

Class: default

Constructors

Constructor

```
new default( config ): PolyvLive
```

Parameters

config

[PlayerConfig](#)

Returns

[PolyvLive](#)

Methods

changeLevel()

```
changeLevel( level ): void
```

切换清晰度

Parameters

level

number

清晰度索引 0 代表流畅

Returns

void

Description

切换清晰度, 切换清晰度后, 会触发 `UPDATE_MEDIA_INFO` 事件

Example

```
<button bind:tap="changeLevel"></button>

player.on(PlayerEventsType.UPDATE_MEDIA_INFO, (mediaInfo: MediaInfo) => {
  const { playType, src } = mediaInfo;
  this.setData({ src });
});

// 调用切换清晰度接口, 触发`UPDATE_MEDIA_INFO`事件更新视频播放地址
function changeLevel() {
  player.changeLevel();
}
```

changeLine()

```
changeLine( line ): void
```

切换线路

Parameters

line

number

线路索引 0 代表线路1

Returns

void

Description

切换线路, 切换线路后, 会触发 UPDATE_MEDIA_INFO 事件

Example

```
<button bind:tap="changeLine"></button>

player.on(PlayerEventsType.UPDATE_MEDIA_INFO, (mediaInfo: MediaInfo) => {
  const { playType, src } = mediaInfo;
  this.setData({ src });
});

// 调用切换线路接口, 触发`UPDATE_MEDIA_INFO`事件更新视频播放地址
function changeLine() {
  player.changeLine(0);
}
```

changePlayCore()

```
changePlayCore( playCore ): void
```

切换延迟内核

Parameters

playCore

number

延迟内核索引 0 代表普通延迟 1 代表无延迟

Returns

void

Description

切换延迟内核, 切换延迟内核后, 会触发 UPDATE_MEDIA_INFO 事件

destroy()

```
destroy(): void
```

销毁播放实例

Returns

```
void
```

getPlayType()

```
getPlayType(): PlayType
```

获取播放类型

Returns

```
PlayType
```

播放类型

Description

可以通过 `getPlayType` 主动获取当前的播放类型

Example

```
const playType = player.getPlayType();

if (playType === PlayType.WAIT_START) {
  console.log('当前暂无直播，显示暂未直播盖图');
}
```

off()

```
off< K >( event , listener ): void
```

取消事件监听

Type Parameters

K

K extends keyof [PlayerEventMap](#)

Parameters

event

K

listener

[PlayerEventMap](#) [**K**]

Returns

void

on()

```
on< K >( event , listener ): void
```

事件监听

Type Parameters

K

K extends keyof [PlayerEventMap](#)

Parameters

event

K

listener

PlayerEventMap [K]

Returns

void

refresh()

```
refresh(): void
```

刷新当前播放流

Returns

void

Description

一般用于直播过程中, 出现卡顿等情况, 主动刷新当前播放流

Example

```
<button bind:tap="refresh"></button>

player.on(PlayerEventsType.UPDATE_MEDIA_INFO, (mediaInfo: MediaInfo) => {
  const { playType, src } = mediaInfo;
  this.setData({ src });
});

// 调用刷新流接口, 触发`UPDATE_MEDIA_INFO`事件更新视频播放地址
function refresh() {
  player.refresh();
}
```

setLivePlayerStateChange()

```
setLivePlayerStateChange( e ): void
```

监测播放组件状态回调

Parameters

e

LivePlayerStateChange

Returns

void

setStatics()

```
setStatics( statics ): void
```

设置统计参数

Parameters

statics

PlayerStatisticsSetting

统计参数

Returns

void

Description

播放中途如设置昵称, 可以通过 `setStatics` 更新统计参数。一般param1用于设置用户唯一Id, param2用于设置用户昵称等信息

Example

```
player.setStatics({  
  param2: '昵称',  
});
```

setUp()

```
setUp(): Promise < void >
```

播放器初始化

Returns

Promise < void >

Description

初始化播放器, 包括鉴权, 解析当前状态等操作