

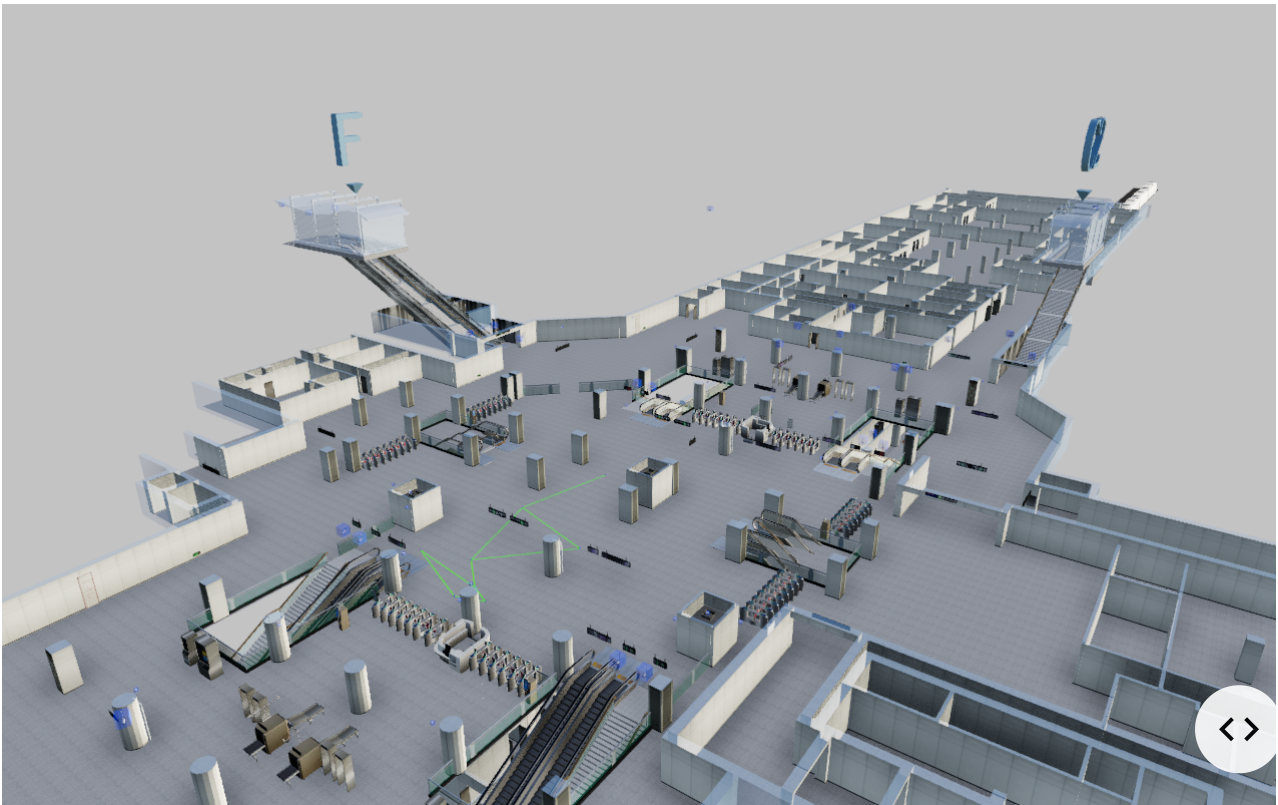
plugin-cps-soonmanager

npm@latest v2.9.10

CPS 平台 生产的场景加载及数据读取。

此插件是基于 **soonmanager2-sync** 插件的扩展，并完全向下兼容。

样例



安装

```
npm install @soonspacejs/plugin-cps-soonmanager
# or
yarn add @soonspacejs/plugin-cps-soonmanager
```

sh

使用方法

```
import SoonSpace from 'soonspacejs';
import CpsSoonmanagerPlugin from '@soonspacejs/plugin-cps-soonmanager';

const ssp = new SoonSpace({
  el: '#view',
  options: {},
  events: {},
});

const cpsSoonmanagerPlugin = ssp.registerPlugin(CpsSoonmanagerPlugin, 'cpsSoo
console.log(cpsSoonmanagerPlugin);
```

属性

path

资源加载的基础路径

- 默认值: ''
- 类型: string

metaData readonly

场景元数据

调用 **fetchMetaData** 方法时会设置此属性

- 默认值: null
- 类型: IMetadata | null

定义

☰ SoonSpace.js 2.x

```
projectId: number;  
version: number;  
name: string;  
projectId: string;  
sceneId: string;  
cover: string | null;  
flatModel: string;  
treeModel: string;  
resource: string;  
exportTime: number;  
environment?: string;  
}
```

treeData readonly

场景树数据

调用 **loadScene** 方法时会设置此属性

- 默认值: `null`
- 类型: `ITreeData[] | null`

定义

```
interface ITreeData {  
  id: string;  
  pid: string | null;  
  name: string;  
  renderType: 'GROUP' | '3D' | 'ROOM' | 'STUB';  
  deviceName: string | null;  
  deviceCode: string | null;  
  matrix: number[];  
  path: string | null;  
  familyId: string | null;  
  children: ITreeData[];  
}
```

ts

提示

除了 `children` 其他字段都会存在每个对象的 `userData` 上

☰ SoonSpace.js 2.x

```
const deviceModel = ssp.getObjectByUserDataProperty('deviceCode', 'kx-1
```

poiData readonly

场景内配置 Poi 数据, 该数据在加载场景 (loadScene) 时自动获取。

- 默认值: `null`
- 类型: `IPoiData[] | null`

定义

```
export enum PoiContentTypeEnum {  
    PANEL = 'PANEL',  
    VIDEO = 'VIDEO',  
    VIDEO_STREAM = 'VIDEO_STREAM',  
}  
  
interface IPoiData {  
    projectId: string;  
    sceneId: string;  
    nodeId: string;  
    poiId: string;  
    name: string;  
    width: number;  
    height: number;  
    x: number;  
    y: number;  
    z: number;  
    rotationX: number;  
    rotationY: number;  
    rotationZ: number;  
    scale: number;  
    dimensional: PoiNodeType;  
    content: string;  
    media: Record<string, string> | null;  
    contentType: PoiContentTypeEnum;  
    display: boolean;
```

ts

☰ SoonSpace.js 2.x

topologyData readonly

拓扑路径数据

调用 **getTopologies** 方法时会设置此属性

- 默认值: `null`
- 类型: `TopologyInfo[]` | `null`

propertiesData readonly

自定义属性数据，根据 `modelId` 分组

调用 **fetchPropertiesData** 方法时会设置此属性

- 默认值: `null`
- 类型: `TPropertiesMap` | `null`

定义

```
interface IProperties {  
  modelId: string;  
  group: string;  
  key: string;  
  value: string | null;  
  label: string | null;  
}  
  
type TPropertiesMap = Map<IProperties['modelId'], IProperties[]>;
```

ts

animationsData readonly

补间动画数据，根据 `modelId` 分组

调用 **fetchAnimationsData** 方法时会设置此属性

≡ SoonSpace.js 2.x

定义

```
interface IKeyframe {  
  id: string;  
  uuid: string;  
  x: number;  
  y: number;  
  z: number;  
  scaleX: number;  
  scaleY: number;  
  scaleZ: number;  
  rotationX: number;  
  rotationY: number;  
  rotationZ: number;  
  easing: AnimationModeType;  
  mode: string;  
  delay: number;  
  duration: number;  
  repeat: number;  
  yoyo: boolean;  
}  
  
/**  
 * 动画  
 */  
interface IAnimations {  
  id: string;  
  uuid: string;  
  modelId: string;  
  name: string;  
  keyframes: IKeyframe[];  
}  
  
type TAnimationsMap = Map<IAnimations['modelId'], IAnimations[]>;
```

modelVisionsData readonly

模型视角数据，根据 `nodeId` 分组

调用 `fetchModelVisionsData` 方法时会设置此属性

☰ SoonSpace.js 2.x

定义

```
interface IModelVisions {  
  id: string;  
  uuid: string;  
  nodeId: string;  
  name: string;  
  code?: any;  
  position: IVector3;  
  rotation: IVector3;  
  target: IVector3;  
}  
  
type TModelVisionsMap = Map<IModelVisions['nodeId'], IModelVisions[]>;
```

提示

Map 的 key 为 "" 时，表示场景视角数据

soonflow readonly

流程执行引擎实例

- 类型: SoonFlow [🔗](#)

flowData readonly

场景中配置好的流程数据，数据可提供给 runFlowById 使用。

- 默认值: `[]`
- 类型: `Array`

方法

≡

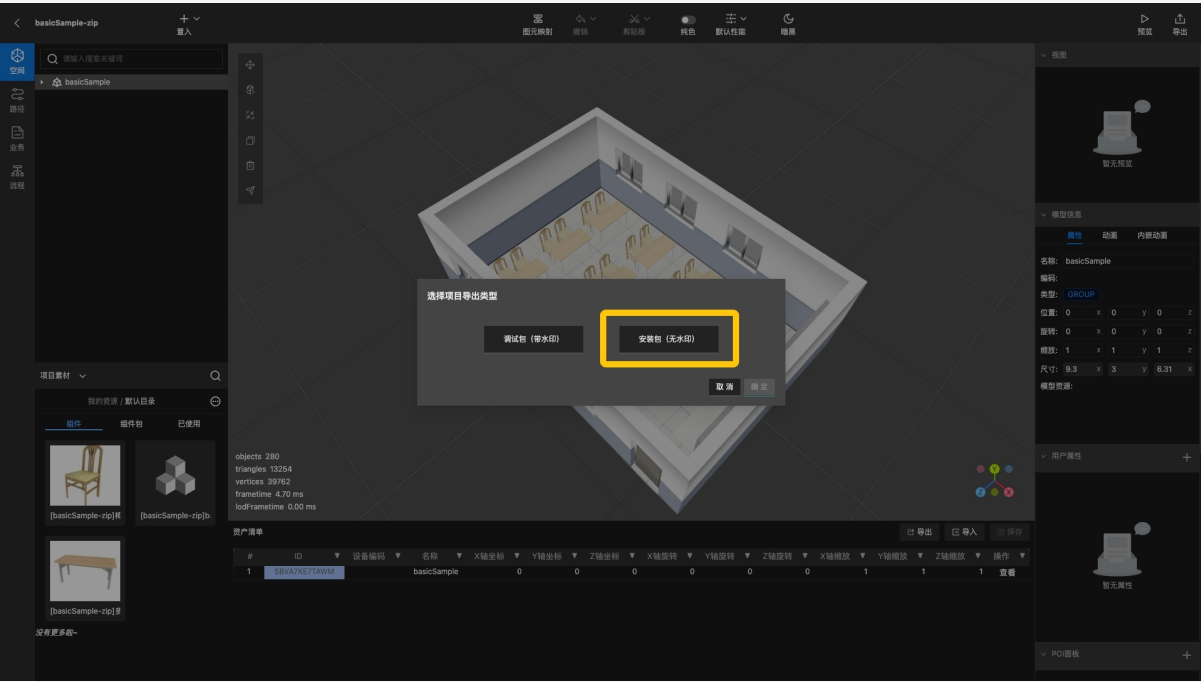
SoonSpace.js 2.x

setKey

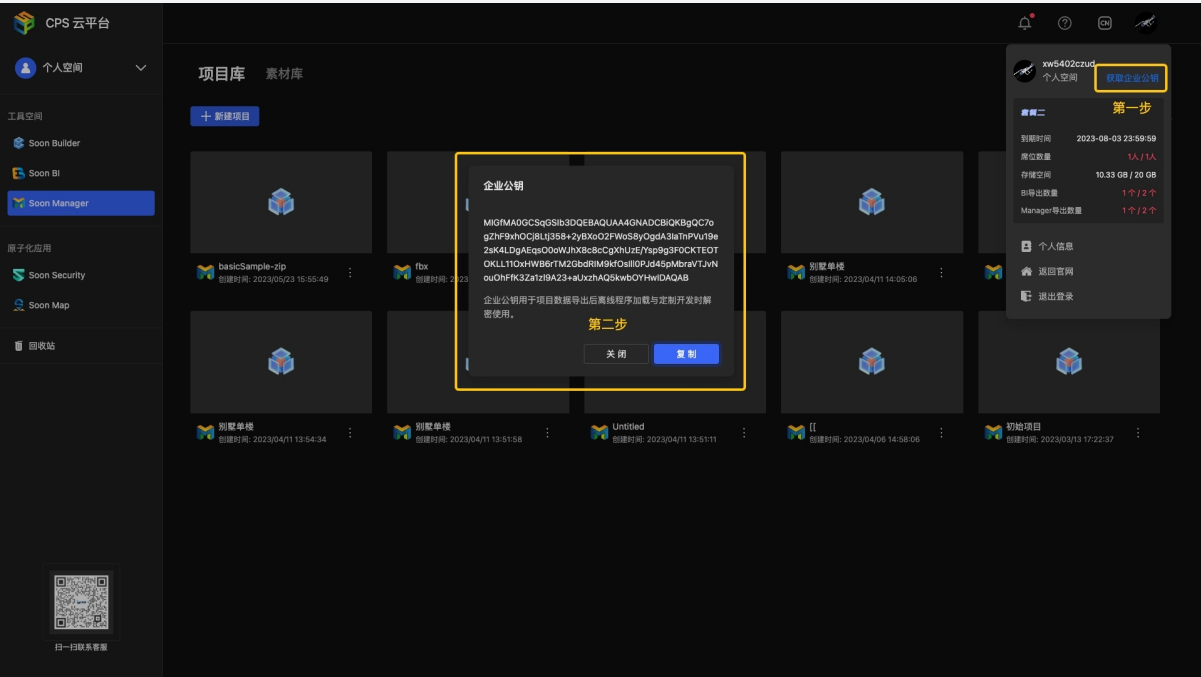
设置企业公钥

提示

如下图，只有使用 **安装包** 需去除场景水印时才需要设置企业公钥。



获取企业公钥, 请联系 CPS 平台企业 管理员 按下图提示操作获取。



整体设计逻辑

☰

SoonSpace.js 2.x

调试包	否	是
安装包	是	否
旧版资源包	否	否

定义

```
function setKey(key: string): void;
```

ts

用法

```
cpsSoonmanagerPlugin.setKey('xxxxxxxxxxxxxxxx');
```

js

注意

需要在调用 `loadScene` 之前调用 `setKey` 方法，否则安装包将无法加载

setPath

设置加载资源的基础路径

定义

```
function setPath(path: string): void;
```

ts

用法

```
cpsSoonmanagerPlugin.setPath('./models');  
// or  
cpsSoonmanagerPlugin.setPath('http://xxx.com/models');
```

js

☰ SoonSpace.js 2.x

插件的其他方法依赖于 `path`，需要先设置才能使用

loadScene

加载场景对象

配合 `loadTargetId` 或 `loadLevel` 使用可以重复调用

定义

```
export enum LoadSceneAlgorithm {  
  BFS = 'BFS', // 广度优先  
  DFS = 'DFS', // 深度优先  
}
```

ts

```
interface ILoadSceneOptions {  
  /**  
   * 平台解密公钥  
   */  
  key?: string;  
  /**  
   * 资源包路径  
   */  
  path?: string;  
  /**  
   * 同步自定义属性  
   */  
  syncProperties?: boolean;  
  /**  
   * 同步模型视角数据  
   */  
  syncModelVisions?: boolean;  
  /**  
   * 计算 bounds tree  
   */  
  needsModelsBoundsTree?: boolean;  
  /**  
   * 应用预设效果  
   */  
}
```

ts

☰ SoonSpace.js 2.x

```
    * 同步场景算法 DFS 与 BFS
    */
    loadSceneAlgorithm?: LoadSceneAlgorithm;
    /**
     * 目标节点 id (DFS时有效)
     */
    loadTargetId?: ITreeData['id'];
    /**
     * 需要加载的层级 (DFS时有效)
     */
    loadLevel?: number;
    /**
     * 加载 poi
     */
    loadPoi?: boolean;
    /**
     * 通过数据源刷新 poi
     */
    refreshPoiByDataSource?: boolean;
}


function loadScene(options?: ILoadSceneOptions): Promise<void>;
```

用法

```
cpsSoonmanagerPlugin.loadScene().then(() => {
  console.log('场景对象加载完成');
});
```

js

提示

某些模型文件可能应用了 **DRACO**  压缩，建议调用 loadScene 方法之前调用 **setModelDracoDecoderPath** 方法设置 DRACO 解压路径

提示

如果你需要使用 Worker 计算 BVH，可以关闭默认行为

☰ SoonSpace.js 2.x

```
type: 'worker',  
workerCreator,  
});  
});
```

具体请查看 [computeModelsBoundsTree](#)

提示

初始化加载大场景时，可以通过 `loadSceneAlgorithm` 参数设置加载场景算法为 `BFS`，可以提升部分加载时间。

```
import { LoadSceneAlgorithm } from '@soonspacejs/plugin-cps-soonmanager' // ts  
  
cpsSoonmanagerPlugin  
  .loadScene({  
    loadSceneAlgorithm: LoadSceneAlgorithm.BFS,  
  })  
  .then(() => {  
    ssp.flyMainViewpoint();  
  
    console.log('场景对象加载完成');  
  });
```

参数

options

- 描述: 场景加载选项
- 必填: **✗**
- 类型: `ILoadSceneOptions`

ILoadSceneOptions

≡

SoonSpace.js 2.x

key	等价于 setKey 方法	string	
path	等价于 setPath 方法	string	
syncProperties	是否同步自定义属性， 开启时自动调用 fetchPropertiesData 方法	boolean	
syncModelVisions	是否同步节点视角数据， 开启时自动调用 fetchModelVisionsData 方法	boolean	
needsModelsBoundsTree	场景加载完成后调用 ssp.computeModelsBoundsTree 方法	boolean	
applyPresetEffects	默认调用 presetEffects 方法	boolean	
loadSceneAlgorithm	加载场景使用的算法	LoadSceneAlgorithm	
loadTargetId	加载的目标树节点id	string	
loadLevel	加载的树层级。 如果设置了loadTargetId， 则以此为起始层。从1开始计算	number	
loadPoi	调用 loadPoi 方法	boolean	
refreshPoiByDataSource	调用 refreshPoiByDataSource 方法	boolean	

提示

自定义属性存储在对象的 `userData.properties` 属性上

分层加载示例

☰

SoonSpace.js 2.x

3F

4F

5F

加载全部

Close Controls

objects 41
triangles 101717
vertices 305151
frametime 16.40 ms

<>

警告

由于场景模型是嵌套的树结构，内部对象的矩阵变换依赖父级，如果先加载内部，可能会出现位置、旋转、缩放的错乱

建议 `loadTargetId` 设置为上层节点的 id

presetEffects

设置预设效果

定义

```
interface IPresetEffectsOptions {  
  hdr?: boolean;  
  ssao?: boolean;  
  directionalLightShadow?: boolean | { angle?: number };  
}  
  
function presetEffects(options?: IPresetEffectsOptions): Promise<void>;
```

ts

SoonSpace.js 2.x

用法

```
js
await cpsSoonmanagerPlugin.loadScene();
await cpsSoonmanagerPlugin.presetEffects({
  hdr: true,
  ssao: true,
  directionalLightShadow: true,
});
```

参数

options

- 描述: 效果参数
- 必填: ✗
- 类型: `IPresetEffectsOptions`

IPresetEffectsOptions

属性	描述	类型	必填	默认值
hdr	使用资源包中预设的hdr 环境	boolean	✗	true
ssao	开启 SSAO 效果	boolean	✗	true
directionalLightShadow	开启平行光阴影	boolean	✗	true

各参数对应的方法

参数	对应的内部方法
hdr	setEnvironment
ssao	setSSAO
directionalLightShadow	createDirectionalLight

注意

☰ SoonSpace.js 2.x

getTopologies

获取拓扑路径数据

定义

```
function getTopologies(): Promise<TopologyInfo[]>;
```

ts

用法

```
const topologiesInfo = await cpsSoonmanagerPlugin.getTopologies();

/**
 * 每个数组元素对应一个拓扑路径
 * 使用获取到的数据直接创建拓扑路径
 */
ssp.createTopology(topologiesInfo[0]);
ssp.createTopology(topologiesInfo[1]);
ssp.createTopology(topologiesInfo[2]);
```

js

sortTopologyNodes

对拓扑路径数据的 nodes 进行排序（只适用于线路）

定义

```
function sortTopologyNodes(topologyInfo: TopologyInfo, startNodeId?: TopologyInfo)
```

ts

用法

```
const [topologyInfo] = await cpsSoonmanagerPlugin.getTopologies();
```

js

≡ SoonSpace.js 2.x

```
const sortedTopologyInfo = cpsSoonmanagerPlugin.sortTopologyNodes(topologyInfo);

ssp.createTopology({
  sortedTopologyInfo,
  imgUrl: 'xxx.png',
  animation: true,
});
```

在播放路径动画或使用**巡检插件**时会按照 nodes 数组的顺序执行

所以可能需要使用此方法对线路的 nodes 排序

playObjectAnimation

根据动画数据播放对象的补间动画

定义

```
type TAnimationsTweenProps = Pick<IKeyframe, 'x' | 'y' | 'z' | 'rotationX' |  

interface IPlayAnimationByIdOptions {  

  autoStopPrevious?: boolean;  

  onUpdate?: (source: TAnimationsTweenProps, tween: Tween<TAnimationsTweenPro  

  onStart?: (tween: Tween<TAnimationsTweenProps>) => void;  

}  

function playObjectAnimation(object: BaseObject3D, animationIndex?: number, o
```

用法

```
const object = ssp.getObjectByUserDataProperty('deviceCode', '111');
cpsSoonmanagerPlugin.playObjectAnimation(object, 0, {
```

☰

SoonSpace.js 2.x

```
onStart: (tween) => {  
    /**  
     * 包含多个帧动画时，每个动画帧开始时 onStart 都会执行  
     */  
    console.log(tween);  
},  
});
```

参数

object

- 描述: 要播放动画的对象
- 必填: ✓
- 类型: BaseObject3D

animationIndex

- 描述: 该动画所在数据列表中的下标
- 必填: ✗
- 默认值: 0
- 类型: number

options

- 描述: 动画播放选项
- 必填: ✗
- 类型: IPlayAnimationByIdOptions

IPlayAnimationByIdOptions

属性	描述	类型
autoStopPrevious	是否自动停止之前的动画	boolean
onUpdate	动画更新回调	IPlayAnimationByIdOptions['onUpdate']
onStart	动画开始回调	IPlayAnimationByIdOptions['onStart']

☰ SoonSpace.js 2.x

动画播放时，可以是多个 `animation` 的组合

所以每次执行新的 `animation` 方法时都会执行 `onStart` 回调并且返回新的 `tween` 实例

stopObjectAnimation

停止由 `playObjectAnimation` 方法触发的补间动画

定义

```
function stopObjectAnimation(object: BaseObject3D): Promise<void>;
```

ts

用法

```
cpsSoonmanagerPlugin.stopObjectAnimation(object);
```

js

flyToSceneFromVisionsData

根据场景视角数据飞向

定义

```
function flyToSceneFromVisionsData(index?: number, options?: AnimationOptions
```

ts

用法

```
cpsSoonmanagerPlugin.flyToSceneFromVisionsData(0);
```

js

参数

☰

SoonSpace.js 2.x

index

- 描述: 该视角所在数据列表中的下标
- 必填: ✖
- 类型: `number`

options

- 描述: 动画参数配置
- 必填: ✖
- 类型: `AnimationOptions`

flyToObjectFromVisionsData

根据对象视角数据飞向

定义

```
function flyToObjectFromVisionsData(object: BaseObject3D, index?: number, options?: AnimationOptions) {}
```

用法

```
const model = ssp.getObjectByUserDataProperty('device', 'xxx');

cpsSoonmanagerPlugin.flyToObjectFromVisionsData(model, 0);
```

参数

object

- 描述: 场景对象
- 必填: ✔
- 类型: `BaseObject3D`

index

☰ SoonSpace.js 2.x

- 类型: `number`

options

- 描述: 动画参数配置
- 必填: **X**
- 类型: `AnimationOptions`

runFlowById

手动执行场景流程，流程 id 可在 `flowData` 中获取。

定义

```
function runFlowById(id: string): void;
```

ts

用法

```
// 假设执行第一条流程  
cpsSoonmanagerPlugin.runFlowById(cpsSoonmanagerPlugin.flowData[0].id);
```

ts

loadPoi

根据 poiData 渲染 Poi

定义

```
function loadPoi(refreshByDataSource: boolean): Promise<void>;
```

ts

用法

☰ SoonSpace.js 2.x

参数

refreshByDataSource

- 描述: 加载完调用 **refreshPoiByDataSource**
- 必填: **X**
- 类型: `boolean`

refreshPoiByDataSource

通过数据源刷新 Poi

定义

```
function refreshPoiByDataSource(): Promise<void>;
```

ts

用法

```
cpsSoonmanagerPlugin.loadPoi();  
cpsSoonmanagerPlugin.refreshPoiByDataSource();
```

ts

fetchMetaData

根据当前 `path` 获取场景元数据

由 **loadScene** 方法调用

定义

```
function fetchMetaData(): Promise<IMetadata>;
```

ts

用法

☰ SoonSpace.js 2.x

```
});
```

fetchTreeData 📄

根据当前 `path` 获取场景树数据

定义

```
function fetchTreeData(): Promise<ITreeData[]>;
```

ts

用法

```
cpsSoonmanagerPlugin.fetchTreeData().then((treeData) => {  
  console.log(treeData);  
});
```

ts

注意

此方法已不适用于加密资源包

fetchPropertiesData

根据当前 `path` 获取自定义属性数据

由 `loadScene` 方法调用

定义

```
function fetchPropertiesData(): Promise<TPropertiesMap>;
```

ts

用法

☰ SoonSpace.js 2.x

```
});
```

fetchAnimationsData

根据当前 `path` 获取补间动画数据

由 `playAnimationById` 方法调用

定义

```
function fetchAnimationsData(): Promise<TAnimationsMap>;
```

ts

用法

```
cpsSoonmanagerPlugin.fetchAnimationsData().then((animationsData) => {  
  console.log(animationsData);  
});
```

ts

fetchModelVisionsData

根据当前 `path` 获取模型视角数据

由 `flyToSceneFromVisionsData` 和 `flyToObjectFromVisionsData` 方法调用

定义

```
function fetchModelVisionsData(): Promise<TModelVisionsMap>;
```

ts

用法

```
cpsSoonmanagerPlugin.fetchModelVisionsData().then((modelVisions) => {  
  console.log(modelVisions);  
});
```

ts

☰

SoonSpace.js 2.x

← plugin-soonmanager2-sync

plugin-cps-scheme →