
App.Renderer.Start().then((res: any) => {
    if (res.success) {
        // Initial scene events
        App.Renderer.RegisterEvent([
            {
                name: 'onVideoReady', func: async function (res: any) {
                    // Video stream link successful
                }
            },
            {
                name: 'onStopedRenderCloud', func: async function (res: any) {
                    // Rendering service interrupted
                }
            }
        ])
        // Scene event callbacks
        App.Renderer.RegisterSceneEvent([
            {
                name: 'OnWdpSceneIsReady', func: async function (res: any) {
                    if (res.result.progress === 100) {
                        // Scene loaded successfully
                    }
                }
            },
            {
                name: 'OnEntityClicked', func: async function (res: any) {
                    // Entity clicked event callback; includes data information and entity object
                }
            },
            {
                name: 'OnMouseEnterEntity', func: async function (res: any) {
                    // Mouse enter entity event callback; includes data information and entity object
                }
            },
            {
                name: 'OnMouseOutEntity', func: async function (res: any) {
                    // Mouse exit entity event callback; includes data information and entity object
                }
            },
            {
                name: 'OnWebJSEvent', func: async function (res: any) {
                    // Receive data sent from the embedded page within the window
                }
            },
            {
                name: 'MeasureResult', func: async function (res: any) {
                    // Measurement tool data callback
                }
            },
            {
                name: 'OnMoveAlongPathEndEvent', func: async function (res: any) {
                    // Overlay movement end information callback
                }
            },
            {
                name: 'OnCameraMotionStartEvent', func: async function (res: any) {
                    // Camera motion start information callback
                }
            },
            {
                name: 'OnCameraMotionEndEvent', func: async function (res: any) {
                    // Camera motion end information callback
                }
            },
            {
                name: 'PickPointEvent', func: async function (res: any) {
                    // Pick point tool data callback
                }
            },
            {
                name: 'OnEntitySelectionChanged', func: async function (res: any) {
                    // Callback for Entities are selected
                }
            },
            {
                name: 'OnEntityReady', func: async function (res: any) {
                    // 3DTilesEntity, WMSEntity, WMTSEntity loaded;
                    // {success: true, message: '', result: { object: object, progress: 100 }}
                }
            }
        ])
    }
}).catch(err => {
    console.log(err)
});
```

Member Function

- **Add(object)**

Purpose: Add entities into scene

```
const path = new App.Path({ "polyline":  { "coordinates": [121.50007292,31.22579403,30]}, "pathStyle":  style })
cosnt res = await App.Scene.Add(path);
console.log(res);
```

- **Covering.Clear()**

Purpose: clear all covering entities

Set camera mode

```
async function SetCameraMode() {

  const res = await App.CameraControl.SetCameraMode('RTS');
  console.log(res);

  /*
   RTS (Real-Time Strategy mode);
 FPS (First-Person Shooter mode);
TPS (Third-Person Shooter mode)
  */
}
```

RTS FPS TPS

Camera mode operations:

Right mouse button Alt + right mouse button N/A N/A Left mouse button Pan camera (move forward, backward, left, or right parallel to the ground) W/↑ Move forward Move forward along the body angle S/↓ Move backward parallel to the ground Move backward Move backward along the body angle A/← Move left parallel to the ground Move left Move left parallel to the ground D/→ Move right parallel to the ground Move right Move right parallel to the ground E/PgUp Ascend N/A Ascend Q/PgDn Descend N/A Descend Space N/A Leap (double the height then land) N/A C N/A Crouch (reduce height by half) N/A Accelerate movement (triple the movement speed) Accelerate movement (triple the movement speed) Accelerate movement (triple the movement speed)

Shortcut key	RTS: Sandbox Mode	FPS: First-Person Mode	TPS: Third-Person Mode
Right-click	Rotate viewpoint	Pan around	Adjust orientation
Alt+Right-click	N/A	N/A	Rotate camera around mouse center
Left-click	Pan camera (move forward, backward, left, or right parallel to the ground)	Pan around	Pan camera (move forward, backward, left, or right parallel to the ground)
W/↑	Move forward parallel to the ground	Move forward	Move forward along the body
S/↓	Move backward parallel to the ground	Move backward	Move backward along the body
A/←	Move left parallel to the ground	Move left	Move left parallel to the ground
D/→	Move right parallel to the ground	Move right	Move right parallel to the ground
E/PgUp	Ascend	N/A	Ascend
Q/PgDn	Descend	N/A	Descend
空格	N/A	Leap (double the height then land)	N/A
C	N/A	Crouch (reduce height by half)	N/A
Shift + movement	Accelerate movement (triple the movement speed)	Accelerate movement (triple the movement speed)	Accelerate movement (triple the movement speed)

Get camera position

```
async function GetCameraPose() {

  const res = await App.CameraControl.GetCameraPose();
  console.log(res);
}
```

Set camera position

```
async function SetCameraPose() {
  const jsondata = {
    "location": [121.48537621,31.23840069,900],
    "rotation": {
      "pitch": -35, // Pitch angle, reference (-90~0)
      "yaw": 0 // Yaw angle, reference (-180~180)
    },
    "flyTime": 1 // Transition duration (unit: seconds)
  }

  const res = await App.CameraControl.SetCameraPose(jsondata);
  console.log(res);
}
```

Reset camera position

```
async function ResetCameraPose() {
  // If the camera's initial state is not ideal, you can adjust the camera's initial state

  const jsondata = {
    "state": 'Default', // Default: Camera's initial state; Last: Last position of the camera
    "flyTime": 1, // Transition duration (unit: seconds)
  }

  const res = await App.CameraControl.ResetCameraPose(jsondata);
  console.log(res);
}
```

Get camera limit values

```
async function GetCameraLimit() {

  const res = await App.CameraControl.GetCameraLimit();
  console.log(res);

}
```

Set camera limit values

```
async function SetCameraLimit() {
  const jsondata = {
    "locationLimit": [ // Set camera position region (optional), defined by at least three coordinates forming a triangular area
      [121.47095414,31.22534628],
      [121.47264982,31.23423431],
      [121.49467492,31.24871524]
    ],
    "pitchLimit": [-80,0], // Pitch angle; Range: [-90~0]
    "yawLimit": [-100,100], // Yaw angle; Range: [-180~180]
    "viewDistanceLimit": [100,1000] // Camera distance from entity distance range
  }

  const res = await App.CameraControl.SetCameraLimit(jsondata);
  console.log(res);
}
```

Set fixed camera limit values

```
async function SetCameraLockLimit() {
  const jsondata = {
    "locationLimit": 100, // The camera can move forward, backward, left, or right when dragging the scene [-100,100] range (unit: meters)
    "pitchLimit": 10, // The current pitch (pitch angle) can move within the range of [-10,0]; Range: [0~90]
    "yawLimit": 20, // The current yaw (yaw angle) can move within the range of [-20,20]; Range: [0~180]
    "viewDistanceLimit": 100 // The current field of view can change within the range of [-100,100] when scrolling the mouse wheel (unit: meters)
  }

  const res = await App.CameraControl.SetCameraLockLimit(jsondata);
  console.log(res);
}
```

Reset camera limit values

```
async function ResetCameraLimit() {
  const res = await App.CameraControl.ResetCameraLimit('Default');
  // Default: Camera's initial Limit; Free: No Limit restrictions

  console.log(res);
}
```

Get Camera Current Infomation

```
App.CameraControl.GetCameraInfo()
```

Update Camera

```
const jsondata = {
    location: [121.48940131,31.25135281,500],
    rotation: { pitch: -30, yaw: 0 },
    pitchLimit: [-90,0], // the limitation of pitch
    yawLimit: [-180,180], // the limitation of yaw
    viewDistanceLimit: [500,600], // the limitation of view distance
    fieldOfView: 90, // the view range of camera
    controlMode: 'RTS', // camera mode
    flyTime: 1 // the duration from current state to expected state (unit : second)
}

await App.CameraControl.UpdateCamera(jsondata)
```

Parameter Description:

Parameter	Type	Required（If option，the default value)	Limitation
location	array		
rotation	object		
-pitch	number	-30	[-90,0]
-yaw	number	0	[-180,180]
pitchLimit	array	[-90,0]	[-90,0]
yawLimit	array	[-180,180]	[-180,180]
viewDistanceLimit	array	[0,10000]	[0,+∞)
fieldOfView	number	90	[0,120]
controlMode	string		

Camera Move

```
App.CameraControl.Move({
    direction: 'right',
    velocity: 10
})
```

Parameter Description:

Parameter	Type	Required（If option，the default value）	Limitation
direction	string	✓	forward backward left right up down
velocity	number	✓	[0,+∞)

Camera Rotate

```
App.CameraControl.Rotate({
    direction: 'right',
    velocity: 10
})
```

Parameter Description:

Parameter	Type	Required（If option，the default value）	Limitation
direction	string	✓	up; down; left; right
velocity	number	✓	[0,+∞)

Camera Around

```
App.CameraControl.Around({
    direction: 'clockwise',
    velocity: 10
})
```

Parameter Description:

Parameter	Type	Required (If option, the default value)	Limitation
direction	string	✓	clockwise anticlockwise
velocity	number	✓	[0,+∞)

Stop Camera

```
await App.CameraControl.Stop()
```

Camera Move Step

```
App.CameraControl.CameraStepMove({
  moveDirection: 'E_Forward', // forward
  step: 0.5, //speed multiplier -1 ~ 1
  bContinuous: true //Whether continuous
});

  /*
  moveDirection:
    E_Forward: Forward,
    E_Backward: Backward,
    E_Left: Left,
    E_Right: Right,
    E_Up: Up,
    E_Down: Down
  */
```

Parameter Description:

Parameter	Type	Required（If option，the default value）	Limitation
moveDirection	string	✓	E_Forward, E_Backward, E_Left, E_Right, E_Up, E_Down
step	number	✓	[-1~1]: speed multiplier
bContinuous	boolean	✓	true, false: Whether continuous

Camera Rotate Step

```
App.CameraControl.CameraStepRotate({
  rotateDirection: 'E_Pitch', //E_Pitch: pitch angle, E_Yaw: yaw angle
  step: 0.5, //speed multiplier -1 ~ 1
  bContinuous: true //Whether continuous
});
```

Parameter Description:

Parameter	Type	Required（If option，the default value）	Limitation
rotateDirection	string	✓	E_Pitch: pitch angle, E_Yaw: yaw angle
step	number	✓	[-1~1]: speed multiplier
bContinuous	boolean	✓	true, false: Whether continuous

Camera Zoom Step

```
App.CameraControl.CameraStepZoom({
  step: 0.5, //speed multiplier -1 ~ 1; (>0 zoom out, <0 zoom in)
  bContinuous: true //Whether continuous
});
```

Parameter Description:

Parameter	Type	Required（If option，the default value)	Limitation
step	number	✓	[-1~1]: speed multiplier, (>0 zoom out, <0 zoom in)
bContinuous	boolean	✓	true, false: Whether continuous

Stop Camera Move, Rotate, Zoom

```
App.CameraControl.StopCameraStepUpdate();
```

Camera Focus To Position

```
const jsondata = {
    "targetPosition": [121.48533665,31.24164246,30],
    "rotation": {
        "pitch": -30, // the preset pitch
        "yaw": 0, //the preset yaw
    },
    "distance": 500, //the distance between camera and target position (unit : meter)
    "flyTime": 1 //the duration from current state to expected state (unit : second)
}
```

```
await App.CameraControl.FlyTo(jsondata)
```

Camera Focus To Entity Node

```
const jsondata = {
 "nodeIds": ['895874688'], // Entity nodeIds
    "rotation": {
        "pitch": -30, //the preset pitch
        "yaw": 0 //the preset pitch
    },
    "distanceFactor": 2, // the entity display ratio on screen, 1 stands for 100% of screen, 2  stands for 50% of screen
    "flyTime": 1, //the duration from current state to expected state (unit : second)
    "entity": [pathObject] // the object of entity
}

await App.CameraControl.Focus(jsondata)
```

Camera Focus To Entity

```
const jsondata = {
    "rotation": {
        "pitch": -30, //the preset pitch
        "yaw": 0 //the preset pitch
    },
    "distanceFactor": 2, // the entity display ratio on screen, 1 stands for 100% of screen, 2  stands for 50% of screen
    "flyTime": 1, //the duration from current state to expected state (unit : second)
    "entity": [pathObject] // the object of entity
}
```

```
await App.CameraControl.Focus(jsondata)
```

Camera Focus To Covering

```
App.CameraControl.FocusToAll({
    types: ['Poi', 'Path'], //Entity type
    bFilterForExclude: true
})

// types
/*
  RealTimeVideo
  Window
  Poi
  Particle
  Text3D
  Viewshed
  Path
  Parabola
  Range
  HeatMap
  ColumnarHeatMap
  SpaceHeatMap
  RoadHeatMap
*/
```

Focus to All Entities

```
App.CameraControl.FocusToAll();
```

Camera Follow Entity

```
const jsondata = {
    "followRotation": {
        "pitch": -20, //the preset pitch
        "yaw": 20 //the preset yaw
    },
    "useRelativeRotation": true, // the mode of "followRotation"; false: the world angle; true: relative to entity
    "distance": 200, // the distance between camera and entity
    "entity": followParticle //the object of entity
}

await App.CameraControl.Follow(jsondata)
```

Stop Camera

```
await App.CameraControl.Stop()
```

Start Camera Roam

```
const jsondata = {
  "frames": [
    {
      "location": [121.49067713,31.11991912,300],
      "rotation": {
        "pitch": -25, //the preset pitch
        "yaw": 0 // the preset yaw
      },
      "time": 20 //the duration to next location
    },
    {
      "location": [121.49060113,31.11431200,300],
      "rotation": {
        "pitch": -15,
        "yaw": 80
      },
      "time": 20
    },
    {
      "location": [121.49687008,31.13777349,300],
      "rotation": {
        "pitch": -20,
        "yaw": 160
      },
      "time": 15
    },
    {
      "location": [121.49441582,31.13728981,300],
      "rotation": {
        "pitch": -15,
        "yaw": 240
      }
    }
  ]
}

const roaming = new App.CameraRoam(jsondata);
await App.Scene.Add(roaming);

// Start camera roam
await App.CameraControl.PlayCameraRoam(roaming);
```

```
// Cache global objects for subsequent operations
cache.set('roaming', roaming);
```

Update Camera Roam

```
const jsondata = {
  "frames": [
    {
      "location": [121.48216421,31.10446008,300],
      "rotation": {
        "pitch": -25, //the preset pitch
        "yaw": -90, //the preset yaw
      },
      "time": 20 //the duration to next location
    },
    {
      "location": [121.48405533,31.12949274,300],
      "rotation": {
        "pitch": -15,
        "yaw": 0
      },
      "time": 20
    },
    {
      "location": [121.48880933,31.13466789,300],
      "rotation": {
        "pitch": -25,
        "yaw": 90
      }
    }
  ]
}

const roam = cache.get('roaming');
await roam.Update(jsondata);
```

```
// Start Camera Roam
await App.CameraControl.PlayCameraRoam(roam);
```

Get Camera Roam

```
const _res = await cache.get('roaming').Get();
console.log(_res);
```

Stop Camera Roam

```
await App.CameraControl.Stop()
```

Add (add entity to the scene)

```
await App.Scene.Add(obj, {
  "calculateCoordZ": {   // Coordinate type and coordinate height; [optional] Highest priority
    "coordZRef": "surface",   // surface; ground; altitude
    "coordZOffset": 50   // Height (unit: meters)
});
```

Parameter description:

Parameter		Type	Required	Value Range	Remarks
object		Array | Object	✓		Object arrays or objects
calculateCoordZ	coordZRef	string	optional	surface, ground, altitude	surface; ground; altitude
	coordZOffset	number	optional		height(unit:meter)

"calculateCoordZ" has the highest priority; defaults to the coordinate coordz in obj if not specified

callback:

```
{
    "success": true, // true, false
    "message": '',
    "result": {
        "object": {}, // "objects": [],
        "sceneChangeInfo": {}
    }
}
```

Update (Unified Update for Multiple Objects of the Same Type)

```
await App.Scene.Update([obj, obj, ...],
    { poiStyle: { ... } }, {
    "calculateCoordZ": {   // Coordinate type and coordinate height; [optional] Highest priority
        "coordZRef": "surface",   // surface;  ground;  altitude
        "coordZOffset": 50   // Height (unit: meters)
    }
});
```

callback:

```
{
    "success": true, // true, false
    "message": '',
    "result": {
        "sceneChangeInfo": {}
    }
}
```

See below for details of the call: Common Entity Behaviors

Create (Bulk Add Entities of the Same Category)

```
await App.Scene.Create({
    ... // Default property values
}, [{
   ... // Bulk property values
}], {
    "calculateCoordZ": {   // Coordinate type and coordinate height; [optional] Highest priority
        "coordZRef": "surface",   // surface;  ground;  altitude
        "coordZOffset": 50   // Height (unit: meters)
      }
   }
);
```

callback:

```
{
    "success": true, // true, false
    "message": '',
    "result": {
        "objects": [obj, ...]
        "sceneChangeInfo": {}
    }
}
```

Creates (Bulk Add Entities of Different Categories)

```
await App.Scene.Creates([{
    ... // Bulk property values
}], {
    "calculateCoordZ": {   // Coordinate type and coordinate height; [optional] Highest priority
        "coordZRef": "surface",   // surface;  ground;  altitude
        "coordZOffset": 50   // Height (unit: meters)
      }
   }
);
```

callback:

```
{
    "success": true, // true, false
    "message": '',
    "result": {
        "objects": [obj, ...]
        "sceneChangeInfo": {}
    }
}
```

GetAll (Get All Entity Objects in the Scene)

```
await App.Scene.GetAll();
```

GetByEids (Get Objects by Eid)

```
await App.Scene.GetByEids(['-9151314316185345952', '-9151314316965221260', ...]);
```

GetByEntityName (Get Entity by Entity Name)

```
await App.Scene.GetByEntityName(['name01', 'name02', ...]);
```

GetByCustomId (Get Entity by Custom ID)

```
await App.Scene.GetByCustomId(['cuId01', 'cuId02', ...]);
```

GetByTypes (Get Entity by Types)

```
await App.Scene.GetByTypes(['Poi', 'Static', ...]);
```

Delete (Bulk Delete Entities)

```
await App.Scene.Delete([obj, obj, ...]);
```

ClearByTypes (Bulk Delete Entities)

```
await App.Scene.ClearByTypes(['xxx', 'xxx', ...]);
/*
  Static
  Skeletal
  Hierarchy
  ISEHierarchy
  Effects
*/

/*
  RealTimeVideo
  Window
  Poi
  Particle
  Effects
  Light
  Text3D
  Viewshed
  Path
  Parabola
  Range
  HeatMap
  ColumnarHeatMap
  SpaceHeatMap
  RoadHeatMap
  Raster
  HighlightArea
*/
```

ClearByObjects (Bulk Delete Entities)

```
await App.Scene.ClearByObjects([obj, obj, ...]);
```

ClearByClearByEids (Bulk Delete Entities)

```
await App.Scene.ClearByEids(['xxx', 'xxx', ...]);
```

Covering.Clear (Delete All Overlays in the Scene)

```
await App.Scene.Covering.Clear();
```

ClearByCustomId (Bulk Delete by Custom ID)

```
await App.Scene.ClearByCustomId(['xxx', 'xxx', ...]);
```

ClearByEntityName (Bulk Delete by Entity Name)

```
await App.Scene.ClearByEntityName(['xxx', 'xxx', ...]);
```

UpdateByCustomId (Bulk Update Same Category Entities by Custom ID)

```
await App.Scene.UpdateByCustomId(['xxx', 'xxx', ...], { poiStyle: { ... } }, {
    "calculateCoordZ": {   // Coordinate type and coordinate height; [optional] Highest priority
        "coordZRef": "surface",   // surface;  ground;  altitude
        "coordZOffset": 50   // Height (unit: meters)
    }
});
```

UpdateByCustomIds (Bulk Update Different Category Entities by Custom ID)

```
await App.Scene.UpdateByCustomId([
        { customId: 'cuId01', poiStyle: { ... } },
        { customId: 'cuId02', pathStyle: { ... } }
    ], {
    "calculateCoordZ": {   // Coordinate type and coordinate height; [optional] Highest priority
        "coordZRef": "surface",   // surface;  ground;  altitude
        "coordZOffset": 50   // Height (unit: meters)
    }
});
```

UpdateByEntityName (Bulk Update Same Category Entities by Entity Name)

```
await App.Scene.UpdateByEntityName(['xxx', 'xxx', ...], { poiStyle: { ... } }, {
    "calculateCoordZ": {   // Coordinate type and coordinate height; [optional] Highest priority
        "coordZRef": "surface",   // surface;  ground;  altitude
        "coordZOffset": 50   // Height (unit: meters)
    }
});
```

UpdateByEntityNames (Bulk Update Different Category Entities by Entity Name)

```
await App.Scene.UpdateByEntityNames([
        { entityName: 'name01', poiStyle: { ... } },
        { entityName: 'name02', pathStyle: { ... } }
    ], {
    "calculateCoordZ": {   // Coordinate type and coordinate height; [optional] Highest priority
        "coordZRef": "surface",   // surface;  ground;  altitude
        "coordZOffset": 50   // Height (unit: meters)
    }
});
```

SetVisibleByObjects (Bulk Show/Hide Entities)

```
await App.Scene.SetVisibleByObjects([obj, obj, ...], false);
```

SetVisible (Bulk Show/Hide Entities)

```
await App.Scene.SetVisible([obj, obj, ...], false);
```

SetLocation (Move Multiple Objects to the Same Location)

```
await App.Scene.SetLocation([obj, obj, ...], { x: 121.50796384, y: 31.23267352, z: 50 });
```

SetLocations (Move Multiple Objects to Different Locations)

```
await App.Scene.SetLocations([
    { object: obj1, location: { x: 121.50796384, y: 31.23267352, z: 50 } },
    { object: obj2, location: { x: 121.52796384, y: 31.25267352, z: 50 } },
]);
```

SetRotator (Rotate Multiple Objects to the Same Angle)

```
await App.Scene.SetRotator([obj, obj, ...], { pitch: 70, yaw: 20, roll: 80 });
```

SetRotators (Rotate Multiple Objects to Different Angles)

```
await App.Scene.SetRotators([
    { object: obj1, rotator: { pitch: 70, yaw: 20, roll: 80 } }
    { object: obj2, rotator: { pitch: 50, yaw: 30, roll: 70 } }
]);
```

SetScale3D (Scale Multiple Objects by the Same Factor)

```
await App.Scene.SetScale3D([obj, obj, ...], { x: 10, y: 50, z: 50 });
```

SetScale3Ds (Scale Multiple Objects by Different Factors)

```
await App.Scene.SetScale3Ds([
    { object: obj, scale3d: { x: 10, y: 50, z: 50 } }
]);
```

SetLocked (Bulk Lock/Unlock Multiple Objects)

```
await App.Scene.SetLocked([obj, obj, ...], false);
```

GetBound (Get the bounding box data of objects; Entity box range)

```
await App.Scene.GetBound([obj, obj, obj]);
```

callback:

```
{
    "success": true,
    "message": "",
    "result": {
        "entitiesBound": {  // Cartesian Coordinates
            "min": [
                575.0857058653336,
                583.5574901356209,
                11.27789306640625
            ],
            "max": [
                1730.7478152403335,
                -1138.2877247081292,
                601.21044921875
            ],
            "isValid": 1
        }
    }
}
```

Convert Cartesian coordinates to GIS coordinates

```
const cartesian = [
    [
        575.0857058653336,
        583.5574901356209,
        11.27789306640625
    ],
    [
        1730.7478152403335,
        -1138.2877247081292,
        601.21044921875
    ],
]
const res = await App.Tools.Coordinate.CartesianToGIS(cartesian);
console.log(res.result.to)
```

Geometry GIS coordinates: point Single point entity

```
const object = new App.Poi({
  "location": [121.46434372, 31.23499129, 200],
  "poiStyle": { ... }
})
```

Parameter description:

Parameter	Type	Required	Remarks
location	array	✅	format: [lng, lat, coordz]; coordz: altitude
poiStyle	JSON	✅	JSON data

Geometry GIS coordinates: polyline multi-point entity

```
const object = new App.Path({
    "polyline": {
        "coordinates": [
            [121.49968476, 31.24861346, 44],
            [121.49956979, 31.25093239, 96],
            [121.47613890, 31.23725069, 39]
        ]
    },
    "pathStyle": { ... }
});
```

Parameter description:

Parameter		Type	Required	Remarks
polyline	coordinates	array	✅	format: [[lng, lat, coordz],[[lng, lat, coordz],....]
pathStyle		JSON	✅	JSON data

Geometry GIS coordinates: polygon2D multipoint entity

```
const object= new App.Range({
    "polygon2D": {
        "coordinates": [
            [  // Outer ring coordinate data
                [121.44988564758069, 31.250519581243555],
                [121.44931229954645, 31.237062463089813],
                [121.47069915607464, 31.23800903013435],
                [121.46964214200186, 31.251854247249092]
            ],
            [  // Inter ring coordinate data
                [121.45523929837454, 31.247795686070997],
                [121.45496451671893, 31.240059486959915],
                [121.46707798490596, 31.24170746459223]
            ]
        ]
    },
    "rangeStyle": { ... }
})
```

Parameter description:

Parameter		Type	Required	Remarks

polyline	coordinates	array	✓	Coordinates, where the first array in the format represents the outer loop and the subsequent arrays represent inner loops.
				[
				    [[lng,lat],[lng,lat],....],
				    [[lng,lat],[lng,lat],....],
				    ....
				]

rangeStyle		JSON	✓	JSON data

Geometry GIS coordinates: points multiple entities

```
const object = new App.HeatMap({
    "points": {
        "features": [
            { "point": [121.49656333, 31.22702479, 49], "value": 89 },
            { "point": [121.46434372, 31.23499129, 60], "value": 62 },
            { "point": [121.49099537, 31.23099794, 22], "value": 54 }
        ]
    },
    "heatMapStyle": { ... }
});
```

Parameter description:

Parameter		Type	Required	Remarks
points.features	point	array	✓	[lng,lat,coordz]
	value	number	✓	Need to be within the range defined by "mappingValueRange"
heatMapStyle		JSON	✓	JSON data

Basic and Custom Properties

3D text Text3D is used as an example, applicable to all entity overlays.

```
const obj = new App.Text3D({
    ... ...
    // Basic properties, all instances have them
    "bLocked": true, // Whether the added entity is locked, not clickable, selectable, etc. (true/false) [Optional]
    "bVisible": true, // Whether the added entity is visible (true/false) [Optional]

    // Custom properties, all instances have them; Define content according to business needs
    "entityName": "myName", // [Optional]
    "customId": "myId", // [Optional]
    "customData": { // [Optional]
        "data": "myCustomData"
    }

})
```

Parameter description:

Parameter	Type	Required (default if optional)	Value Range	Remarks
bLocked	boolean	Optional	true, false	Whether the added entity is locked, not clickable, selectable, etc.
bVisible	boolean	Optional	true, false	Whether the added entity is visible
entityName	string	Optional		Configurable for using EntityName series functionalities
customId	string	Optional		Configurable for using CustomId series functionalities
customData	object	Optional		Configurable according to business needs

Attribute Setting

Keys starting with the letter "b"; Get/Set attributes omit the letter "b", and capitalize the first letter.

```
// Getter:
// Method one:
console.log("bLocked:: ", obj.bLocked);
console.log("bVisible:: ", obj.bVisible);
console.log("entityName:: ", obj.entityName);
console.log("customId:: ", obj.customId);
// Method two:
console.log("GetLocked:: ", await obj.GetLocked());
console.log("GetVisible:: ", await obj.GetVisible());
console.log("GetEntityName:: ", await obj.GetEntityName());
console.log("GetCustom:: ", await obj.GetCustom());
```

```
// Setter:
// Method one:
obj.bLocked = false;
obj.bVisible = false;
obj.entityName = 'newName';
obj.customId= 'newId';

// Method two:
await obj.SetLocked(false);
await obj.SetVisible(false);
await obj.SetEntityName('newName');
await obj.SetCustomId('newId');
```

Receive One: Object Click Event

```
obj.onClick(async ev => {
    const obj = ev.result.object;
    console.log(await obj.Get());
})
```

Receive Two: Event Listener Callback

```
App.Renderer.RegisterSceneEvents([
  {
    name: 'OnEntityClicked', func: async function (res) {
      // Callback for overlay clicked event; contains data information and entity object
      if (res.result.object.entityName === "myName") {
        const jsondata = {
          "text3DStyle": {
            "text": "更新3D文字",
            "color": "a421ffff",
            "type": "plain",
            "outline": 0.2,
            "portrait": false,
            "space": 0.2
          }
        }
        const newObj = res.result.object;
        newObj.Update(jsondata);

        const info = await newObj.Get();
        console.log(info)
      };
    }
  }
])
```

Entity Extension Attributes

Other instance attributes refer to each instance attribute field.

```
// particleObj is the object created with new App.Particle({...});

// Get Particle attributes
async function getParticleAttr () {
    // Method one:
    console.log("location:: ", particleObj.location);
    console.log("type:: ", particleObj.particleType);
    console.log("rotator:: ", particleObj.rotator);
    console.log("scale3d:: ", particleObj.scale3d);

    // Method two:
    console.log("location:: ", await particleObj.GetLocation());
    console.log("type:: ", await particleObj.GetParticleType());
    console.log("rotator:: ", await particleObj.GetRrotator());
    console.log("scale3d:: ", await particleObj.GetScale3d());
}
getParticleAttr();


// Set Particle attributes
async function setParticleAttr () {
    // Method one:
    particleObj.location = [121.46141528,31.23360944,86];
    particleObj.particleType = "vehicle_car_white";
    particleObj.rotator = {
        "pitch": 0, "yaw": 40, "roll": 0
    };
    particleObj.scale3d = [200, 200, 200];

    // Method two:
    await particleObj.SetLocation([121.46141528,31.23360944,86]);
    await particleObj.SetParticleType("vehicle_car_white");
    await particleObj.SetRotator({
        "pitch": 0, "yaw": 40, "roll": 0
    });
    await particleObj.SetScale3d([200, 200, 200]);
}
setParticleAttr();
```

When the key of the attribute field in the instance is "type", the newly mapped field for setting attributes is "sType".

```
// pathObj is the object created with new App.Path({...});

// Get Path attributes
async function getPathAttr (attr) {
    // Method one:
    console.log("coordinates:: ", pathObj.coordinates);
    console.log("sType:: ", pathObj.sType);
    console.log("width:: ", pathObj.width);
    console.log("color:: ", pathObj.color);
    console.log("passColor:: ", pathObj.passColor);

    // Method two:
    console.log("coordinates:: ", await pathObj.GetCoordinates());
    console.log("sType:: ", await pathObj.GetsType());
    console.log("width:: ", await pathObj.GetWidth());
    console.log("color:: ", await pathObj.GetColor());
    console.log("passColor:: ", await pathObj.GetPassColor());
}
getPathAttr();


// Set Path attributes
async function setPathAttr (attr) {
    // Method one:
    pathObj.coordinates = [
      [121.50056782,31.22792919,23],
      [121.49728647,31.22611933,90],
      [121.48236809,31.23146931,60]
    ];
    pathObj.sType = "solid";
    pathObj.width = 50;
    pathObj.color = "ff4b3dff";
    pathObj.passColor = "affff2ff";

    // Method two:
    await pathObj.SetCoordinates([
      [121.50056782,31.22792919,23],
      [121.49728647,31.22611933,90],
      [121.48236809,31.23146931,60]
    ]);
    await pathObj.SetsType("solid");
    await pathObj.SetWidth(50);
    await pathObj.SetColor("ff4b3dff");
    await pathObj.SetPassColor("affff2ff");
}
setPathAttr();
```

Entity Member Functions

Example: Text3D Member Functions

```
const obj = new App.Text3D({ ... });
obj.Update(json);
obj.Get/SetLocation(json);
obj.Get/SetRotator(json);
obj.Get/SetScale3d(json);
obj.Get/SetLocked(boolean);
obj.Get/SetVisible(boolean);
obj.Get/SetEntityName(string);
obj.Get/SetCustomId(string);
obj.Get/SetCustomData(json);
obj.Get();
obj.oType;  //get
obj.bRemoved;  //get
obj.bLocked = boolean;   //get/set
obj.location = [121.49328325, 31.23863899, 10];   //get/set
obj.rotator = {pitch: 0, yaw: 60, roll: 0}   //get/set
obj.scale3d = [5,5,5];   //get/set
obj.bVisible = boolean;   //get/set
obj.entityName = '';   //get/set
obj.customId = '';   //get/set
obj.customData = {};   //get/set
obj.Delete();
obj.onClick(ev => {
    const newObj = ev.result.object;
    console.log(ev);
})
obj.onMouseEnter()(ev => {
    console.log(ev);
})
obj.onMouseOut()(ev => {
    console.log(ev);
})
```

- **Update(data)**

Purpose: Update the Text3D entity

```
data is the same as the parameters used in Add
```

- **Get/SetLocation(jsondata)**

Purpose: Get/Set the location of the Text3D entity

```
const jsondata = [121.48073857, 31.22738813, 67]
```

- **Get/SetRotator(jsondata)**

Purpose: Get/Set the rotation of the Text3D entity

Applicable to individual 3D objects (Particle, Text3D, Viewshed, Effects, Light, model)

```
const jsondata = {
    "pitch": 0, // Pitch angle, reference (-180~180)
    "yaw": 30, // Yaw angle, reference (-180~180)
    "roll": 0 // Roll angle, reference (-180~180)
}
```

- **Get/SetScale3d(jsondata)**

Purpose: Get/Set the scale of the Text3D entity

Applicable to individual 3D objects (Particle, Text3D, Viewshed, Effects, Light, model)

```
const jsondata = [200, 200, 200]; // Scale ratio (x, y, z) axes
```

- **Get/SetLocked(boolean)**

Purpose: Get/Set the locked/unlocked state of the Text3D entity

Locked entities cannot be clicked or box-selected

```
true: locked; false: unlocked
```

- **Get/SetVisible(boolean)**

Purpose: Get/Set the visibility of the Text3D entity

```
true: visible; false: hidden
```

- **Get/SetEntityName(string)**

Purpose: Get/Set the Text3D EntityName (according to business needs)

- **Get/SetCustomId(string)**

Purpose: Get/Set the Text3D CustomId (according to business needs)

- **Get/SetCustomData(json)**

Purpose: Get/Set the Text3D CustomData (according to business needs)

- **Get()**

Purpose: Get the Text3D information

- **oType**

Purpose: Get the type of the Text3D entity

- **bRemoved**

Purpose: Get whether the Text3D entity has been deleted

- **Delete()**

Purpose: Delete the Text3D entity

Events

- **onClick()**
- **onMouseEnter()**
- **onMouseOut()**

Set Entity Outline/Highlight Style

```
async function SetVisualStyle() {
  // Custom outline and highlight style (styleName, hexa color; alpha: highlight effective)
  App.Setting.SetVisualColorStyle('styleName', "0f5dff4c")

  const style = await App.Setting.GetVisualColorStyle();
  console.log(style);


  // Set outline thickness
  App.Setting.SetOutlineThickness(2);  // Minimum value: 2

  const thickness= await App.Setting.GetOutlineThickness();
  console.log(thickness);

}
```

Set Entity Outline

```
async function SetEntityOutline() {
  // Added entity overlays/model objects
  const entityObj1 = cache.get('text3d'),
        entityObj2 = cache.get('particle');

  entityObj1.SetEntityOutline({
    bHighlight: true,
    styleName: "Blue"
  })


  App.Scene.SetEntityOutline({
    entities: [entityObj1,entityObj2],
    bOutline : true,
    styleName: "Blue"
  });
}
```

Set Entity Highlight

```
async function SetEntityHighlight() {
  // Added entity overlays/model objects
  const entityObj1 = cache.get('text3d'),
        entityObj2 = cache.get('particle');

  entityObj1.SetEntityHighlight({
    bHighlight: true,
    styleName: "Blue"
  })

  App.Scene.SetEntityHighlight({
    entities: [entityObj1,entityObj2],
    bHighlight: true,
    styleName: "Blue"
  })

}
```

styleName Color List

	styleName	hexa	color
0	Default	FFBF0077	
1	Black	00000077	
2	DarkBlue	00008B77	

3	MediumBlue	0000CD77	
4	Blue	0000FF77	
5	DarkGreen	00640077	
6	Green	00800077	
7	SpringGreen	00FF7F77	
8	MidnightBlue	19197077	
9	ForestGreen	228B2277	
10	SeaGreen	2E8B5777	
11	LimeGreen	32CD3277	
12	RoyalBlue	4169E177	
13	SteelBlue	4682B477	
14	Maroon	80000077	
15	Purple	80008077	
16	Olive	80800077	
17	Gray	80808077	
18	SkyBlue	87CEEB77	
19	BlueViolet	8A2BE277	
20	DarkRed	8B000077	
21	LightGreen	90EE9077	
22	MediumPurple	9370DB77	
23	DarkViolet	9400D377	
24	PaleGreen	98FB9877	
25	YellowGreen	9ACD3277	
26	Sienna	A0522D77	

27	Brown	A52A2A77	
28	DarkGray	A9A9A977	
29	LightBlue	ADD8E677	
30	GreenYellow	ADFF2F77	
31	PowderBlue	B0E0E677	
32	Silver	C0C0C077	
33	IndianRed	CD5C5C77	
34	Chocolate	D2691E77	
35	LightGray	D3D3D377	
36	Thistle	D8BFD877	
37	Orchid	DA70D677	
38	GoldenRod	DAA52077	
39	Plum	DDA0DD77	
40	LightCyan	E0FFFF77	
41	DarkSalmon	E9967A77	
42	Violet	EE82EE77	
43	LightCoral	F0808077	
44	Wheat	F5DEB377	
45	Salmon	FA807277	
46	Linen	FAF0E677	
47	DeepPink	FF149377	
48	OrangeRed	FF450077	
49	Tomato	FF634777	
50	HotPink	FF69B477	

51	Coral	FF7F5077	
52	DarkOrange	FF8C0077	
53	LightSalmon	FFA07A77	
54	Orange	FFA50077	
55	LightPink	FFB6C177	
56	Pink	FFC0CB77	
57	Gold	FFD70077	
58	FloralWhite	FFFAF077	
59	Snow	FFFAFA77	
60	Yellow	FFFF0077	
61	LightYellow	FFFFE077	
62	Ivory	FFFFF077	
63	White	FFFFFF77	

Camera focus

```
const jsondata = {
    "rotation": {
        "pitch": -30, //-90~0)
        "yaw": 0, //-180~180; 0:East; 90:South; -90:North)
    }
}
```

Parameter Description:

Parameter	Range	Note

rotation	pitch	[-90~0]	
	yaw	[-180~180]	0:East; 90:South; -90:North

Entity

```
const jsondata = {
    "rotator": {
      "pitch": 0, //(-180~180)
      "yaw": 30, //(-180~180)
      "roll": 0 //(-180~180)
    }
}
```

Parameter Description:

Parameter		Range	Note
rotator	pitch	[-180~180]	
	yaw	[-180~180]	
	roll	[-180~180]	

Entity Type Correspondence

Applicable to API operations on demand types in common behaviors; as follows:

```
await App.Scene.GetByTypes(types);
await App.Scene.ClearByTypes(types);
```

Type	Remarks
Tiles	Baseboard Layer

Static	Static Model
Skeletal	Skeletal Model
Hierarchy	Constructive Model
ISEHierarchy	ISE Constructive Model
ModelerWater	Water Surface
ModelerRiver	Riverbank
ModelerEmbank	Embankment
Vegetation	Vegetation Area
CameraStart	Initial Camera State
CameraRoam	Camera Roaming
Bound	Moving Objects Along Path
Environment	Environment
RealTimeVideo	Real-time Video
Window	Window
Poi	POI (Point of Interest)
Particle	Particle Effect
Effects	Particle Effects
Text3D	3D Text
Light	Light
Viewshed	Viewshed
Path	Path
Parabola	Migration Chart
Range	Area Outline
HeatMap	Heat Map
ColumnarHeatMap	Columnar Heat Map
SpaceHeatMap	Point Cloud Heat Map
RoadHeatMap	Path Heat Map
Raster	Raster Image
HighlightArea	Highlight Area

Get Entities by [Type]

```
async function GetByTypes() {
  const types = ['Particle', 'Path'];
  const { result } = await App.Scene.GetByTypes(types);
  console.log(result);
}

// types
/*
  Static          Static model
  Skeletal        Skeletal model
  Hierarchy       Constructive Model
  ISEHierarchy    ISE Constructive Model
  Effects         Particle Effects

  RealTimeVideo    Real-time video
  Window           Window
  Poi              POI
  Particle         Particle
  Effects          Particle effects
  Text3D           3D Text
  Light            Light
  Viewshed         Viewshed
  Path             Path
  Parabola         Parabola
  Range            Range outline
  HeatMap          Heatmap
  ColumnarHeatMap  Columnar heatmap
  SpaceHeatMap     Point cloud heatmap
  RoadHeatMap      Path heatmap
  Raster           Raster
  HighlightArea    Highlight area
*/
```

Get Entities by EntityName

```
async function GetByEntityName() {
  // BasicInfoAtom: { "entityName": "Commercial Office Building", "customId": "myId", "customData": "{'data':'myCustomData'}" }

  const res = await App.Scene.GetByEntityName(["Commercial Office Building"]);
  console.log(res);
}
```

Get Entities by CustomId

```
async function GetByCustomId() {
  // BasicInfoAtom: { "entityName": "Commercial Office Building", "customId": "myId", "customData": "{'data':'myCustomData'}" }

  const res = await App.Scene.GetByCustomId(["myId"]);
  console.log(res);
}
```

Get Entities by Eids

```
async function GetByEids() {
  const Eids = [
    '-9151314316185345952',
    '-9151314316965221260',
    '-9151314316350575262'
  ];

  const res = await App.Scene.GetByEids(Eids);
  console.log(res);
}
```

Get All Entity Objects

```
async function GetAll() {
    const { result } = await App.Scene.GetAll();

  // Example: Get AES static model information
  const modelObj = result?.Static?.[0];
  const model = await modelObj.Get();
  console.log(model);

  // Example: Hide AES static model
  modelObj.SetVisible(false); // true: Show; false: Hide


  // Example: Update Path
  for (i = 0; i < result?.Path.length; i++) {
    const pathObj = result.Path[i];
    pathObj.Update({
      "pathStyle": {
        "type": "solid",
        "width": 20,
        "color": "ffadfbff",
        "passColor": "29ff52ff"
      }
    }, {
      calculateCoordZ: {  //Coordinate type and coordinate height; [Optional] Highest priority
        coordZRef: "surface",  //surface: Surface; ground: Ground; altitude: Altitude
        coordZOffset: 50
      }
    })
  }
}

/* Callback entity types
  Tiles            Baseboard layer
  Static           Static model
  Skeletal         Skeletal model
  Hierarchy        Constructive Model
  ISEHierarchy     ISE Constructive Model
  Effects          Particle Effects
  CameraStart      Initial camera state

  RealTimeVideo    Real-time video
  Window            Window
  Poi               POI
  Particle          Particle
  Effects           Particle effects
  Text3D            3D Text
  Light             Light
  Viewshed          Viewshed
  Path              Path
  Parabola          Parabola
  Range             Range outline
  HeatMap           Heatmap
  ColumnarHeatMap   Columnar heatmap
  SpaceHeatMap      Point cloud heatmap
  RoadHeatMap       Path heatmap
  Raster            Raster
  HighlightArea     Highlight area
*/
```

Entity Landing

```
async function SetSnapTo() {
  // Example: Particle landing
  // particleObj is the object created when new App.Particle({...});

  const res = await particleObj.SnapTo({
    calculateCoordZ: {
      coordZRef: "ground", //surface: Surface; ground: Ground; altitude: Altitude
      coordZOffset: 10 //Height (unit: meter)
    }
  });
}
```

Show/Hide Entities by [Objects]

```
async function SetVisibleByObjects() {
  const objs = [ // Entity objects
    particleObj, pathObj
  ];
  const res = await App.Scene.SetVisibleByObjects(objs,false);
  //true: Show; false: Hide
}
```

Delete Entities by [Type]

```
async function ClearByTypes() {
  const types = ["Particle", "Range"];
  const res = await App.Scene.ClearByTypes(types);

/* types (Note the case)
   Static          Static model
   Skeletal        Skeletal model
   Hierarchy       Constructive Model
   ISEHierarchy    ISE Constructive Model
   Effects         Particle Effects

   RealTimeVideo    Real-time video
   Window           Window
   Poi              POI
   Particle         Particle
   Effects          Particle effects
   Text3D           3D Text
   Light            Light
   Viewshed         Viewshed
   Path             Path
   Parabola         Parabola
   Range            Range outline
   HeatMap          Heatmap
   ColumnarHeatMap  Columnar heatmap
   SpaceHeatMap     Point cloud heatmap
   RoadHeatMap      Path heatmap
   Raster           Raster
   HighlightArea    Highlight area
*/

}
```

Delete Entities by [Objects]

```
async function ClearByObjects() {
  const objs = [ // Entity objects
    particleObj, pathObj
  ];
  const res = await App.Scene.ClearByObjects(objs);
}
```

Delete Entities by [Eids]

```
async function ClearByEids() {
  const Eids = [
      '-9151314316185345952',
      '-9151314316965221260',
     '-9151314316350575262'
  ];
  const res = await App.Scene.ClearByEids(Eids);
}
```

Get entities by Eids

```
async function GetByEids() {

  const res= await App.Scene.GetByEids(["-91513143085896763520","-91513143077246328320"]);
  console.log(res);

  const obj = res.result[0];
  console.log(await obj.Get());
}
```

callback:

```
{
    success: true,
    message: '',
    result: [obj, ...]
}
```

Delete Entities by Eids

```
async function ClearByEids() {
  const Eids = [
      '-9151314316185345952',
      '-9151314316965221260',
     '-9151314316350575262'
  ];
  const res = await App.Scene.ClearByEids(Eids);
}
```

Get entities by EntityName

```
async function GetByEntityName() {
  // BasicInfoAtom: { "entityName": "商业办公楼", "customId": "myId", "customData": "{'data':'myCustomData'}" }

  const res= await App.Scene.GetByEntityName(["商业办公楼"]);
  console.log(res);

  const obj = res.result[0];
  console.log(await obj.Get());
}
```

Update entities by EntityName

```
async function UpdateByEntityName() {
  // 此示例通过EntityName更新路径(path)样式
  const jsondata = {
    "pathStyle": {
      "type": "solid",
      "width": 120,
      "color": "aa6afeff",
      "passColor": "ffe077ff"
    }
  }

  const EntityName = ["myName1", "myName2"];
  const res = await App.Scene.UpdateByEntityName(EntityName, jsondata);
}
```

Get entities by EntityNames

```
async function UpdateByEntityNames() {
  // 此示例通过entityNames更新路径(path)样式
  const jsondata = [
    {
      "pathStyle": {
        "type": "solid",
        "width": 120,
        "color": "c4ff5bff",
        "passColor": "ff1bc8ff"
      },
      "entityName": "myName1" //分类标识
   },
   {
      "pathStyle": {
        "type": "arrow",
        "width": 200,
        "color": "e2faffff",
        "passColor": "faff7dff"
      },
      "entityName": "myName2"
    }
  ]

  const res = await App.Scene.UpdateByEntityNames(jsondata);
}
```

Focus to entities by EntityNames

```
async function FocusByEntityName() {
  const jsondata = {
    "rotation": {
        "pitch": -30, //俯仰角, 参考(-90~0)
        "yaw": 0 //偏航角, 参考(-180~180; 0:东; 90:南; -90:北)
    },
    "distanceFactor": 0.5, //参数范围[0.1~1]; 实体占满屏幕百分比
    "flyTime": 1, //过渡时长(单位:秒)
  }

  const EntityName = ["myName1", "myName2"];
  const res = await App.CameraControl.FocusByEntityName(EntityName, jsondata);
}
```

Delete entities by EntityNames

```
async function ClearByEntityName() {
  const EntityName = ["myName1", "myName2"];
  const res = await App.Scene.ClearByEntityName(EntityName);
}
```

Get entities by CustomId

```
async function GetByCustomId() {
  // BasicInfoAtom: { "entityName": "商业办公楼", "customId": "myId", "customData": "{'data':'myCustomData'}" }

   const res= await App.Scene.GetByCustomId(["myId"]);
   console.log(myId);

   const obj = res.result[0];
   console.log(await obj.Get());
}
```

Update entities by CustomId

```
async function UpdateByCustomId() {
  // update path style by customid
  const jsondata = {
    "pathStyle": {
      "type": "solid",
      "width": 120,
      "color": "aa6afeff",
      "passColor": "ffe077ff"
    }
  }

  const customId = ["myId1", "myId2"];
  const res = await App.Scene.UpdateByCustomId(customId, jsondata);
}
```

Update entities by CustomIds

```
async function UpdateByCustomIds() {
  // update path style by customid
  const jsondata = [
    {
      "pathStyle": {
        "type": "solid",
        "width": 120,
        "color": "c4ff5bff",
        "passColor": "ff1bc8ff"
      },
      "customId": "myId1"
   },
   {
      "pathStyle": {
        "type": "arrow",
        "width": 200,
        "color": "e2faffff",
        "passColor": "faff7dff"
      },
      "customId": "myId2"
    }
  ]

  const res = await App.Scene.UpdateByCustomIds(jsondata);
}
```

Focus to entities by CustomId

```
async function FocusByCustomId() {
  const jsondata = {
    "rotation": {
        "pitch": -30, // Pitch, reference range (-90~0)
        "yaw": 0 // Yaw, reference range (-180~180; 0: East, 90: South, -90: North)
    },
    "distanceFactor": 0.4, // Parameter range [0.1~1]; Percentage of the screen occupied by the entity
    "flyTime": 1, // Transition duration (unit: seconds)
  }

  const customId = ["myId1", "myId2"];
  const res = await App.CameraControl.FocusByCustomId(customId, jsondata);
}
```

Delete entities by CustomId

```
async function ClearByCustomId() {
  const customId = ["myId1", "myId2"];
  const res = await App.Scene.ClearByCustomId(customId);
}
```

Get entities within the screen

```
async function PickScreenEntity() {
  const res = await App.Tools.Picker.GetEntitiesInViewport(['Particle'], false);
  console.log(res);
  // Parameter one: Filter entity types
  // Parameter two: Inverse filtering (true/false)

  // Example
  const entityObj = res.result.objects[0];
  console.log(await entityObj.Get());
}

/* Entity types (pay attention to case)
  Static           Static model
  Skeletal         Skeletal model
  Hierarchy       Constructive Model
  ISEHierarchy    ISE Constructive Model
  Effects         Particle Effects

  RealTimeVideo    Real-time video
  Window           Window
  Poi              POI
  Particle         Particle effect
  Effects          Particle effects
  Text3D           3D Text
  Light            Light
  Viewshed         Viewshed
  Path             Path
  Parabola         Parabola
  Range            Range outline
  HeatMap          Heat map
  ColumnarHeatMap  Columnar heat map
  SpaceHeatMap     Point cloud heat map
  RoadHeatMap      Path heat map
*/
```

Get entities within a specified area of the screen.

```
async function PickEntity() {
  const res = await App.Tools.Picker.PickEntityByRectangle({
    p0: [0,0], // Pixel coordinates of the upper left corner of the screen
    p1: [window.innerWidth, window.innerHeight], // Pixel coordinates of the lower right corner of the screen
    bMustBeFullyEnclosed: false,
    entityTypeFilter: ["Path"], // Entity type
    bFilterForExclude: false, // Inverse filtering (true/false)
    selectMode: 'New' // ['New' (single selection), 'Add' (add selection), 'Subtract' (subtract selection), 'Reverse' (reverse selection)]
  })

  console.log(res);

  const entityObj = res.result[0];
  console.log(await entityObj.Get());
}

/* Entity types (pay attention to case)
  Static           Static model
  Skeletal         Skeletal model
  Hierarchy       Constructive Model
  ISEHierarchy    ISE Constructive Model
  Effects         Particle Effects

  RealTimeVideo    Real-time video
  Window           Window
  Poi              POI
  Particle         Particle effect
  Effects          Particle effects
  Text3D           3D Text
  Light            Light
  Viewshed         Viewshed
  Path             Path
  Parabola          Parabola
  Range             Range outline
  HeatMap           Heat map
  ColumnarHeatMap   Columnar heat map
  SpaceHeatMap      Point cloud heat map
  RoadHeatMap       Path heat map
*/
```

Get entities by mouse selection

```
async function RectPickEntityAction() {
  await App.Tools.Picker.StartRectPick({
    "bMustBeFullyEnclosed": true,
    "entityTypeFilter": [], // Filter entity types
    "bFilterForExclude": false, // Inverse filtering for entityTypeFilter (true/false)
    "selectMode": "New", // ['None', 'New' (single selection), 'Add' (add selection), 'Subtract' (subtract selection), 'Reverse' (reverse selection)]
    "highlightColor": "2afe17",
    "rectangleStyle": {
      "borderColor": "d9ff88", // Selection box color
      "borderThickness": 1 // Selection box line width
    }
  })
}

/* entityTypeFilter (pay attention to case)
  Tiles:            Base layer
  Static            Static model
  Skeletal          Skeletal model
  Hierarchy         Constructive Model
  ISEHierarchy      ISE Constructive Model
  Effects           Particle Effects

  RealTimeVideo     Real-time video
  Window            Window
  Poi               POI
  Particle          Particle effect
  Effects           Particle effects
  Text3D          3D Text
  Light           Light
  Viewshed        Viewshed
  Path            Path
  Parabola        Parabola
  Range           Range outline
  HeatMap         Heat map
  ColumnarHeatMap Columnar heat map
  SpaceHeatMap    Point cloud heat map
  RoadHeatMap     Path heat map
*/
```

End selection by mouse.

```
async function EndRectPickEntityAction () {
   await App.Tools.Picker.EndRectPick();
}
```

Set behavior to get highlighted entities by mouse click.

```
async function PickEntityByClick() {
  const jsondata = {
    "pickFilter": { //[Optional] Entity filtering
      "filterEntityTypes": [], // Filter entity types
      "bFilterForExclude": false, // Inverse filtering for filterEntityTypes (true/false)
      "excludeEntities": [] // Exclude entity objects
      },
    "selectionMode": "New" // ['None', 'New' (single selection), 'Add' (add selection), 'Subtract' (subtract selection), 'Reverse' (reverse selection)]
  }

  await App.Setting.SetDefaultActionSetting(jsondata);
}

/* filterEntityTypes (pay attention to case)
  Tiles:          Base layer
  Static          Static model
  Skeletal         Skeletal model
  Hierarchy        Constructive Model
  ISEHierarchy     ISE Constructive Model
  Effects          Particle Effects

  RealTimeVideo    Real-time video
  Window           Window
  Poi              POI
  Particle         Particle effect
  Effects          Particle effects
  Text3D           3D Text
  Light            Light
  Viewshed         Viewshed
  Path             Path
  Parabola         Parabola
  Range            Range outline
  HeatMap          Heat map
  ColumnarHeatMap  Columnar heat map
  SpaceHeatMap     Point cloud heat map
  RoadHeatMap      Path heat map
*/
```

Gizimo operation

```
async function SetGizmoSetting() {

  /* Overlay entity Gizmo:
    1. Add entities to the scene (e.g., Particle (effects)).
    2. Enable Gizmo mode and entity selection.
    3. Select the added particle, and the entity enters Gizmo mode automatically.
    Note: Poi, Particle (effects), Text3D (3D text), Viewshed (viewshed) are valid.
  */

  /* AES model Gizmo:
    1. Enable Gizmo mode and entity selection.
    2. Select the AES model, and the model enters Gizmo mode automatically.
  */
  // Enable Gizmo mode
  await App.Setting.SetGizmoSetting({
      gizmoState: "Enable", // Toggle Gizmo (Enable/Disable)
      gizmoCoordinateSystem: "World", // World coordinate system
      bPreserveScaleRatio: true // Preserve entity scaling ratio
  })

  // Enable entity selection
  await App.Setting.SetDefaultActionSetting( {
    "selectionMode": "New"
  });

}
```

Set entity outline/highlight style

```
async function SetVisualStyle() {
  // Customizing outline/highlight style (styleName, hexa color; alpha: highlight effect)
  App.Setting.SetVisualColorStyle('styleName', "0f5dff4c");

  const style = await App.Setting.GetVisualColorStyle();
  console.log(style);

  // Setting outline thickness
  App.Setting.SetOutlineThickness(2);  // Minimum value: 2

  const thickness= await App.Setting.GetOutlineThickness();
  console.log(thickness);
}
```

Set entity outline

```
async function SetEntityOutline() {
  // Added model/entity/layer
  const entityObj1 = cache.get('text3d'),
        entityObj2 = cache.get('particle');

  entityObj1.SetEntityOutline({
    bHighlight: true,
    styleName: "Blue"
  })


  App.Scene.SetEntityOutline({
    entities: [entityObj1,entityObj2],
    bOutline : true,
    styleName: "Blue"
  });

}
```

Set entity highlight

```
async function SetEntityHighlight() {
  //  Added model/covering
  const entityObj1 = cache.get('text3d'),
        entityObj2 = cache.get('particle');

  entityObj1.SetEntityHighlight({
    bHighlight: true,
    styleName: "Blue"
  })

  App.Scene.SetEntityHighlight({
    entities: [entityObj1,entityObj2],
    bHighlight: true,
    styleName: "Blue"
  })

}
```

styleName colorlist

	styleName	hexa	color
0	Default	FFBF0077	
1	Black	00000077	
2	DarkBlue	00008B77	
3	MediumBlue	0000CD77	
4	Blue	0000FF77	
5	DarkGreen	00640077	
6	Green	00800077	
7	SpringGreen	00FF7F77	
8	MidnightBlue	19197077	
9	ForestGreen	228B2277	
10	SeaGreen	2E8B5777	
11	LimeGreen	32CD3277	
12	RoyalBlue	4169E177	
13	SteelBlue	4682B477	
14	Maroon	80000077	

15	Purple	80008077	
16	Olive	80800077	
17	Gray	80808077	
18	SkyBlue	87CEEB77	
19	BlueViolet	8A2BE277	
20	DarkRed	8B000077	
21	LightGreen	90EE9077	
22	MediumPurple	9370DB77	
23	DarkViolet	9400D377	
24	PaleGreen	98FB9877	
25	YellowGreen	9ACD3277	
26	Sienna	A0522D77	
27	Brown	A52A2A77	
28	DarkGray	A9A9A977	
29	LightBlue	ADD8E677	
30	GreenYellow	ADFF2F77	
31	PowderBlue	B0E0E677	
32	Silver	C0C0C077	
33	IndianRed	CD5C5C77	
34	Chocolate	D2691E77	
35	LightGray	D3D3D377	
36	Thistle	D8BFD877	
37	Orchid	DA70D677	
38	GoldenRod	DAA52077	

39	Plum	DDA0DD77	
40	LightCyan	E0FFFF77	
41	DarkSalmon	E9967A77	
42	Violet	EE82EE77	
43	LightCoral	F0808077	
44	Wheat	F5DEB377	
45	Salmon	FA807277	
46	Linen	FAF0E677	
47	DeepPink	FF149377	
48	OrangeRed	FF450077	
49	Tomato	FF634777	
50	HotPink	FF69B477	
51	Coral	FF7F5077	
52	DarkOrange	FF8C0077	
53	LightSalmon	FFA07A77	
54	Orange	FFA50077	
55	LightPink	FFB6C177	
56	Pink	FFC0CB77	
57	Gold	FFD70077	
58	FloralWhite	FFFAF077	
59	Snow	FFFAFA77	
60	Yellow	FFFF0077	
61	LightYellow	FFFFE077	
62	Ivory	FFFFF077	

63	White	FFFFFF77	

Get all selected entities

```
async function GetSelection () {

  const res = await App.Scene.GetSelection();
  console.log(res);

}
```

Deselect entity

```
async function RemoveSelection () {
  const res = await App.Scene.GetSelection();
  const someObj = res.result;
  App.Scene.RemoveSelection(someObj);
}
```

Deselect all selected entities

```
async function ClearSelection () {

  App.Scene.ClearSelection();
  App.Setting.SetMode('Runtime');

}
```

Clip heatmap

```
async function SetClipHeatMap() {
  const geo = {
    "coordinates": [
      [
        [121.497223,31.251557,0],
        [121.476501,31.237163,0],
        [121.514851,31.245237,0]
      ]
    ]
  }

  // Clip with geo and optional hole color
  const clipRes = await cache.get('heatmap').Clip(geo, "bd20ffff");

  // Cancel clip
  const unclipRes = await cache.get('heatmap').UnClip();
}
```

Clip columnar heatmap

```
async function SetClipColumnarHeatMap() {
  const geo = {
    "coordinates": [
      [
        [121.497223,31.251557,0],
        [121.476501,31.237163,0],
        [121.514851,31.245237,0]
      ]
    ]
  }

// Clip with geo and optional hole color
  const res = await cache.get('columnarheatmap').Clip(geo, "fef595ff");

  // Cancel clip
  await cache.get('columnarheatmap').UnClip();
}
```

Modify path

```
async function ModifyPath () {
    const jsondata = {
        "method": "add", // add; delete
        "index": [2, 0], // Coordinate point index
        "coordinates": [
            [121.46164010, 31.22692061, 11],
            [121.46881138, 31.22606828, 68]
        ]
    }

    const res = await cache.get('path').Modify(jsondata);
    console.log(res)
}

/*
  method: "add" ::::::
  index: [n], Add coordinate points after index n+1;
  index: [n+1], Add coordinate points beyond the last coordinate point index;

  method: "delete" ::::::
  index: [2], Delete the 2nd coordinate point;
  index: [2,10], Delete data points between [2~10];
*/
```

Modify heatmap

```
async function ModifyHeatMap () {
    const mapdata = [],
        points = [
            [121.49378441, 31.22786931, 21],
            [121.48928671, 31.22207976, 14],
            [121.48146687, 31.25121877, 81],
            [121.46073678, 31.22045260, 42]
        ];
    for (let i = 0; i < points.length; i++) {
        mapdata.push({
            "point": points[i],
            "value": Math.floor(Math.random() * 100)
        })
    }

    const jsondata = {
        "method": "add", // add; delete
        "index": [2, 0], // Coordinate point index
        "features": mapdata
    }

    const res = await cache.get('heatmap').Modify(jsondata);
    console.log(res)
}

/*
  method: "add" ::::::
  index: [n], Add coordinate points after index n+1;
  index: [n+1], Add coordinate points beyond the last coordinate point index;

  method: "delete" ::::::
  index: [2], Delete the 2nd coordinate point;
  index: [2,10], Delete coordinate points between [2~10];
*/
```

Modify columnar heatmap

```
async function ModifyColumnarHeatMap () {
    const mapdata = [],
        points = [
            [121.49378441, 31.22786931, 21],
            [121.48928671, 31.22207976, 14],
            [121.48146687, 31.25121877, 81],
            [121.46073678, 31.22045260, 42]
        ];
    for (let i = 0; i < points.length; i++) {
        mapdata.push({
            "point": points[i],
            "value": Math.floor(Math.random() * 100)
        })
    }

    const jsondata = {
        "method": "add", // add; delete
        "index": [2, 0], // Coordinate point index
        "features": mapdata
    }

    const res = await cache.get('colheatmap').Modify(jsondata);
    console.log(res)
}

/*
  method: "add" ::::::
  index: [n], Add coordinate points after index n+1;
  index: [n+1], Add coordinate points beyond the last coordinate point index;

  method: "delete" ::::::
  index: [2], Delete the 2nd coordinate point;
  index: [2,10], Delete coordinate points between [2~10];
*/
```

Modify space heatmap

```
async function ModifySpaceHeatMap () {
    const mapdata = [],
        points = [
            [121.49378441, 31.22786931, 21],
            [121.48928671, 31.22207976, 14],
            [121.48146687, 31.25121877, 81],
            [121.46073678, 31.22045260, 42]
        ];
    for (let i = 0; i < points.length; i++) {
        mapdata.push({
            "point": points[i],
            "value": Math.floor(Math.random() * 100)
        })
    }

    const jsondata = {
        "method": "add", //add; delete
        "index": [2, 0], // Coordinate point index
        "features": mapdata
    }

    const res = await cache.get('spaceheatmap').Modify(jsondata);
    console.log(res)
}

/*
  method: "add" ::::::
  index: [n], Add coordinate points after index n+1;
  index: [n+1], Add coordinate points beyond the last coordinate point index;

  method: "delete" ::::::
  index: [2], Delete the 2nd coordinate point;
  index: [2,10], Delete coordinate points between [2~10];
*/
```

[Batch] Add Instances (Same type entities)

Example: Batch adding paths (path); Same type entities

```
async function BatchAdd () {
  const normal = { // ========== Common style
    "type": "Path",
    "entityName": "myName",  //optional
    "pathStyle": {
      "width": 100,
      "passColor": "dc0affff"
    }
  }
  const dataArr = [ // ========== Data array
    {
      "polyline": {
        "coordinates": [
          [121.48595648, 31.24834326, 30],
          [121.48600786, 31.24252899, 30],
          [121.50577283, 31.22653989, 30]
        ]
      },
      "customId": "myId1",
      "pathStyle": {
        "type": "arrow",
        "color": "78ffffff"
      }
    },
    {
      "polyline": {
        "coordinates": [
          [121.49709136, 31.22516669, 30],
          [121.49662428, 31.23543741, 30],
          [121.51043061, 31.22969411, 30]
        ]
      },
      "customId": "myId2",
      "pathStyle": {
        "type": "solid",
        "color": "52f1feff"
      }
    }
  ]

  const res = await App.Scene.Create(normal, dataArr,
    {  // [optional] Coordinate type and height; Highest priority
      calculateCoordZ: {
        coordZRef: "surface", //surface; ground; altitude
        coordZOffset: 50 //height (unit: meters)
      }
    }
  )
}
```

[Batch] Add Instances (different types of entities)

Example: Batch adding different types of entities

```
async function BatchAdd3 () {
  const jsondata = [
    //====== Window
    {
      "type": "Window",
      "location": [121.46426478,31.22406702,47],
      "windowStyle": {
        "url": "http://wdpapi.51aes.com/doc-static/images/static/echarts.html",
        "size": [500, 350],
        "offset": [0, 0]
      },
      "bVisible": true,
      "entityName": "myName1",
      "customId": "myId1",
      "customData": {
        "data": "Window"
      }
    },

    //====== Poi
    {
      "type": "Poi",
      "location": [121.46491415,31.21866105,87],
      "poiStyle": {
        "markerNormalUrl": "http://wdpapi.51aes.com/doc-static/images/static/markerNormal.png",
        "markerActivateUrl": "http://wdpapi.51aes.com/doc-static/images/static/markerActive.png",
        "markerSize": [100, 228],
        "labelBgImageUrl": "http://wdpapi.51aes.com/doc-static/images/static/LabelBg.png",
        "labelBgSize": [200, 50],
        "labelBgOffset": [50, 200],
        "labelContent": [" 文本内容A", "ff0000ff", "24"]
      },
      "bVisible": true,
      "entityName": "myName2",
      "customId": "myId2",
      "customData": {
        "data": "Poi"
      }
    },

    //====== Particle
    {
      "type": "Particle",
      "location": [121.49172858,31.22476437,67],
      "rotator": {
        "pitch": 0,
        "yaw": 30,
        "roll": 0
      },
      "bVisible": true,
      "scale3d": [50, 50, 50],
      "particleType": "vehicle_taxi",
      "entityName": "myName3",
      "customId": "myId3",
      "customData": {
        "data": "Particle"
      }
    },

    //====== Text3D
    {
      "type": "Text3D",
      "location": [121.46376561,31.22870602,76],
      "rotator": {
        "pitch": 0,
        "yaw": 30,
        "roll": 0
      },
      "scale3d": [1000, 100, 100],
      "text3DStyle": {
        "text": "3D文字",
        "color": "10ff1bff",
        "type": "plain",
        "outline": 0.4,
        "portrait": false,
        "space": 0.1
      },
      "bVisible": true,
      "entityName": "myName4",
      "customId": "myId3",
      "customData": {
        "data": "Text3D"
      }
    },

    //====== Viewshed
    {
      "type": "Viewshed",
      "location": [121.47315875,31.24472542,27],
      "rotator": {
        "pitch": 0,
        "yaw": 30,
        "roll": 0
      },
      "viewshedStyle": {
        "fieldOfView": 70,
        "radius": 600,
        "outline": true,
        "hiddenColor": "ff136dff",
        "visibleColor": "feaecfff"
      },
      "bVisible": true,
      "entityName": "myName5",
      "customId": "myId5",
      "customData": {
        "data": "Viewshed"
      }
    },

    //====== Path
    {
      "type": "Path",
      "polyline": {
        "coordinates": [
          [121.50114770,31.23691142,93],
          [121.48007773,31.22050415,27],
          [121.47985493,31.24031196,45],
          [121.49030648,31.23537047,33]
        ]
      },
      "pathStyle": {
        "type": "arrow",
        "width": 100,
        "color": "eaffc7ff",
        "passColor": "ff6d96ff"
      },
      "bVisible": true,
      "entityName": "myName6",
      "customId": "myId6",
      "customData": {
        "data": "Path"
      }
    },

    //====== Parabola
    {
      "type": "Parabola",
      "polyline": {
        "coordinates": [
          [121.47607446,31.24372538,84],
          [121.48749492,31.23361823,8]
        ]
      },
      "parabolaStyle": {
        "topHeight": 800,
        "topScale": 1,
        "type": "scanline",
        "width": 20,
        "color": "b1ff8bff",
        "gather": true
      },
      "bVisible": true,
      "entityName": "myName7",
      "customId": "myId7",
      "customData": {
        "data": "Parabola"
      }
    },

    //====== Range
    {
      "type": "Range",
      "polygon2D": {
        "coordinates": [
          [
            [121.47122131,31.24264779],
            [121.48769236,31.23035225],
            [121.50016626,31.22821735]
          ]
        ]
      },
      "rangeStyle": {
        "type": "loop_line",
        "fillAreaType": "block",
        "height": 200,
        "strokeWeight": 10,
        "color": "6fff46ff"
      },
      "bVisible": true,
      "entityName": "myName8",
      "customId": "myId8",
      "customData": {
        "data": "Range"
      }
    },
    //====== HeatMap
    {
      "type": "HeatMap",
      "heatMapStyle": {
        "type": "fit",
        "brushDiameter": 2000,
        "mappingValueRange": [1, 100],
        "gradientSetting": [
          "b4ff25ff", "a174ffff", "2e15feff", "d0ff30ff", "b3ffa4ff"
        ]
      },
      "bVisible": true,
      "entityName": "myName9",
      "customId": "myId9",
      "customData": {
        "data": "HeatMap"
      },
      "points": {
        "features": [
          {
            "point": [121.48783892,31.22955413,91],
            "value": 87
          },
          {
            "point": [121.49360144,31.23134998,41],
            "value": 80
          },
          {
            "point": [121.46524329,31.24312496,77],
            "value": 95
          },
          {
            "point": [121.48712254,31.24286490,34],
            "value": 70
          },
          {
            "point": [121.47944776,31.24252262,86],
            "value": 65
          }
        ]
      }
    },
    //====== ColumnarHeatMap
    {
      "type": "ColumnarHeatMap",
      "columnarHeatMapStyle": {
        "type": "cube",
        "brushDiameter": 1000,
        "mappingValueRange": [1, 100],
        "columnarWidth": 20,
        "mappingHeightRange": [0, 500],
        "enableGap": false,
        "gradientSetting": [
          "f7ffbfff", "ff0083ff", "8991ffff", "a0a7feff", "ff2131ff"
        ]
      },
      "bVisible": true,
      "entityName": "myName10",
      "customId": "myId10",
      "customData": {
        "data": "ColumnarHeatMap"
      },
      "points": {
        "features": [
          {
            "point": [121.49140589,31.25237391,61],
            "value": 87
          },
          {
            "point": [121.48047463,31.22617967,38],
            "value": 80
          },
          {
            "point": [121.48477522,31.23823883,84],
            "value": 95
          },
          {
            "point": [121.48203408,31.25113752,16],
            "value": 70
          },
          {
            "point": [121.49542774,31.23945532,13],
            "value": 65
          }
        ]
      }
    },

    //====== RoadHeatMap
    {
      "type": "RoadHeatMap",
      "roadHeatMapStyle": {
        "width": 50,
        "mappingValueRange": [1, 100],
        "gradientSetting": [
          "ffee1aff", "ffb540ff", "f4ff4bff", "d961feff", "b456ffff"
        ],
        "type": "plane",
        "filter": []
      },
      "bVisible": true,
      "entityName": "myName11",
      "customId": "myId11",
      "customData": {
        "data": "RoadHeatMap"
      },
      "points": {
        "features": [
          {
            "point": [121.47799331,31.23097751,83],
            "value": 87
          },
          {
            "point": [121.49095564,31.22740179,57],
            "value": 80
          },
          {
            "point": [121.46961736,31.24484216,93],
            "value": 95
          },
          {
            "point": [121.49208500,31.25120664,52],
            "value": 70
          },
          {
            "point": [121.48651742,31.22413720,30],
            "value": 65
          }
        ]
      }
    },

    //====== SpaceHeatMap
    {
      "type": "SpaceHeatMap",
      "spaceHeatMapStyle": {
        "brushDiameter": 100,
        "mappingValueRange": [1, 100],
        "gradientSetting": [
          "0000ff", "ff5500", "00ff00", "ffff00", "00ffff"
        ]
      },
      "bVisible": true,
      "entityName": "myName12",
      "customId": "myId12",
      "customData": {
        "data": "SpaceHeatMap"
      },
      "points": {
        "features": [
          {
            "point": [121.48641573,31.23901035,87],
            "value": 87
          },
          {
            "point": [121.46117788,31.22258222,89],
            "value": 80
          },
          {
            "point": [121.47008568,31.22681936,13],
            "value": 95
          },
          {
            "point": [121.49895380,31.22317413,65],
            "value": 70
          },
          {
            "point": [121.47750177,31.22547035,94],
            "value": 65
          }
        ]
      }
    },

    //====== Range
   {
      "type": "Range",
      "circlePolygon2D": {
        "center": [121.49986350,31.24269398,49],
        "radius": 300
      },
      "rangeStyle": {
        "shape": "circle",
        "type": "grid",
        "fillAreaType": "radar",
        "height": 150,
        "strokeWeight": 10,
        "color": "2948feff"
      },
      "bVisible": true,
      "entityName": "myName13",
      "customId": "myId13",
      "customData": {
        "data": "CircleRange"
      }
    },

    //====== light
   {
      "type": "Light",
      "location": [121.49189794,31.24282414,0],
      "rotator": {
        "pitch": 0,
        "yaw": 0,
        "roll": 0
      },
      "scale3d": [100, 100, 100],
      "lightStyle": {
        "intensity": 40,
        "color": "ff00ff",
        "angle": 50,
        "attenuation": 200,
        "shadows": true,
        "haze": true,
        "haze_Intensity": 90
      },
      "bVisible": true,
      "entityName": "myName14",
      "customId": "myId14",
      "customData": {
        "data": "CircleRange"
      }
   }
  ]

  const res = await App.Scene.Creates(jsondata, {
   calculateCoordZ: {   // [optional] Coordinate type and height; Highest priority
      coordZRef: "surface", //surface; ground; altitude
      coordZOffset: 50 //height (unit: meters)
    }
  });
}
```

[Batch] Add Instances (multi objects)

```
async function BatchAdd2 () {
  const objs = [poiObject, pathObject, particleObject...];
  const res = await App.Scene.Add(objs, {
    calculateCoordZ: {   // Coordinate type and height; [optional] Highest priority
      coordZRef: "surface", //surface: surface; ground: ground; altitude: altitude
      coordZOffset: 50 //height (unit: meters)
    }
  });
}
```

[Batch] Set Instance Scale: Same Ratio

Scale multiple objects by the same ratio

```
async function SetScale3D () {
  const obj = [ // Single 3D entity objects (Particle, Text3D, Viewshed, Effects, Light, models)
    particleObj, text3dObj
  ];

  const res = await App.Scene.SetScale3D(obj, {
    x: 400, // Scale ratio
    y: 400,
    z: 400
  });
}
```

[Batch] Set Instance Scale: Different Ratios

Scale multiple objects by different ratios

```
async function SetScale3D () {
    const res = await App.Scene.SetScale3Ds([
        {
            object: cache.get('particle'),  // Single 3D entity objects (Particle, Text3D, Viewshed, Effects, Light, models)
            scale3d: { x: 200, y: 200, z: 200 }
        },
        {
            object: cache.get('text3d'),  // Single 3D entity objects (Particle, Text3D, Viewshed, Effects, Light, models)
            scale3d: { x: 400, y: 400, z: 400 }
        }
    ]);
}
```

[Batch] Set Instance Rotation: Same Angle

Rotate multiple objects to the same angle

```
async function SetRotator () {
  const obj = [ // Single 3D entity objects (Particle, Text3D, Viewshed, Effects, Light, models)
    particleObj, text3dObj
  ];

  const res = await App.Scene.SetRotator(obj, {
    "pitch": 0, // Pitch angle, reference (-180~180)
    "yaw": 60, // Yaw angle, reference (0: north, -180~180)
    "roll": 0 // Roll angle, reference (-180~180)
  });
}
```

[Batch] Set Instance Rotation: Different Angles

Rotate multiple objects to different angles

```
async function SetRotator () {
    const res = await App.Scene.SetRotators([
        {
            object: cache.get('particle'),  // Single 3D entity objects (Particle, Text3D, Viewshed, Effects, Light, models)
            rotator: { "pitch": 0, "yaw": 60, "roll": 0 }
        },
        {
            object: cache.get('text3d'),  // Single 3D entity objects (Particle, Text3D, Viewshed, Effects, Light, models)
            rotator: { "pitch": 0, "yaw": 30, "roll": 0 }
        }
    ]);
}
```

[Batch] Set Instance Location: Same Position

Move multiple objects to the same position

```
async function SetLocation () {
  const obj = [ // Single 3D entity objects (Particle, Text3D, Viewshed, Effects, Light, models)
    particleObj, text3dObj
  ];

  const res = await App.Scene.SetLocation(obj, {
    x: 121.48701062,
    y: 31.23299679,
    z: 100
  });
}
```

[Batch] Set Instance Location: Different Positions

Move multiple objects to different positions

```
async function SetLocation () {
    const res = await App.Scene.SetLocations([
        {
            object: cache.get('particle'),  // Single 3D entity objects (Particle, Text3D, Viewshed, Effects, Light, models)
            location: { x: 121.48814278, y: 31.22652611, z: 100 }
        },
        {
            object: cache.get('text3d'),   // Single 3D entity objects (Particle, Text3D, Viewshed, Effects, Light, models)
            location: { x: 121.48814278, y: 31.24652611, z: 100 }
        }
    ]);
}
```

[Batch] Set Instance Visibility

```
async function SetVisible () {
  const obj = [
    particleObj, text3dObj, pathObj  // Entity objects
  ];

  const res = await App.Scene.SetVisible(obj, false);
  // true: Show; false: Hide
}
```

[Batch] Set Instance Lock/Unlock

```
async function SetLocked() {
  const obj = [
    particleObj, text3dObj, pathObj  // Entity objects
  ];

  await App.Scene.SetLocked(obj, false);
  // true: Lock; false: Unlock
}
```

[Batch] Set Instance Update (same attributes)

```
// This example updates the path style in batch by objects
const jsondata = {
    "pathStyle": {
        "type": "solid",
        "width": 100,
        "color": "99ebffff",
        "passColor": "bdffedff"
    }
}

const obj = [
    path1Obj, path2Obj, path3Obj  // Same type entity objects
];

const res = await App.Scene.Update(obj, jsondata, {
    calculateCoordZ: {  // Coordinate type and coordinate height; [optional] Highest priority
        coordZRef: "surface", // surface: Surface; ground: Ground; altitude: Altitude
        coordZOffset: 50 // Height (unit: meters)
    }
});
```

[Batch] Set Instance Updates (multiple attributes)

```
const particleJson = {
  "rotator": {
    "pitch": 0,
    "yaw": 50,
    "roll": 0
  },
  "scale3d": [60, 60, 60],
  "particleType": "vehicle_car_black"
}

const pathJson = {
  "pathStyle": {
    "type": "solid",
    "width": 80,
    "color": "ff8881ff",
    "passColor": "fff848ff"
  }
}

const res = await App.Scene.Updates([
  { object: cache.get('particle'), jsonData: particleJson }, // {object: object, jsonData: entityJson}
  { object: cache.get('path'), jsonData: pathJson },
], {
  calculateCoordZ: { // Coordinate type and coordinate height; [optional] Highest priority
      coordZRef: "surface", // surface: Surface; ground: Ground; altitude: Altitude
      coordZOffset: 50 // Height (unit: meters)
  }
});

// cache.get('particle'), cache.get('path') The object that was cached when this entity was created
```

[Batch] Delete Instances

```
async function Delete () {
  const obj = [
    particleObj, text3dObj, pathObj  // Entity objects
  ];

  const res = await App.Scene.Delete(obj);
}
```

Entity Move Along Path

Create Bound Entity

```
new App.Bound({ "moving": movedObject,  "path": pathObject,  "boundStyle": { "time": 50,  "bLoop": true,  "bReverse": false, "state": 'play' }, "offset": {}, "rotator": {
```

```
async function CoveringMove () {
    // ========== Add Path
```

```
const { success, result: { object: pathObj } } = await App.Scene.Add(new App.Path({
    "polyline": {
        coordinates: [
            [121.48819022, 31.23292494, 50], [121.48539961, 31.23619302, 50], [121.47025042, 31.23065615, 50]
        ],
    },
    "pathStyle": {
        "type": "arrow", "width": 20, "color": "ff00ffff", "passColor": "00ff00ff"
    }
}));

// ========== Add Entity (Vehicle)
const { result: { object: moveObj } } = await App.Scene.Add(new App.Particle({
    "location": [121.48819022, 31.23292494, 89],
    "rotator": { "pitch": 0, "yaw": 0, "roll": 0 },
    "scale3d": [30, 30, 30],
    "particleType": "vehicle_taxi",
}));

// ========== Move Entity (Vehicle) along Path
const moveObj = new App.Bound({
    "moving": moveObj, // Moving entity object
    "path": pathObj, // Path object
    "boundStyle": {
        "time": 50, // Total duration (unit: seconds)
        "bReverse": false, // Whether to move in reverse (true/false); false to move forward, true to move reversely
        "bLoop": true, // Whether to loop (true/false); true to loop move, false to terminate when ends
        "bVisible": true, // Whether visible (true/false); controls both the visibility of entity and path
        "state": 'play' // play: move; pause: pause; stop: stop
    },
    "offset": { // Clears original settings, match automatically, and adjust site in offset
        "left": 0, // Adjust left and right relative to the path; left +, right - (unit: meters)
        "forward": 0, // Adjust front and back along the path; front +, back - (unit: meters)
        "up": 0 // Adjust perpendicular up and down relative to the path; up +, down - (unit: meters)
    },
    "rotator": { // Clears original settings, match automatically, and adjust angles in rotator
        "pitch": 0, // Pitch angle relative to path, up +, down -; Range [-180~180]
        "yaw": 30, // Yaw angle relative to path , left +, right -; Range [-180~180]
        "roll": 0 // Roll angle relative to path, left +, right -; Range [-180~180]
    }
});
```

```
    await App.Scene.Add(moveObj);
}
```

* When bLoop is false, after the entity moves to the end and stops, if you want to move from the beginning again, please update the state to stop before playing again.

Member Functions

```
// Example
const obj = new App.Bound({ ...});
obj.Update(json); // Same as parameters in Add
obj.SetTime(50);
obj.SetReverse(false); // Whether to move in reverse (true/false); false to move forward, true to move reversely
obj.SetLoop(true); // Whether to loop (true/false); true to loop move, false to terminate when ends
obj.SetState('pause'); // play: move; pause: pause; stop: stop
obj.SetOffset({
      "left": 0, // Adjust left and right relative to the path; left +, right - (unit: meters)
      "forward": 0, // Adjust front and back along the path; front +, back - (unit: meters)
      "up": 0 // Adjust perpendicular up and down relative to the path; up +, down - (unit: meters)
});
obj.SetRotator({
  "pitch": 0, // Pitch angle relative to path, up +, down -; Range [-180~180]
  "yaw": 30, //  Yaw angle relative to path , left +, right -; Range [-180~180]
  "roll": 0 // Roll angle relative to path, left +, right -; Range [-180~180]
});
obj.SetVisible(boolean);// Whether visible (true/false); controls both the visibility of entity and path
obj.Get();
obj.Delete();
```

- **Update(data)**

Purpose: Update Bound Entity

Parameter Description:

```
data() Same as parameters in Add
```

Data-driven entity movement

Specify destination points and movement time for objects. The objects will move from their current positions to the destination points over the specified time. When the array contains movement information for multiple different objects, they will start their first movement almost simultaneously. For each object, after reaching its destination point, it will automatically move to the next destination point.

```
async function MoveLinear () {
    const res = await App.Scene.Add(new App.Poi({
        "location": [121.49295594, 31.25143325, 4],
        "poiStyle": {
            "markerNormalUrl": "http://wdpapi.51aes.com/doc-static/images/static/markerNormal.png",
            "markerActivateUrl": "http://wdpapi.51aes.com/doc-static/images/static/markerActive.png",
            "markerSize": [50, 114],
            "labelBgImageUrl": "http://wdpapi.51aes.com/doc-static/images/static/LabelBg.png",
            "labelBgSize": [115, 22],
            "labelBgOffset": [25, 100],
            "labelContent": ["Data-driven Movement", "bc8affff", "12"]
        }
    }));

    const obj = res.result.object;

    const entityObj = [
        {
            objects: [obj],
            location: [121.50108135, 31.24556635, 71],
            time: 0 // Time required for the entity to move to this location
        },
        {
            objects: [obj],
            location: [121.51409703, 31.24399604, 31],
            time: 5
        },
        {
            objects: [obj],
            location: [121.51240709, 31.25260645, 0],
            time: 10
        },
        {
            objects: [obj],
            location: [121.50848679, 31.23589336, 3],
            time: 5
        },
        {
            objects: [obj],
            location: [121.48741336, 31.22861979, 47],
            time: 15
        },
    ]

    await App.Tools.MoveLinear.Move(entityObj, {
        "calculateCoordZ": {  // Coordinate type and coordinate height; [Optional] Highest priority
            "coordZRef": "surface",//surface: surface; ground: ground; altitude: altitude
            "coordZOffset": 50 // Altitude (in meters)
        }
    });
}
```

Entity Click Event

```
async function SetClickPath () {
    let flag = true, __winObj = null;

    //add path
    const entityObj = new App.Path({
        "polyline": {
            "coordinates": [
                [121.49921961, 31.23764884, 77],
                [121.46326121, 31.22644542, 31],
                [121.49408610, 31.24848319, 73]
            ]
        },
        "pathStyle": {
            "type": "arrow",
            "width": 100,
            "color": "ff2620ff",
            "passColor": "c117feff"
        }
    });
    await App.Scene.Add(entityObj).then(async res => {
        if (res.success) {
            // focus
            const jsondata = {
                "rotation": {
                    "pitch": -40,
                    "yaw": 0,
                },
                "distanceFactor": 0.8,
                "flyTime": 1,
                "entity": [entityObj]
            }
            await App.CameraControl.Focus(jsondata);

            res.result.object.onClick(async ev => {
                // click path, then update path style
                const upjsondata = {
                    "pathStyle": {
                        "type": "solid",
                        "width": 120,
                        "color": "ffd8e5ff",
                        "passcolor": "36fe7eff"
                    }
                }
                ev.result.object.Update(upjsondata);


                // click path, then add windowthrough toggling
                if (flag) {
                    flag = false;
                    const entityObj = new App.Window({
                        "location": ev.result.position,
                        "windowStyle": {
                            "url": "http://wdpapi.51aes.com/doc-static/images/static/echarts.html",
                            "size": [500, 350],
                            "offset": [0, 0]
                        }
                    })
                    const res = await App.Scene.Add(entityObj);
                    __winObj = res.result.object;
                } else {
                    __winObj.Delete();
                    flag = true;
                }
            })
        }
    })
}
```

Entity Hover Event

```
async function MouseEnterOut () {
  // Mouse hover events
  App.Renderer.UnRegisterSceneEvents([
    "OnMouseEnterEntity", "OnMouseOutEntity"
  ])
  App.Renderer.RegisterSceneEvents([
    { name: 'OnMouseEnterEntity', func: function (res) {} },
    { name: 'OnMouseOutEntity', func: function (res) {} }
  ])

  // Example: Add path entity
  const entityObj = new App.Path({
    "polyline": {
      "coordinates": [
        [121.47635644, 31.11820445, 25],
        [121.47139989, 31.11113569, 40],
        [121.50090474, 31.12182567, 80]
      ]
    },
    "pathStyle": {
      "type": "solid",
      "width": 50,
      "color": "0099ffff",
      "passColor": "e950ffff"
    }
  });

  await App.Scene.Add(entityObj).then(async res => {
    if (res.success) {
      // Mouse hover in
      res.result.object.onMouseEnter(async ev => {
        console.error("onMouseEnter", ev);
      })

      // Mouse hover out
      res.result.object.onMouseOut(async ev => {
        console.error("onMouseOut", ev);
      })

    }
  })
}
```

RealTimeVideo

Add RealTimeVideo object

```
new App.RealTimeVideo({ "location":  [121.50007292,31.22579403,30], "realTimeVideoStyle": style })
```

```
async function AddRealTimeVideo() {
  const realTimeVideo = new App.RealTimeVideo({
    "location": [121.50007292,31.22579403,30],
    "realTimeVideoStyle": {
      "url": "rtsp://admin:admin123456@121.63.247.105:20037/h264/ch1/sub/av_stream",
      "resolution": [400,300], //window size (unit:pixel)
      "offset": [0,0], //window offset relative to location(unit:pixel; x>0,y>0 shift to the right and up)
      "state": "pause",  //play; pause
      "overlapOrder": 1, // Overlapping hierarchy; the higher the value, the more floating on top; range [1~10]
      "bokeh": 0.5, //Edge enhancement (unit: proportion); range [0,1]
      "conrnerShift":[
          [40,-40],[0,-40],[10,0],[0,0]
       ] // Corner offset (unit: pixel); Fixed point order [top left, top right, bottom left, bottom right]; Each corner's original position is [0,0]; X>0 to the right, Y
    },
    "bVisible": true,
    "entityName": "myName",
    "customId": "myId1",
    "customData": {
      "data": "myCustomData"
    }
  })

  const res = await App.Scene.Add(realTimeVideo,{
    calculateCoordZ: {
      coordZRef: "surface", //surface:;ground;altitude
      coordZOffset: 50 //height(unit:meter)
    }
  });
}
```

Parameter description:

Parameter	Type	Required(default if optional)	Value Range
location	array	✓	
realTimeVideoStyle	object	✓	
url	string	✓	
size	array		positive integer
offset	array		[0,0]; x>0,y>0 offset to the right and up (x,y unit: pixel)
state	string		play, pause
overlapOrder	int		[1,10]; Overlapping hierarchy; the higher the value, the more floating on top
bokeh	number	[0,1]	Edge enhancement (unit: proportion)
conrnerShift	Array	[[0,0], [0,0], [0,0], [0,0]]	Corner offset (unit: pixel); Fixed point order [top left, top right, bottom left, bottom right]; Each corner's original position is [0,0]; X>0 to the right, Y>0 upwards
bVisible	boolean		true/false; Whether it is visible
entityName	string		
customId	string		
customData	object		
data	string		

Member Function

```
// example
  const obj = new App.RealTimeVideo({...});
  obj.Update(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
```

- **Update(data)**

Purpose: update RealTimeVideo state

Parameter description:

```
const data = {
  "realTimeVideoStyle": {
    "state": "pause"  //play; pause
  }
}
```

- **SetVisible()**

Purpose: show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose: get object data

- **Delete()**

Purpose: delete object

Window

Add Window object

```
new App.Window({ "location": [121.50007292,31.22579403,30], "windowStyle": style })
```

```
async function AddWindow() {
  const window = new App.Window({
    "location": [121.50007292,31.22579403,30],
    "windowStyle": {
      "url": "http://wdpapi.51aes.com/doc-static/images/static/echarts.html",
      "size": [500,350], //window size (unit:pixel)
      "offset": [0,0] //window offset relative to location(unit:pixel; x>0,y>0 shift to the right and up)
    },
    "entityName": "myName",
    "customId": "myId1",
    "customData": {
      "data": "myCustomData"
    },
    "bVisible": true,
    "visible2D": {
      "camera": {    //visibility affected by camera
        "hideDistance": 2000,  //define the entity hide distance between entity and camera
        "hideType": "default", // the display mode if entity is hided(none:not display; default:display as a circle)
        "scaleMode": "2D" //the display mode if the entity is shown(2D; 3D)
      },
 "entity": {
        "overlapOrder": 1 //Z-index; The higher the value, the more it floats to the top layer; Range [1~10].
      }
    }
  })

  const res = await App.Scene.Add(window,{
    calculateCoordZ: {
      coordZRef: "surface", //surface;ground;altitude
      coordZOffset: 50 //height(unit:meter)
    }
  });
}
```

Parameter description:

Parameter	Type	Required(default if optional)	Value Range

location	array	✓	
windowStyle	object	✓	
url	string	✓	
size	array		positive integer
offset	array	[0,0]	integer
bVisible	boolean	true	
entityName	string		
customId	string		
customData	object		
data	string		

Member Function

```
// example
  const obj = new App.Window({...});
  obj.Update(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
```

- **Update(data)**

Purpose：update window object

Parameter description：

```
data() same as Add()
```

- **SetVisible()**

Purpose：show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose: get object data

- **Delete()**

Purpose: delete object

Communication between window and 3D world

```
w51_event("name", "args");
```

w51_event is a native JavaScript function belonging to the UE built-in event delegation method, through which the js event information on the embedded webpage of the window is sent back to the UE engine, and return this information through the event registration function

w51_event accepts two parameters - event name and event message

Sample code in window url page

```
async function w51_event() {
    const jsondata = {
        "squadName": "Super hero WdpApi",
        "active": true,
        "definitions": {
            "state": { "enum": ["CA", "NY", "... etc ..."] }
        },
        "properties": {
            "first_name": { "type": "string" },
            "shipping_address": { "$ref": "/schemas/address" },
        }
    }

    button.addEventListener('click', (ev) => {
        w51_event('EventKey', jsondata);
    })
}
```

Sample code in rendering app

```
App.Renderer.registerSceneEvent([
    {
        name: 'OnWebJSEvent', func: function (data) {
            console.log(data);
            // { "event_name": "OnWebJSEvent", "result": { "name": "自定义name", "args": "自定义数据" }}
        }
    }
])
```

① The window object is the Chrome kernel framework embeded in app(Chrome 80); prohibit the right mouse button and alert event during the development
② Recommend to set content size as percentage

Poi

Add POI object

```
new App.Poi({ "location":  [121.50007292,31.22579403,30], "poiStyle": style })
```

```
async function AddPOI () {
  const poi = new App.Poi({
    "location": [121.50007292,31.22579403,30],
    "poiStyle": {
        "markerNormalUrl": "http://wdpapi.51aes.com/doc-static/images/static/markerNormal.png",//normal marker picture url address
        "markerActivateUrl": "http://wdpapi.51aes.com/doc-static/images/static/markerActive.png",//active marker picture url address(by mouse click or hover)
         //support 2 address style:
         //online address:sample"http://wdpapi.51aes.com/doc-static/images/static/markerNormal.png"
         //local address:sample"file:///D:/xxx/markerNormal.png"; D: the driver letter of online sear machine
        "markerSize": [100,228], //marker picture size(unit:pixel)
        "labelBgImageUrl": "http://wdpapi.51aes.com/doc-static/images/static/LabelBg.png", //label picture url address
        "labelBgSize": [200,50], //label picture size(unit:pixel)
        "scrollSpeed": 2, //Text scrolling speed (0: no scrolling)
        "textBoxWidth": 80, //Text box width (default 100)
        "labelBgOffset": [50,200], // Label can be shifted up, down, left or right; When [0,0], top left corner of label aligns to the location (x,y units: pixels)
        "labelContent": [" 文本内容A","ff0000ff","24"], //label text
       "labelContentOffset": [5,5], // labeContent can be shifted up, down, left or right; When [0,0], upper left corner of labelContent aligns to the upper left corner o
        "labelTop": true //label is on the top of marker
    },
    "entityName": "myName",
    "customId": "myId1",
    "customData": {
      "data": "myCustomData"
    },
    "bVisible": true,
    "visible2D": {
      "camera": {   //visibility affected by camera
        "hideDistance": 2000,  //define the entity hide distance between entity and camera
        "hideType": "default", //the display mode if entity is hided(none:not display; default:display as a circle)
        "scaleMode": "2D" // Whether affected by perspective (2D: no affect; 3D: affect); Perspective includes size adaption、overlapOrder affect
      },
      "interaction": { //visibility affected by interaction
        "clickTop": true, //whether display on the top if POI is clicked
        "hoverTop": true  //whether display on the top if POI is hovered
      },
  "entity": {
        "overlapOrder": 1 // settings across 2D covering types, effective when scaleMode is set 2D; Larger the value, the more forward the covering; Range [1~10]
      }
    }
  })

  const res = await App.Scene.Add(poi,{
    calculateCoordZ: {
      coordZRef: "surface", //surface;ground;altitude
      coordZOffset: 50 //height(unit:meter); This setting overrides the z value in location. If you want to use location , delete this line
    } //
  });
}
```

Parameter description:

Parameter		Type	Required	Value Range	Remarks
location		array	✅		Coordinate position
poiStyle	markerNormalUrl	string	✅		Normal state, image URL address, supports two forms: · Online address: e.g., "http://wdpapi.51aes.com/doc-static/images/static/markerNormal.png" · Local address: e.g., "file:///D:/xxx/markerNormal.png"; D: indicates the drive of the online seat
	markerActivateUrl	string	✅		Activated state, triggered by mouse hover or click
	markerSize	string	✅		Marker size (width, height in pixels)

		labelBgImageUrl	string	No		Image URL address
		labelBgSize	array	No	Positive integers	Label image size (width, height in pixels)
		labelBgOffset	array	No	Any integers	Offset of the entire label's top-left corner relative to the center point of the marker (coordinates center point) (x, y in pixels); Note: x is positive to the right, y is positive upwards
		labelContent	array	No	["Text content", "ff0000ff", "50"]	Rich text; Format: ["text", "color", "size"] color: in HEXA format
		scrollSpeed	int	No		Text scrolling speed (0: do not scroll)
		textBoxWidth	int	No		Text box width (default 100)
		labelContentOffset	array	No	[5,5]	Offset of the label content relative to the label's top-left corner (x, y in pixels); Note: x is positive to the right, y is positive downwards
		labelTop	boolean	No	true, false	Whether the label is on top of the marker
bVisible			boolean	No		Whether visible (true/false)
visible2D	camera	hideDistance	number	No	Positive integers	Defines the distance at which the entity is hidden (in meters); if the camera exceeds this distance, the entity will be hidden
		hideType	string	No	none, default	Entity exceeding display distance (none: do not display; default: circular display)
		scaleMode	string	No	2D, 3D	Whether affected by the camera's perspective (2D: not affected; 3D: affected)
	interaction	clickTop	boolean	No	true, false	When clicked (priority higher than hover), needs to be displayed on top
		hoverTop	boolean	No	true, false	When hovered over, needs to be displayed on top
	entity	overlapOrder	array	No	[1~10]	Overlap order; larger numbers float to the top; range [1~10]
entityName			string	No		Entity name
customId			string	No		Entity ID, for easy indexing in future operations
customData		data	string	No		Entity data, can be extended freely

Member Function

```
// 示例
  const obj = new App.Poi({...});
  obj.Update(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
  obj.onClick(ev => {
    const newObj = ev.result.object;
    console.log(ev);
  })
```

- **Update(data)**

Purpose: update window object

Parameter description:

```
data() same as Add()
```

- **SetVisible()**

Purpose: show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose: get object data

- **Delete()**

Purpose: delete object

Event

- **onClick()**
- **onMouseEnter()**
- **onMouseOut()**

Particle

Add particle object

```
new App.Particle({ "location":  [121.50007292,31.22579403,30], "rotator": { "pitch": 0, "yaw": 30, "roll": 0 }, "scale3d": [100, 30, 30], "particleType": style })
```

```
async function AddParticle() {
  const particle = new App.Particle({
    "location":  [121.50007292,31.22579403,30],
    "rotator": {
      "pitch": 0, // Pitch angle, reference (-180~180)
      "yaw": 0, // Yaw angle, reference (-180~180)
      "roll": 0 // Roll angle, reference (-180~180)
    },
    "scale3d": [30, 30, 30],  //scale size,[x,y,z]
    "particleType": "vehicle_taxi", //type
    "bVisible": true,
    "entityName": "myName",
    "customId": "myId1",
    "customData": {
      "data": "myCustomData"
    }
  });

  const res = await App.Scene.Add(particle,{
    calculateCoordZ: {
      coordZRef: "surface", //surface;ground;altitude
      coordZOffset: 50 //height(unit:meter)
    }
  });
}
```

Pameter description:

Parameter	Type	Required(default if optional)	Value Range
location	array	✓	
rotator	object		
pitch	number	0	[-180,180]
yaw	number	0	[-180,180]
roll	number	0	[-180,180]
scale3d	array	[1,1,1]	positive integer

particleType	string	flame	flame, 3dmark_build_loop, 3dmark_build, 3dmark_camera_loop, 3dmark_camera, 3dmark_sign, 3dmark_warning, vehicle_car, vehicle_car_black, vehicle_car_white, vehicle_taxi, shield, fire, arrow, alarm, circle, pyramid, marker_cube, marker_pyramid, marker_site, marker_cone, tool_wrench, weather_tornado, circle_glass, circle_compass, circle_outside, circle_inside, circle_scan, circle_diffuse, circle_area, circle_area2, circle_flash
bVisible	boolean	true	
entityName	string		
customId	string		
customData	object		
data	string		

Member Function

```
// 示例
  const obj = new App.Particle({...});
  obj.Update(json);
  obj.SetRotator(json);
  obj.SetScale3d(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
  obj.onClick(ev => {
    const newObj = ev.result.object;
    console.log(ev);
  })
```

- **Update(data)**

Purpose：update object

Parameter description：

```
data() same as Add()
```

- **SetRotator(jsondata)**

Purpose：rotate object angle

```
const jsondata = {
    "pitch": 0, //pitch angle
    "yaw": 30, //yaw angle
    "roll": 0 //roll angle
}
```

- **SetScale3d(jsondata)**

Purpose: set object scale

```
const jsondata = [200,200,200];
```

- **SetVisible()**

Purpose: show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose: get object data

- **Delete()**

Purpose: delete object

Event

- **onClick()**
- **onMouseEnter()**
- **onMouseOut()**

Effects

Add Effects object

```
new App.Effects({ "location":  [121.50007292,31.22579403,30], "rotator": { "pitch": 0, "yaw": 30, "roll": 0 }, "scale3d": [100, 30, 30], "speed": 1, "seedId": "" })
```

```
async function AddEffects() {
  const entityObj = new App.Effects({
    "location": [121.51132810,31.23485399,52],
    "rotator": {
      "pitch": 0, // Pitch angle, reference (-180~180)
      "yaw": 0, // Yaw angle, reference (-180~180)
      "roll": 0 // Roll angle, reference (-180~180)
    },
    "scale3d": [1, 1, 1],
    "bVisible": true, // Whether visible (true/false)
    "speed": 1, // Animation speed
    "seedId": "ac2a41915c7c7097be7dc64602e0e4fb", // Model number (obtained from DaaS)
    "entityName": "myName",
    "customId": "myId1",
    "customData": {
      "data": "myCustomData"
    }
  });

  const res = await App.Scene.Add(entityObj,{
    calculateCoordZ: {
      coordZRef: "surface", // Surface, ground, or altitude
      coordZOffset: 50 // Height (in meters)
    }
  });
}
```

Pameter description:

Parameter	Type	Required(default if optional)	Value Range
location	array	✅	
rotator	object		
pitch	number	0	[-180,180]
yaw	number	0	[-180,180]
roll	number	0	[-180,180]
scale3d	array	[1,1,1]	positive integer
speed	number	false	The speed of particle motion is usually measured in meters per second (m/s).

seedId	string	true	The model number is obtained from the DaaS platform.
bVisible	boolean	true	
entityName	string		
customId	string		
customData	object		
data	string		

Member Function

```
// example
  const obj = new App.Particle({...});
  obj.Update(json);
  obj.SetRotator(json);
  obj.SetScale3d(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
  obj.onClick(ev => {
    const newObj = ev.result.object;
    console.log(ev);
  })
```

- **Update(data)**

Purpose: update object

Parameter description:

```
data() same as Add()
```

- **SetRotator(jsondata)**

Purpose: rotate object angle

```
const jsondata = {
    "pitch": 0, //pitch angle
    "yaw": 30, //yaw angle
    "roll": 0 //roll angle
}
```

- **SetScale3d(jsondata)**

Purpose: set object scale

```
const jsondata = [200,200,200];
```

- **SetVisible()**

Purpose: show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose: get object data

- **Delete()**

Purpose: delete object

Event

- **onClick()**
- **onMouseEnter()**
- **onMouseOut()**

Light

Add Light object

```
new App.Light({ "location":  [121.50007292,31.22579403,30], "rotator": { "pitch": 0, "yaw": 30, "roll": 0 }, "scale3d": [100, 30, 30], "lightType": style })
```

```
async function AddLight () {
  const entityObj = new App.Light({
    "location": [121.51594199,31.23935862,25],
    "rotator": {
      "pitch": 0, //pitch angle: value range [-180~180]
      "yaw": 0, //Yaw angle: value range [-180 ~ 180]
      "roll": 0 //roll angle: value range [-180 ~ 180]
    },
    "scale3d": [1, 1, 1],
    "bVisible": true, //Whether it is visible (true/false)
    "lightStyle": {
      "intensity": 40, //Light intensity (0~100)
      "color": "ffb841ff", //Light color (HEXA color value)
      "angle": 50, //Light angle (0~50)
      "attenuation": 200, //Light attenuation length (unit: meter)
      "shadows": true, //Whether to enable shadows (true/false)
      "haze": true, //Whether to enable haze (true/false)
      "haze_Intensity": 90 //Haze intensity (0~100)
    }
  });

  const res = await App.Scene.Add(entityObj,{
    "calculateCoordZ": {  //coord type and coordZ; [optional] highest priority
      "coordZRef": "surface",//surface; ground; altitude
      "coordZOffset": 200 //coordZ Offset (unit: meters)
    }
  });
}
```

Pameter description:

Parameter		Type	Required(default if optional)	Value Range
location		array	✓	
rotator	pitch	number		[-180,180]
	yaw	number		[-180,180]
	roll	number		[-180,180]
scale3d		array		[1,1,1]

bVisible		boolean		
lightStyle	intensity	int	✓	[0, 100]
	color	string		HEXA color
	angle	int		[0, 50]: Light angle
	attenuation	int	✓	int: Light attenuation length (unit: meter)
	shadows	boolean	✓	true, false: Whether to enable shadows
	haze	boolean	✓	true, false: Whether to enable haze
	haze_Intensity	boolean	✓	[0, 100]: Haze intensity

Member Function

```
const obj = new App.Light({ ...});
obj.Update(json);
obj.SetRotator(json);
obj.SetScale3d(json);
obj.SetVisible(boolean);
obj.Get();
obj.Delete();
obj.onClick(ev => {
  const newObj = ev.result.object;
  console.log(ev);
})
```

- **Update(data)**

Purpose: update object

Parameter description:

```
data()  same as Add()
```

- **SetRotator(jsondata)**

Purpose: rotate object angle

```
const jsondata = {
    "pitch": 0, //pitch angle
    "yaw": 30, //yaw angle
    "roll": 0 //roll angle
}
```

- **SetScale3d(jsondata)**

Purpose: set object scale

```
const jsondata = [200,200,200]
```

- **SetVisible()**

Purpose: set object scale

```
boolean()
```

- **Get()**

Purpose: get object data

- **Delete()**

Purpose: delete object

Event

- **onClick()**
- **onMouseEnter()**
- **onMouseOut()**

Text3D

Add Text3D object

```
new App.Text3D({ "location":  [121.50007292,31.22579403,30], "rotator": { "pitch": 0, "yaw": 30, "roll": 0 }, "scale3d": [100, 30, 30], "text3DStyle": style })
```

```
async function AddText3D () {
  const text3d = new App.Text3D({
    "location": [121.46434372,31.23499129,60],
    "rotator": {
      "pitch": 0, // Pitch angle, reference (-180~180)
      "yaw": 0, // Yaw angle, reference (-180~180)
      "roll": 0 // Roll angle, reference (-180~180)
    },
    "scale3d": [100, 30, 30],//scale size,[x,y,z]
    "text3DStyle": {
      "text": "3D文字",//text content
      "color": "ff00ffff", //color: HEXA,or RGBA(0,0,0,0.8)
      "type": "plain", //style(plain; reflection; metal)
      "outline": 0.4, //outline(unit: pecerntage)
      "portrait": false, //whether to portrait
      "space": 0.1 // space(unit:meter)
    },
    "bVisible": true,
    "entityName": "myName",
    "customId": "myId1",
    "customData": {
      "data": "myCustomData"
    }
  });

  const res = await App.Scene.Add(text3d,{
    calculateCoordZ: {
      coordZRef: "surface", //surface;ground;altitude
      coordZOffset: 50 //height(unit:meter)
    }
  });
}
```

Pameter description:

Parameter	Type	Required(default if optional)	Value Range
location	array	✅	
rotator	object		

pitch	number	0	[-180,180]
yaw	number	0	[-180,180]
roll	number	0	[-180,180]
scale3d	array	[1,1,1]	正整数
text3DStyle	object		
text	string		
color	string	ff0000ff	HEXA,RGBA
type	string	plain	plain; reflection; metal
outline	number	0	[0,1]
portrait	boolean	false	true,false
space	number	0	positive integer
bVisible	boolean	true	
entityName	string		
customId	string		
customData	object		
data	string		

Member Function

```
// 示例
  const obj = new App.Text3D({...});
  obj.Update(json);
  obj.SetRotator(json);
  obj.SetScale3d(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
  obj.onClick(ev => {
    const newObj = ev.result.object;
    console.log(ev);
  })
```

- **Update(data)**

Purpose: update object

Parameter description:

```
data() same as Add()
```

- **SetRotator(jsondata)**

Purpose: rotate object angle

```
const jsondata = {
    "pitch": 0,
    "yaw": 30,
    "roll": 0
}
```

- **SetScale3d(jsondata)**

Purpose: set object scale

```
const jsondata = [200,200,200];
```

- **SetVisible()**

Purpose: show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose: get object data

- **Delete()**

Purpose: delete object

Event

- **onClick()**

- **onMouseEnter()**

- **onMouseOut()**

Viewshed

Add Viewshed object

```
new App.Viewshed({ "location":  [121.50007292,31.22579403,30], "rotator": { "pitch": 0, "yaw": 30, "roll": 0 }, "viewshedStyle": style })
```

```
async function AddViewshed () {
  const viewshed = new App.Viewshed({
    "location": [121.47025042,31.23065615,90],
    "rotator": {
      "pitch": 0, // Pitch angle, reference (-180~180)
      "yaw": 0, // Yaw angle, reference (-180~180)
      "roll": 0 // Roll angle, reference (-180~180)
    },
    "viewshedStyle": {
      "fieldOfView": 70, //field of view
      "radius": 600, //radius(unit:meter)
      "outline": true, //whether to outline
      "hiddenColor": "75fe97ff", //invisible area color
      "visibleColor": "3cff71ff" //visible area color
    },
    "bVisible": true,
    "entityName": "myName",
    "customId": "myId1",
    "customData": {
      "data": "myCustomData"
    }
  });

  const res = await App.Scene.Add(viewshed,{
    calculateCoordZ: {
      coordZRef: "ground", //surface;ground;altitude
      coordZOffset: 10 //height(unit:meter)
    }
    });
  }
```

Parameter description:

Parameter	类型	Required(default if optional)	Value Range
location	array	✓	
rotator	object		
pitch	number	0	[-180,180]
yaw	number	0	[-180,180]
roll	number	0	[-180,180]
viewshedStyle	object		
fieldOfView	number	90	[0,120]
radius	number	10	positive integer
outline	boolean	true,false	true,false
hiddenColor	string	ff0000ff	HEXA,RGBA
visibleColor	string	00ff00ff	HEXA,RGBA
bVisible	boolean	true	
entityName	string		
customId	string		
customData	object		
data	string		

Member Function

```
// 示例
  const obj = new App.Viewshed({...});
  obj.Update(json);
  obj.SetRotator(json);
  obj.SetScale3d(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
  obj.onClick(ev => {
    const newObj = ev.result.object;
    console.log(ev);
  })
```

- **Update(data)**

Purpose: update object

Parameter description:

```
data() same as Add()
```

- **SetRotator(jsondata)**

Purpose: rotate object angle

```
const jsondata = {
    "pitch": 0,
    "yaw": 30,
    "roll": 0
}
```

- **SetScale3d(jsondata)**

Purpose: set object scale

```
const jsondata = [200,200,200];
```

- **SetVisible()**

Purpose: show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose: get object data

- **Delete()**

Purpose: delete object

Event

- **onClick()**

Path

Add Path Object

```
new App.Path({ "polyline":  { "coordinates": [121.50007292,31.22579403,30]}, "pathStyle":  style })
```

```
async function AddPath () {
  const path = new App.Path({
    "polyline": {
      "coordinates": [
        [121.49968476,31.24861346,44],
        [121.49956979,31.25093239,96],
        [121.47613890,31.23725069,39]
      ]
    },
    "pathStyle": {
      "type": "arrow",//path style
      "width": 100,//path width,(unit:meter; unit:pixel when style is "adaptive_solid")
      "color": "b4fed7ff", //HEXA, or RGBA(0,0,0,0.8)
      "passColor": "ffb3deff"// the entity pass color, applied to "move along path"
    },
    "bVisible": true,
    "entityName": "myName",
    "customId": "myId1",
    "customData": {
      "data": "myCustomData"
    }
  });

  const res = await App.Scene.Add(path,{
    calculateCoordZ: {
      coordZRef: "surface", //surface:表面;ground:地面;altitude:海拔
      coordZOffset: 50 //高度(单位:米)
    }
  });
}
```

Parameter description:

Parameter	Type	Required(default if optional)	Value Range
polyline	object	✓	
coordinates	array	✓	
pathStyle	object		
type	string	solid	fit_solid, adaptive_solid, none, solid, arrow, arrow_dot, dashed_dot, arrow_dashed, flash, scan_line, brimless_arrow, railway, round_pipe, square_pipe, dashed_line
width	number	1	positive integer
color	string	FF0000FF	HEXA,RGBA
passColor	string		HEXA,RGBA
bVisible	boolean	true	true,false
entityName	string		
customId	string		
customData	object		
data	string		

adaptive_solid fit_solid round_pipe square_pipe railway brimless_arrow dashed_line solid arrow arrow_dot arrow_dashed dashed_dot flash scan_line

Member Function

```
// 示例
  const obj = new App.Path({...});
  obj.Update(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
  obj.onClick(ev => {
    const newObj = ev.result.object;
    console.log(ev);
  })
```

- **Update(data)**

Purpose: update object

Parameter description:

```
data() same as Add()
```

- **SetVisible()**

Purpose: show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose: get object data

- **Delete()**

Purpose: delete object

Event

- **onClick()**
- **onMouseEnter()**
- **onMouseOut()**

Parabola

Add Parabola object

```
new App.Parabola({ "polyline":  { "coordinates": [121.50007292,31.22579403,30]}, "parabolaStyle": style })
```

```
async function AddParabola () {
  const parabola = new App.Parabola({
    "polyline": {
      "coordinates": [
        [121.49968476,31.24861346,44],
        [121.47025042,31.23065615,90]
      ]
    },
    "parabolaStyle": {
      "topHeight": 800, //height of arc top(unit:meter)
      "topScale": 1, //horizontal ratio of arc top(unit: percentage)
      "type": "scanline", // style
      "width": 20, //width(unit:meter)
      "color": "ff3fafff", //HEXA, or RGBA(0,0,0,0.8)
      "gather": true //true: aggregates inward, false: expands outward
    },
    "bVisible": true,
    "entityName": "myName",
    "customId": "myId1",
    "customData": {
      "data": "myCustomData"
    }
  });

  const res = await App.Scene.Add(parabola,{
    calculateCoordZ: {
      coordZRef: "surface", //surface;ground;altitude
      coordZOffset: 50 //height(unit:meter)
    }
  });
}
```

Parameter description:

Parameter	Type	**Required(default if optional)**	**Value Range**
polyline	object	✓	
coordinates	array	✓	

parabolaStyle	object		
topHeight	number	10	positive integer
topScale	number	0.5	[0,1]
type	string	arrow	arrow,solid,scanline
width	number	1	positive integer
color	string	ff0000ff	HEXA,RGBA
gather	boolean	false	true,false
bVisible	boolean	true	true,false
entityName	string		
customId	string		
customData	object		
data	string		

Member Function

```
// 示例
  const obj = new App.Parabola({...});
  obj.Update(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
  obj.onClick(ev => {
    const newObj = ev.result.object;
    console.log(ev);
  })
```

- **Update(data)**

Purpose：update object

Parameter description：

```
data() same as Add()
```

- **SetVisible()**

Purpose: show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose: get object data

- **Delete()**

Purpose: delete object

Event

- **onClick()**
- **onMouseEnter()**
- **onMouseOut()**

Range

Add Range object

```
new App.Range({ "polygon2D": { "coordinates": [] }, "rangeStyle": style })
```

```
async function AddRange () {
  const range = new App.Range({
    "polygon2D": {
      "coordinates": [
        [  //outer ring coordinate
          [121.44988564758069,31.250519581243555],
          [121.44931229954645,31.237062463089813],
          [121.47069915607464,31.23800903013435],
          [121.46964214200186,31.251854247249092]
        ],
        [  //inner ring coordinate
          [121.45523929837454,31.247795686070997],
          [121.45496451671893,31.240059486959915],
          [121.46707798490596,31.24170746459223]
        ]
      ]
    },
    "rangeStyle": {
      "type": "loop_line", //range style
      "fillAreaType": "block", //range bottom style
      "height": 200, //range height(unit:meter)
      "strokeWeight": 10, //range bottom contour width(unit:meter)(note:invalid when range with inner ring)
      "color": "ff3772ff" //HEXA, or RGBA
    },
    "bVisible": true,
    "entityName": "myName",
    "customId": "myId1",
    "customData": {
      "data": "myCustomData"
    }
  })

  //tip::: add object
  const res = await App.Scene.Add(range,{
    calculateCoordZ: {
      coordZRef: "surface", //surface;ground;altitude
      coordZOffset: 10 //height(unit:meter)
    }
  });
}
```

Parameter description:

Parameter	Type	Required(default if optional)	Value Range
polygon2D	object	✅	
coordinates	array	✅	coordinates, the first array in the format is the outer ring, followed by the inner ring: [[[x,y,z],[x,y,z],....], [[x,y,z],[x,y,z],....], ...

			]
rangeStyle	object		
type	string	wave	none, wave, loop_line, grid, stripe, bias, box_wave_line, box_wave, box_solid_line, box_solid
fillAreaType	string	none	none, solid, block, block2, dot, dot2, dot3, dash_line, radar
height	number	10	positive integer
strokeWeight	number	1	positive integer
color	string	ff0000ff	HEXA,RGBA
bVisible	boolean	true	true;false
entityName	string		
customId	string		
customData	object		
data	string		

Member Function

```
// 示例
  const obj = new App.Range({...});
  obj.Update(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
  obj.onClick(ev => {
    const newObj = ev.result.object;
    console.log(ev);
  })
```

- **Update(data)**

Purpose: update object

Parameter description:

```
data() same as Add()
```

- **SetVisible()**

Purpose: show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose: get object data

- **Delete()**

Purpose: delete object

Event

- **onClick()**
- **onMouseEnter()**
- **onMouseOut()**

Circular Range

Add circular range object

```
new App.Range({ "circleStyle": {} "rangeStyle": {} })
```

```
async function AddRange () {
  const entityObj = new App.Range({
    "circlePolygon2D": {
      "center": [121.49885272, 31.24683565, 46],
      "radius": 200
    },
    "rangeStyle": {
      "shape": "circle",
      "type": "loop_line", // Type
      "fillAreaType": "block", // Bottom area fill type
```

```
        "height": 200, // Fence height (in meters)
        "strokeWeight": 10, // Bottom outline width (in meters)
        "color": "ff3772ff" // HEXA or rgba(0,0,0,0.8)
      },
      "bVisible": true,
      "entityName": "myName",
      "customId": "myId1",
      "customData": {
        "data": "myCustomData"
      }
    })

    // Add entity to the scene
    const res = await App.Scene.Add(entityObj, {
      calculateCoordZ: {
        coordZRef: "surface", // Surface; ground; altitude
        coordZOffset: 10 // Height (in meters)
      }
    });
  }
```

Parameter description:

Parameter	Type	Required(default if optional)	Value Range	Note
circlePolygon2D	object	✓		
center	array			center coordinates
radius	number			radius,unit: meter
rangeStyle	object			
shape	string	✓	circle	outline type:(circle)
type	string		none, wave, loop_line, grid, stripe, bias,box_wave_line, box_wave, box_solid_line, box_solid	range type
fillAreaType	string		none, solid, block, block2, dot, dot2, dot3, dash_line, radar	the bottom surface type
height	number		positive integer	extruded height of range
strokeWeight	number		positive integer	stroke line width
color	string		HEXA,RGBA	coloe
bVisible	boolean			
entityName	string			
customId	string			
customData	object			

data	string			

Member Function

```
// 示例
  const obj = new App.Range({...});
  obj.Update(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
  obj.onClick(ev => {
    const newObj = ev.result.object;
    console.log(ev);
  })
```

- **Update(data)**

Purpose：update object

Parameter description：

```
data() same as Add()
```

- **SetVisible()**

Purpose：show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose：get object data

- **Delete()**

Purpose：delete object

Event

- **onClick()**
- **onMouseEnter()**
- **onMouseOut()**

HeatMap

Add HeatMap object

```
new App.HeatMap({ "points": { "features": [{ "point": [], "value": 90 }] }, "heatMapStyle": style })
```

```
async function AddHeatMap () {
  const mapdata = [],
        points = [
          [121.49656333,31.22702479,49],
          [121.46434372,31.23499129,60],
          [121.49099537,31.23099794,22],
          [121.47780699,31.23877183,79]
        ];
  for (let i = 0; i < points.length; i++) {
      mapdata.push({
        "point": points[i],
        "value": Math.floor(Math.random() * 100)
      })
  }
  const heatmap = new App.HeatMap({
    "heatMapStyle": {
      "type": "fit", //style type(fit: projection; plane: plane)
      "brushDiameter": 2000, //single heat point brush diameter(unit:meter)
      "mappingValueRange": [1,100], //normalized heat point value range
      "gradientSetting": [
        //define visual color that related with heat point value(from lowest value to highest)
        "c9ff6fff","d153feff","01edffff","feb539ff","ffd30fff"
      ]
    },
    "bVisible": true,
    "entityName": "myName",
    "customId": "myId1",
    "customData": {
      "data": "myCustomData"
    },
    "points": {
      features: mapdata
    }
  });

  const res = await App.Scene.Add(heatmap,{
    calculateCoordZ: {
      coordZRef: "surface", //surface;ground;altitude
      coordZOffset: 10 //heihgt(unit:meter)
    }
  });
}
```

Parameter description:

Parameter	Type	**Required(default if optional)**	**Value Range**
points	object	✓	
features	object	✓	
point	array	✓	
value	number	✓	within the range defined in "mappingValueRange"
heatMapStyle	object		
type	string	plane	fit，plane
brushDiameter	number	10	positive integer
mappingValueRange	array	[1,100]	number
gradientSetting	array	"c9ff6fff","d153feff","01edffff","feb539ff","ffd30fff"	HEXA
bVisible	boolean	true	true,false
entityName	string		
customId	string		
customData	object		
data	string		

Member Function

```
// 示例
  const obj = new App.HeatMap({...});
  obj.Update(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
  obj.onClick(ev => {
    const newObj = ev.result.object;
    console.log(ev);
  })
```

- **Update(data)**

Purpose: update object

Parameter description:

```
data() same as Add()
```

- **SetVisible()**

Purpose: show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose: get object data

- **Delete()**

Purpose: delete object

Event

- **onClick()**
- **onMouseEnter()**
- **onMouseOut()**

ColumnarHeatMap

Add ColumnarHeatMap object

```
new App. ColumnarHeatMap({ "points": { "features": [{ "point": [], "value": 90 }] }, "columnarHeatMapStyle": style })
```

```
async function AddColumnarHeatMap () {
  const mapdata = [],
        points = [
          [121.48766129,31.23586660,35],
          [121.49968476,31.24861346,44],
          [121.49956979,31.25093239,96],
          [121.47613890,31.23725069,39]
        ];
  for (let i = 0; i < points.length; i++) {
      mapdata.push({
        "point": points[i],
        "value": Math.floor(Math.random() * 100)
      })
  }
  const columnarheatmap = new App.ColumnarHeatMap({
    "columnarHeatMapStyle": {
      "type": "cube", //appearance type (cube, cylinder, needle, frame)
      "brushDiameter": 550, //single heat point brush diameter(unit:meter)
      "mappingValueRange": [1,100], //normalized heat point value range
      "columnarWidth": 5, //single column width(note: single heat point contains many columns)
      "mappingHeightRange": [0,500], //normalized heat point height range(unit:meter)
      "enableGap": false, //whether a gap between columns
      "gradientSetting": [
        //define visual color that related with heat point value(from lowest value to highest)
        "ffae12ff","8f62ffff","60ff4bff","a207ffff","ff15c8ff"
      ]
    },
    "bVisible": true,
    "entityName": "myName",
    "customId": "myId1",
    "customData": {
      "data": "myCustomData"
    },
    "points": {
      "features": mapdata
    }
  });

  const res = await App.Scene.Add(columnarheatmap,{
    calculateCoordZ: {
      coordZRef: "surface", //surface;ground;altitude
      coordZOffset: 10 //height(unit:meter)
    }
  });
}
```

Parameter description:

Parameter	Type	**Required(default if optional)**	**Value Range**
points	object	✓	
features	object	✓	
point	array	✓	
value	number	✓	within the range defined in "mappingValueRange"
columnarheatStyle	object		
type	string	cube	cube, cylinder, needle, frame
brushDiameter	number	10	positive integer
columnarWidth	number	5	positive integer
mappingValueRange	array	[1,100]	integer
mappingHeightRange	array	[1,100]	positive integer
enableGap	boolean	false	true,false
gradientSetting	array	"c9ff6fff","d153feff","01edffff","feb539ff","ffd30fff"	HEXA
bVisible	boolean	true	true,false
entityName	string		
customId	string		
customData	object		
data	string		

Member Function

```
// 示例
  const obj = new App.ColumnarHeatMap({...});
  obj.Update(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
  obj.onClick(ev => {
    const newObj = ev.result.object;
    console.log(ev);
  })
```

- **Update(data)**

Purpose：update object

Parameter description：

```
data() same as Add()
```

- **SetVisible()**

Purpose：show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose：get object data

- **Delete()**

Purpose：delete object

Event

- **onClick()**
- **onMouseEnter()**
- **onMouseOut()**

ColumnarHeatMap

Add ColumnarHeatMap object

```
new App. ColumnarHeatMap({ "points": { "features": [{ "point": [], "value": 90 }] }, "columnarHeatMapStyle": style })
```

```
async function AddColumnarHeatMap () {
  const mapdata = [],
        points = [
          [121.48766129,31.23586660,35],
          [121.49968476,31.24861346,44],
          [121.49956979,31.25093239,96],
          [121.47613890,31.23725069,39]
        ];
  for (let i = 0; i < points.length; i++) {
      mapdata.push({
        "point": points[i],
        "value": Math.floor(Math.random() * 100)
      })
  }
  const columnarheatmap = new App.ColumnarHeatMap({
    "columnarHeatMapStyle": {
      "type": "cube", //appearance type (cube, cylinder, needle, frame)
      "brushDiameter": 550, //single heat point brush diameter(unit:meter)
      "mappingValueRange": [1,100], //normalized heat point value range
      "columnarWidth": 5, //single column width(note: single heat point contains many columns)
      "mappingHeightRange": [0,500], //normalized heat point height range(unit:meter)
      "enableGap": false, //whether a gap between columns
      "gradientSetting": [
        //define visual color that related with heat point value(from lowest value to highest)
        "ffae12ff","8f62ffff","60ff4bff","a207ffff","ff15c8ff"
      ]
    },
    "bVisible": true,
    "entityName": "myName",
    "customId": "myId1",
    "customData": {
      "data": "myCustomData"
    },
    "points": {
      "features": mapdata
    }
  });

  const res = await App.Scene.Add(columnarheatmap,{
    calculateCoordZ: {
      coordZRef: "surface", //surface;ground;altitude
      coordZOffset: 10 //height(unit:meter)
    }
  });
}
```

Parameter description:

Parameter	Type	**Required(default if optional)**	**Value Range**
points	object	✓	
features	object	✓	
point	array	✓	
value	number	✓	within the range defined in "mappingValueRange"
columnarheatStyle	object		
type	string	cube	cube, cylinder, needle, frame
brushDiameter	number	10	positive integer
columnarWidth	number	5	positive integer
mappingValueRange	array	[1,100]	integer
mappingHeightRange	array	[1,100]	positive integer
enableGap	boolean	false	true,false
gradientSetting	array	"c9ff6fff","d153feff","01edffff","feb539ff","ffd30fff"	HEXA
bVisible	boolean	true	true,false
entityName	string		
customId	string		
customData	object		
data	string		

Member Function

```
// 示例
  const obj = new App.ColumnarHeatMap({...});
  obj.Update(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
  obj.onClick(ev => {
    const newObj = ev.result.object;
    console.log(ev);
  })
```

- **Update(data)**

Purpose：update object

Parameter description：

```
data() same as Add()
```

- **SetVisible()**

Purpose：show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose：get object data

- **Delete()**

Purpose：delete object

Event

- **onClick()**
- **onMouseEnter()**
- **onMouseOut()**

RoadHeatMap

Create RoadHeatMap object

```
new App. RoadHeatMap({ "points": { "features": [{ "point": [], "value": 90 }] }, "RoadHeatMapStyle": style })
```

```
async function AddRoadHeatMap () {
    const mapdata = [],
        points = [
          [121.46434372,31.23499129,60],
          [121.49099537,31.23099794,22],
          [121.47780699,31.23877183,79]
        ];
  for (let i = 0; i < points.length; i++) {
      mapdata.push({
        "point": points[i],
        "value": Math.floor(Math.random() * 100)
      })
  }
  const roadheatmap = new App.RoadHeatMap({
    "roadHeatMapStyle": {
      "width": 50, //width
      "mappingValueRange": [1,100], //normalized heat point value range
      "gradientSetting": [
         //define visual color that related with heat point value(from lowest value to highest)
        "ff91fdff","cdff75ff","ff9e79ff","ff07a2ff","fea587ff"
      ],
      "type": "plane", //type(fit: projecrtion; plane:plane)
      "filter":["water"] //applied layer when "type":"fit"(Note:null applied to all layers)
      //common filter: "building","terrain","water","road","tree"
    },
    "bVisible": true,
    "entityName": "myName",
    "customId": "myId1",
    "customData": {
      "data": "myCustomData"
    },
    "points": {
      "features": mapdata
    }
  });
  const res = await App.Scene.Add(roadheatmap,{
    calculateCoordZ: {
      coordZRef: "surface", //surface;ground;altitude
      coordZOffset: 50 //height(unit:meter)
    }
  });
}
```

Parameter description:

Parameter	Type	**Required(default if optional)**	**Value Range**
points	object	✓	
features	object	✓	
point	array	✓	
value	number	✓	within the range defined in "mappingValueRange"
roadHeatMapStyle	object		
width	number	10	positive integer
type	string	plane	fit, plane
mappingValueRange	array	[1,100]	integer
gradientSetting	array	"ff91fdff","cdff75ff","ff9e79ff","ff07a2ff","fea587ff"	HEXA
filter	array		"building","road"..
bVisible	boolean	true	true,false
entityName	string		
customId	string		
customData	object		
data	string		

Member Function

```
// 示例
  const obj = new App.RoadHeatMap({...});
  obj.Update(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
  obj.onClick(ev => {
    const newObj = ev.result.object;
    console.log(ev);
  })
```

- **Update(data)**

Purpose：update object

Parameter description

```
data() same as Add()
```

- **SetVisible()**

Purpose：show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose：get object data

- **Delete()**

Purpose：delete object

Event

- **onClick()**
- **onMouseEnter()**
- **onMouseOut()**

Raster

Add Raster object

```
new App. Raster({ "rasterStyle": style })
```

```
async function AddRaster() {
    const raster = new App.Raster({
    "rasterStyle": {
      "path": "http://wdpapi.51aes.com/doc-static/images/static/raster/raster.tif", //TIF format
         //support 2 address style:
         //online address:sample"http://wdpapi.51aes.com/doc-static/images/static/raster/raster.tif"
         //local address:sample"file:///D:/xxx/raster.tif"; D: the driver letter of online seat machine
      "type": "fit", //type (fit:projection; plane)
      "gradientSetting": [
        //define visual color that related with value(from lowest value to highest)
        "91ffd5ff","ff1af5ff","ff0455ff","ff71d3ff","fed500ff"
      ]
    },
    "bVisible": true,
    "entityName": "myName",
    "customId": "myId1",
    "customData": {
      "data": "myCustomData"
    }
  });

  const res = await App.Scene.Add(raster,{
    calculateCoordZ: {
      coordZRef: "surface", //surface;ground;altitude
      coordZOffset: 10 //height(unit:meter)
    }
  });
}
```

Parameter description:

Parameter	Type	**Required(default if optional)**	Value Range
rasterStyle	object	✓	
path	string	✓	
type	string	plane	fit, plane

gradientSetting	array	"91ffd5ff","ff1af5ff","ff0455ff","ff71d3ff","fed500ff"	HEXA
bVisible	boolean	true	true,false
entityName	string		
customId	string		
customData	object		
data	string	否	

Member Function

```
// 示例
  const obj = new App.Raster({...});
  obj.Update(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
  obj.onClick(ev => {
    const newObj = ev.result.object;
    console.log(ev);
  })
```

- **Update(data)**

Purpose: update object

Parameter description

```
data() same as Add()
```

- **SetVisible()**

Purpose: show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose：get object data

- **Delete()**

Purpose：delete object

Event

- **onClick()**
- **onMouseEnter()**
- **onMouseOut()**

HighlightArea

Add HighlightArea object

```
new App. HighlightArea({ "polygon2D": { "coordinates": [] }, "highlightAreaStyle": style })
```

```
async function AddHighlightArea () {
  const highlightarea = new App.HighlightArea({
    "polygon2D": {
      "coordinates": [
        [
          [121.44988564758069,31.250519581243555],
          [121.44931229954645,31.237062463089813],
          [121.47069915607464,31.23800903013435],
          [121.46964214200186,31.251854247249092]
        ]
      ]
    },
    "highlightAreaStyle": {
      "interiorColor": "cbba89ff", //inside color
      "exteriorColor": "00ffffff", //outside color
      "exteriorOutlineColor": "ff00ffff", //outside contour color
      "exteriorSaturation": 10, //outside color saturation
      "exteriorBrightness": 15, //outside color brightness
      "exteriorContrast": 10 //outside color contrast
    },
    "bVisible": true
  });

  const res = await App.Scene.Add(highlightarea,{
    calculateCoordZ: {
      coordZRef: "surface", //surface;ground;altitude
      coordZOffset: 50 //height(unit:meter)
    }
  });
}
```

Parameter description:

Parameter	Type	**Required(default if optional)**	**Value Range**
polygon2D	object	✓	
coordinates	array	✓	
highlightAreaStyle	object		
interiorColor	string	"cbba89ff"	HEXA
exteriorColor	string	"00ffffff"	HEXA
exteriorOutlineColor	string	"ff00ffff"	HEXA
exteriorSaturation	number	10	(-100, 100)
exteriorBrightness	number	15	(-100, 100)
exteriorContrast	number	10	(-100, 100)
bVisible	boolean	true	

Member Function

```
// 示例
  const obj = new App.HighlightArea({...});
  obj.Update(json);
  obj.SetVisible(boolean);
  obj.Get();
  obj.Delete();
```

- **Update(data)**

Purpose: update object

Parameter description

```
data() same as Add()
```

- **SetVisible()**

Purpose: show and hide object

```
boolean() true:show ; false: hide
```

- **Get()**

Purpose: delete object

- **Delete()**

Purpose: delete object

Pick entity by clicking

Get node ids of model by clicking

```
async function PickEntityByClick () {
  const jsondata = {
    "pickFilter": { //Entity filtering for specific entities [Optional]
      "filterEntityTypes": [], //Filter entity types
      "excludeEntities": [], //Exclude entity objects
      "bFilterForExclude": false //Reverse filtering of filterEntityTypes (true/false)
    },
    "selectionMode": "New" //['None', 'New'(Single), 'Add'(Add), 'Subtract'(Subtract), 'Reverse'(Reverse)]
  }

  await App.Setting.SetDefaultActionSetting(jsondata);
}

// filterEntityTypes (Pay attention to capitalization)
/*
  Tiles: Layer;
  Static: Static model;
  Skeletal: Skeletal model
*/
```

Set the outline style and highlight style of model

```
async function SetVisualStyle() {
  // Set custom outline and highlight styles (styleName, hex color; alpha: highlight valid)
  App.Setting.SetVisualColorStyle('styleName', "0f5dff4c")

  const style = await App.Setting.GetVisualColorStyle();
  console.log(style);

  // Set outline thickness
  App.Setting.SetOutlineThickness(2);  //Minimum value: 2

  const thickness = await App.Setting.GetOutlineThickness();
  console.log(thickness);
}
```

styleName

	styleName	hexa	color
0	Default	FFBF0077	
1	Black	00000077	
2	DarkBlue	00008B77	
3	MediumBlue	0000CD77	
4	Blue	0000FF77	
5	DarkGreen	00640077	
6	Green	00800077	
7	SpringGreen	00FF7F77	
8	MidnightBlue	19197077	
9	ForestGreen	228B2277	
10	SeaGreen	2E8B5777	
11	LimeGreen	32CD3277	
12	RoyalBlue	4169E177	
13	SteelBlue	4682B477	
14	Maroon	80000077	
15	Purple	80008077	
16	Olive	80800077	
17	Gray	80808077	
18	SkyBlue	87CEEB77	
19	BlueViolet	8A2BE277	
20	DarkRed	8B000077	
21	LightGreen	90EE9077	
22	MediumPurple	9370DB77	

23	DarkViolet	9400D377	
24	PaleGreen	98FB9877	
25	YellowGreen	9ACD3277	
26	Sienna	A0522D77	
27	Brown	A52A2A77	
28	DarkGray	A9A9A977	
29	LightBlue	ADD8E677	
30	GreenYellow	ADFF2F77	
31	PowderBlue	B0E0E677	
32	Silver	C0C0C077	
33	IndianRed	CD5C5C77	
34	Chocolate	D2691E77	
35	LightGray	D3D3D377	
36	Thistle	D8BFD877	
37	Orchid	DA70D677	
38	GoldenRod	DAA52077	
39	Plum	DDA0DD77	
40	LightCyan	E0FFFF77	
41	DarkSalmon	E9967A77	
42	Violet	EE82EE77	
43	LightCoral	F0808077	
44	Wheat	F5DEB377	
45	Salmon	FA807277	
46	Linen	FAF0E677	

47	DeepPink	FF149377	
48	OrangeRed	FF450077	
49	Tomato	FF634777	
50	HotPink	FF69B477	
51	Coral	FF7F5077	
52	DarkOrange	FF8C0077	
53	LightSalmon	FFA07A77	
54	Orange	FFA50077	
55	LightPink	FFB6C177	
56	Pink	FFC0CB77	
57	Gold	FFD70077	
58	FloralWhite	FFFAF077	
59	Snow	FFFAFA77	
60	Yellow	FFFF0077	
61	LightYellow	FFFFE077	
62	Ivory	FFFFF077	
63	White	FFFFFF77	

Get AES layers

```
async function GetLayers () {
  const { result } = await App.Scene.GetTiles();
  const tilesObj = result?.Tiles?.[0];
  if (tilesObj) {
    const res = await App.Scene.Tiles.GetLayers(tilesObj);
    console.log(res);
    /*
      {
        "layers": [
          {
            "layer": "Terrain",
            "bVisible": true
          },
          {
            "layer": "Building",
            "bVisible": true
          },
          {
            "layer": "Road",
            "bVisible": true
          },
          {
            "layer": "Water",
            "bVisible": true
          }
        ]
      }
    }
  */
}
```

Set the outline of layer

```
async function SetLayersOutline() {

  App.Scene.Tiles.SetLayersOutline({
    layers: ['Building'],
    bOutline: true,
    styleName: "Blue"
  })
}
```

Set highlight style of layer

```
async function SetLayersHighlight() {

  App.Scene.Tiles.SetLayersHighlight({
    layers: ['Building'],
    bHighlight: true,
    styleName: "Blue"
  })

}
```

Set layer visibility

```
async function SetLayersVisibility() {

  App.Scene.Tiles.SetLayersVisibility({
    layers: ['Building'],
    bVisible: false
  })
}

/* layers:
    Terrain
    Building
    Road
    Water
    Tree
*/
```

Set model outline

```
async function SetNodesOutline() {

  App.Scene.TilesNode.SetNodesOutline({
    nodeIds: ['895874688'],
    bOutline: true,
    styleName: "Blue"
  })

}
```

Set model highlight style

```
async function SetNodesHighlight() {

  App.Scene.TilesNode.SetNodesHighlight({
    nodeIds: ['895874688'],
    bHighlight: true,
    styleName: "Blue"
  })

}
```

Set model visibility

```
async function SetNodesVisibility() {

  App.Scene.TilesNode.SetNodesVisibility({
    nodeIds: ['895874688'],
    bVisible: false
  })

}
```

Get the AES layer object

```
async function GetLayers () {
  const { result } = await App.Scene.GetTiles();
  const tilesObj = result?.Tiles?.[0];

  cache.set('tilesObj', tilesObj);

}
```

Set the model outline

```
async function SetNodesOutline() {

  tilesObj.SetNodesOutline({
    nodeIds: ['895874688'],
    bOutline: true,
    styleName: "Blue"
  })
}
```

Set the model highlight

```
async function SetNodesHighlight() {

  tilesObj.SetNodesHighlight({
    nodeIds: ['895874688'],
    bHighlight: true,
    styleName: "Blue"
  })

}
```

Model visibility settings

```
async function SetNodesHighlight() {

  tilesObj.SetNodesVisibility({
    nodeIds: ['895874688'],
    bVisible: false
  })

}
```

AES static model

Add static model entity

```
new App.Static({ "location":  [121.50007292,31.22579403,30], "rotator": { "pitch": 0, "yaw": 30, "roll": 0 }, "scale3d": [100, 30, 30], ... })
```

```
async function AddStatic () {
  const Static = new App.Static({
    "location": [121.49992450, 31.10650030, 66],
    "rotator": {
      "pitch": 0, // Pitch angle, reference (-180~180)
      "yaw": 0, // Yaw angle, reference (-180~180)
      "roll": 0 // Roll angle, reference (-180~180)
    },
    "scale3d": [30, 30, 30],
    "bVisible": true,
    "seedId": "9ab0dfa9cc0d811dd04e5f8f688d7080", //obtained from DaaS
  });

  const res = await App.Scene.Add(Static);
  console.log(res);
}
```

Parameter Description:

Parameter	Type	Required	Range	Note
location	array	Yes		coordinate
rotator	object	No		
pitch	number	No	[-180,180]	Pitch angle, 0 is horizontal
yaw	number	No	[-180,180]	Yaw angle
roll	number	No	[-180,180]	Roll angle, 0 is vertical to the ground
scale3d	array	No	Positive integer	scale
bVisible	boolean	No		Visibility (true/false)
seedId	string	Yes		Model ID (obtained from DaaS)

Member Function

```
const obj = new App.Static({ ...});
obj.Update(json);
obj.SetRotator(json);
obj.SetScale3d(json);
obj.SetVisible(boolean);
obj.Get();
obj.Delete();
obj.onClick(ev => {
  const newObj = ev.result.object;
  console.log(ev);
})
```

- **Update(data)**

Purpose: update object

Parameter description

```
data() same as Add()
```

- **SetRotator(jsondata)**

Purpose: rotate object

Parameter description

```
const jsondata = {
  "pitch": 0, //(-180~180)
  "yaw": 30, //(-180~180)
  "roll": 0 //(-180~180)
}
```

- **SetScale3d(jsondata)**

Purpose: change the scale of object

Parameter description

```
const jsondata = [50,50,50]; //(x,y,z)
```

- **SetVisible()**

Purpose: show and hide object

```
boolean() true:show; false:hide
```

- **Get()**

Purpose: get object data

- **Delete()**

Purpose: delete object

Event

- **Click()**

AES Skeletal model

Add Skeletal model entity

```
new App.Skeletal({ "location":  [121.50007292,31.22579403,30], "rotator": { "pitch": 0, "yaw": 30, "roll": 0 }, "scale3d": [100, 30, 30], ... })
```

```
async function AddSkeletal() {
  const skeletal = new App.Skeletal({
    "location": [121.49992450,31.10650030,66],
    "rotator": {
      "pitch": 0, // Pitch angle, reference (-180~180)
      "yaw": 0, // Yaw angle, reference (-180~180)
      "roll": 0 // Roll angle, reference (-180~180)
    },
    "scale3d": [30, 30, 30],
    "bVisible": true,
    "seedId": "11cf4fabacb485caaa58ec8b1362047d",
    "animSequenceIndex": 0,
    "bPause": false,
    "bLoop": true,
    "playRate": 1,
    "playInterval": {
      "min": 0,
      "max": 100
    },
    "animSequences": []
  });

  const res = await App.Scene.Add(skeletal);
  console.log(res);
}
```

Parameter Description:

Parameter	Type	Required	Range	Note
location	array	Yes		coordinate
rotator	object	No		
pitch	number	No	[-180,180]	Pitch angle, 0 is horizontal
yaw	number	No	[-180,180]	Yaw angle
roll	number	No	[-180,180]	Roll angle, 0 is vertical to the ground
scale3d	array	No	Positive integer	scale
bVisible	boolean	No		Visibility (true/false)
seedId	string	Yes		Model ID (obtained from DaaS)
animSequenceIndex	string	No		The index of the animation that needs to be played (when there are multiple animation segments)
bPause	boolean	No		Whether to play
bLoop	boolean	No		Whether to loop
playRate	number	No		Playback progress (rate), supports any floating-point number
playInterval	object	No		The frame range for playing the animation segment, such as 10-100
min	number	No		
max	number	No		
animSequences	array			The animation segments included in the current skeletal asset

Member Function

```
const obj = new App.Static({ ...});
obj.Update(json);
obj.SetRotator(json);
obj.SetScale3d(json);
obj.SetVisible(boolean);
obj.Get();
obj.Delete();
obj.onClick(ev => {
  const newObj = ev.result.object;
  console.log(ev);
})
```

- **Update(data)**

Purpose: update object

Parameter description

```
data() same as Add()
```

- **SetRotator(jsondata)**

Purpose: rotate object

Parameter description

```
const jsondata = {
  "pitch": 0, //(-180~180)
  "yaw": 30, //(-180~180)
  "roll": 0 //(-180~180)
}
```

- **SetScale3d(jsondata)**

Purpose: change the scale of object

Parameter description

```
const jsondata = [50,50,50]; //(x,y,z)
```

- **SetVisible()**

Purpose：show and hide object

```
boolean() true:show; false:hide
```

- **Get()**

Purpose：get object data

- **Delete()**

Purpose：delete object

Event

- **Click()**

Set Model Outline

```
async function SetNodesOutline() {

  App.Scene.Project.SetNodesOutline({
    nodeIds: ['688'],
    bOutline: true,
    styleName: "Blue"
  })

}
```

Set Model Highlight

```
async function SetNodesHighlight() {

  App.Scene.Project.SetNodesHighlight({
    nodeIds: ['688'],
    bHighlight: true,
    styleName: "Blue"
  })

}
```

Set Model Visibility

```
async function SetNodesVisibility() {

  App.Scene.Project.SetNodesVisibility({
    nodeIds: ['688'],
    bVisible: false
  })

}
```

Retrieve Project Layer Object

```
async function ProjectLayers() {
  const { result } = await App.Scene.GetProject();
  const projectObj= result?.Project?.[0];

  cache.set('projectObj ', projectObj);
}
```

Set Model Outline

```
async function SetNodesOutline() {

  projectObj.SetNodesOutline({
    nodeIds: ['688'],
    bOutline: true,
    styleName: "Blue"
  })

}
```

Set Model Highlight

```
async function SetNodesHighlight() {

  projectObj.SetNodesHighlight({
    nodeIds: ['688'],
    bHighlight: true,
    styleName: "Blue"
  })

}
```

Set Model Visibility

```
async function SetNodesHighlight() {

  projectObj.SetNodesVisibility({
    nodeIds: ['688'],
    bVisible: false
  })

}
```

Get model object

```
// Click the model entity to get the eid
const Eids = [
'-9151314316185345952',
];
const { result } = await App.Scene.GetByEids(Eids);

// Cache global objects for subsequent operations
cache.set('model', result[0]);
```

Update model

```
const jsondata = {
"location": [undefined],
"rotator": {
"pitch": 0, // pitch angle; value range [-180~180]
"yaw": 30, // yaw angle; value range [-180~180]
"roll": 0 //Roll angle; value range [-180~180]
},
"scale3d": [20, 20, 20],
"bVisible": true, //Is it visible (true/false)
"name": "Project model test",
"customData": {
"data": "myCustomData"
}
}
const res = await cache.get('model').Update(jsondata,{
"calculateCoordZ": { //[Optional] Coordinate type and coordinate altitude; Highest priority
"coordZRef": "surface",//surface: surface; ground: ground; altitude: altitude
"coordZOffset": 50 //Height (unit: meter)
}
});
```

Set rotation

```
const rotator = {
"pitch": 0, //Pitch angle; value range [-180~180]
"yaw": 40, //Yaw angle; value range [-180~180]
"roll": 0 //Roll angle; value range [-180~180]
}

const res = await cache.get('model').SetRotator(rotator);
```

Set scale

```
/**
* @param {Array} scale - scale ratio (x, y, z) axis
*/
```

```
const res = await cache.get('model').SetScale3d([30,30,30]);
```

Set location

```
/**
* @param {Array} location - location coordinates
*/

const location = [121.46573032,31.21639509,12];
const res = await cache.get('model').SetLocation(location);
```

Show and hide models

```
/**
* @param {boolean} bVisible - true: show; false: hide
*/

const res = await cache.get('model').SetVisible(false);
```

Get information

```
const res = await cache.get('model').Get();
```

Add Material

Obtain SeedId from official material library in WDP editor. Create a material instance to this case.

Acquire the unique MaterialEid of the instance for the subsequent use of this material.

```
const material = new App.Material({
  "seedId": "xxx" // Get corresponding material SeedID from official materials list in editor
});
const res = await App.Scene.Add(material);
console.log(res);
if (res.success) {
  cache.set('material', res.result.object);
  console.log('materialEid: ', res.result.object.materialEid); // Recommended: Use material.materialEid to get the materialEid of this instance
}
```

Start Pick Materials Tool

Effective for static official models, skeletal animation models, and hierarchical models, and can be used continuously across models.

After activating the tool, when the mouse hovers, the same material in the model pointed to by the mouse is highlighted.

Click to obtain the unique identifier of the material of the model (including eid+meshName+materialIndex).

```
const res = await App.Tools.PickerMaterial.Start({
  "bContinuous": true, // true: continuously get, false: get only once
  "func": (res) => {console.log(res)}
});
console.log(res);
```

Get Picked Materials

Obtain the unique identifiers of the materials of the various existing models that were just clicked on (including eid+meshName+materialIndex) in a JSON array format, which can be directly copied into the material replacement parameters.

```
const res = await App.Tools.PickerMaterial.GetMaterials();
console.log(res);
if (res.success) {
  console.log(res.result);
}
```

End Pick Materials Tool

Turn off the mouse hover highlight effect and stop obtaining the material identifiers of existing models.

```
const res = await App.Tools.PickerMaterial.End();
console.log(res);
```

Set Overide Material

Replace the existing model materials with material instance created in the case study.

```
const res = await App.DataModel.Material.SetModelMaterial({
  "TargetMaterials": [
      {
        "eid": "xxx", // Corresponding static, skeletal, hierarchy model eid
        "meshName": "xxx",
        "materialIndex": -1
      },
      //...
  ], // Get arrays of target model materials and paste it here
  "MaterialEid": xxxxx // Replace multiple materials of multiple models with one material instances created
});
console.log(res);
```

Batch Assign Materials

```
// Data can be obtained via the following API:
// App.Tools.PickerMaterial.Start;
// App.Tools.PickerMaterial.GetMaterials

const { result: [ { meshName, materialIndex } = {} ] = [] } = await App.Tools.PickerMaterial.GetMaterials();

const jsondata = [
    {
        "obj": modelObj, // Model instance
        "newMaterialsInfo": [
            {
                "MeshName": meshName,
                "MaterialIndex": materialIndex,
                "MIEid": cache.get('material').materialEid
            }
        ]
    }
]

const res = await App.DataModel.Material.Apply(jsondata);
```

Set Entity mesh slot Highlight

```
// Data can be obtained via the following API:
const { result } = await App.DataModel.Material.GetList([modelObj]);
const jsondata = [
  {
    entity: modelObj,
    meshName: "Root_Mesh",
    MaterialIndex: 2,
    bHighlight: true
  }
]

const res = await App.DataModel.Material.SetEntitySlotsHighlight(jsondata);
```

Gets the material object applied by the entity

```
const { result } = await modelObj.GetMaterial();
```

Environment

Skylight

```
// get skylight time
await App.Environment.GetSkylightTime();

// set skylight time
await App.Environment.SetSkylightTime('12:30', 3, false);
parameter 1: switch to the time; parameter 2: duration of switching; parameter 3: whether to apply real time(the real time is online seat machine time)
```

Weather

```
// get whether
await App.Environment.GetSceneWeather();

// set whether
await App.Environment.SetSceneWeather('Sunny', 3, false);
parameter 1: switch to the weather; parameter 2: duration of switching; parameter 3: whether to apply real weather(only valid in China mainland)
```

Parameter	Type	Required(default if optional)	Value Range
parameter 1	string	✓	Sunny,Cloudy,PartlyCloudy,Overcast,LightRain,ModerateRain,HeavyRain,LightSnow,ModerateSnow,HeavySnow,Foggy,Sand,Haze

Set Scene Style

```
const style = "comic";
const res = await App.Scene.SetSceneStyle(style);
```

Parameter	Type	Required(default if optional)	Value Range
style	string	✓	comic, sketch, dark, ashy, false

Widget

Get API Version

```
await App.System.GetInfomation();
```

RTC

```
await App.Renderer.GetStats();
```

Set Render Mode

```
await App.Renderer.SetRendererMode('fixed', [2560,1440]);
// parameter 1: full(adaptive to the container), fixed(the number from parameter 2);  parameter 2: resolution(limit range from [100~7680, 100~4320])
// note: Google Chrome supports up to 4K: 4096 * 2160; 51Browser supports up to 8K: 7680 * 4320
```

Set Frame Rate

```
await App.Renderer.SetFrameRateLimit(30); //limit range [2~60]
```

Set Bitrate

```
await App.Renderer.SetBitrate(10); //limit range [5~15]
```

Set Audio

```
await App.Setting.SetAudioVolume(50); //limit range[0~100]
```

Mini Map

```
const _url = "http://wdpapi.51aes.com/doc-static/images/static/MiniMap/"
const jsondata = {
    "type": "manual",
    "source": {
        "bg": _url + "Minimap.png",  //background image(note: fixed resolution 4096x4096)
        "needle": _url + "Minimap_needle.png", //needle image
        "mask": _url + "Minimap_mask.jpg", //mask image
        "frame": _url + "Minimap_outline.png" //frame image
    },
    "mappingAnchors": [
       //map background image`s left upper corner and right bottom corner with coordiante
        [121.54900666925667, 31.175366234862334],
        [121.42595948541654, 31.068307985569852]
    ],
    "display": {
        "position": [300, 100], //position on real-time video(unit:pixel; Note: The upper left corner of the screen resolution of 1920 * 1080 is the reference point, the
        "size": 300, //size on real-time video(unit:pixel; Note: Based on screen resolution 1920 * 1080)
        "anchors": "leftTop" // Affects position reference point & screen stretch reference point
        // Stretch: under different resolutions, the minimap position will be stretched by short side adaptive. When the screen ratio of development and display is differ
        // leftTop, leftMiddle, leftDown
        // middleTop, middleCenter, middleDown
        // rightTop, rightMiddle, rightDown
    }
}

const res = await App.Tools.MiniMap.Start(jsondata);
console.log(res)
```

```
//end mini map
const res = await App.Tools.MiniMap.End();
console.log(res)
```

Compass

```
const jsondata = {
    "source": {
        "bg": "http://wdpapi.51aes.com/doc-static/images/static/compass_bg.png", //compass background map
        "needle": "http://wdpapi.51aes.com/doc-static/images/static/compass_needle.png" //center Pointe
    },
    "display": {
       "position": [300,100], //position (unit:pixel; Note: The upper left corner of the screen resolution of 1920 * 1080 is the reference point, the reference point can
        "size": 300, //size on real-time video(unit:pixel; Note: Based on screen resolution 1920 * 1080)
        "anchors": "leftTop" // Affects position reference point & screen stretch reference point
        // Stretch: under different resolutions, the compass position will be stretched by short side adaptive. When the screen ratio of development and display is differ
        // leftTop, leftMiddle, leftDown
        // middleTop, middleCenter, middleDown
        // rightTop, rightMiddle, rightDown
    }
}

const res = await App.Tools.Compass.Start(jsondata);
console.log(res)



//End Compass
const res = await App.Tools.Compass.End();
console.log(res)
```

Tools

GIS coordinates to Cartesian coordinates

```
async function GISToCartesian () {
  const coord = [
    [121.478818,31.24251593,94],
    [121.47274158,31.22456944,95],
    [121.48836526,31.22625219,68],
    [121.49542527,31.2341436,58]
  ]

  const res = await App.Tools.Coordinate.GISToCartesian(coord);
  console.log(res.result.to)
}
```

Cartesian coordinates to GIS coordinates

```
async function CartesianToGIS () {
  const cartesian = [
    [5000, 5000, 20],
    [50000, 10000, 20],
    [70000, 70000, 20],
    [10000, 70000, 20]
  ]

  const res = await App.Tools.Coordinate.CartesianToGIS(cartesian);
  console.log(res.result.to)
}
```

GIS coordinates to screen coordinates

```
async function GISToScreenPos () {
  const coord = [  //Note: The coordinates must be within the screen area
    [121.47274158,31.22456944,95],
    [121.48836526,31.22625219,68],
    [121.49542527,31.2341436,58]
  ]

  const res = await App.Tools.Coordinate.GISToScreenPos(coord);
  console.log(res.result.to)
}
```

Coordinate Geometry Assistance

```
async function AddCoordAide () {
  const coords = [
    [121.48874015,31.23789002,66],
    [121.48598707,31.24150321,1],
    [121.503394,31.24398916,49],
    [121.49355523,31.23633749,38],
    [121.51654214,31.23608135,29],
    [121.4942991,31.23271274,70]
  ]

  await App.DataModel.Geometry.StartShowCoord(coords, 'surface');
  // surface; ground; altitude


  // Close the coordinate point assistant.
  App.DataModel.Geometry.EndShowCoord();
}
```

CAD coordinates to latitude and longitude coordinates

```
async function CADGeoRef () {
  const { result } = await App.Tools.Coordinate.CreateCADGeoRef({
    cadPoints: [
      [1363051.25,1183997.5,0],
      [-258250.0,-495587.5,0]
    ],
    worldPoints: [
      [121.49969494,31.22487886,76],
      [121.47813416,31.22210366,70]
    ]
  })

  const geoObj = result.object;
  const res = await App.Tools.Coordinate.LocalToGlobalGeoRef(geoObj,
    [
      [1532640.0,1752786.25,0],
      [1363051.25,1183997.5,0],
      [1231995.625,493529.375,0],
      [173426.875,-244088.125,0],
      [-2944.375,-21562.5,0],
      [-1984983.125,-733952.5,0],
      [-1143592.5,-114504.375,0],
      [-5587255.0,747446.875,0],
      [-258250.0,-495587.5,0]
    ]
  );

  console.log(res.result.to);

}
```

Pick Point Tools

```
// Start Pick Point
await App.Tools.PickerPoint.StartPickPoint(true, true, "surface");
arguments ①: true:show coordinate information; false:hide coordinate information
arguments ②: true:show point marker; false:hide point marker
arguments ③: surface; ground; altitude

// Get Picked Points
```

```
await App.Tools.PickerPoint.GetPickedPoints(0);
// 0:surface; 1:ground; 2:altitude

// End Pick Point
await App.Tools.PickerPoint.EndPickPoint();
```

Measure

```
// Measure Start
await App.Tools.Measure.Start();

// Measure ENd
await App.Tools.Measure.End();
```

Section

```
async function StartSection () {
  const res = await App.Scene.Section.Start({
      "coordZRef": "surface", //surface; ground; altitude
      "strokeColor": "56a8ff", //The color of the cut object outline (HEX color value)
      "strokeWeight": 0.8, //The width of the cut object outline[0~1]
      "invert": false, //The cut object (true: visible inside; false: visible outside)
      "transform": {
        "location": [121.51117039,31.2503413,30], //The position corresponding to the bottom center of the section body
        "rotator": {
          "pitch": 0, //pitch, reference(-180~180)
          "yaw": 30, //yaw, reference(0 positive north, -180~180)
          "roll": 0 //roll, reference(-180~180)
        },
        "scale3d": [1, 1, 1]
```

```
    }
  })
}
```

```
async function EndSection () {
  const res = await App.Scene.Section.End();
}
```

Project CustomizeApi

```
const jsondata = {
  "apiClassName": "WdpCameraControlAPI",
  "apiFuncName": "GetCameraInfo",
  "args": {
      // json data
   }
}
const res = await App.Customize.RunCustomizeApi(jsondata);
console.log(res);
```

Get Scene Snapshot

```
/**
* @param {number[]} resolution - The resolution of the image, represented as [width, height].
* @param {number} quality - The quality of the snapshot, ranging from [0, 1], where 0 represents the lowest quality and 1 represents the highest quality.
*/
await App.Renderer.GetSnapshot([1920, 1080], 0.8);
```

RGBA to HEXA

```
/**
* RGBA object to 16-digit color string
* @param {string} decimal - RGBA: The value range for A is 0~1.
* @param {string} integer - RGBA: The value range for A is 0~255.
*/

const color = {
  r: 100,
  g: 50,
  b: 200,
  a: 1
}

const res = await App.Tools.Color.RgbaToHexa(color, 'decimal');
```

HEXA to RGBA

```
/**
* 16-digit color string to RGBA object
* @param {string} decimal - RGBA: The value range for A is 0~1.
* @param {string} integer - RGBA: The value range for A is 0~255.
*/

const res = await App.Tools.Color.HexaToRgba('f38929ff', 'decimal');
```