aws

Web 客户端SDK开发者指南

Amazon DCV

Amazon DCV: Web 客户端SDK开发者指南

Table of Contents

什么是亚马逊 DCV Web 客户端SDK？

> **Note**
>
> 亚马逊以前DCV被称为 NICEDCV.

Amazon DCV 是一种高性能的远程显示协议。它允许您在不同的网络条件下，将远程桌面和应用程序流从任何云或数据中心安全地传送到任何设备。通过将 Amazon DCV 与 Amazon 配合使用EC2，您可以在亚马逊实例上远程运行图形密集型应用程序。EC2然后，您可以将结果流式传输到更适中的客户端计算机，从而消除对昂贵的专用工作站的需求。

Amazon DCV Web Client SDK 是一个 JavaScript 库，您可以使用它来开发自己的亚马逊DCV网络浏览器客户端应用程序。您的最终用户可以使用这些应用程序连接正在运行的 Amazon DCV 会话并与之交互。

使用 Amazon DCV Web Client SDK 作为构建块，您可以构建自定义 Web 应用程序，使用户可以随时随地即时访问其桌面或应用程序，其响应灵敏且流畅的性能与本机安装的应用程序几乎没有区别。

本指南介绍如何使用亚马逊 DCV Web Client SDK 来构建您的自定义 Web 浏览器客户端应用程序，以便在工作流程中与亚马逊DCV会话进行交互。

主题

先决条件

在开始使用亚马逊 DCV Web 客户端之前SDK，请确保您熟悉亚马逊DCV和亚马逊DCV会话。有关更多信息，请参阅《Amazon DCV 管理员指南》。

亚马逊DCV网络客户端SDK支持亚马逊DCV服务器版本 2020 及更高版本。

支持的特征

您可以构建支持以下 Amazon DCV 功能的自定义 Web 浏览器客户端应用程序：

- 连接到 Windows 亚马逊DCV服务器
- 连接 Linux 亚马逊DCV服务器
- 管理流式处理模式
- 传输文件
- 从会话打印
- 复制和粘贴
- 立体声 2.0 音频播放
- 立体声 2.0 音频录制（在 Windows 服务器上）
- 触摸屏
- 触控笔（在 Linux、Windows 10 和 Windows Server 2019 服务器上）
- 多显示器支持

有关这些功能的更多信息，请参阅 Amazon DCV 用户指南中的支持的功能。

浏览器支持

Amazon DCV Web 客户端SDK支持 JavaScript (ES6)，可以从 JavaScript 或TypeScript 应用程序中使用。

Amazon DCV 网络客户端SDK支持以下网络浏览器：

浏览器	版本
Google Chrome	最新的三个主要版本
Mozilla Firefox	最新的三个主要版本
Microsoft Edge	最新的三个主要版本
Apple Safari for macOS	最新的三个主要版本

版本控制约定

Amazon DCV Web 客户端SDK版本定义为以下格式：*major*.*minor*.*patch*。版本控制约定通常遵循语义版本控制模型。主要版本变化（例如从 1.x.x 变为 2.x.x）表示已引入重大更改，这些更改可能需要更改代码和执行计划的部署。次要版本变化（例如从 1.1.x 变为 1.2.x）是向后兼容的，但可能包含弃用的组件。

开始使用 Amazon DCV Web 客户端 SDK

Amazon DCV Web 客户端SDK由主dcv.js文件和一些辅助组件组成。所有文件都分发在一个压缩档案中，可以从 Amazon DCV 网站下载。

要开始使用亚马逊 DCV Web 客户端 SDK

1. Amazon DCV Web Client SDK 档案采用安全签名进行数字GPG签名。要验证档案的签名，必须导入NICEGPG密钥。为此，请打开终端窗口并导入NICEGPG密钥。

   ```
   $ wget https://d1uj6qtbmh3dt5.cloudfront.net/NICE-GPG-KEY
   ```

   ```
   $ gpg --import NICE-GPG-KEY
   ```

2. 从亚马逊DCV网站下载亚马逊网络客户端SDK档案和亚马逊DCV网络客户端SDK档案签名。DCV
3. 使用签名验证 Amazon DCV Web Client SDK 档案的签名。

   ```
   $ gpg --verify
   signature_filename.zip.sign
   archive_filename.zip
   ```

   例如：

   ```
   $ gpg --verify nice-dcv-web-client-sdk-1.8.4-840.zip.sign nice-dcv-web-client-
   sdk-1.8.4-840.zip
   ```

4. 如果签名验证成功，请提取 Amazon DCV Web Client SDK 档案的内容并将提取的目录放在您的 Web 服务器上。例如：

   ```
   $ unzip
   archive_filename.zip
   -d /
   path_to
   /
   server_directory
   /
   ```

> **Important**
>
> - 在您的网络服务器SDK上部署 Amazon DCV Web 客户端时，必须保留文件夹结构。
> - 使用 Amazon DCV Web UI 时SDK，请注意 `DCVViewer` React 组件期望此包中的 third-party-licenses .txt 和.txt 文件出现在嵌入式 Web URL 服务器的路径中。EULA应修改 third-party-licenses .txt 文件，使其还包含 Amazon DCV Web Client SDK 软件包中相应文件的内容，可能还包括使用用户应用程序使用的库中的任何其他许可信息。

连接到 Amazon DCV 服务器并获取第一帧

以下教程向您展示如何为自定义 Web 客户端准备HTML页面、如何进行身份验证和连接到 Amazon DCV 服务器，以及如何接收来自亚马逊DCV会话的第一帧流媒体内容。

主题

第 1 步：准备HTML页面

在您的网页中，您必须加载所需的 JavaScript 模块，并且必须在您希望亚马逊DCV网络客户端SDK从远程亚马逊DCV服务器提取内容流的`id`位置添加一个有效的`<div>`HTML元素。

例如：

```
<!DOCTYPE html>
<html lang="en" style="height: 100%;">
  <head>
    <title>DCV first connection</title>
  </head>
  <body style="height: 100%;">
    <div id="root" style="height: 100%;"></div>
    <div id="dcv-display"></div>
    <script type="module" src="index.js"></script>
  </body>
```

```
</html>
```

步骤 2：验证身份、连接并获取第一帧

本节介绍如何完成用户身份验证过程、如何连接亚马逊DCV服务器以及如何从亚马逊DCV服务器接收第一帧内容。

首先，从`index.js`文件中导入 Amazon DCV Web 客户端SDK。它可以作为通用模块定义 (UMD) 模块导入，如下所示：

```
import "./dcvjs/dcv.js"
```

否则，从版本开始`1.1.0`，也可以将其作为 ECMAScript Module (ESM) 从相应的包中导入，如下所示：

```
import dcv from "./dcvjs/dcv.js"
```

定义用于存储身份验证对象、连接对象和 Amazon DCV 服务器的变量URL。

```
let auth,
    connection,
    serverUrl;
```

加载脚本时，记录 Amazon DCV Web Client SDK 版本，在加载页面时，调用该`main`函数。

```
console.log("Using Amazon DCV Web Client SDK version " + dcv.version.versionStr);
document.addEventListener('DOMContentLoaded', main);
```

`main` 函数设置日志级别，并启动身份验证过程。

```
function main () {
  console.log("Setting log level to INFO");
  dcv.setLogLevel(dcv.LogLevel.INFO);

  serverUrl = "https://your-dcv-server-url:port/";

  console.log("Starting authentication with", serverUrl);

  auth = dcv.authenticate(
    serverUrl,
    {
```

```
      promptCredentials: onPromptCredentials,
      error: onError,
      success: onSuccess
    }
  );
}
```

promptCredentials、error 和 success 函数是在身份验证过程中必须定义的必需回调函数。

如果 Amazon DCV 服务器提示您输入证书，则promptCredentials回调函数会收到来自亚马逊DCV服务器的请求的凭证质询。如果将 Amazon DCV 服务器配置为使用系统身份验证，则必须提供登录凭证。以下代码示例假设用户名是 my_dcv_user，密码是 my_password。

如果身份验证失败，则error回调函数会收到来自 Amazon DCV 服务器的错误对象。

如果身份验证成功，success回调函数将收到一组情侣，其中包括允许my_dcv_user用户在 Amazon DCV 服务器上连接的每个会话的会话 ID (sessionIdauthToken) 和授权令牌 ()。以下代码示例调用 connect 函数，并连接到数组中返回的第一个会话。

> **Note**
>
> 在以下代码示例中，将 MY_DCV_USER 替换为您自己的用户名，并将 MY_PASSWORD 替换为您自己的密码。

```
function onPromptCredentials(auth, challenge) {
  // Let's check if in challege we have a username and password request
  if (challengeHasField(challenge, "username") && challengeHasField(challenge,
 "password")) {
    auth.sendCredentials({username: MY_DCV_USER, password: MY_PASSWORD})
  } else {
    // Challenge is requesting something else...
  }
}

function challengeHasField(challenge, field) {
  return challenge.requiredCredentials.some(credential => credential.name === field);
}

function onError(auth, error) {
  console.log("Error during the authentication: " + error.message);
}
```

```
// We connect to the first session returned
function onSuccess(auth, result) {
  let {sessionId, authToken} = {...result[0]};

  connect(sessionId, authToken);
}
```

连接到 Amazon DCV 服务器。当从 Amazon DCV 服务器收到第一帧时，将`firstFrame`调用回调方法。

```
function connect (sessionId, authToken) {
  console.log(sessionId, authToken);

  dcv.connect({
    url: serverUrl,
    sessionId: sessionId,
    authToken: authToken,
    divId: "dcv-display",
    callbacks: {
      firstFrame: () => console.log("First frame received")
    }
  }).then(function (conn) {
    console.log("Connection established!");
    connection= conn;
  }).catch(function (error) {
    console.log("Connection failed with error " + error.message);
  });
}
```

奖励：自动创建HTML登录表单

在调用 `promptCredentials` 回调函数时，将会返回 `challenge` 对象。它包括一个名为的属性`requiredCredentials`，该属性是一个对象数组，Amazon DCV 服务器请求的每个证书都有一个对象。每个对象包含请求的凭证的名称和类型。您可以使用`challenge`和`requiredCredentials`对象自动创建HTML登录表单。

以下代码示例说明了如何执行该操作。

```
let form,
    fieldSet;
```

```
function submitCredentials (e) {
  var credentials = {};
  fieldSet.childNodes.forEach(input => credentials[input.id] = input.value);
  auth.sendCredentials(credentials);
  e.preventDefault();
}

function createLoginForm () {
  var submitButton = document.createElement("button");

  submitButton.type = "submit";
  submitButton.textContent = "Login";

  form = document.createElement("form");
  fieldSet = document.createElement("fieldset");

  form.onsubmit = submitCredentials;
  form.appendChild(fieldSet);
  form.appendChild(submitButton);

  document.body.appendChild(form);
}

function addInput (name) {
  var type = name === "password" ? "password" : "text";

  var inputField = document.createElement("input");
  inputField.name = name;
  inputField.id = name;
  inputField.placeholder = name;
  inputField.type = type;
  fieldSet.appendChild(inputField);
}

function onPromptCredentials (_, credentialsChallenge) {
  createLoginForm();
  credentialsChallenge.requiredCredentials.forEach(challenge =>
 addInput(challenge.name));
}
```

使用 Amazon DCV 功能

Amazon DCV 功能的可用性取决于为亚马逊DCV会话配置的权限和客户端 Web 浏览器的功能。

Amazon DCV 会话中可用的功能由为该会话指定的权限管理。这意味着，即使 Amazon DCV Web Client SDK 支持某项功能，也可能会根据会话管理员定义的权限阻止对该功能的访问。有关更多信息，请参阅《亚马逊DCV管理员指南》中的配置亚马逊DCV授权。

了解 featuresUpdate 回调函数

当亚马逊DCV会话中某项功能的可用性发生变化时，Amazon DCV Web Client 会使用您在建立连接时指定的`featuresUpdate`回调函数SDK通知您。例如：

```
featuresUpdate: function (connection, list) {
  ...
},
```

该回调函数仅通知您可用性发生变化的功能。`list` 参数是一个字符串数组，它仅包含更新的功能的名称。例如，如果会话的音频输入功能可用性发生变化，则该参数仅包含 `["audio-in"]`。如果会话的剪贴板复制和粘贴功能的可用性以后发生变化，则该参数仅包含 `["clipboard-copy", "clipboard-paste"]`。

处理功能更新

`featuresUpdate` 回调函数仅通知您一个或多个功能的可用性发生变化。要了解更新了哪些功能，您必须使用 `connection.queryFeature` 方法查询该功能。在收到变化通知后，可以随时执行该操作。该方法返回 `Promise`，它解析为请求的功能的更新状态。`status` 值始终是关联的，并且它具有一个名为 `enabled` 的布尔值（`true | false`）属性。某些功能可能在 `status` 值中具有其他属性。如果尚未更新该功能的可用性，则会拒绝该功能。

以下示例代码说明了如何执行该操作。

```
// Connection callback called
function featuresUpdate (_, list) {
  if (list.length > 0) {
    list.forEach((feat) => {
      connection.queryFeature(feat).then(status => console.log(feat, "is",
 status.enabled)));
    });
  }
}
```

使用亚马逊 DCV Web 用户界面 SDK

以下教程向您展示了如何针对亚马逊DCV服务器进行身份验证、连接该服务器以及如何通过亚马逊 DCV Web 用户界面呈现 `DCVViewer` React 组件SDK。

主题

先决条件

您需要安装 `React`、`ReactDOM`、`Cloudscape Design Components React`、`Cloudscape Design Global Styles` 和 `Cloudscape Design Design Tokens`。

```
$ npm i react react-dom @cloudscape-design/components @cloudscape-design/global-styles
 @cloudscape-design/design-tokens
```

您还需要下载 `Amazon DCV Web Client SDK`。请参阅开始使用 Amazon DCV Web 客户端 SDK阅读有关如何执行此操作的 step-by-step指南。

您必须为导入模块创建别名，因为该`dcv`模块是 Amazon DCV Web UI 的外部依赖项SDK。例如，如果使用 webpack 捆绑 Web 应用程序，您可以使用 resolve.alias 选项，如下所示：

```
const path = require('path');

module.exports = {
  //...
  resolve: {
    alias: {
      dcv: path.resolve('path', 'to', 'dcv.js'),
    },
  },
};
```

如果您使用 rollup 进行捆绑，您可以安装并使用 @rollup/plugin-alias，如下所示：

```
import alias from '@rollup/plugin-alias';
```

```
const path = require('path');

module.exports = {
  //...
  plugins: [
    alias({
      entries: [
        { find: 'dcv', replacement: path.resolve('path', 'to', 'dcv.js') },
      ]
    })
  ]
};
```

第 1 步：准备HTML页面

在你的网页中，你必须加载所需的 JavaScript 模块，并且你应该有一个有效的 HTML元素，你的应用程序的入口组件将在id哪里呈现。

例如：

```
<!DOCTYPE html>
<html lang="en" style="height: 100%;">
  <head>
    <title>DCV first connection</title>
  </head>
  <body style="height: 100%;">
    <div id="root" style="height: 100%;"></div>
    <script type="module" src="index.js"></script>
  </body>
</html>
```

步骤 2：验证身份、连接并渲染 **DCVViewer** React 组件。

本节介绍如何完成用户身份验证过程、如何连接 Amazon DCV 服务器以及如何呈现 DCVViewer React 组件。

首先，从 index.js 文件中导入 React、ReactDOM 和顶级 App 组件。

```
import React from "react";
import ReactDOM from 'react-dom';
import App from './App';
```

渲染应用程序的顶级容器节点。

```
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById("root")
);
```

在App.js文件中，将亚马逊 DCV Web 客户端SDK作为ESM模块导入，从亚马逊 DCV Web UI SDK 导入 DCVViewer React 组件React以及Cloudscape Design Global Styles软件包。

```
import React from "react";
import dcv from "dcv";
import "@cloudscape-design/global-styles/index.css";
import {DCVViewer} from "./dcv-ui/dcv-ui.js";
```

以下示例演示了如何在身份验证成功的情况下对亚马逊DCV服务器进行身份验证并通过亚马逊 DCV Web UI SDK 呈现 DCVViewer React 组件。

```
const LOG_LEVEL = dcv.LogLevel.INFO;
const SERVER_URL = "https://your-dcv-server-url:port/";
const BASE_URL = "/static/js/dcvjs";

let auth;

function App() {
  const [authenticated, setAuthenticated] = React.useState(false);
  const [sessionId, setSessionId] = React.useState('');
  const [authToken, setAuthToken] = React.useState('');
  const [credentials, setCredentials] = React.useState({});

  const onSuccess = (_, result) => {
    var { sessionId, authToken } = { ...result[0] };

    console.log("Authentication successful.");

    setSessionId(sessionId);
    setAuthToken(authToken);
    setAuthenticated(true);
    setCredentials({});
  }
```

```
  const onPromptCredentials = (_, credentialsChallenge) => {
    let requestedCredentials = {};

    credentialsChallenge.requiredCredentials.forEach(challenge =>
requestedCredentials[challenge.name] = "");
    setCredentials(requestedCredentials);
  }

  const authenticate = () => {
    dcv.setLogLevel(LOG_LEVEL);

    auth = dcv.authenticate(
      SERVER_URL,
      {
        promptCredentials: onPromptCredentials,
        error: onError,
        success: onSuccess
      }
    );
  }

  const updateCredentials = (e) => {
    const { name, value } = e.target;
    setCredentials({
      ...credentials,
      [name]: value
    });
  }

  const submitCredentials = (e) => {
    auth.sendCredentials(credentials);
    e.preventDefault();
  }

  React.useEffect(() => {
    if (!authenticated) {
      authenticate();
    }
  }, [authenticated]);

  const handleDisconnect = (reason) => {
    console.log("Disconnected: " + reason.message + " (code: " + reason.code + ")");
    auth.retry();
```

```
    setAuthenticated(false);
  }

  return (
    authenticated ?
    <DCVViewer
      dcv={{
        sessionId: sessionId,
        authToken: authToken,
        serverUrl: SERVER_URL,
        baseUrl: BASE_URL,
        onDisconnect: handleDisconnect,
        logLevel: LOG_LEVEL
      }}
      uiConfig={{
        toolbar: {
          visible: true,
          fullscreenButton: true,
          multimonitorButton: true,
        },
      }}
    />
    :
    <div
      style={{
        height: window.innerHeight,
        backgroundColor: "#373737",
        display: 'flex',
        alignItems: 'center',
        justifyContent: 'center',
      }}
    >
      <form>
        <fieldset>
          {Object.keys(credentials).map((cred) => (
            <input
              key={cred}
              name={cred}
              placeholder={cred}
              type={cred === "password" ? "password" : "text"}
              onChange={updateCredentials}
              value={credentials[cred]}
            />
          ))}
```

```
        </fieldset>
        <button
          type="submit"
          onClick={submitCredentials}
        >
          Login
        </button>
      </form>
    </div>
  );
}

const onError = (_, error) => {
  console.log("Error during the authentication: " + error.message);
}

export default App;
```

promptCredentials、error 和 success 函数是在身份验证过程中必须定义的必需回调函数。

如果 Amazon DCV 服务器提示您输入证书，则promptCredentials回调函数会收到来自亚马逊DCV服务器的请求的凭证质询。如果 Amazon DCV 服务器配置为使用系统身份验证，则必须以用户名和密码的形式提供凭证。

如果身份验证失败，则error回调函数会收到来自 Amazon DCV 服务器的错误对象。

如果身份验证成功，success回调函数将收到一组情侣，其中包括允许用户在 Amazon DCV 服务器上连接的每个会话的会话 ID (sessionIdauthToken) 和授权令牌 ()。上面的代码示例更新 React 状态，以在成功进行身份验证时渲染 DCVViewer 组件。

要了解有关此组件接受的属性的更多信息，请参阅 Amazon DCV Web UI SDK 参考。

要了解自签名证书的更多信息，请参阅自签名证书的重定向说明。

从 AWS-UI 更新到 Cloudscape 设计系统

从SDK版本1.3.0开始，我们将DCVViewer组件从 AWS-UI更新为其演变：Cloud scape Design。

Cloudscape 使用的视觉主题与 AWS-UI 不同，但底层代码库保持不变。因此，根据 DCVViewer 迁移您的应用程序应该很容易。要迁移，请将已安装的与 AWS-UI 相关的NPM软件包替换为关联的 Cloudscape 软件包：

AWS-UI 软件包名称	Cloudscape 软件包名称
@awsui/components-react	@cloudscape-design/components
@awsui/global-styles	@cloudscape-design/global-styles
@awsui/collection-hooks	@cloudscape-design/collection-hooks
@awsui/design-tokens	@cloudscape-design/design-tokens

有关迁移的更多详细信息，请参阅 AWS-UI GitHub 文档页面。

SDK参考

本节提供了 Amazon DCV Web 客户端的描述、语法和用法示例SDK。

主题

DCV模块

实现DCV协议客户端的模块。

公开

方法

列出

authenticate(url, callbacks) → {Authentication}

启动指定的 Amazon DCV 服务器终端节点的身份验证过程。

参数：

名称	Type	描述
`url`	字符串	正在运行的 Amazon DCV 服务器的主机名和端口，格式如下：`https://dcv_host_address:port` 。例如：`https://my-dcv-server:8443` 。
`callbacks`	authenticationCallbacks	在身份验证过程中可调用的回调。

返回值:

- Authentication 对象。

类型

身份验证

连接（配置）→ {承诺。 < 连接 >|承诺。 < {code: ConnectionErrorCode，消息：字符串} >}

连接到指定的 Amazon DCV 服务器终端节点。如果连接成功，则返回一个 Connection 对象。如果连接失败，则返回一个错误对象。

参数：

名称	Type	描述
`config`	ConnectionConfig	物 ConnectionConfig 体。

返回值:

- Connection 对象或错误对象。

类型

承诺。 < 连接 > | 承诺。 < {code: ConnectionErrorCode，消息：字符串} >

setLogHandler（处理程序）→ {void}

设置自定义日志处理函数。如果覆盖默认日志处理程序，在使用浏览器控制台调试时，原始日志条目位置将丢失。

参数：

名称	Type	描述
handler	函数	自定义日志处理函数。处理函数包含级别（数字）、levelName（字符串）、域（字符串）和消息（字符串）。

返回值:

类型

void

setLogLevel（等级）→ {无效}

设置日志级别。只有在使用默认日志处理程序时，才需要使用该方法。

参数：

名称	Type	描述
level	LogLevel	要使用的日志级别。

返回值:

类型

void

成员

列出

- （常量）AudioError：AudioErrorCode
- （常量）AuthenticationError：AuthenticationErrorCode
- （常量）ChannelError：ChannelErrorCode
- （常量）ClosingReasonError：ClosingReasonErrorCode
- （常量）ConnectionError：ConnectionErrorCode
- （常量）CustomChannelError：CustomChannelErrorCode
- （常量）DisplayConfigError：DisplayConfigErrorCode
- （常量）FileStorageError：FileStorageErrorCode
- （常量）LogLevel：LogLevel
- （常量）MultiMonitorError：MultiMonitorErrorCode
- （常量）ResolutionError：ResolutionErrorCode
- （常量）TimezoneRedirectionError：TimezoneRedirectionErrorCode
- （常量）TimezoneRedirectionSetting：TimezoneRedirectionSettingCode
- （常量）TimezoneRedirectionStatus：TimezoneRedirectionStatusCode
- (constant) version
- （常量）ScreenshotError：ScreenshotErrorCode
- （常量）WebcamError：WebcamErrorCode

（常量）AudioError：AudioErrorCode

AudioError 代码枚举。

类型：

- AudioErrorCode

（常量）AuthenticationError：AuthenticationErrorCode

AuthenticationError 代码枚举。

类型：

- AuthenticationErrorCode

（常量）ChannelError：ChannelErrorCode

ChannelError 代码枚举。

类型：

- ChannelErrorCode

（常量）ClosingReasonError：ClosingReasonErrorCode

ClosingReasonError 代码枚举。

类型：

- ClosingReasonErrorCode

（常量）ConnectionError：ConnectionErrorCode

ConnectionError 代码枚举。

类型：

- ConnectionErrorCode

（常量）CustomChannelError：CustomChannelErrorCode

CustomChannelError 代码枚举。

类型：

- CustomChannelErrorCode

（常量）DisplayConfigError：DisplayConfigErrorCode

DisplayConfigError 代码枚举。

类型：

- DisplayConfigErrorCode

（常量）FileStorageError：FileStorageErrorCode

FileStorageError 代码枚举。

类型：

- FileStorageErrorCode

（常量）LogLevel：LogLevel

可用的SDK日志级别。

类型：

- LogLevel

（常量）MultiMonitorError：MultiMonitorErrorCode

MultiMonitorError 代码枚举。

类型：

- MultiMonitorErrorCode

（常量）ResolutionError：ResolutionErrorCode

ResolutionError 代码枚举。

类型：

- ResolutionErrorCode

（常量）TimezoneRedirectionError：TimezoneRedirectionErrorCode

TimezoneRedirectionError 代码枚举。

类型：

- TimezoneRedirectionErrorCode

（常量）TimezoneRedirectionSetting：TimezoneRedirectionSettingCode

TimezoneRedirectionSetting 代码枚举。

类型：

- TimezoneRedirectionSettingCode

（常量）TimezoneRedirectionStatus：TimezoneRedirectionStatusCode

TimezoneRedirectionStatus 代码枚举。

类型：

- TimezoneRedirectionStatusCode

(constant) version

Amazon DCV 版本包含主要版本、次要版本、补丁版、修订版、扩展版和versionStr。

属性：

名称	Type	描述
major	整数	主要版本号。
minor	整数	次要版本号。
patch	整数	补丁版本号。
revision	整数	修订号。

名称	Type	描述
extended	字符串	扩展的字符串。
versionStr	字符串	串联的主要版本号、次要版本号、补丁号和修订号，形式为 major.minor.patch+build.revision 。

（常量）ScreenshotError：ScreenshotErrorCode

ScreenshotError 代码枚举。

类型：

- ScreenshotErrorCode

（常量）WebcamError：WebcamErrorCode

WebcamError 代码枚举。

类型：

- WebcamErrorCode

类型和回调定义

列出

- AudioErrorCode
- authenticationCallbacks
- AuthenticationErrorCode
- authErrorCallback（身份验证，错误）
- authPromptCredentials回调（身份验证、质询）
- authSuccessCallback（身份验证，authenticationData）
- 频道
- ChannelErrorCode

- clipboardEventCallback（事件）
- ClosingReasonErrorCode
- Colorspace
- connectionCallbacks
- ConnectionConfig
- ConnectionErrorCode
- createDirectory（路径）
- CustomChannelErrorCode
- dataChannelCallback（信息）
- deleteFile（路径）
- deviceChangeEvent回调 ()
- disconnectCallback（原因）
- displayAvailabilityCallback（状态，displayId）
- DisplayConfigErrorCode
- displayLayoutCallback(serverWidth,serverHeight, 头)
- feature
- featuresUpdateCallback(featuresList)
- fileDownloadCallback(fileResource)
- filePrintedCallback(printResource)
- filestorage
- filestorageEnabledCallback（已启用）
- FileStorageErrorCode
- firstFrameCallback(resizeEnabled， relativeMouseMode已启用，displayId)
- idleWarningNotification回调 (disconnectionDateTime)
- collaboratorListCallback（合作者）
- licenseNotificationCallback（通知）
- list(path)
- LogLevel
- 监控

AudioErrorCode

模块中可用的 AudioError 代码枚举 DCV

- `SETTING_AUDIO_FAILED`
- `CHANNEL_NOT_AVAILABLE`

类型：

- number

authenticationCallbacks

身份验证回调

类型：

- 对象

属性：

名称	Type	描述
promptCredentials	authPromptCredentials回调	在询问用户凭证时调用的回调函数。
error	authErrorCallback	在身份验证失败时调用的回调函数。
success	authSuccessCallback	在身份验证成功时调用的回调函数。

AuthenticationErrorCode

模块中可用的 AuthenticationError 代码枚举 DCV

- INVALID_MESSAGE
- UNKNOWN_AUTH_MODE
- SESSION_NOT_AVAILABLE
- NO_SESSIONS
- WRONG_CREDENTIALS
- SASL_CHALLENGE
- SASL_AUTH_MECHANISM
- FAILED_COMMUNICATION
- AUTHENTICATION_REJECTED
- GENERIC_ERROR
- WRONG_CREDENTIALS_FORMAT
- WRONG_CREDENTIALS_TYPE
- UNREQUESTED_CREDENTIALS
- MISSING_CREDENTIAL

类型：

- number

authErrorCallback（身份验证，错误）

在身份验证失败时调用的回调函数。

参数：

名称	Type	描述
`authentication`	身份验证	Authentication 对象。
`error`	对象	身份验证过程引发的错误对象。 名称 / Type / 描述 `code` / AuthenticationErrorCode / 错误代码。 `message` / 字符串 / 错误消息。

authPromptCredentials回调（身份验证、质询）

在询问用户凭证时调用的回调函数。用户必须提供请求的凭证以回答质询。

参数：

名称	Type	描述
`authentication`	身份验证	Authentication 对象。
`challenge`	对象	质询。

名称	Type	描述
		名称 Type 描述 requiredCredentials Array.<Object> 请求的凭证对象的数组。 名称 Type 描述 name 字符串 请求的凭证的名称。 type 字符串 请求的凭证的类型。

authSuccessCallback（身份验证，authenticationData）

在身份验证成功时调用的回调函数。

参数：

<table>
<tr><th>名称</th><th>Type</th><th>描述</th></tr>
<tr><td>authentication</td><td>身份验证</td><td>Authentication 对象。</td></tr>
<tr><td>authenticationData</td><td>Array.<Object></td><td>包含 Amazon DCV 会话IDs和身份验证令牌的对象数组。<table><tr><th>名称</th><th>Type</th><th>描述</th></tr><tr><td>sessior</td><td>字符串</td><td>亚马逊会DCV话 ID。</td></tr><tr><td>authTok</td><td>字符串</td><td>Amazon DCV 会话的身份验证令牌。</td></tr></table></td></tr>
</table>

频道

可以指定的可用通道。

类型：

- "clipboard" | "display" | "input" | "audio" | "filestorage"

ChannelErrorCode

模块中可用的 ChannelError 代码枚举 DCV

- ALREADY_OPEN
- INITIALIZATION_FAILED
- REJECTED

类型：

- number

clipboardEventCallback（事件）

在生成 `clipboardEvent` 时调用的回调函数。

参数：

名称	Type	描述
`event`	对象	有关剪贴板事件的信息。 名称 | Type | Attributes | 描述 `name` | 已建立 | 复制 | 粘贴 dataSilert autoCone | newDailable | | autoPaDone | remotor | pasteAbleD | 始终存在。事件名称。

名称	Type	描述

名称	Type	Attribu s	描述
clipb Data	Object | string		剪贴板中的数据。
autoC	布尔值	<可选>	指示是否启用从会话剪贴板到本地客户端剪贴板的自动复制。
maxDa ze	numbe	<可选>	可以在剪贴板中放置的最大数据量。
error	字符串	<可选>	错误信息（如果适

名称	Type	描述
		名称 Type Attributes 描述 用 ）。

ClosingReasonErrorCode

模块中可用的 ClosingReasonError 代码枚举 DCV

- TRANSPORT_ERROR
- NO_ERROR
- GENERIC_ERROR
- INTERNAL_SERVER_ERROR
- PROTOCOL_ERROR
- AUTHORIZATION_DENIED
- AUTHORIZATION_REVOKED
- ACCESS_REJECTED
- IDLE_TIMEOUT_EXPIRED
- DISCONNECT_BY_OWNER
- DISCONNECT_BY_USER
- EVICTED
- EXTERNAL_PROTOCOL_CONNECTION_EVICTED
- DISCONNECTION_REQUESTED

类型：

- number

Colorspace

可以指定的可用色彩空间。

类型：

- “RGB" | "YUV_ REC6 01" |" YUV _ REC7 09"

connectionCallbacks

在出现连接错误时可调用的回调。

类型：

- 对象

属性：

名称	Type	描述
`disconnect`	disconnectCallback	在连接结束时调用的回调函数。
`displayLayout`	displayLayoutCallback	在更改显示布局或分辨率时调用的回调函数。
`displayAvailability`	displayAvailabilityCallback	在显示器的可用性发生变化时调用的回调函数。
`firstFrame`	firstFrameCallback	从 Amazon DCV 服务器收到第一帧时要调用的回调函数。
`filePrinted`	filePrintedCallback	在 Amazon DCV 服务器上打印文件时要调用的回调函数。
`fileDownload`	fileDownloadCallback	当文件准备好从 Amazon DCV 服务器下载时要调用的回调函数。
`dataChannel`	dataChannelCallback	当 Amazon DCV 服务器发送有关数据通道可用性的通知时要调用的回调函数。

名称	Type	描述
licenseNotification	licenseNotificationCallback	当 Amazon DCV 服务器发送有关许可证状态的通知时要调用的回调函数。
idleWarningNotification	idleWarningNotification回调	当 Amazon DCV 服务器发送空闲超时警告时要调用的回调函数。
collaboratorList	collaboratorListCallback	亚马逊DCV服务器发送合作者列表时要调用的回调函数（自亚马逊DCV网络客户端 1.1.0 SDK 版起）。
qualityIndicatorState	qualityIndicatorState回调	在连接质量指标改变状态时调用的回调函数。
filestorageEnabled	filestorageEnabledCallback	在启用或禁用文件存储时调用的回调函数。
featuresUpdate	featuresUpdateCallback	在功能状态发生变化时调用的回调函数。
clipboardEvent	clipboardEventCallback	在生成 clipboardEvent 时调用的回调函数。
deviceChangeEvent	deviceChangeEvent回调	在触发 deviceChange 事件时调用的回调函数。
screenshot	screenshotCallback	在 screenshot 可用时调用的回调函数。

ConnectionConfig

Amazon DCV 连接的配置。

类型：

- 对象

属性：

名称	Type	描述
`url`	字符串	正在运行的 Amazon DCV 服务器的主机名和端口，格式如下：`https://dcv_host_address:port` 。例如：`https://my-dcv-server:8443` 。
`sessionId`	字符串	亚马逊会DCV话 ID。
`authToken`	字符串	在连接到服务器时使用的身份验证令牌。
`baseUrl`	字符串	URL从中加载SDK文件的绝对值或相对值。
`resourceBaseUrl`	字符串	访问DCV资源的绝对值或相对URL值。
`enabledChannels`	Array.<Channel>	指示可以启用的通道列表。如果未指定或提供空数组，它默认为所有可用的通道。
`losslessColorspace`	Colorspace	指示将使用的色彩空间。如果未指定，则默认为 “RGB”。
`divId`	字符串	wher div e 中对象HTMLDOM 的 ID SDK 应使用远程流创建画布。
`volumeLevel`	整数	首选的音量。有效范围是 0 到 100。

名称	Type	描述
clipboardAutoSync	布尔值	表示兼容的 Web 浏览器是否启用了从 Amazon DCV 会话剪贴板自动复制到本地客户端剪贴板的功能。
dynamicAudioTuning	布尔值	表示在建立连接时是否根据 Amazon DCV 服务器的音频设置动态调整音频。
clientHiDpiScaling	布尔值	指示是否根据客户端缩放画布 DPI。
highColorAccuracy	布尔值	指示是否应使用高色彩精度（如果可用）。如果未指定，它默认为 false。
enableWebCodecs	布尔值	表示是否 WebCodecs 应使用（如果有）。如果未指定，则默认为 false。
observers	connectionCallbacks	用于调用与连接相关的事件的回调函数。
callbacks	connectionCallbacks	与 observers 属性相同，但每个回调都包含 Connection 对象以作为第一个参数。

ConnectionErrorCode

模块中可用的 ConnectionError 代码枚举 DCV

- ALREADY_OPEN
- INVALID_CONFIG
- INITIALIZATION_FAILED
- REJECTED
- MAIN_CHANNEL_ALREADY_OPEN

- GENERIC_ERROR（自DCV服务器 2021.0 起）
- INTERNAL_SERVER_ERROR（自DCV服务器 2021.0 起）
- AUTHENTICATION_FAILED（自DCV服务器 2021.0 起）
- PROTOCOL_ERROR（自DCV服务器 2021.0 起）
- INVALID_SESSION_ID（自DCV服务器 2021.0 起）
- INVALID_CONNECTION_ID（自DCV服务器 2021.0 起）
- CONNECTION_LIMIT_REACHED（自DCV服务器 2021.0 起）
- SERVER_UNREACHABLE（自DCV服务器 2022.1 起）
- GATEWAY_BUSY
- UNSUPPORTED_CREDENTIAL（自DCV服务器 2022.2 起）
- TRANSPORT_ERROR

类型：

- number

createDirectory（路径）

参数：

名称	Type	描述
path	字符串	我们要在其中创建目录的服务器上的绝对路径。它还应包括目标目录的名称。

CustomChannelErrorCode

模块中可用的 CustomChannelError 代码枚举 DCV

- TRANSPORT_ERROR

类型：

- number

dataChannelCallback（信息）

当 Amazon DCV 服务器发送有关数据通道可用性的通知时要调用的回调函数。

参数：

名称	Type	描述
`info`	对象	有关数据通道的信息。 名称 / Type / 描述 `name` / 字符串 / 数据通道的名称。 `token` / 字符串 / 数据通道的身份验证令牌。

deleteFile（路径）

参数：

名称	Type	描述
`path`	字符串	服务器上的绝对路径，指定我们要删除的文件。

deviceChangeEvent回调 ()

在触发 `deviceChange` 事件时调用的回调函数。

disconnectCallback（原因）

在连接结束时调用的回调函数。

参数：

名称	Type	描述
reason	对象	断开连接原因。 名称 / Type / 描述 code / number / 原因代码。 message / 字符串 / 原因消息。

displayAvailabilityCallback（状态，displayId）

在显示器的可用性发生变化时调用的回调函数。

参数：

名称	Type	描述
status	对象	显示器的状态。 名称 / Type / 描述 enabled / 布尔值 / 指示是否启用显示器。 closed / 布尔值 / 指示显示器是否关闭。
displayId	number	显示器的标识符。

DisplayConfigErrorCode

模块中可用的 DisplayConfigError 代码枚举 DCV

- `INVALID_ARGUMENT`
- `UNSUPPORTED_OPERATION`
- `NO_CHANNEL`

类型：

- number

displayLayoutCallback(serverWidth,serverHeight, 头)

在更改显示布局或分辨率时调用的回调函数。

参数：

名称	Type	描述
`serverWidth`	number	主显示器的宽度（以像素为单位）。
`serverHeight`	number	主显示器的高度（以像素为单位）。
`heads`	Array.<Monitor>	Amazon DCV 服务器支持的显示头。

feature

功能值。

- `display` - 指示单显示器视频流的可用性。
- `display-multi` - 指示多显示器视频流的可用性。
- `high-color-accuracy`-表示色彩精度高（自 Amazon DCV Web Client SDK 版本 1.1.0 起）。
- `mouse` - 指示鼠标功能的可用性。
- `keyboard` - 指示键盘功能的可用性。

- `keyboard-sas`-表示SAS序列（控制 + Alt + 删除）功能的可用性。
- `relative-mouse` - 指示相对鼠标模式的可用性。
- `clipboard-copy`-表示从 Amazon DCV 服务器到客户端的剪贴板复制功能是否可用。
- `clipboard-paste`-表示从客户端到 Amazon DCV 服务器的剪贴板粘贴功能是否可用。
- `audio-in` - 指示使用麦克风的音频输入功能的可用性。
- `audio-out` - 指示音频播放功能的可用性。
- `webcam` - 指示网络摄像头流功能的可用性。
- `file-download`-表示从 Amazon DCV 服务器向客户端下载文件功能的可用性。
- `file-upload`-表示可以将文件从客户端上传到 Amazon DCV 服务器。
- `timezone-redirection`-表示时区重定向功能的可用性（自 Amazon DCV Web Client SDK 版本 1.3.0 起）。

类型：

- 字符串

featuresUpdateCallback(featuresList)

在功能状态发生变化时调用的回调函数。

参数：

名称	Type	描述
`featuresList`	Array.<feature>	一系列已更改的功能。

fileDownloadCallback(fileResource)

当文件准备好从 Amazon DCV 服务器下载时要调用的回调函数。

参数：

名称	Type	描述
`fileResource`	对象	有关准备好下载的文件的信息。

名称	Type	描述
		名称 / Type / 描述 id / 字符串 / 文件的标识符。 url / 字符串 / URL用于下载文件。 domain / 字符串 / 资源域。 token / 字符串 / 用于下载文件的身份验证令牌。令牌也包含在URL。

filePrintedCallback(printResource)

在 Amazon DCV 服务器上打印文件时要调用的回调函数。

参数：

名称	Type	描述
printResource	对象	有关打印的文件的信息。

名称	Type	描述
		名称 / Type / 描述 id / 字符串 / 打印的文件的标识符。 url / 字符串 / URL用于下载打印的文件。 domain / 字符串 / 资源域。此处为 printer。 token / 字符串 / 用于下载打印的文件的身份验证令牌。令牌也包含在URL。

filestorage

允许在文件系统上浏览和执行操作的对象。

类型：

- 对象

属性：

名称	Type	描述
`list`	list	该函数允许列出服务器上的提供路径中存在的项目（文件和目录）。
`createDirectory`	createDirectory	该函数允许在服务器上的指定路径中创建目录。
`retrieveFile`	retrieveFile	该函数允许将文件下载到服务器上的指定路径本地。
`deleteFile`	deleteFile	该函数允许删除服务器上的指定路径中的文件。
`renameFile`	renameFile	该函数允许将文件从指定源路径重命名为指定目标路径。
`renameDirectory`	renameDirectory	该函数允许将目录从指定源路径重命名为绝对目标路径。
`storeFile`	storeFile	该函数允许将本地文件上传到服务器上的提供路径。

filestorageEnabledCallback（已启用）

在启用文件存储时调用的回调函数。仅 Internet Explorer 11 上的延迟通道。

参数：

名称	Type	描述
`enabled`	布尔值	指示是否启用了文件存储。

FileStorageErrorCode

模块中可用的 FileStorageError 代码枚举 DCV

- `CANCELLED`
- `ABORTED`
- `INVALID_ARGUMENT`
- `NOT_IMPLEMENTED`
- `ERROR`
- `ALREADY_EXIST`
- `NOT_FOUND`

类型：

- number

firstFrameCallback(resizeEnabled， relativeMouseMode已启用，displayId)

从 Amazon DCV 服务器收到第一帧时要调用的回调函数。为每个显示器发出。

参数：

名称	Type	描述
`resizeEnabled`	布尔值	指示服务器是否支持调整客户端显示布局的大小。
`relativeMouseModeEnabled`	布尔值	指示服务器是否支持相对鼠标模式。
`displayId`	number	显示器的标识符。

idleWarningNotification回调 (disconnectionDateTime)

当 Amazon DCV 服务器发送空闲超时警告时要调用的回调函数。

参数：

名称	Type	描述
`disconnectionDateTime`	日期	断开连接日期和时间。

collaboratorListCallback（合作者）

Amazon DCV 服务器发送合作者列表时要调用的回调函数。

参数：

名称	Type	描述
`collaborators`	Array.<Object>	包含有关协作者的信息的对象列表。 名称 / Type / 描述 `usernam` / 字符串 / 协作者的用户名。 `owner` / 布尔值 / 指示协作者是否为会话所有者。 `connectnId` / number / 指示服务器为连接分配的 ID。

licenseNotificationCallback（通知）

当 Amazon DCV 服务器发送有关许可证状态的通知时要调用的回调函数。

参数：

名称	Type	描述
`notification`	对象	通知。 名称 / Type / 描述 `product` / 字符串 / 该DCV产品。 `status` / 字符串 / 许可证的状态。 `message` / 字符串 / 消息。 `leftDay` / number / 许可证过期前的天数。 `isDemo` / 布尔值 / 指示许可证是否为演示许可证。 `numUnlicensed` / number / 未许可的连接数。 `licensiMode` / 字符串 / 许可模式。

名称	Type	描述
		名称 Type 描述 documentationUrl 字符串 URL 用于文档。

list(path)

参数：

名称	Type	描述
path	字符串	我们要列出内容的服务器上的绝对路径。

LogLevel

可用的SDK日志级别。

类型：

- TRACE | DEBUG | INFO | WARN | ERROR | SILENT

监控

类型：

- 对象

属性：

名称	Type	描述
name	字符串	显示头的名称。

<table>
<tr><th>名称</th><th>Type</th><th>描述</th></tr>
<tr><td><code>rect</code></td><td>对象</td><td>有关显示头的信息。<table><tr><th>名称</th><th>Type</th><th>描述</th></tr><tr><td>x</td><td>number</td><td>显示头的初始 x 坐标。</td></tr><tr><td>y</td><td>number</td><td>显示头的初始 y 坐标。</td></tr><tr><td><code>width</code></td><td>number</td><td>显示头的宽度（以像素为单位）。</td></tr><tr><td><code>height</code></td><td>number</td><td>显示头的高度（以像素为单位）。</td></tr></table></td></tr>
<tr><td><code>primary</code></td><td>布尔值</td><td>指示显示头是否为主显示头。这是从远程操作系统（如果可用）中确定的。</td></tr>
<tr><td><code>dpi</code></td><td>number</td><td>显示屏头的。DPI</td></tr>
</table>

MultiMonitorErrorCode

模块中可用的 MultiMonitorError 代码枚举 DCV

- NO_DISPLAY_CHANNEL
- MAX_DISPLAY_NUMBER_REACHED
- INVALID_ARGUMENT
- DISPLAY_NOT_OPENED_BY_SERVER
- REQUEST_TIMEOUT
- GENERIC_ERROR
- NO_ERROR

类型：

- number

qualityIndicatorState回调（状态）

在连接质量指标改变状态时调用的回调函数。

参数：

名称	Type	描述
state	Array.<Object>	有关连接质量的信息。 名称 / Type / 描述 name / 字符串 / 指标的名称。 status / NORMAL | WARNIN | CRITICA / 状态描述。 changec / 布尔值 / 指示状态是否发生变化。

renameDirectory（src，dest）

参数：

名称	Type	描述
`src`	字符串	服务器上的绝对源路径，指定我们要重命名的目录。
`dest`	字符串	服务器上的绝对目标路径，指定目标路径和目录名。

renameFile（src，dest）

参数：

名称	Type	描述
`src`	字符串	服务器上的绝对源路径，指定我们要重命名的文件。
`dest`	字符串	服务器上的绝对目标路径，指定目标路径和文件名。

ResolutionErrorCode

模块中可用的 ResolutionError 代码枚举 DCV

- `INVALID_ARGUMENT`
- `NO_CHANNEL`
- `NOT_IMPLEMENTED`

类型：

- number

retrieveFile（路径）

参数：

名称	Type	描述
`path`	字符串	服务器上的绝对路径，指定我们要下载到本地的文件。

screenshotCallback（屏幕截图）

在屏幕截图可用时调用的回调函数。

参数：

名称	Type	描述
`screenshot`	byte[]	PNG格式的屏幕截图缓冲区，`null`或者屏幕截图检索失败的情况。

ScreenshotErrorCode

模块中可用的 ScreenshotError 代码枚举 DCV

- `NO_CHANNEL`
- `GENERIC_ERROR`

类型：

- number

serverInfo

类型：

- 对象

属性：

名称	Type	描述
name	字符串	软件的名称。
version	对象	软件版本号。 名称 / Type / 描述 major / number / 主要版本号。 minor / number / 次要版本号。 revisic / number / 修订版本号。
os	字符串	操作系统。
arch	字符串	架构。
hostname	字符串	主机名。

stats

类型：

- 对象

属性：

名称	Type	描述
fps	number	当前的每秒帧数。

名称	Type	描述
`traffic`	number	当前的流量（以位/秒为单位）。
`peakTraffic`	number	自建立连接以来的流量峰值（以位/秒为单位）。
`latency`	number	当前的延迟（以毫秒为单位）。
`currentChannels`	number	自建立连接以来打开的通道数。
`openedChannels`	number	当前打开的通道数。
`channelErrors`	number	报告错误的通道数。

storeFile（文件，目录）

参数：

名称	Type	描述
`file`	文件	我们要上传到服务器的文件对象（有关更多信息，请参阅 https://developer.mozilla.org/en-US/docs/Web/ API / File）。
`dir`	字符串	我们要将文件上传到的服务器上的绝对路径。

TimezoneRedirectionErrorCode

模块中可用的 TimezoneRedirectionError 代码枚举 DCV

- `INVALID_ARGUMENT`

- NO_CHANNEL
- USER_CANNOT_CHANGE

类型：

- number

TimezoneRedirectionSettingCode

模块中可用的 TimezoneRedirectionSetting 代码枚举 DCV

- ALWAYS_OFF
- ALWAYS_ON
- CLIENT_DECIDES

类型：

- number

TimezoneRedirectionStatusCode

模块中可用的 TimezoneRedirectionStatus 代码枚举 DCV

- SUCCESS
- PERMISSION_ERROR
- GENERIC_ERROR

类型：

- number

WebcamErrorCode

模块中可用的 WebcamError 代码枚举 DCV

- SETTING_WEBCAM_FAILED
- CHANNEL_NOT_AVAILABLE

类型：

- number

Connection 类

调用 dcv 模块的 connect 方法获取的 Connection 类。有关说明如何使用该类的示例，请参阅入门一节。

公开

- 方法

方法

列出

- attachDisplay(赢了，displayConf) → {承诺。 <number>|承诺。 < {code: MultiMonitorErrorCode，消息：字符串} >}
- captureClipboardEvents（启用，获胜，displayId）→ {void}
- detachDisplay(displayId) → {无效}
- disconnect() → {void}
- disconnectCollaborator(connectionId) → {无效}
- enableDisplayQuality更新（启用）→ {无效}
- enableHighPixel密度（启用）→ {void}
- enableTimezoneRedirection（启用）→ {Promise|Promise。 < {code: TimezoneRedirectionErrorCode，消息：字符串} >}
- enterRelativeMouse模式 () → {void}
- getConnectedDevices() → {承诺。 <数组。 < MediaDeviceInfo >>|承诺。 < {message: string} >}
- getFileExplorer() → {承诺。 < 文件存储 >|承诺。 < {code: ChannelErrorCode，消息：字符串} >}
- getServerInfo() → {serverInfo}
- getScreenshot() → {Promise|Promise。 < {code: ScreenshotErrorCode，消息：字符串} >}
- getStats() → {统计}
- latchModifierKey（密钥、位置、isDown）→ {布尔值}

- openChannel（名称、、回调authToken、命名空间）→{Promise|Promise。 <{code: ChannelErrorCode，消息：字符串} >}
- queryFeature(featureName) →{承诺。 <{启用：布尔值，远程？：字符串，autoCopy？：布尔值，autoPaste？：布尔值，serviceStatus？：字符串，可用吗？：布尔值} >|承诺。 <{message: string} >}
- registerKeyboardShortcuts（快捷方式）→{void}
- requestDisplayConfig(highColorAccuracy) →{Promise|Promise。 <{code: DisplayConfigErrorCode，消息：字符串} >}
- requestDisplayLayout（布局）→{Promise|Promise。 <{code: ResolutionErrorCode，消息：字符串} >}
- requestResolution（宽度、高度）→{Promise|Promise。 <{code: ResolutionErrorCode，消息：字符串} >}
- sendKeyboardEvent（事件）→{布尔值}
- sendKeyboardShortcut（快捷方式）→{无效}
- setDisplayQuality（min，maxopt）→{void}
- setDisplayScale(scaleRatio,displayId) →{Promise|Promise。 <{code: ResolutionErrorCode，消息：字符串} >} (DEPRECATED)
- setKeyboardQuirks（怪癖）→{void}
- setMaxDisplay分辨率 (maxWidth,maxHeight) →{无效}
- setMicrophone（启用）→{Promise|Promise。 <{code: AudioErrorCode，消息：字符串} >}
- setMinDisplay分辨率 (minWidth,minHeight) →{无效}
- setUploadBandwidth（值）→{数字}
- setVolume（音量）→{无效}
- setMicrophone（启用，deviceId）→{Promise|Promise。 <{code: AudioErrorCode，消息：字符串} >}
- setWebcam（启用，deviceId）→{Promise|Promise。 <{code: WebcamErrorCode，消息：字符串} >}
- syncClipboards() →{布尔值}

attachDisplay(赢了，displayConf) →{承诺。 <number>|承诺。 <{code: MultiMonitorErrorCode，消息：字符串} >}

将特定的显示器连接到窗口。您无法连接主显示器。如果成功，该函数返回 `displayId`。

参数：

<table>
<tr><th>名称</th><th>Type</th><th>描述</th></tr>
<tr><td>win</td><td>对象</td><td>必须将显示器连接到的窗口。</td></tr>
<tr><td>displayConf</td><td>对象</td><td>显示器的配置。
<table>
<tr><th>名称</th><th>Type</th><th>Attributes</th><th>描述</th></tr>
<tr><td>displ</td><td>numbe</td><td><可选></td><td>显示器的ID。</td></tr>
<tr><td>displ vName</td><td></td><td></td><td>显示器div的名称。</td></tr>
</table>
</td></tr>
</table>

返回值:

Promise。如果被拒绝，Promise 返回一个错误对象。

类型

承诺。 <number>| 承诺。 < {code: MultiMonitorErrorCode，消息：字符串} >

captureClipboardEvents（启用，获胜，displayId）→ {void}

开始或停止侦听复制粘贴事件。对于交互式剪贴板（对于粘贴，始终如此），我们需要开始侦听复制/粘贴事件。仅在需要时（例如，在显示模式时）开始和停止侦听可能是非常有用的。

参数：

名称	Type	Attributes	描述
`enabled`	布尔值		要开始侦听事件，请指定 `true`。要停止侦听事件，请指定 `false`。
`win`	对象	<可选>	在其中侦听事件的窗口。如果省略，则使用默认窗口。
`displayId`	number	<可选>	应侦听事件的显示器的 ID。如果省略，则使用窗口的默认显示器。

返回值:

类型

void

detachDisplay(displayId) → {无效}

断开连接特定的显示器。无法断开连接主显示器。

参数：

名称	Type	描述
`displayId`	number	要断开连接的显示器的 ID。

返回值:

类型

void

disconnect() → {void}

断开与 Amazon DCV 服务器的连接并关闭连接。

返回值:

类型

void

disconnectCollaborator(connectionId) → {无效}

请求断开与提供的连接 ID 关联的合作者的连接（自 Amazon DCV Web Client SDK 版本 1.1.0 起）。

参数：

名称	Type	描述
`connectionId`	布尔值	将断开的连接的 ID。

返回值:

类型

void

enableDisplayQuality更新（启用）→ {无效}

为不接收更新的流区域启用或禁用显示质量更新。禁用显示质量更新将减少带宽使用量，但也会降低显示质量。

参数：

名称	Type	描述
`enable`	布尔值	要启用显示质量更新，请指定 `true`。要禁用显示质量更新，请指定 `false`。

返回值:

类型

void

enableHighPixel密度（启用）→ {void}

在客户端上启用或禁用高像素密度。

参数：

名称	Type	描述
`enable`	布尔值	是否应启用高像素密度。

返回值:

类型

void

enableTimezoneRedirection（启用）→ {Promise|Promise。 < {code: TimezoneRedirectionErrorCode，消息：字符串} >}

启用或禁用时区重定向。在启用后，客户端请求服务器将服务器桌面时区与客户端时区匹配。

参数：

名称	Type	描述
`enable`	布尔值	要启用时区重定向，请指定 `true`。要禁用时区重定向，请指定 `false`。

返回值:

Promise。如果被拒绝，Promise 返回一个错误对象。

类型

承诺。 <number>| 承诺。 < {code: TimezoneRedirectionErrorCode，消息：字符串} >

enterRelativeMouse模式 () → {void}

启用相对鼠标模式。

返回值:

类型

void

getConnectedDevices() → {承诺。 <数组。 < MediaDeviceInfo >>|承诺。 < {message: string} >}

请求连接到客户端计算机的媒体设备的列表。

返回值:

如果成功，它将返回一个解析为 MediaDeviceInfo 对象数组的 Promise。欲了解更多信息，请参阅 https://developer.mozilla.org/en-US/docs/Web/API/MediaDeviceInfo。如果被拒绝，Promise 返回一个错误对象。

类型

承诺。 <数组。 < MediaDeviceInfo >> | 承诺。 < {message: string} >

getFileExplorer() → {承诺。 < 文件存储 >|承诺。 < {code: ChannelErrorCode，消息：字符串} >}

获取用于管理 Amazon DCV 服务器文件存储的对象。

返回值:

Promise。如果完成，则解析为文件资源管理器对象；如果被拒绝，则解析为错误对象。

类型

承诺。 < 文件存储 > | 承诺。 < {code: ChannelErrorCode，消息：字符串} >

getServerInfo() → {serverInfo}

获取有关 Amazon DCV 服务器的信息。

返回值:

有关服务器软件的信息。

类型

serverInfo

getScreenshot() → {Promise|Promise。 < {code: ScreenshotErrorCode，消息：字符串} >}

以PNG格式检索远程桌面的屏幕截图。屏幕截图将在screenshotCallback观察者中返回。 `null`如果失败，将改为返回。

返回值:

在处理了请求时解析的 Promise。如果被拒绝，我们将收到一个错误对象。

类型

承诺 | 承诺。 < {code: ScreenshotErrorCode，消息：字符串} >

getStats() → {统计}

获取有关 Amazon DCV 服务器的统计信息。

返回值:

有关流式传输统计信息的信息。

类型

stats

latchModifierKey（密钥、位置、isDown）→ {布尔值}

为允许的修饰键发送单个键盘 keydown 或 keyup 事件。

参数：

名称	Type	描述
key	控制 | Alt | | Meta AltGraph | OS | Shift	要发送的键。
location	KeyboardEvent. 位置	键的位置。欲了解更多信息，请参阅 https://developer.mozilla.org/en-US/docs/Web/KeyboardEvent/API/location。
isDown	布尔值	如果要注入的按键事件是 keydown（true）或 keyup（false）。

返回值:

如果请求的组合有效，该函数返回 true，否则，该函数返回 false。

类型

布尔值

openChannel（名称、、回调authToken、命名空间）→ {Promise|Promise。 < {code: ChannelErrorCode，消息：字符串} >}

如果连接是在 Amazon DCV 服务器上创建的，则会在连接上打开自定义数据通道。

参数：

名称	Type	描述
`name`	字符串	通道的名称。
`authToken`	字符串	用于连接到通道的身份验证令牌。
`callbacks`	对象	onMessage 和要 onClose 调用的回调函数。
`namespace`	字符串	通道的命名空间。自亚马逊 DCV网络客户端 SDK 1.2.0 和 Amazon DCV Server 2022.1 起可用。

返回值:

Promise。如果被拒绝，我们将收到一个错误对象。

类型

承诺 | 承诺。 < {code: ChannelErrorCode，消息：字符串} >

queryFeature(featureName) → {承诺。 < {启用：布尔值，远程？：字符串，autoCopy？：布尔值，autoPaste？：布尔值，serviceStatus？：字符串，可用吗？：布尔值} >|承诺。 < {message: string} >}

查询特定 Amazon DCV 服务器功能的状态。

参数：

名称	Type	描述
`featureName`	feature	要查询的功能的名称。

返回值:

Promise。如果已解析，该函数返回一个 `status` 对象，该对象始终包含 `enabled` 属性，并且还可能包含其他属性。如果被拒绝，该函数返回一个 `error` 对象。

类型

{承诺。 < {启用：布尔值，远程？：字符串，autoCopy？：布尔值，autoPaste？：布尔值，serviceStatus？：字符串，可用吗？：布尔值} > | 承诺。 < {message: string} >

registerKeyboardShortcuts（快捷方式）→ {void}

注册键盘快捷键。

参数：

名称	Type	描述
`shortcuts`	Array.<Object>	要注册的键和映射的数组。 名称 Type 描述 sequenc Array.<Object> 要注册的键盘快捷键。 名称 Type 描述 k K 用boardE v 户.key

名称	Type	描述
		名称 Type 描述 名称 T 描述 按下的键的值。欲了解更多信息，请参阅 https://developer.mozilla.org/en-US/docs/Web/KeyboardEvent/API/key。

名称	Type	描述
		名称 Type 描述 名称 T 描述 l K 要发送的键的数组。键盘上的键的位置。欲了解更多信息，请参阅 https://developer.mozilla.o

名称	Type	描述
		名称 Type 描述 名称 T 描述 rg/en-US/docs/Web/KeyboardEvent/API/location。 output Array.<Object> 快捷键执行的预期操作。 名称 T 描述 k k v 用户按下的键的值。欲了解更 boardE t.key

名称	Type	描述
		名称 Type 描述 多信息，请参阅 https://developer.mozilla.org/en-US/docs/Web/KeyboardEvent/API/key。 keyboardE 要发送的键的数组。键盘上

名称	Type	描述
		名称 Type 描述 名称 T 描述的键的位置。欲了解更多信息，请参阅 https://developer.mozilla.org/en-US/docs/Web/KeyboardEvent/API/location。

返回值:

类型

void

requestDisplayConfig(highColorAccuracy) → {Promise|Promise。 < {code: DisplayConfigErrorCode，消息：字符串} >}

向 Amazon DCV 服务器请求更新的显示配置。自亚马逊DCV网络客户端 SDK 1.1.0 和 Amazon DCV Server 2022.0 起可用。

参数：

名称	Type	描述
highColorAccuracy	布尔值	是否应请求高色彩精度。

返回值:

Promise。如果被拒绝，Promise 返回一个错误对象。

类型

承诺 | 承诺。 < {code: DisplayConfigErrorCode，消息：字符串} >

requestDisplayLayout（布局）→ {Promise|Promise。 < {code: ResolutionErrorCode，消息：字符串} >}

为连接请求更新的显示布局。

参数：

名称	Type	描述
layout	Array.<Monitor>	请求的内容显示在布局中。

返回值:

Promise。如果被拒绝，我们将收到一个错误对象。

类型

承诺 | 承诺。 < {code: ResolutionErrorCode，消息：字符串} >

requestResolution（宽度、高度）→ {Promise|Promise。 < {code: ResolutionErrorCode，消息：字符串} >}

向 Amazon DCV 服务器请求更新的显示分辨率。

参数：

名称	Type	描述
`width`	number	请求的宽度（以像素为单位）。最小的允许值为 `0`。
`height`	number	请求的高度（以像素为单位）。最小的允许值为 `0`。

返回值:

Promise。如果被拒绝，Promise 返回一个错误对象。

类型

承诺 | 承诺。 < {code: ResolutionErrorCode，消息：字符串} >

sendKeyboardEvent（事件）→ {布尔值}

发送键盘快捷键事件。有关键盘事件的更多信息，请参阅 https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent。有效的键盘事件包括：`keydown`、`keypress` 和 `keyup`。有关这些事件的更多信息，请参阅 https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent#events。

参数：

名称	Type	描述
event	KeyboardEvent	要发送的键盘事件。

返回值:

如果事件无效，该函数返回 false。如果事件有效，该函数返回 true。

类型

布尔值

sendKeyboardShortcut（快捷方式）→ {无效}

发送键盘快捷键。可以使用该函数发送完整 keydown 或 keyup 序列。例如，在发送 Ctrl + Alt + Del 时，将发送所有按键的 keydown 事件，然后发送 keyup 事件。即使您希望发送单个键，也可以使用该函数。

参数：

名称	Type	描述
shortcut	Array.<Object>	要发送的键的数组。 名称 / Type / 描述 key / Keyboard vent.key / 用户按下的键的值。欲了解更多信息，请参阅 https://d

名称	Type	描述
		名称 Type 描述
		eveloper.mozilla.org/en-US/docs/Web/KeyboardEvent/API/key。
		locatic Keyboarc vent. 位置 要发送的键的数组。键盘上的键的位置。欲了解更多信息，请参阅 https://developer.mozilla.org/en-US/docs/Web/KeyboardEvent/API/location。

返回值:

类型

void

setDisplayQuality (min , maxopt) → {void}

设置用于连接的图像质量。有效范围是 0 到 100，其中 1 为最低图像质量，100 为最高图像质量。指定 0 将保留当前值。

参数：

名称	Type	Attributes	描述
min	number		最低图像质量。
max	number	<可选>	最高图像质量。

返回值:

类型

void

setDisplayScale(scaleRatio,displayId) → {Promise|Promise。 < {code: ResolutionErrorCode，消息：字符串} >} (DEPRECATED)

自版本 1.3.0 起已弃用。无需再设置显示比例。将在内部自动管理鼠标坐标。

通知 Amaz DCV on 显示屏已在客户端缩放。可以使用该函数通知服务器，需要缩放鼠标事件以与客户端的显示比率匹配。

参数：

名称	Type	描述
scaleRatio	float	要使用的缩放比率。必须是严格的正数。

名称	Type	描述
displayId	number	要缩放的显示器的 ID。

返回值:

Promise。如果被拒绝，Promise 返回一个错误对象。

类型

承诺 | 承诺。 < {code: ResolutionErrorCode，消息：字符串} >

setKeyboardQuirks（怪癖）→ {void}

设置客户端计算机的键盘特性。

参数：

名称	Type	描述
quirks	对象	要启用或禁用的键盘特性。 名称 Type 描述 macOptiToAlt 布尔值 对于 macOS，要将 Option 键映射到 Alt，请指定 true。否则，请指定 false。

名称	Type	描述
		名称 Type 描述 macCommandToCtrl 布尔值 对于macOS，要将Command键映射到Ctrl，请指定true。否则，请指定false。

返回值:

类型

void

setMaxDisplay分辨率 (maxWidth,maxHeight) → {无效}

设置用于连接的最大显示分辨率。

参数：

名称	Type	描述
maxWidth	number	最大显示宽度（以像素为单位）。最小的允许值为 0。
maxHeight	number	最大显示高度（以像素为单位）。最小的允许值为 0。

返回值:

类型

void

setMicrophone（启用）→ {Promise|Promise。 < {code: AudioErrorCode，消息：字符串} >}

启用或禁用麦克风。

参数：

名称	Type	描述
`enable`	布尔值	要启用麦克风，请指定 `true`。要禁用麦克风，请指定 `false`。

返回值:

Promise。如果被拒绝，Promise 返回一个错误对象。

类型

承诺 | 承诺。 < {code: AudioErrorCode，消息：字符串} >

setMinDisplay分辨率 (minWidth,minHeight) → {无效}

设置用于连接的最小显示分辨率。某些应用程序可能需要使用最低显示分辨率。如果所需的最小分辨率大于客户端支持的最大分辨率，则使用调整大小策略。请谨慎使用该函数。调整大小策略可能会导致鼠标和触摸输入系统的精度下降。

参数：

名称	Type	描述
`minWidth`	number	最小显示宽度（以像素为单位）。最小的允许值为 `0`。
`minHeight`	number	最小显示高度（以像素为单位）。最小的允许值为 `0`。

返回值:

类型

void

setUploadBandwidth（值）→ {数字}

设置用于将文件上传到 Ama DCV zon 服务器的最大带宽。

参数：

名称	Type	描述
`value`	number	最大带宽限制（以 kbps 为单位）。有效范围是 1024 kbps 到 102400 kbps。

返回值:

- 设置的带宽限制。如果在服务器上禁用了文件存储功能，则为 `null`。

类型

number

setVolume（音量）→ {无效}

设置用于音频的音量。有效范围是 0 到 100，其中 0 为最低音量，100 为最高音量。

参数：

名称	Type	描述
`volume`	number	要使用的音量。

返回值:

类型

void

setMicrophone（启用，deviceId）→ {Promise|Promise。 < {code: AudioErrorCode，消息：字符串} >}

[实验性-将来可能会改变] 启用或禁用麦克风。

参数：

名称	Type	描述
`enable`	布尔值	要启用麦克风，请指定 `true`。要禁用麦克风，请指定 `false`。
`deviceId`	字符串	麦克风的设备 ID。如果 deviceId 未提供，`default` deviceId 则使用。

返回值:

Promise。如果被拒绝，Promise 返回一个错误对象。

类型

承诺 | 承诺。 < {code: AudioErrorCode，消息：字符串} >

setWebcam（启用，deviceId）→ {Promise|Promise。 < {code: WebcamErrorCode，消息：字符串} >}

启用或禁用网络摄像头。

参数：

名称	Type	描述
`enable`	布尔值	要启用网络摄像头，请指定 `true`。要禁用网络摄像头，请指定 `false`。
`deviceId`	字符串	网络摄像头的设备 ID。

返回值:

承诺，如果成功，将解析为连接/分离的网络摄像头。deviceId如果被拒绝，Promise 返回一个错误对象。

类型

承诺。 <string>| 承诺。 < {code: WebcamErrorCode，消息：字符串} >

syncClipboards() → {布尔值}

将本地客户端剪贴板与远程 Amazon DCV 服务器剪贴板同步。浏览器必须支持自动复制。

返回值:

如果剪贴板已同步，该函数返回 `true`。如果剪贴板尚未同步，或者浏览器不支持自动复制，该函数返回 `false`。

类型

布尔值

Authentication 类

必须调用 dcv 模块的 `authenticate` 方法，以使用 Authentication 类获取身份验证令牌。有关说明如何使用该类的示例，请参阅入门一节。

公开

- 方法

方法

列出

- retry() → {void}
- sendCredentials（凭证）→ {无效}

retry() → {void}

重试身份验证过程。

返回值:

类型

void

sendCredentials（凭证）→ {无效}

将客户端提供的身份验证凭证发送到 Amazon DCV 服务器。

参数：

名称	Type	描述
`credentials`	对象	包含提供的凭证的对象。凭证必须具有质询中指定的相同名称和相同类型。

返回值:

类型

void

Resource 类

Resource 类可以获取或丢弃刚刚打印或下载的相应文件。在执行这些操作时，将分别调用相应的观察者函数 `filePrinted` 和 `fileDownload` 并将资源对象作为唯一参数。可以接受或拒绝此类资源，以便获取或丢弃它们引用的文件。

公开

- 方法

方法

列出

- 接受 (urlParameters) → {无效}
- decline() → {void}

接受 (urlParameters) → {无效}

在本地下载资源。

参数：

名称	Type	描述
`urlParameters`	对象	可选对象，其中包含传递给获取资源的请求的URL搜索参数的键/值对。

返回值:

类型

void

decline() → {void}

丢弃资源。

返回值:

类型

void

亚马逊 DCV Web 用户界面 SDK

一个 JavaScript React 组件库，目前正在导出一个名为的 React 组件`DCVViewer`，该组件连接到 Amazon DCV Server 并呈现工具栏以与远程流进行交互。

公开

- 组件

组件

列出

- DCVViewer

DCVViewer

React 组件，渲染工具栏及其所有可用于与远程流交互的功能。

属性：

列出

- dcv
- uiConfig

dcv

名称	Type	必需	描述
dcv	对象	是	该对象定义建立与 Amazon DCV Server 的连接所需的属性，设置日志级别以及加载亚马逊 DCV Web Client SDK 资产和访问DCV资源的起点。URL

名称	Type	必需	描述
ses	Strir	是	亚马逊会 DCV 话 ID。
aut	Strir	是	在连接到服务器时使用的身份

名称	Type	必需	描述
			名称 Type 必需 描述 验证令牌。 ser Strir 是 正在运行的亚马逊DCV服务器的主机名和端口，格式如下：https://dcv_host_a

名称	Type	必需	描述
			名称 Type 必需 描述 ddress:port。例如：https: //:8443 my-dcv-server。 bas Strir 是 URL从中加载SDK文件的绝对值或相对值。 res Strir 否（默 访问DCV ase

名称	Type	必需	描述
			名称 Type 必需 描述 认值： 资源的绝对值或相对URL值。 onDect 函数 否（默认值：=> {}） 在断开与Amazon DCV服务器的连接并关闭连接时调

名称	Type	必需	描述
			名称 Type 必需 描述 用的回调函数。 log Logl 否（默认值：INF 查看器中使用的日志级别 evel.

uiConfig

名称	Type	必需	描述
uiConfig	对象	否（默认值：{}）	该对象定义属性以配置工具栏是否可见，以及是否在工具栏上显示全屏和多显示器按钮。

名称	Type	必需	描述
			名称 Type 必需 描述 too 对象 否（默认值：） 该对象定义工具栏配置选项。 隐藏 Type 描述 可折叠 Type 描述 默认值：true 定义是显示还是隐藏工具栏。

名称	Type	必需	描述
			名称 / Type / 必需 / 描述

名称	Type	必需	描述
fullscreenButton	布尔值	否	默认值：true 定义是显示还是隐藏工具栏上的全屏按钮。
multimonitorButton	布尔值	否	默认值：true 定义是显

名称	Type	必需	描述
			名称 Type 必需 描述 toolbarMultiMonitorButton boolean 隐藏 显示还是隐藏工具栏上的多显示器按钮。

Amazon DCV Web 客户端的发行说明和文档历史记录 SDK

本页提供了 Amazon DCV Web Client 的发行说明和文档历史记录SDK。

主题

Amazon DCV Web 客户端SDK发行说明

本节按发布日期提供亚马逊 DCV Web 客户端SDK的发行说明。

主题

1.8.4 — 2024 年 10 月 1 日

内部版本号	新功能	更改和错误修复
• 语义版本：1.8.4 • Build：840	添加了以下功能： • 已重命名为 “Amazon DCV Web 客户端SDK” • 为高 dpi 显示器添加了新的API enableHighPixel密度 • 添加了在兼容浏览器中选择麦克风的实验API setMicrophone 功能 • 添加了新的连接错误 GATEWAY UNSUPPORTED _ BUSY CREDENTIAL、_ 和 TRANSPORT _ ERROR • 添加了新的关闭原因 EXTERNAL PROTOCOL CONNECTION _ _ EVICTED 和 DISCONNECTION _ REQUESTED	• 改进了网络摄像头处理能力 • 改进了音频播放处理 • 改进了 WebCodecs 操控性 • 改进了麦克风和网络摄像头的即插即用 • 改进了多显示器时的远程窗口拖动 • 文件存储上传和下载权限现在可以正确传播 • 对渲染的次要修复

版本	发布说明
• 语义版本：1.8.4 • Build：840	更改和错误修复 •

1.5.10 – 2023 年 12 月 19 日

版本	发布说明

版本	发布说明
• 语义版本：1.5.10 • 内部版本：684	更改和错误修复 • 修复流式传输解码错误

1.5.6 - 2023 年 11 月 9 日

版本	发行说明
• 语义版本：1.5.6 • 内部版本：659	更改和错误修复 • 提高了流解码和渲染性能 • 删除了对 Internet Explorer 11 的支持

1.4.4 - 2023 年 6 月 29 日

版本	发布说明
• 语义版本：1.4.4 • 内部版本：573	更改和错误修复 • Viewer UI 组件现在使用支持它的浏览器`navigator.keyboard.lock` API上的，在全屏模式下处理特殊按键。 • 修复了一个问题，该问题可能导致在使用 Chrome 114 或更高版本时出现颜色错误。 • 改进了 WebCodecs 检测。 • 修复了进入窗口时鼠标按钮状态问题。 •

版本	发布说明
	修复了一个问题，该问题可能导致修饰键在 macOS 上保持按下状态。 • 在恶劣网络条件下提高了音频可靠性。 • 修复了内存泄漏。 • 改进了日志以包括时间和级别。

1.4.0 - 2023 年 3 月 28 日

版本	发布说明
• 语义版本：1.4.0 • 内部版本：476	新功能 • 为 `FileStorage` 对象添加新的 `uploadFiles` 方法以上传多个文件。 • 查看器 UI 组件现在支持拖放以启动文件上传。 • WebCodecs 浏览器API现在还用于音频和网络摄像头。 更改和错误修复 • 修复了与来自同一页面的重复连接相关的内存泄漏。 • `setUploadBandwidth` 现在允许最多 1 Gbps 的值。

版本	发布说明
	• 优化了 UI 组件渲染。 • 修复了 Windows 上对动画光标的支持。 • 修复了在同一操作同时存在文本和图像数据时的剪贴板支持问题。 • 提高了网络摄像头的稳健性API：请求正在进行时无法更改设置，`webcam.setEnabled` 现在可以跟踪请求正在进行的设备ID并返回Promise。查看器 UI 组件在出现错误时显示通知。

1.3.1 - 2022 年 12 月 9 日

版本	发布说明
• 语义版本：1.3.1 • 内部版本：413	更改和错误修复 • 修复了一个问题，该问题可能导致时区重定向 UI 与服务器不同步。 • 修复了多次重新连接后的内存泄漏。 • 修复了一个问题，该问题导致断开连接时出现空白页。 • 修复了导致音频解码器关闭时控制台发出警告的错误。

1.3.0 - 2022 年 11 月 11 日

版本	发布说明
• 语义版本：1.3.0 • 内部版本：407	新功能 • 采用 Cloudscape（ https://cloudscape.design) for the UI Viewer component。 • 添加了对时区重定向的支持。 更改和错误修复 • 修复了DCV查看者聚焦时异步剪贴板上缺少更新的问题。 • 在客户端缩放显示比例时，不再需要使用 `setDisplayScale` 函数。 • 现在，在卸载 `DCVViewer` 组件时，它自动调用 `disconnect()` 。

1.2.1 - 2022 年 7 月 21 日

版本	发布说明
• 语义版本：1.2.1 • 内部版本：358	更改和错误修复 • 修复了导致无法连接到 Amazon DCV 服务器 2019.1 及更早版本的问题。

1.2.0 - 2022 年 6 月 29 日

版本	发布说明
• 语义版本：1.2.0 • 内部版本：352	更改和错误修复 • 修复了收到的帧大于支持的最大分辨率（4096x2160（时的崩溃错误。 • 资源对象（作为参数传递给 `fileDownload` 和 `filePrinted` 观察者）现在具有 `accept` 和 `decline` 方法，可以为对象调用这些方法以分别下载和丢弃资源。 • 修复了断开连接时的自动剪贴板同步小错误。

1.1.3 - 2022 年 5 月 23 日

版本	发布说明
• 语义版本：1.1.3 • 内部版本：329	更改和错误修复 • 修复了指定 web-url-path选项时无法成功连接的问题。

1.1.2 - 2022 年 5 月 19 日

版本	发布说明
• 语义版本：1.1.2 •	更改和错误修复 •

版本	发布说明
内部版本：322	修复了一个问题，该问题可能导致在连接后输入无法正常工作。 • 修复了扩展比率大于 1 时的鼠标坐标。

1.1.1 - 2022 年 3 月 23 日

版本	发布说明
• 语义版本：1.1.1 • 内部版本：309	更改和错误修复 • 与服务器通信超时时报告 `Transport Error`。 • 修复了流式传输大分辨率内容时反复出现的解码错误。

1.1.0 - 2022 年 2 月 23 日

版本	发布说明
• 语义版本：1.1.0 • 内部版本：295	新功能 • 发布带有 `DCVViewer` React 组件的亚马逊 DCV Web 用户界面SDK库。 • 将 Amazon DCV Web 客户端导出为SDK UMD和 ES 模块。 • 添加了高色彩精度支持。 •

版本	发布说明
	添加了列出连接到会话的客户端并与其进行交互的功能。添加了连接和断开连接通知。 更改和错误修复 • 改进了 WebCodecs 解码支持。 • 各种键盘改进。 • 修复了禁用剪贴板时导致无法打开第二个屏幕的错误。

1.0.4 - 2021 年 12 月 20 日

版本	发布说明
• 语义版本：1.0.4 • 内部版本：249	新功能 • 支持从同一页面中打开多个连接。 • Support SDK 从中加载CDN.

1.0.3 - 2021 年 9 月 1 日

版本	发布说明
• 语义版本：1.0.3 • 内部版本：202	新功能 •

版本	发布说明
	对的实验性支持 WebCodecs。默认禁用该功能，必须使用新属性 `enableWebCodecs` 通过 `ConnectionConfig` 对象启用该功能。 • 剪贴板：在基于 Chromium 的浏览器上添加了对 `image/png` 数据类型的支持。 • 添加了观察者/回调以将服务器的屏幕截图作为PNG图像获取（需要亚马逊DCV服务器2021.2）。 更改和错误修复 • 改进了键盘修饰键处理。

1.0.2 - 2021 年 7 月 30 日

版本	发布说明
• 语义版本：1.0.2 • 内部版本：167	• 修复了触控笔事件的压力检测。 • 改进了 Chrome 上的韩语键盘布局支持。

1.0.1 - 2021 年 5 月 31 日

版本	发布说明
• 语义版本：1.0.1 •	• 修复了连接错误传播和关闭原因 •

版本	发布说明
内部版本：141	修复了文件存储块进度更新 • 改进了网络摄像头处理 • 改进了音频输入处理

1.0.0 - 2021 年 3 月 24 日

版本	发布说明
• 语义版本：1.0.0 • 内部版本：81	Amazon DCV Web 客户端的初始版本SDK。

文档历史记录

下表描述了此版本的 Amazon DCV Web 客户端的文档SDK。

更改	描述	日期
亚马逊 DCV Web 客户端 1. SDK 8.4 版	亚马逊 DCV Web 客户端 SDK 1.8.4 现已推出。有关更多信息，请参阅 SDKv.1.8.4。	2024年10月1日
亚马逊 DCV Web 客户端 1. SDK 5.6 版	亚马逊DCV网络客户端 SDK 1.5.6 现已推出。有关更多信息，请参阅 SDKv.1.5.6。	2023 年 11 月 9 日
亚马逊 DCV Web 客户端 1. SDK 4.4 版	亚马逊 DCV Web 客户端 SDK 1.4.4 现已推出。有关更多信息，请参阅 SDKv.1.4.4。	2023 年 6 月 29 日

更改	描述	日期
亚马逊 DCV Web 客户端 1. SDK 4.0 版	亚马逊DCV网络客户端 SDK 1.4.0 现已推出。有关更多信息，请参阅 SDKv.1.4.0。	2023 年 3 月 28 日
亚马逊 DCV Web 客户端 1. SDK 3.1 版	亚马逊 DCV Web 客户端 SDK 1.3.1 现已推出。有关更多信息，请参阅 SDKv.1.3.1。	2022 年 12 月 9 日
亚马逊 DCV Web 客户端 1. SDK 3.0 版	亚马逊 DCV Web 客户端 SDK 1.3.0 现已推出。有关更多信息，请参阅 SDKv.1.3.0。	2022 年 11 月 11 日
亚马逊 DCV Web 客户端 1.2. SDK 0 版	亚马逊 DCV Web 客户端 SDK 1.2.0 现已推出。有关更多信息，请参阅 SDKv.1.2.0。	2022 年 6 月 29 日
亚马逊 DCV Web 客户端 1.1. SDK 0 版	亚马逊 DCV Web 客户端 SDK 1.1.0 现已推出。有关更多信息，请参阅 SDKv.1.1.0。	2022 年 2 月 23 日
亚马逊 DCV Web 客户端 1.0.1 SDK 版	更正了一些错别字。修复了一些小错误，参见 SDKv.1.0.1。	2021 年 5 月 31 日
初始版本	该内容的第一版。	2021 年 3 月 24 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。