NICE DCV

Web 客户端 SDK 开发人员指南

NICE DCV: Web 客户端 SDK 开发人员指南

Table of Contents

什么是 NICE DCV Web 客户端 SDK？

NICE DCV 是一种高性能远程显示协议。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 NICE DCV 与 Amazon EC2 结合使用，您可以在 Amazon EC2 实例上远程运行图形密集型应用程序。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。

NICE DCV Web 客户端 SDK 是一个 JavaScript 库，你可以用它来开发自己的 NICE DCV Web 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到正在运行的 NICE DCV 会话并与之交互。

使用 NICE DCV Web Client SDK 作为构建块，您可以构建自定义 Web 应用程序，让用户能够从任何地方即时访问其桌面或应用程序，并具有与本地安装的应用程序几乎无法区分的响应和流畅的性能。

本指南介绍了如何使用 NICE DCV Web Client SDK 构建自定义 Web 浏览器客户端应用程序，以便在工作流程中与 NICE DCV 会话进行交互。

主题

先决条件

在开始使用 NICE DCV Web 客户端 SDK 之前，请确保熟悉 NICE DCV 和 NICE DCV 会话。有关更多信息，请参阅 。NICE DCV 管理员指南.

NICE DCV Web 客户端 SDK 支持 NICE DCV 服务器版本 2020 及更高版本。

支持的功能

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

- Connect 到 Windows NICE DCV 服务器
- Connect 到 Linux NICE 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，请参阅支持的功能中的NICE DCV 用户指南.

支持浏览器

NICE DCV 网络客户端软件开发工具包支持 JavaScript (ES6)，可以从 JavaScript 或 TypeScript 应用程序中使用它。

NICE DCV Web 客户端 SDK 支持以下 Web 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

NICE DCV Web 客户端 SDK 版本按以下格式定义：*major*.*minor*.*patch*. 版本控制约定通常遵守语义版本控制模型. 主要版本的更改，例如来自`1.x.x`到`2.x.x`，表示已经引入了可能需要更改代码和计划部署的重大更改。次要版本的更改，例如来自`1.1.x`到`1.2.x`，向后兼容，但可能包括弃用的元素。

入门 NICE DCV Web 客户端 SDK

NICE DCV Web 客户端 SDK 包括一个主dcv.js文件和一些辅助组件。所有文件都分发在压缩存档中，可以从网站 NICE.

开始使用 NICE DCV Web 客户端 SDK

1. NICE DCV Web 客户端 SDK 归档文件使用安全 GPG 签名进行数字签名的。要验证档案的签名，您必须导入 NICE GPG 密钥。要执行此操作，请打开一个终端窗口并导入 NICE GPG 密钥。

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

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

2. 下载NICE DCV Web 客户端 SDK 存档和NICE DCV Web 客户端 SDK 存档签名来自 的网站 NICE.
3. 使用签名验证 NICE DCV Web 客户端 SDK 存档的签名。

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

例如：

```
$ gpg --verify nice-dcv-web-client-sdk-1.1.3-329.zip.sign nice-dcv-web-client-
sdk-1.1.3-329.zip
```

4. 如果签名验证成功，请提取 NICE DCV Web Client SDK 归档的内容，然后将提取的目录放在 Web 服务器上。例如：

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

Important

- 在 Web 服务器上部署 NICE DCV Web 客户端 SDK 时，必须保留文件夹结构。
- 使用 NICE DCV Web UI SDK 时，请注意DCVViewerReact 组件希望此软件包中的 EULA.txt 和 third-party-licenses.txt 文件将出现在嵌入式 Web 服务器的 URL 路径中。应该修改 third-party-licenses.txt 文件，以包含来自 NICE DCV Web Client SDK 软件包的相应文件的内容，以及消费用户应用程序使用的库中可能的任何其他许可证信息。

Connect 到 NICE DCV 服务器并获得第一帧

以下教程向您展示了如何为自定义 Web 客户端准备 HTML 页面、如何进行身份验证并连接到 NICE DCV 服务器，以及如何从 NICE DCV 会话接收第一帧流媒体内容。

主题

第 1 步：准备您的 HTML 页面

在您的网页中，您必须加载所需的 JavaScript 模块，你必须添加`<div>`具有有效的 HTML 元素`id`您希望 NICE DCV Web 客户端 SDK 从远程 NICE DCV 服务器中绘制内容流的位置。

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>
```

第 2 步：进行身份验证、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 NICE DCV 服务器以及如何从 NICE DCV 服务器接收第一帧内容。

首先，来自`index.js`文件导入 NICE 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`，它也可以从相应的软件包中作为 ECMascript 模块 (ESM) 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 NICE DCV 服务器 URL 的变量。

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

在脚本加载时，记录 NICE DCV Web 客户端 SDK 版本，然后在页面加载时调用`main`function.

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

这些区域有：`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
    }
  );
}
```

这些区域有：promptCredentials、error, 和success函数是必须在身份验证过程中定义的回调函数。

如果 NICE DCV 服务器提示输入凭据，promptCredentials回调函数从 NICE DCV 服务器接收请求的凭据质询。如果 NICE 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据。以下代码示例假设用户名为my_dcv_user而且密码是my_password.

如果身份验证失败，则error回调函数从 NICE DCV 服务器接收错误对象。

如果身份验证成功，则success回调函数接收一组包含会话 ID (sessionId) 和授权令牌 (authToken) 对于每个会话my_dcv_user允许用户在 NICE DCV 服务器上连接到。以下代码示例调用 connect 函数并连接到阵列中返回的第一个会话。

```
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);
}
```

Connect NICE DCV 服务器。这些区域有：firstFrame从 NICE DCV 服务器接收第一帧时调用回调方法。

```
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);
  });
}
```

奖金：自动创建 HTML 登录表单

这些区域有：challenge对象在promptCredentials调用回调函数。它包括一个名为requiredCredentials这是一组对象-NICE DCV 服务器请求的每个凭据一个对象。每个对象都包含所请求的凭据的名称和类型。您可以使用challenge和requiredCredentials对象以自动创建 HTML 登录表单。

以下代码示例演示了如何执行此操作。

```
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 NICE DCV 功能

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

NICE 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 NICE DCV Web Client SDK 支持某项功能，但根据会话管理员定义的权限，可能会阻止对该功能的访问。有关更多信息，请参阅 。配置 NICE DCV 授权中的NICE DCV 管理员指南.

了解 FureUpdate 回调函数

当 NICE DCV 会话中某项功能的可用性发生变化时，NICE DCV Web 客户端 SDK 会使用`featuresUpdate`您在建立连接时指定的回调函数。例如：

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

回调函数只通知您可用性已更改的功能。这些区域有：`list`参数是一个字符串数组，它只包含更新的要素的名称。例如，如果会话的音频输入功能的可用性发生了变化，则该参数仅包括`["audio-in"]`. 如果稍后会话的剪贴板复制和粘贴功能的可用性发生了变化，则该参数仅包括`["clipboard-copy", "clipboard-paste"]`.

处理功能更新

这些区域有：`featuresUpdate`回调函数只会通知您一个或多个功能的可用性已发生变化。要知道哪些功能已更新，必须使用`connection.queryFeature`方法。这可以在收到变更通知后随时完成。此方法返回`Promise`这将解析为请求的功能的更新状态。这些区域有：`status`value 始终是关联的，它有一个布尔值 (`true`|`false`) 名为的属性`enabled`. 某些功能可能在`status`值。如果该功能的可用性尚未更新，则会被拒绝。

以下示例代码演示了如何执行此操作。

```
// 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 NICE DCV Web UI SDK

以下教程向您展示了如何对 NICE DCV 服务器进行身份验证、连接到该服务器并渲染`DCVViewer`来自 NICE DCV Web UI SDK 的 React 组件。

主题

先决条件

你需要安装React、ReactDOM、AWS UI Components React、AWS UI Global Styles和AWS UI Design Tokens.

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

您还需要下载NICE DCV Web Client SDK. 请参阅入门 NICE DCV Web 客户端 SDK (p. 3)阅读 step-by-step 指南如何执行此操作。

您必须创建一个别名才能导入dcv模块，因为它是 NICE DCV Web UI SDK 的外部依赖关系。例如，如果你使用 webpack 来捆绑你的 Web 应用程序，你可以使用解决.alias像这样的选项：

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

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

如果你使用汇总来进行捆绑，你可以安装@rollup /插件别名，然后像这样使用它：

```
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：准备您的 HTML 页面

在您的网页中，您必须加载所需 JavaScript 模块你应该有 具有有效的 HTML 元素id您的应用程序的入口组件将在其中呈现。

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>
```

第 2 步：进行身份验证、连接和渲染DCVViewerReact 组件。

本节介绍如何完成用户身份验证过程、如何连接 NICE DCV 服务器以及如何渲染DCVViewerReact 组件。

首先，来自index.js导入文件React、ReactDOM和你的顶级App组件。

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

渲染应用的顶级容器节点。

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

在App.js文件中，将 NICE DCV Web 客户端 SDK 导入为 ESM 模块，DCVViewer来自 NICE DCV Web UI SDK 的 React 组件，React和AWS UI Global Styles程序包。

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

下面是一个示例，展示了如何对 NICE DCV 服务器进行身份验证并渲染DCVViewer来自 NICE DCV Web UI SDK 的 React 组件，前提是身份验证成功。

```
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;
```

这些区域有：promptCredentials、error, 和success函数是必须在身份验证过程中定义的回调函数。

如果 NICE DCV 服务器提示输入凭据，promptCredentials回调函数从 NICE DCV 服务器接收请求的凭据质询。如果 NICE 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据。

如果身份验证失败，则error回调函数从 NICE DCV 服务器接收错误对象。

如果身份验证成功，则success回调函数接收一组包含会话 ID (sessionId) 和授权令牌 (authToken) 对于允许用户在 NICE DCV 服务器上连接的每个会话。上面的代码示例更新了 React 状态以呈现DCVViewer成功身份验证时的组件。

要了解有关此组件所接受的属性的更多信息，请参阅NICE DCV Web UI SDK 参考.

SDK 参考

本节为 NICE DCV Web 客户端 SDK 提供说明、语法和使用示例。

主题

DCV 模块

实现 DCV 协议的客户端的模块。

Exise

方法

列表

身份验证（url，回调）→{身份验证 (p. 54)}

启动指定 NICE DCV 服务器终端节点的身份验证过程。

参数：

名称	类型	说明
`url`	字符串	运行中的 NICE 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`	身份验证回调/回调 (p. 18)	在身份验证过程中可以调用的回调。

返回值:

-身份验证对象。

类型

身份验证 (p. 54)

连接（配置）→ {Promise。 <Connection (p. 38)>| 承诺。 < {代码: 连接错误代码 (p. 25)，消息:string} >}

连接到指定的 NICE DCV 服务器端点。如果连接成功，则会返回连接对象。如果连接失败，它会返回错误对象。

参数：

名称	类型	说明
`config`	连接配置 (p. 24)	这些区域有： ConnectionConfig 对象。

返回值:

-连接对象或错误对象。

类型

承诺。 <Connection (p. 38)> | 承诺。 < {代码: 连接错误代码 (p. 25)，消息:string} >

setLogHandler（处理程序）→ {void}

设置自定义日志处理程序函数。覆盖默认日志处理程序时，使用浏览器控制台进行调试时，原始日志条目的位置将丢失。

参数：

名称	类型	说明
`handler`	函数	自定义日志处理程序函数。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（级别）→ {void}

设置日志级别。只有在使用默认日志处理程序时必需。

参数：

名称	类型	说明
`level`	日志级别 (p. 33)	要使用的日志级别。

返回值:

类型

void

成员

列表

- （常量）AudioError：音频错误码 (p. 14)
- （常量）身份验证错误：身份验证错误代码 (p. 14)
- （常量）渠道错误：渠道错误代码 (p. 15)
- （常量）关闭原因错误：关闭原因错误代码 (p. 15)
- （常量）连接错误：连接错误代码 (p. 15)
- （常量）自定义渠道错误：自定义渠道错误代码 (p. 15)
- （常量）DisplayConfigError：DisplayConfig 错误代码 (p. 15)
- （常量）文件存储错误：文件存储错误代码 (p. 15)
- （常量）LogLevel 别：日志级别 (p. 15)
- （常量）MultiMonitoreRror：MultiMonitor 错误码 (p. 16)
- （常量）解决错误：解决错误代码 (p. 16)
- （常量）版本 (p. 16)
- （常量）ScreenShoArror：屏幕截图恐怖代码 (p. 16)
- （常量）网络摄像头错误：WebCamera Rcode (p. 17)

（常量）AudioError：音频错误码 (p. 18)

这些区域有： AudioError 枚举。

类型:

- 音频错误码 (p. 18)

（常量）身份验证错误：身份验证错误代码 (p. 19)

这些区域有： AuthenticationError 枚举。

类型:

- 身份验证错误代码 (p. 19)

（常量）渠道错误：渠道错误代码 (p. 21)

这些区域有： ChannelError 枚举。

类型:

- 渠道错误代码 (p. 21)

（常量）关闭原因错误：关闭原因错误代码 (p. 22)

这些区域有： ClosingReasonError 枚举。

类型:

- 关闭原因错误代码 (p. 22)

（常量）连接错误：连接错误代码 (p. 25)

这些区域有： ConnectionError 枚举。

类型:

- 连接错误代码 (p. 25)

（常量）自定义渠道错误：自定义渠道错误代码 (p. 26)

这些区域有： CustomChannelError 枚举。

类型:

- 自定义渠道错误代码 (p. 26)

（常量）DisplayConfigError：DisplayConfig 错误代码 (p. 28)

这些区域有： DisplayConfigError 枚举。

类型:

- DisplayConfig 错误代码 (p. 28)

（常量）文件存储错误：文件存储错误代码 (p. 31)

这些区域有： FileStorageError 枚举。

类型:

- 文件存储错误代码 (p. 31)

（常量）LogLevel 别：日志级别 (p. 33)

可用的 SDK 日志级别。

类型:

- 日志级别 (p. 33)

（常量）MultiMonitoreRror：MultiMonitor 错误码 (p. 34)

这些区域有： MultiMonitorError 枚举。

类型:

- MultiMonitor 错误码 (p. 34)

（常量）解决错误：解决错误代码 (p. 35)

这些区域有： ResolutionError 枚举。

类型:

- 解决错误代码 (p. 35)

（常量）版本

NICE DCV 版本包含主要版、次要版本、修补程序、修订版、扩展版和 version Str。

属性：

名称	类型	说明
`major`	integer	主要版本号。
`minor`	integer	次要版本号。
`patch`	integer	修补程序版本号。
`revision`	integer	修订版号。
`extended`	字符串	扩展字符串。
`versionStr`	字符串	表单中的主要、次要、修补程序和修订版号的连接`major.minor.patch+build.revision`.

（常量）ScreenShoArror：屏幕截图恐怖代码 (p. 36)

这些区域有： ScreenshotError 枚举。

类型:

- 屏幕截图恐怖代码 (p. 36)

（常量）网络摄像头错误：WebCamera Rcode (p. 38)

这些区域有： WebcamError 枚举。

类型:

- WebCamera Rcode (p. 38)

类型和回调定义

列表

音频错误码

这些区域有： AudioError DCV 模块中可用的代码枚举

- `SETTING_AUDIO_FAILED`
- `CHANNEL_NOT_AVAILABLE`

类型:

- number

验证回调

身份验证回调

类型:

- 对象

属性：

名称	类型	说明
`promptCredentials`	AuthPrompt凭据回调 (p. 19)	当用户要求凭据受到质疑时要调用的回调函数。
`error`	authError 回调 (p. 19)	身份验证失败时要调用的回调函数。

名称	类型	说明
success	authSucessCallback (p. 20)	身份验证成功时要调用的回调函数。

身份验证错误代码

这些区域有： AuthenticationError DCV 模块中可用的代码枚举

- 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）

身份验证失败时要调用的回调函数。

参数：

名称	类型	说明
authentication	身份验证 (p. 54)	身份验证对象。
error	对象	身份验证过程引发的错误对象。 名称 / 类型 / 说明 code / 身份验证错误代码 (p. 19) / 错误代码。 message / 字符串 / 错误消息。

AuthPrimt凭据回调（身份验证、质询）

当用户要求凭据受到质疑时要调用的回调函数。用户必须通过提供请求的凭据来回答质询。

参数：

名称	类型	说明
authentication	身份验证 (p. 54)	身份验证对象。
challenge	对象	挑战。 名称：requiredCredentials 类型：数组。 <Object> 说明：请求的凭据对象数组。 名称：name 类型：字符串 说明：请求的凭证的名称。 名称：type 类型：字符串 说明：请求的凭据的类型。

AuthSucessCallback（身份验证、身份验证数据）

身份验证成功时要调用的回调函数。

参数：

名称	类型	说明
authentication	身份验证 (p. 54)	身份验证对象。
authenticationData	数组。 <Object>	包括 NICE DCV 会话 ID 和身份验证令牌的对象数组。 名称：sessionId 类型：字符串 说明：漂亮的 DCV 会话 ID。

名称	类型	说明
		名称 | 类型 | 说明 authToken | 字符串 | NICE DCV 会话的身份验证令牌。

通道

可以指定的可用频道。

类型:

- “剪贴板” | “显示” | “输入” | “音频” | “文件存储”

渠道错误代码

这些区域有： ChannelError DCV 模块中可用的代码枚举

- `ALREADY_OPEN`
- `INITIALIZATION_FAILED`
- `REJECTED`

类型:

- number

clipBoarEvent 回调（事件）

当`clipboardEvent`已生成。

参数：

名称	类型	说明
`event`	对象	有关剪贴板事件的信息。 名称 | 类型 | 属性 | 说明 `name` | 成立 | 复制 | 粘贴 | DataSizeAlert | autoCopyDone | 新数据可用 | | 始终在场。事件名称。

名称	类型	说明
		名称 / 类型 / 属性 / 说明 autoPasteDone | RemoteError | 粘贴可用数据 clipboardData 对象 | 字符串 剪贴板中的数据。 autoCopy 布尔值 <optional> 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功能。 maxDataSize number <optional> 可在剪贴板中放置的最大数据量。 error 字符串 <optional> 错误信息（如果适用）。

关闭原因错误代码

这些区域有： ClosingReasonError DCV 模块中可用的代码枚举

- `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`

类型:

- number

ColorSpace

可以指定的可用色彩空间。

类型:

- “RGB” | “YUV_REC601" | “YUV_REC709”

连接回调

发生连接错误时可以调用的回调函数。

类型:

- 对象

属性：

名称	类型	说明
`disconnect`	断开回调 (p. 27)	连接结束时调用的回调函数。
`displayLayout`	DisplayLayout 回调 (p. 28)	更改显示布局或分辨率时要调用的回调函数。
`displayAvailability`	显示可用性回调 (p. 27)	显示器的可用性发生变化时要调用的回调函数。
`firstFrame`	第一帧回调 (p. 31)	从 NICE DCV 服务器接收第一帧时要调用的回调函数。
`filePrinted`	文件打印的回调 (p. 29)	在 NICE DCV 服务器上打印文件时要调用的回调函数。
`fileDownload`	文件下载回调 (p. 29)	准备好从 NICE DCV 服务器下载文件时要调用的回调函数。
`dataChannel`	数据频道回调 (p. 26)	NICE 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。
`licenseNotification`	许可通知回调 (p. 32)	NICE 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。

名称	类型	说明
idleWarningNotification	IdleWarning 通知回调 (p. 32)	NICE DCV 服务器发送空闲超时警告时要调用的回调函数。
collaboratorList	协作者列表回调 (p. 32)	NICE DCV 服务器发送协作者列表时要调用的回调函数（自 NICE DCV Web 客户端 SDK 1.1.0 版起）。
qualityIndicatorState	质量指标状态回调 (p. 34)	连接质量指示器改变状态时要调用的回调函数。
filestorageEnabled	启用文件存储的回调 (p. 31)	启用或禁用文件存储时要调用的回调函数。
featuresUpdate	功能更新回调 (p. 29)	要在功能状态发生变化时调用的回调函数。
clipboardEvent	clipBoarEtivity 回调 (p. 21)	当clipboardEvent已生成。
deviceChangeEvent	设备ChangeEvent 回调 (p. 27)	当deviceChange触发事件。
screenshot	屏幕截图回调 (p. 36)	当screenshot可用。

连接配置

NICE DCV 连接的配置。

类型:

- 对象

属性：

名称	类型	说明
url	字符串	运行中的 NICE 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	漂亮的 DCV 会话 ID。
authToken	字符串	连接到服务器时使用的身份验证令牌。
baseUrl	字符串	从中加载 SDK 文件的绝对或相对 URL。
resourceBaseUrl	字符串	用于访问 DCV 资源的绝对或相对 URL。
enabledChannels	数组。 <Channel (p. 21)>	表示可以启用的频道列表。如果未指定或提供空阵列，则默认为所有可用频道。

名称	类型	说明
losslessColorspace	ColorSpace (p. 23)	表示将使用的色彩空间。如果未指定，则默认为 “RGB”。
divId	字符串	的 IDdivHTML DOM 中的对象，其中 SDK 应该用远程流创建画布。
volumeLevel	integer	首选的音量级别。有效范围为 0 至 100。
clipboardAutoSync	布尔值	指示兼容的 Web 浏览器是否启用了从 NICE DCV 会话剪贴板到本地客户端剪贴板的自动复制功能。
dynamicAudioTuning	布尔值	指示建立连接时是否根据 NICE DCV 服务器音频设置动态调整音频。
clientHiDpiScaling	布尔值	指示是否根据客户的 DPI 缩放画布。
highColorAccuracy	布尔值	指示是否应使用高色彩准确度（如果可用）。如果未指定，则默认为false.
enableWebCodecs	布尔值	指示是否 WebCodecs 应使用（如果可用）。如果未指定，则默认为 false。
observers	连接回调 (p. 23)	用于调用与连接相关的事件的回调函数。
callbacks	连接回调 (p. 23)	与observers属性，但是每个回调都包括Connection (p. 38)对象作为第一个参数。

连接错误代码

这些区域有： ConnectionError DCV 模块中可用的代码枚举

- ALREADY_OPEN
- INVALID_CONFIG
- INITIALIZATION_FAILED
- REJECTED
- MAIN_CHANNEL_ALREADY_OPEN
- GENERIC_ERROR（自从 DCV 服务器 2021.0 以来）
- INTERNAL_SERVER_ERROR（自从 DCV 服务器 2021.0 以来）
- AUTHENTICATION_FAILED（自从 DCV 服务器 2021.0 以来）
- PROTOCOL_ERROR（自从 DCV 服务器 2021.0 以来）
- INVALID_SESSION_ID（自从 DCV 服务器 2021.0 以来）
- INVALID_CONNECTION_ID（自从 DCV 服务器 2021.0 以来）
- CONNECTION_LIMIT_REACHED（自从 DCV 服务器 2021.0 以来）

类型:

- number

创建目录（路径）

参数：

名称	类型	说明
path	字符串	我们想要创建目录的服务器上的绝对路径。它还应包含目标目录的名称。

自定义渠道错误代码

这些区域有： CustomChannelError DCV 模块中可用的代码枚举

- TRANSPORT_ERROR

类型:

- number

dataChannel 回调（信息）

NICE 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	类型	说明
info	对象	有关数据通道的信息。 名称 / 类型 / 说明 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。

Delete 文件（路径）

参数：

名称	类型	说明
path	字符串	服务器上标识我们要删除的文件的绝对路径。

设备更改事件回调 ()

当一个deviceChange触发事件。

断开回调（原因）

连接结束时调用的回调函数。

参数：

名称	类型	说明
reason	对象	断开连接的原因。 名称 | 类型 | 描述 code | number | 原因代码。 message | 字符串 | 原因消息。

显示可用性回调（状态，DisplayID）

显示器的可用性发生变化时要调用的回调函数。

参数：

名称	类型	说明
status	对象	显示器的状态。 名称 | 类型 | 说明 enabled | 布尔值 | 指示显示是否已启用。 closed | 布尔值 | 指示显示器是否已关闭。

名称	类型	说明
displayId	number	显示器的标识符。

DisplayConfig 错误代码

这些区域有： DisplayConfigError DCV 模块中可用的代码枚举

- INVALID_ARGUMENT
- UNSUPPORTED_OPERATION
- NO_CHANNEL

类型:

- number

DisplayLayout 回调（服务器宽度、服务器高度、头）

更改显示布局或分辨率时要调用的回调函数。

参数：

名称	类型	描述
serverWidth	number	主显示器的宽度 (像素)。
serverHeight	number	主显示器的高度 (像素)。
heads	数组。 <显示器 (p. 33)>	NICE DCV 服务器支持的显示头。

功能

功能值。

- display-表示单显示器视频流的可用性。
- display-multi-表示多显示器视频流的可用性。
- high-color-accuracy-表示色彩准确度高（自 NICE DCV Web 客户端 SDK 版本 1.1.0 以来）的可用性。
- mouse-表示鼠标功能的可用性。
- keyboard-表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-表示相对鼠标模式的可用性。
- clipboard-copy-表示从 NICE DCV 服务器到客户端的剪贴板复制功能的可用性。
- clipboard-paste-表示从客户端到 NICE DCV 服务器的剪贴板粘贴功能的可用性。
- audio-in-指示使用麦克风的音频输入功能的可用性。
- audio-out-表示音频播放功能的可用性。
- webcam-表示网络摄像头流媒体功能的可用性。
- file-download-表示从 NICE DCV 服务器到客户端的文件下载功能的可用性。
- file-upload-表示从客户端到 NICE DCV 服务器的文件上传功能的可用性。

类型:

- 字符串

功能更新回调（功能列表）

要在功能状态发生变化时调用的回调函数。

参数：

名称	类型	说明
`featuresList`	数组。 <功能 (p. 28)>	一系列已更改的功能。

文件下载回调（文件资源）

准备好从 NICE DCV 服务器下载文件时要调用的回调函数。

参数：

名称	类型	说明
`fileResource`	对象	有关可供下载的文件的信息。 名称 / 类型 / 说明 `id` / 字符串 / 文件的标识符。 `url` / 字符串 / 用于下载文件的 URL。 `domain` / 字符串 / 资源域。 `token` / 字符串 / 用于下载文件的身份验证令牌。该令牌还包含在 URL 中。

文件打印回调（PrintResource）

在 NICE DCV 服务器上打印文件时要调用的回调函数。

参数：

名称	类型	说明
`printResource`	对象	有关打印文件的信息。

名称	类型	说明
		名称 / 类型 / 说明 id / 字符串 / 打印文件的标识符。 url / 字符串 / 用于下载打印文件的 URL。 domain / 字符串 / 资源域。在本例中，printer. token / 字符串 / 用于下载打印文件的身份验证令牌。该令牌还包含在 URL 中。

文件存储

允许在文件系统上探索和执行操作的对象。

类型:

- 对象

属性：

名称	类型	说明
list	列表 (p. 33)	允许列出服务器上提供的路径上存在的项目（文件和目录）的函数。
createDirectory	创建目录 (p. 26)	允许在服务器上指定路径上创建目录的函数。
retrieveFile	检索文件 (p. 36)	允许在服务器上指定路径本地下载文件的函数。
deleteFile	删除文件 (p. 27)	允许删除服务器上指定路径处的文件的函数。
renameFile	重命名文件 (p. 35)	允许将文件从指定源路径重命名到指定目标路径的函数。

名称	类型	说明
`renameDirectory`	重命名目录 (p. 35)	允许将目录从指定源路径重命名为绝对目标路径的函数。
`storeFile`	存储文件 (p. 37)	允许将本地文件上传到服务器上提供的路径的函数。

启用文件存储的回调（已启用）

启用文件存储时要调用的回调函数。仅限 Internet Explorer 11 上的懒惰频道。

参数：

名称	类型	说明
`enabled`	布尔值	指示是否启用文件存储。

文件存储错误代码

这些区域有： FileStorageError DCV 模块中可用的代码枚举

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

类型:

- number

第一帧回调（已启用调整大小、已启用相对鼠标模式、DisplayID）

从 NICE DCV 服务器接收第一帧时要调用的回调函数。为每个显示器发出。

参数：

名称	类型	说明
`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。

名称	类型	说明
displayId	number	显示器的标识符。

IdleWarning 通知回调（断开连接日期时间）

NICE 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	类型	说明
disconnectionDateTime	日期	断开连接的日期和时间。

协作者列表回调（合作者）

NICE DCV 服务器发送协作者列表时要调用的回调函数。

参数：

名称	类型	说明
collaborators	数组。 <Object>	包含有关协作者的信息的对象列表。 名称 / 类型 / 说明 username / 字符串 / 合作者的用户名。 owner / 布尔值 / 指示协作者是否是会话所有者。 connectionId / number / 指示服务器分配给连接的 ID。

许可证通知回调（通知）

NICE 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	类型	说明
notification	对象	通知功能。

名称	类型	说明
		名称 / 类型 / 说明 product / 字符串 / DCV 产品。 status / 字符串 / 许可证的状态。 message / 字符串 / 消息。 leftDays / 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 / 未许可连接的数量。 licensingMode / 字符串 / 许可模式。 documentationUrl / 字符串 / 文档的 URL。

列表（路径）

参数：

名称	类型	说明
path	字符串	我们想要列出内容的服务器上的绝对路径。

日志级别

可用的 SDK 日志级别。

类型:

- TRACE | 调试 | 信息 | 警告 | 错误 | 静音

监控

类型:

- 对象

属性：

名称	类型	说明
name	字符串	显示头的名称。
rect	对象	有关显示头的信息。 名称 / 类型 / 描述 x / number / 首次x坐标显示头。 y / number / 首次y坐标显示头。 width / number / 显示头的宽度 (像素)。 height / number / 显示头的高度 (像素)。
primary	布尔值	指示显示头是否为主显示头。这取决于远程操作系统（如果可用）。
dpi	number	显示头的 DPI。

MultiMonitor 错误码

这些区域有： MultiMonitorError DCV 模块中可用的代码枚举

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

类型:

- number

质量指标状态回调（州）

连接质量指示器改变状态时要调用的回调函数。

参数：

名称	类型	说明
state	数组。 <Object>	有关连接质量的信息。 名称 | 类型 | 说明 name | 字符串 | 指标的名称。 status | 普通 | 警告 | 关键 | 状态的描述。 changed | 布尔值 | 指示状态是否更改。

重命名目录（src、dest）

参数：

名称	类型	说明
src	字符串	服务器上标识我们要重命名的目录的绝对源路径。
dest	字符串	服务器上指定目标路径和目录名称的绝对目标路径。

重命名文件（src、dest）

参数：

名称	类型	说明
src	字符串	服务器上标识我们要重命名的文件的绝对源路径。
dest	字符串	服务器上指定目标路径和文件名的绝对目标路径。

解决错误代码

这些区域有： ResolutionError DCV 模块中可用的代码枚举

- INVALID_ARGUMENT
- NO_CHANNEL

类型:

- number

检索文件（路径）

参数：

名称	类型	说明
`path`	字符串	服务器上标识我们要在本地下载的文件的绝对路径。

ScreenShot 回调（截图）

截图可用时要调用的回调函数。

参数：

名称	类型	说明
`screenshot`	byte[]	PNG 格式的屏幕截图缓冲区，或`null`如果屏幕截图检索失败。

屏幕截图恐怖代码

这些区域有： ScreenshotError DCV 模块中可用的代码枚举

- `NO_CHANNEL`
- `GENERIC_ERROR`

类型:

- number

ServerInfo

类型:

- 对象

属性：

名称	类型	说明
`name`	字符串	软件的名称。
`version`	对象	软件版本号。

名称	类型	说明
		名称 | 类型 | 描述 major | number | 主要版本号。 minor | number | 次要版本号。 revision | 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	类型	描述
fps	number	每秒当前帧数。
traffic	number	当前流量（以位/秒为单位）。
peakTraffic	number	自连接建立以来，流量的峰值（以位/秒为单位）。
latency	number	当前延迟（以毫秒为单位）。
currentChannels	number	自建立连接以来已打开的频道数。
openedChannels	number	当前打开的通道数量。
channelErrors	number	报告错误的频道数。

StoreFile（文件、目录）

参数：

名称	类型	说明
file	文件	我们想要上传到服务器的文件对象（有关更多信息，请参阅

名称	类型	说明
		https://developer.mozilla.org/en-US/docs/Web/API/File ）。
`dir`	字符串	我们要上传文件的服务器上的绝对路径。

WebCamera Rcode

这些区域有： WebcamError DCV 模块中可用的代码枚举

- `SETTING_WEBCAM_FAILED`
- `CHANNEL_NOT_AVAILABLE`

类型:

- number

连接类

通过调用`connect`方法 (p. 13)的`dcv`模块。有关展示如何使用它的示例，请参阅入门 (p. 4)部分。

Exise

- 方法 (p. 12)

方法

列表

- 附上显示（赢，DisplayConf ）→{承诺。 <number>| 承诺。 < {代码: MultiMonitor 错误码，消息:string} >} (p. 39)
- CaptureClipBoar 事件（已启用、赢、DisplayId ）→{void} (p. 40)
- 分离显示 (DisplayID) →{void} (p. 40)
- 断开 () →{void} (p. 40)
- 断开协作者（ConnecconnectionId ）→{void} (p. 41)
- 启用显示质量更新（启用 ）→{void} (p. 41)
- 进入相对鼠标模式 () →{void} (p. 41)
- 获取已连接的设备 () →{Promise。 < 数组。 <MediaDeviceInfo>>| 承诺。 < {message: string} >} (p. 42)
- getFileExplorer () →{Promise。 <文件存储>| 承诺。 < {代码: 渠道错误代码，消息:string} >} (p. 42)
- 获取 ServerInfo () →{ServerInfo} (p. 42)
- getScreen () →{Promise| 承诺。 < {代码: 屏幕截图恐怖代码，消息:string} >} (p. 42)
- getStats () →{统计数据} (p. 43)
- latchModifierKey（键、位置、iSDOWN ）→{布尔值} (p. 43)
- OpenChannel（姓名、AuthToken、回调 ）→{Promise| 承诺。 < {代码: 渠道错误代码，消息:string} >} (p. 43)
- 查询功能（功能重命名 ）→{Promise。 < {启用：布尔值，远程？：字符串，自动复制？ ：布尔值，自动粘贴？ ：布尔值，ServiceStatus？ ：字符串，可用？：布尔值} >| 承诺。 < {message: string} >} (p. 44)

附上显示（ 赢，DisplayConf ）→ {承诺。 <number>| 承诺。 < {代码: MultiMonitor 错误码 (p. 34)，消息:string} >}

将特定的显示器附加到窗口。你不能连接主显示器。如果成功，该函数将`displayId`.

参数：

名称	类型	说明
`win`	对象	显示器必须连接到的窗口。
`displayConf`	对象	显示器的配置。 名称 | 类型 | 属性 | 说明 `displayId` | number | <optional> | 显示器的 ID。 `displayDivName` | | | 显示 div 的名称。

返回值:

承诺。如果拒绝，承诺会返回错误对象。

类型

承诺。 <number>| 承诺。 < {代码: MultiMonitor 错误码 (p. 34)，消息:string} >

CaptureClipBoar 事件（已启用、赢、DisplayId）→ {void}

开始或停止监听复制粘贴事件。对于交互式剪贴板（总是在粘贴的情况下），我们需要开始监听复制/粘贴事件。只有在需要时（例如，在显示模式时）才开始和停止监听可能会有用。

参数：

名称	类型	属性	说明
`enabled`	布尔值		要开始收听事件，请指定`true`. 要停止监听事件，请指定`false`.
`win`	对象	<optional>	监听事件的窗口。如果省略，则会使用默认窗口。
`displayId`	number	<optional>	应监听事件的显示器的 ID。如果省略，则将使用默认显示窗口。

返回值:

类型

void

分离显示 (DisplayID) → {void}

分离特定的显示器。无法分离主显示器。

参数：

名称	类型	描述
`displayId`	number	要分离的显示器的 ID。

返回值:

类型

void

断开 () → {void}

断开与 NICE DCV 服务器的连接并关闭连接。

返回值:

类型

void

断开协作者（ConnecconnectionId ）→ {void}

请求断开与提供的连接 ID 连接的协作者的连接（自 NICE DCV Web 客户端 SDK 版本 1.1.0 起）。

参数：

名称	类型	说明
`connectionId`	布尔值	将断开连接的 ID。

返回值:

类型

void

启用显示质量更新（启用）→ {void}

为未接收更新的直播区域启用或禁用显示质量更新。禁用显示质量更新可减少带宽使用量，但也会降低显示质量。

参数：

名称	类型	说明
`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

进入相对鼠标模式 () → {void}

启用相对鼠标模式。

返回值:

类型

void

获取已连接的设备 () → {Promise。 < 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 Promise，该 Promise 解析为 MediaDeviceInfo 对象。有关更多信息，请参阅 https://developer.mozilla.org/en-US/docs/Web/API/MediaDeviceInfo。如果拒绝，承诺会返回错误对象。

类型

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

getFileExplorer () → {Promise。 <文件存储 (p. 30)>| 承诺。 < {代码: 渠道错误代码 (p. 21)，消息:string} >}

获取用于管理 NICE DCV 服务器的文件存储的对象。

返回值:

承诺。如果已完成，则解析文件资源管理器对象；如果拒绝，则解析错误对象。

类型

承诺。 <文件存储 (p. 30)> | 承诺。 < {代码: 渠道错误代码 (p. 21)，消息:string} >

获取 ServerInfo () → {ServerInfo (p. 36)}

获取有关 NICE DCV 服务器的信息。

返回值:

有关服务器软件的信息。

类型

ServerInfo (p. 36)

getScreen () → {Promise| 承诺。 < {代码: 屏幕截图恐怖代码 (p. 36)，消息:string} >}

以 PNG 格式检索远程桌面的屏幕截图。屏幕截图将在屏幕截图回调 (p. 36)观察员。`null`如果失败，将返回。

返回值:

承诺是否处理了请求。如果被拒绝，我们会收到错误对象。

类型

承诺 | 承诺。 < {代码: 屏幕截图恐怖代码 (p. 36)，消息:string} >

getStats () → {统计数据 (p. 37)}

获取有关 NICE DCV 服务器的统计信息。

返回值:

有关流式传输统计信息。

类型

统计数据 (p. 37)

latchModifierKey (键、位置、iSDOWN) → {布尔值}

发送单个键盘keydown要么keyup允许的修饰符的事件。

参数：

名称	类型	说明
key	控制 | Alt | AltGraph | 元 | 操作系统 | Shift	要发送的钥匙。
location	键盘事件。位置	钥匙的位置。有关更多信息，请参阅 https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/location。
isDown	布尔值	如果要注入的关键事件是 keydown (true) 或者一个 keyup (false)。

返回值:

如果请求的组合有效，该函数返回。true，否则它会返回false.

类型

布尔值

OpenChannel (姓名、AuthToken、回调) → {Promise| 承诺。 < {代码: 渠道错误代码 (p. 21)，消息:string} >}

如果在 NICE DCV 服务器上创建了自定义数据通道，则在连接上打开它。

参数：

名称	类型	说明
name	字符串	通道的名称。
authToken	字符串	用于连接频道的身份验证令牌。

名称	类型	说明
callbacks	对象	onMessage 和 onClose 回调函数要调用。

返回值:

承诺。如果被拒绝，我们会收到错误对象。

类型

承诺 | 承诺。 < {代码: 渠道错误代码 (p. 21)，消息:string} >

查询功能（功能重命名）→ {Promise。 < {启用：布尔值，远程？：字符串，自动复制？ ：布尔值，自动粘贴？ ：布尔值，ServiceStatus？ ：字符串，可用？：布尔值} >| 承诺。 < {message: string} >}

查询特定 NICE DCV 服务器功能的状态。

参数：

名称	类型	说明
featureName	功能 (p. 28)	要查询的功能的名称。

返回值:

承诺。如果解决，该函数将status始终包含enabled财产，可能还有其他财产。如果拒绝，该函数将error对象。

类型

{Promise。 < {启用：布尔值，远程？：字符串，自动复制？ ：布尔值，自动粘贴？ ：布尔值，ServiceStatus？ ：字符串，可用？：布尔值} > | 承诺。 < {message: string} >

注册键盘快捷键（快捷键）→ {void}

注册键盘快捷键。

参数：

名称	类型	说明
shortcuts	数组。 <Object>	要注册的密钥和映射的数组。 名称 / 类型 / 说明 sequence / 数组。 <Object> / 要注册的键盘快捷键。

名称	类型	说明
		名称 / 类型 / 说明 key — KeyboardEvent.key — 用户按下的键的值。有关更多信息，请参阅 https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key。 location — 键盘事件位置 — 要发送的密钥数组。键在键盘上的位置。有关更多信息，请

<table>
<tr><th>名称</th><th>类型</th><th>说明</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>名称</th><th>类型</th><th>说明</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>名称</th><th>类型</th><th>说明</th></tr>
<tr><td></td><td></td><td>参阅 https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/location。</td></tr>
</table>
</td></tr>
<tr><td><code>output</code></td><td>数组。<Object></td><td>快捷方式要执行的预期操作。
<table>
<tr><th>名称</th><th>类型</th><th>说明</th></tr>
<tr><td><code>key</code></td><td>KeyboardEvent.key</td><td>用户按下的键的值。有关更多信息，请参阅 https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key。</td></tr>
<tr><td><code>location</code></td><td>键盘事</td><td>要发送on</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

名称	类型	说明
		名称 类型 说明 名称 类型 说明 的密钥数组。键在键盘上的位置。有关更多信息，请参阅 https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/location。 件位置

返回值:

类型

void

请求 DisplayConfig（高色彩准确度）→ {Promise| 承诺。 < {代码: DisplayConfig 错误代码 (p. 28)，消息:string} >}

从 NICE DCV 服务器请求更新的显示配置。自 NICE DCV Web 客户端 SDK 1.1.0 和 NICE DCV 服务器 2022.0 以来可用。

参数：

名称	类型	说明
highColorAccuracy	布尔值	是否应该要求高的色彩准确度。

返回值:

承诺。如果拒绝，承诺会返回错误对象。

类型

承诺 | 承诺。 < {代码: DisplayConfig 错误代码 (p. 28)，消息:string} >

请求显示布局（布局）→ {Promise| 承诺。 < {代码: 解决错误代码 (p. 35)，消息:string} >}

请求连接的更新显示布局。

参数：

名称	类型	说明
layout	数组。 <显示器 (p. 33)>	请求的布局将显示在布局中。

返回值:

承诺。如果被拒绝，我们会收到错误对象。

类型

承诺 | 承诺。 < {代码: 解决错误代码 (p. 35)，消息:string} >

请求解决方案 (宽度、高度) → {Promise | Promise。 < {代码: 解决错误代码 (p. 35)，消息:string} >}

从 NICE DCV 服务器请求更新的显示分辨率。

参数：

名称	类型	描述
width	number	要请求的宽度，以像素为单位。允许的最小值为0.
height	number	请求的高度（以像素为单位）。允许的最小值为0.

返回值:

承诺。如果拒绝，承诺会返回错误对象。

类型

承诺 | 承诺。 < {代码: 解决错误代码 (p. 35)，消息:string} >

sendkeyBoarEvent（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	类型	说明
`event`	键盘/Event	要发送的键盘事件。

返回值:

如果事件无效，函数返回`false`. 如果事件有效，该函数返回。`true`.

类型

布尔值

SendKeyboard 快捷方式（快捷方式）→ {void}

发送键盘快捷键。使用此函数发送完整`keydown`要么`keyup`序列。例如，发送 Ctrl + Del 将`keydown`所有钥匙的事件，然后是`keyup`事件. 即使你想发送单个密钥，也可以使用此功能。

参数：

名称	类型	说明
`shortcut`	数组。 <Object>	要发送的密钥数组。 名称 | 类型 | 说明 `key` | KeyboardEvent.key | 用户按下的键的值。有关更多信息，请参阅 https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key。

名称	类型	说明
		名称 / 类型 / 说明 location / 键盘事件。位置 / 要发送的密钥数组。键在键盘上的位置。有关更多信息，请参阅 https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/location。

返回值:

类型

void

设置显示质量（min，maxopt）→ {void}

设置用于连接的图像质量。有效范围为0到100，与1是最低的图像质量100是最高的图像质量。指定0保留当前值。

参数：

名称	类型	属性	说明
min	number		最低图像质量。
max	number	<optional>	最高的图像质量。

返回值:

类型

void

设置 DisplayScale（缩放比例、DisplayId）→ {Promise| 承诺。 <{代码: 解决错误代码 (p. 35)，消息:string} >}

通知 NICE DCV 显示屏在客户端缩放。使用此功能可以通知服务器它需要缩放鼠标事件以匹配客户端的显示比率。

参数：

名称	类型	说明
`scaleRatio`	float	要使用的缩放比率。必须是严格的正数。
`displayId`	number	要缩放的显示器的 ID。

返回值:

承诺。如果拒绝，承诺会返回错误对象。

类型

承诺 | 承诺。 < {代码: 解决错误代码 (p. 35)，消息:string} >

setkeyboard Quirks（怪癖）→ {void}

为客户端计算机设置键盘怪癖。

参数：

名称	类型	说明
`quirks`	对象	要启用或禁用的键盘怪癖。 名称 | 类型 | 说明 `macOptionKeyIsAlt` | 布尔值 | 要将 Option 键映射到 macOS 的 Alt 键，请指定`true`. 否则，请指定`false`。 `macCommandKeyIsCtrl` | 布尔值 | 要将 Command 键映射到 macOS 的 Ctrl 键，请指定`true`. 否则，请指定`false`。

返回值:

类型

void

设置最大显示分辨率（最大宽度、最大高度）→ {void}

设置用于连接的最大显示分辨率。

参数：

名称	类型	描述
`maxWidth`	number	以像素为单位的最大显示宽度。允许的最小值为`0`.
`maxHeight`	number	以像素为单位的最大显示高度。允许的最小值为`0`.

返回值:

类型

void

Set 麦克风（启用）→ {Promise| 承诺。 < {代码: 音频错误码 (p. 18)，消息:string} >}

启用或禁用麦克风。

参数：

名称	类型	说明
`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:

承诺。如果拒绝，承诺会返回错误对象。

类型

承诺 | 承诺。 < {代码: 音频错误码 (p. 18)，消息:string} >

setminDisplayDisplay 分辨率（minWidth，minHeight）→ {void}

设置用于连接的最小显示分辨率。某些应用程序可能需要最低显示分辨率。如果所需的最低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。请仔细使用此功能。调整大小策略可能会导致鼠标和触摸输入系统的精确度较低。

参数：

名称	类型	描述
`minWidth`	number	以像素为单位的最小显示宽度。允许的最小值为0.
`minHeight`	number	以像素为单位的最小显示高度。允许的最小值为0.

返回值:

类型

void

setUpload带宽（值）→ {数字}

设置用于将文件上传到 NICE DCV 服务器的最大带宽。

参数：

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

返回值:

-设定的带宽限制。`null`是否在服务器上禁用了文件存储功能。

类型

number

setVolume（卷）→ {void}

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

参数：

名称	类型	描述
`volume`	number	要使用的卷级别。

返回值:

类型

void

setWebcam（启用，设备 ID）→ {Promise| 承诺。 < {代码: WebCamera Rcode (p. 38)，消息:string} >}

启用或禁用网络摄像头。

参数：

名称	类型	说明
`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，将解析为已连接/分离的网络摄像头设备 Id。如果拒绝，承诺会返回错误对象。

类型

承诺 | 承诺。 < {代码: WebCamera Rcode (p. 38)，消息:string} >

syncClipBoards () → {布尔值}

将本地客户端剪贴板与远程 NICE 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，则函数返回`true`. 如果剪贴板尚未被 sycnhronized，或者如果浏览器不支持自动复制，则该函数将返回`false`.

类型

布尔值

身份验证类

必须使用身份验证类来获取身份验证令牌，方法是调用`authenticate`方法 (p. 12)的`dcv`模块。有关展示如何使用它的示例，请参阅入门 (p. 4)部分。

Exise

- 方法 (p. 12)

方法

列表

- 重试 () → {void} (p. 55)
- Send凭据（凭据）→ {void} (p. 55)

重试 () → {void}

重试身份验证过程。

返回值:

类型

void

Send凭据（凭据）→ {void}

将客户端提供的身份验证凭据发送到 NICE DCV 服务器。

参数：

名称	类型	说明
`credentials`	对象	包含提供的凭据的对象。凭证必须具有相同的名称，且类型必须与质询中指定的类型相同。

返回值:

类型

void

NICE DCV Web UI SDK

一个 JavaScript React 组件库，目前导出了一个名为`DCVViewer`它连接到 NICE DCV 服务器并渲染工具栏以与远程流进行交互。

Exise

- 组件 (p. 55)

组件

列表

- dcvViewer (p. 55)

dcvViewer

React 组件呈现工具栏的所有功能对于与远程流进行交互都很有用。

属性：

列表

- DCV (p. 56)
- uiconFfig (p. 58)

DCV

名称	类型	必需	说明
dcv	对象	是	定义建立与 NICE DCV 服务器的连接所需属性的对象，设置日志级别和 URL 加载 NICE DCV Web Client SDK 资产并访问 DCV 资源的位置。 名称 / 类型 / 必需 / 描述 sessionId / 字符串 / 是 / 漂亮的 DCV 会话 ID。 authToken / 字符串 / 是 / 连接到服务器时使用的身份验证令牌。 serverUrl / 字符串 / 是 / 正在运行的 NICE DCV 服务器的主机名和端口

名称	类型	必需	说明
			名称 / 类型 / 必需 / 描述 格式如下：https://dcv_host_address:p 例如：https://my-dcv-server:8443。 baseUrl — 字符串 — 是 — 从中加载 SDK 文件的绝对或相对 URL。 resourceBaseUrl — 字符串 — 否（默认值："") — 用于访问 DCV 资源的绝对或相对 URL。 onDisconnect — 函数 — 否（默认值：() => {}） — 断开与 NICE DCV 服务器的连

名称	类型	必需	说明
			名称 / 类型 / 必需 / 描述 接时调用的回调函数，连接已关闭。 logLevel / 日志级别 (p. 38) / 不（默认值）: LogLevel.INFO) / 在查看器中使用的日志级别。

uiconFfig

名称	类型	必需	说明
uiConfig	对象	不（默认值：{}）	定义属性的对象，以配置工具栏是否可见以及是否在其上显示全屏和多显示器按钮。 名称 / 类型 / 必需 / 说明 toolbar / 对象 / 不（默认值：{}） / 定义工具栏配置选项的

名称	类型	必需	说明
			名称 / 类型 / 必需 / 说明 对象。 描述 viewOnly 布尔值 用于定义是显示还是隐藏工具栏的选项。 默认值：true fullscreenButton 布尔值 用于定义是显示还是隐藏工具栏上的全屏按钮的选项。 默认值：true multimonitorBu 布尔值 用于定义 默认值：true

名称	类型	必需	说明
			名称 / 类型 / 必需 / 说明 描述是显示还是隐藏工具栏上的多显示器按钮的选项。

NICE DCV Web 客户端开发工具包的版本说明和文档历史记

本页面提供了 NICE DCV Web 客户端开发工具包的版本说明和文档历史记录。

主题

NICE DCV Web 客户端 SDK 发行说明

本节提供了按发布日期为 NICE DCV Web 客户端 SDK 的发行说明。

主题

2022 年 5 月 23 日 —

版本	发布说明
• 语义版本：1.1.3 • 构建：329	更改和错误修复 • 修复了在指定 web-url-path 选项。

2022 年 5 月 19 日

版本	发布说明
• 语义版本：1.1.2 • 构建：322	更改和错误修复 • 修复了连接后可能导致输入无法正常工作的问题。 • 修复了比例比率大于 1 时的鼠标坐标。

2022 年 3 月 23 日 —

版本	发布说明
• 语义版本：1.1.1 • 构建：309	更改和错误修复 • 报告`Transport Error`当与服务器的通信超时时。 • 修复了流式传输大分辨率时反复出现的解码错误。

2022 年 2 月 23 日

版本	发布说明
• 语义版本：1.1.0 • 构建：295	新 功能 • 发布 NICE DCV Web UI SDK 库`DCVViewer`React 组件。 • 将 NICE DCV Web 客户端 SDK 同时导出为 UMD 和 ES 模块。 • 添加了高色彩准确度支持。 • 添加了列出与会话连接的客户端并与之交互的功能。添加了连接和断开连接的通知。 更改和错误修复 • 改进了 webcodecs 解码支持。 • 各种键盘改进。 • 修复禁用剪贴板时无法打开第二个屏幕的错误。

2021 年 12 月 20 日 -12 月 20 日

版本	发布说明
• 语义版本：1.0.4 • 构建：249	新 功能 • Support 从同一页面打开多个连接。 • Support 从 CDN 加载 SDK。

2021 年 9 月 1 日 1.0.3

版本	发布说明
• 语义版本：1.0.3 • 构建：202	新 功能

版本	发布说明
	• 对 WebCoDecs 的实验性支持。这在默认情况下处于禁用状态，必须通过`ConnectionConfig`使用新属性的对象`enableWebCodecs`. • 剪贴板：增加了对的支持`image/png`基于 Chromium 的浏览器上的数据类型。 • 添加了观察员/回调功能以 PNG 图像的形式获取服务器的屏幕截图（需要 NICE DCV 服务器 2021.2）。 更改和错误修复 • 改进了键盘修饰符的处理。

1.0.2 — 2021 年 7 月 30 日

版本	发布说明
• 语义版本：1.0.2 • 构建：167	• 修复了手写笔事件的压力检测。 • 改进了 Chrome 上对韩语键盘布局的支持。

2021 年 5 月 31 日 —

版本	发布说明
• 语义版本：1.0.1 • 构建：141	• 修复了连接错误和关闭原因的传播 • 修复了文件存储块进度更新 • 改进了网络摄像 • 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 • 构建：81	NICE DCV Web 客户端 SDK 的初始版本。

文档历史记录

下表介绍此版本的 NICE DCV Web 客户端开发工具包的文档。

更改	说明	日期
Nice DCV Web 客户端开发工具包 1.1.0 版本	Nice DCV Web 客户端开发工具包 1.1.0 现已推出。有关更	2022 年 2 月 23 日

更改	说明	日期
	多信息，请参阅 。开发工具包 v.1.1.0 (p. 62).	
NICE DCV Web 客户端 SDK 版本 1.0.1	修复了一些拼写错误。修复了小错误，请参阅开发工具包 v.1.0.1 (p. 63).	2021 年 5 月 31 日
首次发布	此内容的第一版。	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。