

# 3DCAT-JSSDK 开发文档-V0.3.0

- JSSDK npm 地址

目录

## 3DCAT-JSSDK 开发文档-V0.3.0

### 1、快速启动

#### 2、启动器(Launcher)

- 2.1、构造属性(Constructors)
- 2.2、属性(Attributes)
- 2.3、方法(Methods)
- 2.4、示例(Examples)
  - 2.4.1、获取实时连接各种状态

#### 3、连接管理(Connection)

- 3.1、属性(Attributes)
- 3.2、方法(Methods)
- 3.3、事件(Events)
- 3.4、示例(Examples)
  - 3.4.1、调节码率

#### 4、播放管理(LivePlayer)

- 4.1、属性(Attributes)
- 4.2、方法(Methods)
- 4.3、示例(Examples)
  - 4.3.1、调节播放实例显示模式
    - 4.3.1.1、拉伸模式
    - 4.3.1.2、裁剪模式
    - 4.3.1.3、自适应模式

#### 5、鼠标键盘控制相关接口(可选扩展)

- 5.1、示例
- 5.2、键盘事件(Keyboard)
  - 5.2.1、构造属性(Constructors)
- 5.3、鼠标移动事件(MouseMove)
  - 5.3.1、构造属性(Constructors)
- 5.4、鼠标按钮事件(MouseButton)
  - 5.4.1、构造属性(Constructors)
- 5.5、缩放事件(WheelScroll)
  - 5.5.1、构造属性(Constructors)
- 5.6、触控事件(TouchSet)
  - 5.6.1、构造属性(Constructors)
- 5.7、文本传输事件(TextInput)
  - 5.7.1、构造属性(Constructors)
- 5.8、鼠标控制管理相关(PointerManager)
  - 5.8.1、构造属性(Constructors)
  - 5.8.2、属性(Attributes)
  - 5.8.3、方法(Methods)

#### 6、多种接入场景 demo/参考

- 6.1、快速接入
  - 6.1.1、公有化
  - 6.1.2、私有化
- 6.2、错误代码使用说明

- 6.3、接入消息双向通信(ue4、unity)
- 6.4、进入有效画面后初始化
- 6.5、接入微信小程序
- 6.6、投屏交互接入
- 6.7、替换加载页面各阶段提示
- 6.8、去除/替换加载流程 UI 页面
- 6.9、接入多点触控需求 (UE4)
- 6.10、浏览器直接接入 sdk
- 6.11、聚焦应用输入框, 发送内容(复制粘贴)
- 6.12、调节码率
- 6.13、消息通信传输中文乱码问题
- 6.14、获取连接状态具体说明
- 6.15、RTCPeerConnection、RTCDataChannel 状态说明
- 6.16、WebRTC 调试参考
- 6.17、键盘事件自定义
- 6.18、自定义工具栏(实现上报功能)

## 7、更新历史

- v0.3.0 (2022-07-04)
- v0.2.15 (2022-07-01)
- v0.2.14 (2022-06-28)
- v0.2.13 (2022-06-24)
- v0.2.12 (2022-06-22)
- v0.2.11 (2022-06-21)
- v0.2.10 (2022-06-18)
- v0.2.9 (2022-05-16)
- v0.2.8 (2022-05-12)
- v0.2.7 (2022-05-10)
- v0.2.6 (2022-04-21)
- v0.2.5 (2022-04-18)
- v0.2.4 (2022-04-15)
- v0.2.3 (2022-04-02)
- v0.2.2 (2022-03-25)
- v0.2.1 (2022-03-07)
- v0.2.0 (2022-03-02)
- v0.1.1 (2022-02-28)
- v0.1.0 (2022-02-24)
- v0.1.0-beta.6 (2022-01-06)
- v0.1.0-beta.5 (2021-12-06)
- v0.1.0-beta.4 (2021-11-23)
- v0.1.0-beta.3 (2021-11-09)
- v0.1.0-beta.2 (2021-05-11)
- v0.1.0-beta.1 (2021-04-17)
- v0.1.0-beta.0 (2021-02-19)

# 1、快速启动

- 以公有化启动流程为例

```
import { Launcher } from "live-cat";
const appKey = "xxxxxxxxxxxxxxxxxxxx";
const address = "https://app.3dcat.live";
const bootstrap = async () => {
  try {
    const launch = new Launcher({
      baseOptions: {
```

```

    address,
    appKey,
    startType: 1, // 1:普通链接 | 3:投屏链接
  },
});
const player = document.querySelector("body");
document.querySelector("body").style.width = "100vw";
document.querySelector("body").style.height = "100vh";
await launch.automata(player);
} catch (error) {
  console.error(error);
}
};

window.addEventListener("DOMContentLoaded", () => {
  if (
    navigator.userAgent.includes("miniProgram") ||
    navigator.userAgent.includes("MicroMessenger")
  ) {
    //微信浏览器/微信小程序环境
    document.addEventListener("WeixinJSBridgeReady", bootstrap, false);
  } else {
    bootstrap();
  }
});

```

## 2、启动器(Launcher)

### 2.1、构造属性(Constructors)

属性	说明	类型	是否必填
baseOptions	基础配置		必填
address	连接地址	string	必填
appKey	appKey	string	必填
startType	1: 普通连接   3: 投屏连接	number	必填
appSecret	密钥	string	-
isCastScreenMaster	(投屏)是否为主控	boolean	-
castScreenMaxQty	(投屏)最大并发数(最大为 5)	number	-
castScreenNickname	(投屏)昵称	string	-
castScreenPwd	(投屏)投屏密钥	boolean	-
joinType	(投屏)加入方式 (1: 密钥   3: 链接)	number	-
extendOptions	拓展配置		-

onPlay 属性	开始播放回调 说明	() => void 类型	- 是否必填
ueMultiTouch	是否开启 UE4 应用多点触控	boolean	-
minBitrate	初始化最小码率(默认为: 2000)	number	-
maxBitrate	初始化最大码率(默认为: 5000)	number	-
startBitrate	初始化码率(默认为: 4000)	number	-
exeParameter	自定义参数	string	-
openWebRTCStats	开启 WebRTC 日志打印	boolean	-
holdToolDisplay	是否保持工具栏显示	boolean	-
toolOption	工具栏选项		-
dropTools	删除工具栏	DropToolsType	-
toolsIndex	工具栏位置(从 0 开始)	number[]	-
platform	平台 (1:pc   2:移动端   3:全部)	number	-
extendTools	新增工具栏	ExtendToolsType[]	-
icon	图标(base64/url)	string	-
text	标题	string	-
platform	平台 (1:pc   2:移动端   3:全部)	number	-
order	新增位置(从 0 开始)	number	-
onClick	点击触发事件	() => void	-
eventOption	事件选项		-
enableKeyBoard	是否挂载键盘事件(默认为 true)	boolean	-

## 2.2、属性(Attributes)

属性名	说明	类型	默认值
runningState.isOrientationMatch	获取屏幕方向变化(false:横屏 true:竖屏)	boolean	true
moveSensitivity	获取鼠标或者触摸移动的敏感度	number(0: 低, 1: 中, 2: 高)	0

## 2.3、方法(Methods)

方法名	说明	参数	回调参数
getConnection	获取连接实例	-	连接实例 (Connection)
getPlayer	获取播放实例	-	播放实例 (LivePlayer)
report	获取实时连接状态(详细参考示例)	-	object
toggleStatistics	控制连接状态显示状态(详细参考示例)	-	object
toggleFullscreen	切换全屏(横屏) <a href="#">支持情况参考</a>	-	void
destory	断开全部连接	[errMsg]	void
setMoveSensitivity	设置鼠标或者触摸移动的敏感度	number(0: 低, 1: 中, 2: 高)	void
handleSubscribe	手动挂载鼠标、键盘、触控等事件	HTMLElement	void
handleUnsubscribe	手动取消鼠标、键盘、触控等事件	-	void

## 2.4、示例(Examples)

### 2.4.1、获取实时连接各种状态

```
launcher.toggleStatistics();
window.setInterval(() => {
  const { fps, bitrate, packetLossRate, latency, averageJitterBufferDelay } =
    launcher.report();
  console.log(`
    FPS: ${fps}
    bitrate: ${bitrate}kbps
    latency: ${latency}ms
    averageJitterBufferDelay: ${averageJitterBufferDelay}ms
    packetLossRate: ${(packetLossRate * 100).toFixed(3)}%
  `);
}, 1000);
```

## 3、连接管理(Connection)

## 3.1、属性(Attributes)

属性名	说明	类型	默认值
dc	获取 RTCDataChannel 实例	<a href="#">RTCDataChannel</a>	-
event	Connection 事件(参考 3.3 事件(Events))	object	-
pc	获取 RTCPeerConnection 实例	<a href="#">RTCPeerConnection</a>	-

## 3.2、方法(Methods)

方法名	说明	参数	回调参数
emitUIInteraction	发送消息到应用程序	string	Promise<boolean>
send	发送消息(第二个参数为 true 可重置超时时间)	(string Blob ArrayBuffer,boolean)	void
changeBandwidth	调节码率(详细参考示例)	number	void

## 3.3、事件(Events)

事件名	说明	回调参数
connect	可视化服务连接中	-
dataChannelConnected	连接成功, 资源加载中	-
close	连接中断回调	<a href="#">CloseEvent</a>
disconnect	信令断开回调	string
interaction	接收应用端返回数据	string
afk	超时断开	-

## 3.4、示例(Examples)

### 3.4.1、调节码率

```

Connection.changeBandwidth(
    8000, // start bitrate kbps
    10000, // max bitrate kbps
    5000 // min bitrate kbps
);
Connection.changeBandwidth(
    10000 // medians bitrate kbps
);

```

## 4、播放管理(LivePlayer)

### 4.1、属性(Attributes)

属性名	说明	类型	默认值
videoElement	播放实例节点	<a href="#">HTMLVideoElement</a>	-
playerElement	播放实例父级容器	<a href="#">HTMLDivElement</a>	-
videoVolume	获取 video 当前音量值 (应用声音)	number	0
displayRect	获取显示区域的参数	object	-

### 4.2、方法(Methods)

方法名	说明	参数	回调参数
handleChangeCanResize	设置播放实例显示模式自适应(详细看示例)	boolean	void
resizePlayer	初始化播放实例显示模式	-	void
showLoadingText	设置加载中文字	string	void
showTextOverlay	设置播放页遮盖文字	string	void
setVideoVolume	设置 video 播放音量值 (应用声音)	number	void
handleChangeLandscapeType	设置显示模式(详细参考 4.3 示例)	number(1:自适应 2:拉伸 3:裁剪)	void
handleChangeBgImage	设置应用加载背景图	string	void
handleChangeOrientationLock	移动端调节旋转	boolean(false:横屏 true:竖屏)	void

## 4.3、示例(Examples)

### 4.3.1、*调节播放实例显示模式*

默认为自适应模式，如需初始化其他显示模式，建议写在 Launcher->onPlay 钩子

#### 4.3.1.1、**拉伸模式**

```
livePlayer.handleChangeLandscapeType(2);
```

#### 4.3.1.2、**裁剪模式**

```
livePlayer.handleChangeLandscapeType(3);
```

#### 4.3.1.3、**自适应模式**

```
livePlayer.handleChangeLandscapeType(1);
```

# 5、鼠标键盘控制相关接口(可选扩展)

## 5.1、示例

```
import {
  Keyboard,
  MouseMove,
  MouseButton,
  WheelScroll,
  TouchSet,
  TextInput,
  PointerManager,
} from "ray-streaming";

//键盘事件
connection.send(
  new Keyboard(27, false, false, false, false, false, false, true).dumps()
);
//鼠标移动事件
connection.send(new MouseMove(100, 300, 3, 4).dumps());

//鼠标按钮事件
connection.send(new MouseButton(1, true).dumps());

//缩放事件
connection.send(new WheelScroll(10, true).dumps());
```



```
//触控事件
connection.send(new TouchSet(0, [{ x: 10, y: 10, id: 1 }]).dumps());

//文本传输事件
connection.send(new TextInput("test text").dumps());

//鼠标控制管理相关
new PointerManager(target);
```

## 5.2、键盘事件(Keyboard)

### 5.2.1、构造属性(Constructors)

属性	说明	类型
keycode	键码	number
alt	按下 alt	boolean
shift	按下 shift	boolean
ctrl	按下 ctrl	boolean
nlock	按下 nlock	boolean
clock	按下 clock	boolean
slock	按下 slock	boolean
down	按下	boolean

## 5.3、鼠标移动事件(MouseMove)

### 5.3.1、构造属性(Constructors)

属性	说明	类型
x	x	number
y	y	number
dx	dx	number
dy	dy	number

## 5.4、鼠标按钮事件(MouseButton)

### 5.4.1、构造属性(Constructors)

属性	说明	类型
mouseButtonType	left = 1,middle=2, right=3	number
down	按下	boolean

## 5.5、缩放事件(WheelScroll)

### 5.5.1、构造属性(Constructors)

属性	说明	类型
step	step	number
forward	forward	boolean

## 5.6、触控事件(TouchSet)

### 5.6.1、构造属性(Constructors)

属性	说明	类型
touchType	down = 0,update=1, up=2	number
touchList	TouchPoint: {x: number y: number id: number} //id: touch identifier	TouchPoint[]

## 5.7、文本传输事件(TextInput)

### 5.7.1、构造属性(Constructors)

属性	说明	类型
text	聚焦应用输入框时能够快速发送内容，比如实现复制粘贴功能	string

## 5.8、鼠标控制管理相关(PointerManager)

### 5.8.1、构造属性(Constructors)

属性	说明	类型
target	宿主容器	HTMLElement

属性	说明	类型
onExit	全局退出或者中断回调	() => void

### 5.8.2、属性(Attributes)

属性名	说明	类型	默认值
cursor	光标实例	base64:string,show:boolean,x:number,y:number	cursor

### 5.8.3、方法(Methods)

方法名	说明	参数	回调参数
setCursor	设置鼠标大小/形状	Cursor	void
destory	销毁	-	void

## 6、多种接入场景 demo/参考

### 6.1、快速接入

#### 6.1.1、公有化

```
import { Launcher } from "live-cat";
const appKey = "xxxxxxxxxxxxxxxxxxxx";
const address = "https://app.3dcat.live";
const bootstrap = async () => {
  try {
    const launch = new Launcher({
      baseOptions: {
        address,
        appKey,
        startType: 1,
      },
    });
    const player = document.querySelector("body");
    //容器必须指定宽高，否则不显示
    document.querySelector("body").style.width = "100vw";
    document.querySelector("body").style.height = "100vh";
    await launch.automata(player);
  } catch (error) {
    console.error(error);
  }
}
```

```

};
window.addEventListener("DOMContentLoaded", () => {
  if (
    navigator.userAgent.includes("miniProgram") ||
    navigator.userAgent.includes("MicroMessenger")
  ) {
    //微信浏览器/微信小程序环境
    document.addEventListener("WeixinJSBridgeReady", bootstrap, false);
  } else {
    bootstrap();
  }
});

```

## 6.1.2、私有化

```

import { LauncherPrivate } from "live-cat";
const appKey = "xxxxxxxxxxxxxxxxxxxx";
const address = "私有化服务器地址";
const bootstrap = async () => {
  try {
    const launch = new LauncherPrivate({
      address,
      appKey,
    });
    const player = document.querySelector("body");
    //容器必须指定宽高，否则不显示
    document.querySelector("body").style.width = "100vw";
    document.querySelector("body").style.height = "100vh";
    await launch.automata(player);
  } catch (error) {
    console.error(error);
  }
};
window.addEventListener("DOMContentLoaded", () => {
  if (
    navigator.userAgent.includes("miniProgram") ||
    navigator.userAgent.includes("MicroMessenger")
  ) {
    //微信浏览器/微信小程序环境
    document.addEventListener("WeixinJSBridgeReady", bootstrap, false);
  } else {
    bootstrap();
  }
});

```

## 6.2、错误代码使用说明

- 以公有化为例
- [错误代码说明文档](#)

```

import { Launcher } from "live-cat";
const appKey = "xxxxxxxxxxxxxxxxxxxx";
const address = "https://app.3dcat.live";

```

```

const bootstrap = async () => {
  try {
    const launch = new Launcher({
      baseOptions: {
        address,
        appKey,
        startType: 1,
      },
    });

    const player = document.querySelector("body");
    document.querySelector("body").style.width = "100vw";
    document.querySelector("body").style.height = "100vh";
    await launch.automata(player).catch((code) => {
      //错误代码
      console.error(code);
    });
  } catch (error) {
    console.error(error);
  }
};

window.addEventListener("DOMContentLoaded", () => {
  if (
    navigator.userAgent.includes("miniProgram") ||
    navigator.userAgent.includes("MicroMessenger")
  ) {
    //微信浏览器/微信小程序环境
    document.addEventListener("WeixinJSBridgeReady", bootstrap, false);
  } else {
    bootstrap();
  }
});

```

## 6.3、接入消息双向通信(ue4、unity)

- 以公有化为例
- 应用需要集成相应插件: ue4, unity

```

import { Launcher } from "live-cat";
const appKey = "xxxxxxxxxxxxxxxxxxxx";
const address = "https://app.3dcat.live";
const bootstrap = async () => {
  try {
    const launch = new Launcher({
      baseOptions: {
        address,
        appKey,
        startType: 1,
      },
    });

    const player = document.querySelector("body");
    document.querySelector("body").style.width = "100vw";
    document.querySelector("body").style.height = "100vh";
    await launch.automata(player);
    const connection = launch.getConnection();
    //web端->应用

```

```

connection.emitUIInteraction("text").then((res) => {
  console.log(res ? "发送成功" : "发送失败");
});
//应用->web端
connection.event.interaction.on((msg) => {
  console.log("收到应用发来的消息:", msg);
});
} catch (error) {
  console.error(error);
}
};
window.addEventListener("DOMContentLoaded", () => {
  if (
    navigator.userAgent.includes("miniProgram") ||
    navigator.userAgent.includes("MicroMessenger")
  ) {
    //微信浏览器/微信小程序环境
    document.addEventListener("WeixinJSBridgeReady", bootstrap, false);
  } else {
    bootstrap();
  }
});

```

## 6.4、进入有效画面后初始化

```

import { Launcher } from "live-cat";
const appKey = "xxxxxxxxxxxxxxxxxxxx";
const address = "https://app.3dcat.live";
const bootstrap = async () => {
  try {
    const launch = new Launcher({
      baseOptions: {
        address,
        appKey,
        startType: 1,
      },
      extendOptions: {
        onPlay: () => {
          // 初始化操作
        },
      },
    });
  };

  const player = document.querySelector("body");
  document.querySelector("body").style.width = "100vw";
  document.querySelector("body").style.height = "100vh";
  await launch.automata(player);
} catch (error) {
  console.error(error);
}
};
window.addEventListener("DOMContentLoaded", () => {
  if (
    navigator.userAgent.includes("miniProgram") ||
    navigator.userAgent.includes("MicroMessenger")
  ) {
    //微信浏览器/微信小程序环境

```

```
document.addEventListener("WeixinJSBridgeReady", bootstrap, false);
} else {
  bootstrap();
}
});
```

## 6.5、接入微信小程序

### ■ 步骤

1. 使用 jssdk 开发，打包成 html 部署到线上(生产环境需要域名解析)
  2. 微信小程序通过 web-view 嵌入 html 地址(域名链接)
- 开发 html 时候，微信(小程序)环境需要注意监听 WeixinJSBridgeReady 事件，然后开始对接 jssdk。

```
window.addEventListener("DOMContentLoaded", () => {
  if (
    navigator.userAgent.includes("miniProgram") ||
    navigator.userAgent.includes("MicroMessenger")
  ) {
    //微信浏览器/微信小程序环境
    document.addEventListener("WeixinJSBridgeReady", bootstrap, false);
  } else {
    bootstrap();
  }
});
```

## 6.6、投屏交互接入

- 投屏交互分为主控端、观看端（可被赋予临时操控权）
- 待补充

## 6.7、替换加载页面各阶段提示

### 1. 替换 "可视化服务启动中"

- 运行 await launch.automata(player)后，调用 launch.getPlayer()，获取 livePlayer，执行如下代码。

```
livePlayer.showLoadingText("xxxxxx");
```

### 2. 替换 "可视化服务连接中"

- 运行 await launch.automata(player) 后，调用 launch.getPlayer()，获取 livePlayer；调用 launch.getConnection()，获取 connection，执行如下代码。

```
connection.event.connect.on(() => livePlayer.showLoadingText("xxxxxx"));
```

### 3. 替换 "连接成功，资源加载中"

- 运行 `await launch.automata(player)` 后，调用 `launch.getPlayer()`，获取 `livePlayer`；调用 `launch.getConnection()`，获取 `connection`，执行如下代码。

```
connection.event.dataChannelConnected.on(() =>
  livePlayer.showLoadingText("xxxxxx")
);
```

#### 4. 替换所有的异常报错

- 运行 `await launch.automata(player)` 后，执行如下代码。

```
launch.destory("xxxx");
//or
livePlayer.showTextOverlay("xxxxxx");
```

## 6.8、去除/替换加载流程 UI 页面

- 挂载容器的时候 (`await launch.automata(player)`)，把 `player` 元素隐藏掉，展示业务动画，在出现有效画面后，在 `onPlay` 回调中把 `player` 元素显示出来，参考代码如下。
- 注意自动播放问题：[Autoplay policy in Chrome](#)，可能会出现资源加载中一直进不去，需要引导用户点击 `player` 元素再隐藏。

```
//以公有化为例
const launch = new Launcher({
  baseOptions: {
    address,
    appKey,
    startType: 1,
  },
  extendOptions: {
    onPlay: () => {
      //开始播放回调
      player.style.display = "block";
    },
  },
});
const player = document.querySelector("body");
document.querySelector("body").style.width = "100vw";
document.querySelector("body").style.height = "100vh";
document.querySelector("body").style.display = "none";
await launch.automata(player);
```

## 6.9、接入多点触控需求 (UE4)

- 接入 3DCAT 的 [UE4 插件](#)
- 设置 `ueMultiTouch` 为 `true`，参考代码如下。

```
//以公有化为例
const launch = new Launcher({
  baseOptions: {
    address,
    appKey,
```



```
    startType: 1,
  },
  extendOptions: {
    ueMultiTouch: true,
  },
});
const player = document.querySelector("body");
document.querySelector("body").style.width = "100vw";
document.querySelector("body").style.height = "100vh";
await launch.automata(player);
```

## 6.10、浏览器直接接入 sdk

- [参考链接](#)
- 需要引入 jssdk 的 umd 版本，以及相关依赖模块，具体见参考链接。

## 6.11、聚焦应用输入框，发送内容(复制粘贴)

- 运行 await launch.automata(player)后，调用 launch.getConnection()，获取 connection，执行如下代码。

```
//开发可以自定义检测到键盘的粘贴事件后执行
connection.send(new TextInput("test text").dumps());
```

## 6.12、调节码率

- 运行 await launch.automata(player)后，执行如下代码。
- 建议：用户的下行网络最好高于最高码率的 50%,体验最佳。

```
launcher.changeBandwidth(
  8000, // 初始码率(8M)
  10000, // 最高码率(10M)
  5000 // 最低码率(5M)
);
//or
launcher.changeBandwidth(
  10000 // 码率(均值)
);
```

## 6.13、消息通信传输中文乱码问题

- 遇到 web 端->应用，传输中文消息出现乱码问题,参考如下代码。
- 将 string 转为 base64，再传输过去，应用那边再解析 base64 成 string

```
const text = "text测试内容";
const base64Text = window.btoa(encodeURIComponent(text));
connection.emitUIInteraction(base64Text).then((res) => {
  console.log(res ? "发送成功" : "发送失败");
});
```

## 6.14、获取连接状态具体说明

- 通过 launcher 的 toggleStatistics 方法可以获取连接状态的数据，参考代码如下：

```
launcher.toggleStatistics();
window.setInterval(() => {
  const {
    fps,
    bitrate,
    packetLossRate,
    latency,
    averageJitterBufferDelay,
    framesDecoded,
    framesDropped,
    framesReceived,
    keyFramesDecoded,
  } = launcher.report();
  console.log(`
    FPS: ${fps}
    bitrate: ${bitrate}kbps
    latency: ${latency}ms
    averageJitterBufferDelay: ${averageJitterBufferDelay}ms
    packetLossRate: ${(packetLossRate * 100).toFixed(3)}%
  `);
}, 1000);
```

## 6.15、RTCPeerConnection、RTCDataChannel 状态说明

- RTCPeerConnection：通过 connection.pc 获取

```
//打印ICE的状态
console.log("RTCPeerConnection.connectionState", connection.dc.connectionState);
```

属性	说明	类型
new	表示至少有一个 ICE 连接 (RTCIceTransport (en-US) 或 RTCDtlsTransport (en-US) 对象) 处于 new 状态，并且没有连接处于以下状态：connecting、checking、failed、disconnected，或者这些连接都处于 closed 状态	string
connecting	表示至少有一个 ICE 连接处于正在建立连接	string
connected	表示每一个 ICE 连接要么正在使用 (connected 或 completed 状态)，要么已被关闭 (closed 状态)；并且，至少有一个连接处于 connected 或 completed 状态	string
disconnected	表示至少有一个 ICE 连接处于 disconnected 状态，并且没有连接处于 failed、connecting 或 checking 状态。	string

属性	说明	类型
failed	表示至少有一个 ICE 连接处于 failed 的状态	string
closed	表示 RTCPeerConnection 已关闭	string

- RTCDataChannel: 通过 connection.dc 获取

```
//打印数据通道状态
console.log("RTCDataChannel.readyState", connection.dc.readyState);
```

属性	说明	类型
connecting	正在初始化连接, 正在启动	string
open	数据通道已经建立	string
closing	正在关闭数据通道	string
closed	数据通道已经关闭	string

## 6.16、WebRTC 调试参考

-在任一浏览器标签地址打开 "chrome://webrtc-internals", 即可看到 WebRTC 的完整调试数据

## 6.17、键盘事件自定义

1. 启动器初始化的时候取消挂载默认键盘事件, OnPlay 执行自定义的键盘事件。参考如下:

```
let connection, player, ele;
const launch = new Launcher({
  baseOptions: { address, appKey, startType: 1 },
  extendOptions: {
    eventOption: {
      enableKeyBoard: false,
    },
    onPlay: () => {
      createKeyboardStream(ele);
    },
  },
});
await launch.automata(document.body);
connection = launch.getConnection();
player = launch.getPlayer();
ele = player.videoElement;
```

2. 然后自定义键盘事件, 比如实现组合键, 参考代码如下:

```
function createKeyboardStream(target) {
  //防止按到tab键切换dom
  target.tabIndex = -1;
  const onInteract = () => target.focus();
```

```

target.addEventListener("touchstart", onInteract, { passive: true });
target.addEventListener("mouseenter", onInteract);
target.addEventListener("click", onInteract);

target.addEventListener("keydown", (e) => {
  e.preventDefault();
  let capsLock = e.getModifierState("CapsLock");
  let shift = e.getModifierState("Shift");
  //includes " ~ ( ) _ { } : " < > ? "
  let signalKeyCode = [186, 187, 188, 189, 190, 191, 192, 219, 220, 221, 222];
  if (
    (capsLock || shift) &&
    (signalKeyCode.includes(e.keyCode) ||
      (e.keyCode >= 65 && e.keyCode <= 90) ||
      (e.keyCode >= 48 && e.keyCode <= 57))
  ) {
    connection.send(new TextInput(e.key, true).dumps(), true);
  }
  connection.send(Keyboard.fromKeyboardEvent(e, true).dumps(), true);
});

target.addEventListener("keyup", (e) => {
  e.preventDefault();
  connection.send(Keyboard.fromKeyboardEvent(e, false).dumps(), true);
});
}

```

## 6.18、自定义工具栏(实现上报功能)

- 可以自行新增工具栏，以上报功能为例

1. 启动器初始化的时候指定上报栏目。参考如下：

```

const launch = new Launcher({
  baseOptions: { address, appKey, startType: 1 },
  extendOptions: {
    toolOption: {
      extendTools: [
        {
          icon: "data:image/png;base64xxxxxxxx",
          text: "上报错误",
          platform: 3,
          order: 0,
          onClick: () => {
            API.post("/api/feed/xxxxxx", { error: "xxxxxx" });
          },
        },
      ],
    },
  },
});

```

# 7、更新历史

## v0.3.0 (2022-07-04)

1. U 修复一些已知问题，整合多个模块固定版本。

## v0.2.15 (2022-07-01)

1. F 原有方法/属性改动
  - 改动 1：启动器构造属性结构调整，分为 baseOptions、extendOptions
2. A 支持开启 WebRTC 调试打印
3. A 支持 PC 端/移动端工具栏可拔插
4. A 支持失败情况下挂载工具栏
5. A 支持自定义键盘事件(组合按键)

## v0.2.14 (2022-06-28)

1. U 原有方法/属性改动
  - 改动 1：启动器的 changeBandwidth 方法 -> 连接管理的 changeBandwidth 方法
  - 改动 2：播放管理的 setVideoVolume 方法
2. U 启动器的 handleSubscribe 方法参数调整
3. U 裁剪模式示例改动，新增播放管理裁剪方法 handleChangeLandscapeType
4. A 支持设置/获取播放音量
5. A 补充设置鼠标样式
6. A 支持设置应用加载背景图 handleChangeBgImage
7. A 新增移动端调节旋转 handleChangeOrientationLock
8. A 新增获取屏幕方向变化 runningState.isOrientationMatch
9. A 支持自定义参数传输 exeParameter
10. A 支持新增自定义 PC 端工具栏
11. A 支持 send 接口，支持重置超时时间

## v0.2.13 (2022-06-24)

1. F 修复一些已知问题

## v0.2.12 (2022-06-22)

1. F 修复一些已知问题

## v0.2.11 (2022-06-21)

- 1. F 修复一些已知问题

## v0.2.10 (2022-06-18)

- 1. F 修复一些已知问题

## v0.2.9 (2022-05-16)

- 1. F 修复一些已知问题

## v0.2.8 (2022-05-12)

- 1. A 私有化兼容 V1.2.1版本

## v0.2.7 (2022-05-10)

- 1. A 私有化兼容 V1.2.3版本
- 2. A 支持私有化自定义loading **加载动画、背景图、猫头工具 logo**
- 3. U 优化多点触控问题(UE)
- 4. U 优化 launcher 启动流程

## v0.2.6 (2022-04-21)

- 1. A 新增支持**多点触控(UE)**
- 2. F 修复 query 类型校验问题

## v0.2.5 (2022-04-18)

- 1. F 修复一些已知问题

## v0.2.4 (2022-04-15)

- 1. F 修复 react 版本兼容问题

## v0.2.3 (2022-04-02)

1. F 修复一些已知问题

## v0.2.2 (2022-03-25)

1. A 新增获取实例 token 方法(getToken)
2. A 新增错误捕获
3. U 优化移动端样式
4. U 允许状态、输入等控件拖拽
5. F 修复一些已知问题

## v0.2.1 (2022-03-07)

1. A 新增全屏切换(toggleFullscreen)
2. F 修复超时断开问题

## v0.2.0 (2022-03-02)

1. A 支持私有化启动(LauncherPrivate)

## v0.1.1 (2022-02-28)

1. A 新增调节播放实例显示模式方法(handleChangeCanResize、resizePlayer)
2. A 新增获取播放实例容器属性

## v0.1.0 (2022-02-24)

1. A 新增码率修改
2. A 新增 launcher 一键销毁(destory)
3. A 支持一推多流功能(demo 文档待补充)
4. U 废除 gatherStatistics、getRTT 方法
5. U 重构连接的状态统计, 改用 report
6. F 修复应用通讯信息丢失问题

## v0.1.0-beta.6 (2022-01-06)

1. A 新增 PC 端虚拟控件
2. A 新增 PC 端虚拟输入框

## v0.1.0-beta.5 (2021-12-06)

1. A 新增移动端移动敏感度设置

## v0.1.0-beta.4 (2021-11-23)

1. A 新增移动端屏幕显示模式

## v0.1.0-beta.3 (2021-11-09)

1. A 新增移动端键鼠映射功能
2. A 新增移动端码率设置功能
3. U 优化虚拟控件

## v0.1.0-beta.2 (2021-05-11)

1. A 新增黑屏检测

## v0.1.0-beta.1 (2021-04-17)

1. A 新增虚拟控件
2. A ~~新增获取连接的状态统计 (gatherStatistics)~~
3. A ~~新增获取延迟信息 (getRTT)~~

## v0.1.0-beta.0 (2021-02-19)

1. A PeerConnection/DataChannel/WebSocket 连接管理
2. A 为指定视频元素创建事件监听 (attachListener)
3. A 发送界面交互消息 (emitUIInteraction)
4. A 观测宿主元素尺寸变化调整播放器位置 (使用 ResizeObserver)
5. A 展示加载动画、遮罩信息等额外元素
6. A 控制视频元素的播放以及自动播放