



ウェブクライアント SDK デベロッパーガイド

NICE DCV



NICE DCV: ウェブクライアント SDK デベロッパーガイド

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標とトレードドレスは、Amazon 以外の製品またはサービスとの関連において、顧客に混乱を招いたり、Amazon の名誉または信用を毀損するような方法で使用することはできません。Amazon が所有していない他のすべての商標は、それぞれの所有者の所有物であり、Amazon と提携、接続、または後援されている場合とされていない場合があります。

Table of Contents

NICE DCV ウェブクライアント SDK とは	1
前提条件	1
サポートされている機能	2
ブラウザのサポート	2
バージョンニング規則	3
はじめに	4
NICE DCV サーバーに接続して最初のフレームを取得する	5
ステップ 1: HTML ページを準備する	5
ステップ 2: 最初のフレームの認証、接続、取得を行う	6
ボーナス: HTML ログインフォームを自動的に作成する	9
NICE DCV 機能の操作	10
FeatureUpdate コールバック関数について	10
機能更新の処理	11
NICE DCV ウェブ UI SDK の使用	11
前提条件	12
ステップ 1: HTML ページを準備する	13
ステップ 2: DCVViewer React コンポーネントの認証、接続、レンダリングを行う	13
AWS-UI から Cloudscape Design System へのアップデート	17
SDK リファレンス	19
DCV モジュール	19
方法	19
メンバー	22
データ型とコールバックの定義	26
接続クラス	65
方法	19
認証クラス	95
方法	19
リソースクラス	96
方法	19
NICE DCV ウェブ UI SDK	97
コンポーネント	98
リリースノートとドキュメント履歴	108
リリースノート	108
1.5.6 — 2023 年 11 月 9 日	109

1.4.4 — 2022 年 6 月 29 日	109
1.4.0 — 2023 年 3 月 28 日	110
1.3.1 — 2022 年 12 月 9 日	111
1.3.0 — 2022 年 11 月 11 日	112
1.2.1 — 2022 年 7 月 21 日	113
1.2.0 — 2022 年 6 月 29 日	113
1.1.3 — 2022 年 5 月 23 日	114
1.1.2 — 2022 年 5 月 19 日	114
1.1.1 — 2022 年 3 月 23 日	114
1.1.0 — 2022 年 2 月 23 日	115
1.0.4 — 2021 年 12 月 20 日	116
1.0.3 — 2021 年 9 月 1 日	116
1.0.2 — 2021 年 7 月 30 日	117
1.0.1 — 2021 年 5 月 31 日	117
1.0.0 — 2021 年 3 月 24 日	118
ドキュメント履歴	118
.....	CXX

NICE DCV ウェブクライアント SDK とは

NICE DCV は、高性能のリモートディスプレイプロトコルです。さまざまなネットワーク条件で、リモートデスクトップやアプリケーションストリーミングをクラウドやデータセンターからあらゆるデバイスへ安全に配信できます。NICE DCV と Amazon EC2 を使用すると、グラフィックスを多用するアプリケーションを Amazon EC2 インスタンス上でリモートで実行できます。結果をより控えめなクライアントマシンにストリーミングできるため、高価な専用ワークステーションが不要になります。

NICE DCV ウェブクライアント SDK とは、独自の NICE DCV ウェブブラウザクライアントアプリケーションの開発に使用できる JavaScript ライブラリです。エンドユーザーは、これらのアプリケーションを使用して、実行中の NICE DCV セッションへの接続や操作を行うことができます。

NICE DCV ウェブクライアント SDK を構築ブロックとして使用することで、ユーザーがどこからでもデスクトップやアプリケーションに瞬時にアクセスできるカスタマイズされたウェブアプリケーションを構築できます。これにより、ネイティブにインストールされたアプリケーションとほぼ同等の応答性と流動パフォーマンスを実現できます。

このガイドでは、NICE DCV ウェブクライアント SDK を使用して、カスタムのウェブブラウザクライアントアプリケーションを構築し、ワークフロー内で NICE DCV セッションを操作する方法について説明します。

トピック

- [前提条件](#)
- [サポートされている機能](#)
- [ブラウザのサポート](#)
- [バージョンニング規則](#)

前提条件

NICE DCV ウェブクライアント SDK の使用を開始する前に、NICE DCV と NICE DCV セッションについて理解しておいてください。詳細については、「[NICE DCV 管理者ガイド](#)」を参照してください。

NICE DCV ウェブクライアント SDK では、NICE DCV サーバーバージョン 2020 以降がサポートされています。

サポートされている機能

次の NICE DCV 機能がサポートされているカスタムのウェブブラウザクライアントアプリケーションを構築できます。

- Windows NICE DCV サーバーへの接続
- Linux NICE DCV サーバーへの接続
- ストリーミングモードの管理
- ファイルの転送
- セッションから印刷
- コピーアンドペースト
- ステレオ 2.0 オーディオ再生
- ステレオ 2.0 オーディオ録音 (Windows サーバーの場合)
- タッチスクリーン
- スタイラス (Linux、Windows 10、Windows Server 2019 の各サーバー)
- マルチモニターをサポート

これらの機能の詳細については、「NICE DCV ユーザーガイド」の「[サポートされている機能](#)」を参照してください。

ブラウザのサポート

NICE DCV ウェブクライアント SDK は JavaScript (ES6) に対応しており、JavaScript や TypeScript アプリケーションから利用できます。

NICE DCV ウェブクライアント SDK では、次のウェブブラウザがサポートされています。

ブラウザ	バージョン
Google Chrome	最新の 3 つのメジャーバージョン
Mozilla Firefox	最新の 3 つのメジャーバージョン
Microsoft Edge	最新の 3 つのメジャーバージョン

ブラウザ	バージョン
MacOS 版 Apple Safari	最新の 3 つのメジャーバージョン

バージョンニング規則

NICE DCV ウェブクライアント SDK のバージョンは、*major.minor.patch* の形式で定義されています。バージョンニング規則は、通常、[セマンティックバージョンニングモデル](#)に準拠しています。メジャーバージョンの変更 (1.x.x から 2.x.x への変更など) は、互換性を破る変更が導入され、それによってコードの変更や計画されたデプロイが必要になる可能性があることを示します。マイナーバージョンの変更 (1.1.x から 1.2.x への変更など) については、下位互換性がありますが、非推奨の要素が含まれている可能性があります。

NICE DCV ウェブクライアント SDK の開始方法

NICE DCV アーカイブクライアント SDK は、メインの `dcv.js` ファイルといくつかの補助コンポーネントで構成されます。すべてのファイルは、[NICE ウェブサイト](#) からダウンロードできる圧縮アーカイブ内に配布されます。

NICE DCV ウェブクライアント SDK を開始するには

1. NICE DCV ウェブクライアント SDK アーカイブは、安全な GPG 署名によりデジタル署名が施されています。アーカイブの署名を検証するには、NICE GPG キーをインポートする必要があります。そのためには、ターミナルウィンドウを開いて NICE GPG キーをインポートします。

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

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

2. NICE DCV ウェブクライアント SDK アーカイブと NICE DCV ウェブクライアント SDK アーカイブ署名を [NICE ウェブサイト](#) からダウンロードします。
3. 署名を使用して NICE DCV ウェブクライアント SDK アーカイブの署名を検証します。

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

例:

```
$ gpg --verify nice-dcv-web-client-sdk-1.5.6-659.zip.sign nice-dcv-web-client-  
sdk-1.5.6-659.zip
```

4. 署名が正常に検証された場合は、NICE DCV ウェブクライアント SDK アーカイブの内容を抽出し、抽出したディレクトリをウェブサーバーに配置します。例:

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



```
/  
server_directory  
/
```

⚠ Important

- NICE DCV ウェブクライアント SDK をウェブサーバーにデプロイするときは、フォルダ構造を維持する必要があります。
- NICE DCV ウェブ UI SDK を使用する場合、DCVViewer React コンポーネントは、このパッケージの EULA.txt ファイルと third-party-licenses.txt ファイルが埋め込みのウェブサーバーの URL パスに存在することを想定しています。third-party-licenses.txt ファイルを変更して、NICE DCV ウェブクライアント SDK パッケージの対応するファイルの内容と、場合によっては利用するユーザーアプリケーションが使用するライブラリからのその他のライセンス情報も含める必要があります。

NICE DCV サーバーに接続して最初のフレームを取得する

次のチュートリアルでは、カスタムウェブクライアント用の HTML ページを準備する方法、NICE DCV サーバーを認証して接続する方法、および NICE DCV セッションからストリーミングコンテンツの最初のフレームを受信する方法について説明します。

トピック

- [ステップ 1: HTML ページを準備する](#)
- [ステップ 2: 最初のフレームの認証、接続、取得を行う](#)
- [ボーナス: HTML ログインフォームを自動的に作成する](#)

ステップ 1: HTML ページを準備する

ウェブページで、必要な JavaScript モジュールをロードする必要があり、さらに、有効な id がある <div> HTML 要素を、NICE DCV ウェブクライアント SDK で NICE DCV サーバーからコンテンツストリームを描画させる場所に追加する必要があります。

例:

```
<!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>
```

ステップ 2: 最初のフレームの認証、接続、取得を行う

このセクションでは、ユーザー認証プロセスを完了する方法、NICE DCV サーバーを接続する方法、および NICE DCV サーバーからコンテンツの最初のフレームを受信する方法について説明します。

まず、`index.js` ファイルから NICE DCV ウェブクライアント SDK をインポートします。以下のようにユニバーサルモジュール定義 (UMD) モジュールとしてインポートできます。

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

または、バージョン 1.1.0 以降の場合、以下のように対応するパッケージから ECMAScript モジュール (ESM) としてインポートすることもできます。

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

認証オブジェクト、接続オブジェクト、および NICE DCV サーバー URL の保存に使用する変数を定義します。

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

スクリプトのロード時に、NICE DCV ウェブクライアント SDK のバージョンをログに記録し、ページのロード時に `main` 関数を呼び出します。

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

main 関数により、ログレベルの設定と認証プロセスの開始が実行されます。

```
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 関数、error 関数、success 関数は、認証プロセスで定義する必要がある必須コールバック関数です。

NICE DCV サーバーから認証情報が要求された場合、promptCredentials コールバック関数に、NICE DCV サーバーから要求された認証情報チャレンジが送られます。システム認証が使用されるように NICE DCV サーバーが設定されている場合、認証情報を提供する必要があります。次のコードサンプルでは、仮に、ユーザー名を my_dcv_user、パスワード my_password にしています。

認証に失敗した場合、error コールバック関数に NICE DCV サーバーからエラーオブジェクトが送られます。

認証が成功すると、success コールバック関数に、my_dcv_user ユーザーが NICE DCV サーバーに接続できる各セッションのセッション ID (sessionId) と認証トークン (authToken) が含まれているカップルの配列が送られます。次のコードサンプルでは、接続関数を呼び出して、配列で返された最初のセッションに接続します。

Note

以下のコード例では、MY_DCV_USER を自分のユーザー名に、MY_PASSWORD を自分のパスワードに置き換えてください。

```
function onPromptCredentials(auth, challenge) {
  // Let's check if in challenge 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);
}
```

Linux NICE DCV サーバーに接続します。firstFrame コールバック関数は、NICE DCV サーバーから最初のフレームが届いたときに呼び出されます。

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

ボーナス: HTML ログインフォームを自動的に作成する

challenge オブジェクトは `promptCredentials` コールバック関数が呼び出されたときに返されます。これには、オブジェクトの配列である `requiredCredentials` という名前のプロパティが含まれています。NICE DCV サーバーからリクエストされる認証情報 1 件につきオブジェクトは 1 つです。各オブジェクトには、リクエストされた認証情報の名前とタイプが含まれます。challenge オブジェクトと `requiredCredentials` オブジェクトを使用して、HTML ログインフォームを自動的に作成することができます。

次のコードサンプルでは、その実行方法を説明しています。

```
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));
}
```

NICE DCV 機能の操作

NICE DCV 機能の可用性は、NICE DCV セッションに対して設定されたアクセス許可と、クライアントのウェブブラウザの機能に左右されます。

NICE DCV セッションで使用できる機能は、そのセッションに対して指定されたアクセス許可によって管理されます。機能が NICE DCV ウェブクライアント SDK でサポートされていても、セッション管理者が定義したアクセス許可に基づいて、その機能へのアクセスが妨げられることがあります。詳細については、「NICE DCV 管理者ガイド」の「[NICE DCV 認可の設定](#)」を参照してください。

FeatureUpdate コールバック関数について

NICE DCV セッションの機能の可用性が変わると、接続の確立時に指定した `featuresUpdate` コールバック関数により、NICE DCV ウェブクライアント SDK から通知を受けます。例:

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

コールバック関数は、可用性が変化した機能のみを通知します。list パラメータは一連の文字列で、更新された機能の名前のみが含まれます。例えば、セッションでオーディオ入力機能の可用性が変化した場合、パラメータには ["audio-in"] のみが含まれます。後に、セッションでクリッ

プボードのコピー/貼り付け機能の可用性が変わった場合は、パラメータに ["clipboard-copy", "clipboard-paste"] のみが含まれます。

機能更新の処理

featuresUpdate コールバック関数は、可用性が変化した機能のみを通知します。どの機能が更新されたかを知るには、connection.queryFeature メソッドを使用して機能のクエリを行う必要があります。これは、変更通知の受信後にいつでも実行できます。このメソッドでは、解決を行う Promise が、リクエストされた機能の更新ステータスに返されます。status 値は常に関連付けられ、enabled というブール値 (true|false) プロパティを含みます。場合によっては、status 値に追加のプロパティがある機能もあります。機能の可用性が更新されていない場合は、拒否されません。

次のコード例では、その実行方法を説明しています。

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

NICE DCV ウェブ UI SDK の使用

以下のチュートリアルでは、NICE DCV サーバーに対して認証を行い、接続して NICE DCV ウェブ UI SDK から DCVViewer React コンポーネントをレンダリングする方法を説明します。

トピック

- [前提条件](#)
- [ステップ 1: HTML ページを準備する](#)
- [ステップ 2: DCVViewer React コンポーネントの認証、接続、レンダリングを行う](#)
- [AWS-UI から Cloudscape Design System へのアップデート](#)

前提条件

React、ReactDOM、Cloudscape Design Components React、Cloudscape Design Global Styles、Cloudscape Design Design Tokens をインストールする必要があります。

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

また、NICE DCV Web Client SDK のダウンロードも必要になります。その手順については、「[NICE DCV ウェブクライアント SDK の開始方法](#)」のステップごとのガイドをご覧ください。

NICE DCV ウェブ UI SDK の外部依存関係であるため、dcv モジュールをインポートするためのエイリアスを作成する必要があります。例えば、webpack を使用してウェブアプリケーションをバンドルする場合、以下のように [resolve.alias](#) オプションを使用できます。

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

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

バンドルにロールアップを使用している場合は、[@rollup /plugin-alias](#) をインストールして以下のように使用できます。

```
import alias from '@rollup/plugin-alias';
const path = require('path');

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



```
};
```

ステップ 1: HTML ページを準備する

ウェブページには、必要な JavaScript モジュールをロードする必要があります。また、アプリケーションのエントリコンポーネントがレンダリングされる、有効な id を持つ <div> HTML 要素が必要です。

例:

```
<!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>
```

ステップ 2: DCVViewer React コンポーネントの認証、接続、レンダリングを行う

このセクションでは、ユーザー認証プロセスを完了する方法、NICE DCV サーバーを接続する方法、DCVViewer React コンポーネントをレンダリングする方法を説明します。

まず、index.js ファイルから React、ReactDOM、および最上位の App コンポーネントをインポートします。

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

アプリケーションの最上位のコンテナノードをレンダリングします。

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

```
);
```

App.js ファイルで、ESM モジュールとしての NICE DCV ウェブクライアント SDK、NICE DCV ウェブ UI SDK の DCVViewer React コンポーネント、React および Cloudscape Design Global Styles パッケージをインポートします。

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

以下は、認証が成功した場合に NICE DCV サーバーに対して認証を行い、NICE DCV ウェブ UI SDK から DCVViewer React コンポーネントをレンダリングする方法の例です。

```
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>
</div>
```

```
        </button>
      </form>
    </div>
  );
}

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

export default App;
```

`promptCredentials` 関数、`error` 関数、`success` 関数は必須コールバック関数であり、認証プロセスで定義する必要があります。

NICE DCV サーバーから認証情報が要求された場合、`promptCredentials` コールバック関数に、NICE DCV サーバーから要求された認証情報チャレンジが送られます。システム認証が使用されるように NICE DCV サーバーが設定されている場合、ユーザー名とパスワードの形式で認証情報を提供する必要があります。

認証に失敗した場合、`error` コールバック関数に NICE DCV サーバーからエラーオブジェクトが送られます。

認証が成功すると、`success` コールバック関数に、ユーザーが NICE DCV サーバーに接続できる各セッションのセッション ID (`sessionId`) と認証トークン (`authToken`) が含まれるカップルの配列が送られます。上記のコードサンプルは、認証に成功すると React ステータスを更新し、`DCVViewer` コンポーネントをレンダリングします。

このコンポーネントで承認されるプロパティの詳細については、「[NICE DCV ウェブ UI SDK のリファレンス](#)」をご覧ください。

自己署名証明書の詳細については、「[自己署名証明書によるリダイレクトの説明](#)」をご覧ください。

AWS-UI から Cloudscape Design System へのアップデート

SDK バージョン 1.3.0 以降、`DCVViewer` コンポーネントを AWS-UI から進化した [Cloudscape Design](#) にアップデートしました。

Cloudscape は AWS-UI とは異なるビジュアルテーマを使用していますが、基盤となるコードベースに変わりはありません。したがって、`DCVViewer` ベースのアプリケーションの移行は容易です。移

行するには、インストールした AWS-UI 関連の NPM パッケージを関連する Cloudscape パッケージに置き換えます。

AWS-UI パッケージ名	Cloudscape パッケージ名
@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

移行の詳細については、[AWS-UI GitHub ドキュメントページ](#)を参照してください。

SDK リファレンス

このセクションでは、NICE DCV ウェブクライアント SDK の説明、構文、および使用例を示します。

トピック

- [DCV モジュール](#)
- [接続クラス](#)
- [認証クラス](#)
- [リソースクラス](#)
- [NICE DCV ウェブ UI SDK](#)

DCV モジュール

DCV プロトコルのクライアント側を実装するモジュール。

エクスポート

- [方法](#)
- [メンバー](#)
- [データ型とコールバックの定義](#)

方法

リスト

- [authenticate\(url, callbacks\) → {Authentication}](#)
- [connect\(config\) → {Promise.<Connection>|Promise.<{code: ConnectionErrorCode, message: string}>}](#)
- [setLogHandler\(handler\) → {void}](#)
- [setLogLevel\(level\) → {void}](#)

`authenticate(url, callbacks) → {Authentication}`

指定した NICE DCV サーバーエンドポイントの認証プロセスを開始します。

パラメータ:

名前	型	説明
url	string	実行中の NICE DCV サーバーのホスト名とポートを次の形式で指定します: https://dcv_host_address:port 例: https://my-dcv-server:8443 。
callbacks	authenticationCallbacks	認証プロセス中に呼び出すことができるコールバック。

戻り値:

- Authentication オブジェクト。

型

[Authentication](#)

connect(config) → {Promise.<[Connection](#)>|Promise.<{code: [ConnectionErrorCode](#), message: string}>}

指定した NICE DCV サーバーエンドポイントに接続します。接続が成功した場合は Connection オブジェクトを返します。接続が失敗した場合はエラーオブジェクトを返します。

パラメータ:

名前	型	説明
config	ConnectionConfig	ConnectionConfig オブジェクト。

戻り値:

- Connection オブジェクト、またはエラーオブジェクト。

型

Promise.<[Connection](#)> | Promise.<{code: [ConnectionErrorCode](#), message: string}>

setLogHandler(handler) → {void}

カスタムログハンドラ関数を設定します。デフォルトのログハンドラをオーバーライドすると、ブラウザコンソールでデバッグするときに元のログエントリの位置が失われます。

パラメータ:

名前	型	説明
handler	function	カスタムログハンドラ関数。ハンドラ関数には、level (数値)、levelName (文字列)、domain (文字列)、message (文字列) が含まれます。

戻り値:

型

void

setLogLevel(level) → {void}

ログレベルを設定します。これは、デフォルトのログハンドラが使用されている場合にのみ必要です。

パラメータ:

名前	型	説明
level	LogLevel	使用するログレベル。

戻り値:

型

void

メンバー

リスト

- [\(constant\) AudioError :AudioErrorCode](#)
- [\(定数\) AuthenticationError :AuthenticationErrorCode](#)
- [\(定数\) ChannelError :ChannelErrorCode](#)
- [\(定数\) ClosingReasonError :ClosingReasonErrorCode](#)
- [\(定数\) ConnectionError :ConnectionErrorCode](#)
- [\(定数\) CustomChannelError :CustomChannelErrorCode](#)
- [\(定数\) DisplayConfigError :DisplayConfigErrorCode](#)
- [\(定数\) FileStorageError :FileStorageErrorCode](#)
- [\(定数\) LogLevel :LogLevel](#)
- [\(定数\) MultiMonitorError :MultiMonitorErrorCode](#)
- [\(定数\) ResolutionError :ResolutionErrorCode](#)
- [\(定数\) TimezoneRedirectionError :TimezoneRedirectionErrorCode](#)
- [\(定数\) TimezoneRedirectionSetting :TimezoneRedirectionSettingCode](#)
- [\(定数\) TimezoneRedirectionStatus :TimezoneRedirectionStatusCode](#)
- [\(定数\) バージョン](#)
- [\(定数\) ScreenshotError :ScreenshotErrorCode](#)
- [\(定数\) WebcamError :WebcamErrorCode](#)

(constant) AudioError :[AudioErrorCode](#)

AudioError コードの列挙型。

型:

- [AudioErrorCode](#)

(定数) AuthenticationError : [AuthenticationErrorCode](#)

AuthenticationError コードの列挙型。

型:

- [AuthenticationErrorCode](#)

(定数) ChannelError : [ChannelErrorCode](#)

ChannelError コードの列挙型。

型:

- [ChannelErrorCode](#)

(定数) ClosingReasonError : [ClosingReasonErrorCode](#)

ClosingReasonError コードの列挙型。

型:

- [ClosingReasonErrorCode](#)

(定数) ConnectionError : [ConnectionErrorCode](#)

ConnectionError コードの列挙型。

型:

- [ConnectionErrorCode](#)

(定数) CustomChannelError : [CustomChannelErrorCode](#)

CustomChannelError コードの列挙型。

型:

- [CustomChannelErrorCode](#)

(定数) DisplayConfigError :[DisplayConfigErrorCode](#)

DisplayConfigError コードの列挙型。

型:

- [DisplayConfigErrorCode](#)

(定数) FileStorageError :[FileStorageErrorCode](#)

FileStorageError コードの列挙型。

型:

- [FileStorageErrorCode](#)

(定数) LogLevel :[LogLevel](#)

使用可能な SDK ログレベル。

型:

- [LogLevel](#)

(定数) MultiMonitorError :[MultiMonitorErrorCode](#)

MultiMonitorError コードの列挙型。

型:

- [MultiMonitorErrorCode](#)

(定数) ResolutionError :[ResolutionErrorCode](#)

ResolutionError コードの列挙型。

型:

- [ResolutionErrorCode](#)

(定数) TimezoneRedirectionError : [TimezoneRedirectionErrorCode](#)

TimezoneRedirectionError コードの列挙型。

型:

- [TimezoneRedirectionErrorCode](#)

(定数) TimezoneRedirectionSetting : [TimezoneRedirectionSettingCode](#)

TimezoneRedirectionSetting コードの列挙型。

型:

- [TimezoneRedirectionSettingCode](#)

(定数) TimezoneRedirectionStatus : [TimezoneRedirectionStatusCode](#)

TimezoneRedirectionStatus の列挙型。

型:

- [TimezoneRedirectionStatusCode](#)

(定数) バージョン

NICE DCV バージョンで、major、minor、patch、revision、extended、versionStr があります。

プロパティ:

名前	型	説明
major	integer	メジャーバージョン番号。
minor	integer	マイナーバージョン番号。
patch	integer	パッチバージョン番号。
revision	integer	リビジョン番号。

名前	型	説明
extended	string	拡張文字列。
versionStr	string	メジャー番号、マイナー番号、パッチ番号、リビジョン番号を形式 major.minor.patch+build.revision で連結したもの。

(定数) ScreenshotError : [ScreenshotErrorCode](#)

ScreenShotError コードの列挙型。

型:

- [ScreenshotErrorCode](#)

(定数) WebcamError : [WebcamErrorCode](#)

WebcameError コードの列挙型。

型:

- [WebcamErrorCode](#)

データ型とコールバックの定義

リスト

- [AudioErrorCode](#)
- [authenticationCallbacks](#)
- [AuthenticationErrorCode](#)
- [authErrorCallback\(authentication, error\)](#)
- [authPromptCredentialsCallback\(authentication, challenge\)](#)
- [authSuccessCallback\(authentication, authenticationData\)](#)
- [チャンネル](#)

- [ChannelErrorCode](#)
- [clipboardEventCallback\(event\)](#)
- [ClosingReasonErrorCode](#)
- [Colorspace](#)
- [connectionCallbacks](#)
- [ConnectionConfig](#)
- [ConnectionErrorCode](#)
- [createDirectory\(path\)](#)
- [CustomChannelErrorCode](#)
- [dataChannelCallback\(info\)](#)
- [deleteFile\(path\)](#)
- [deviceChangeEventCallback](#)
- [disconnectCallback\(reason\)](#)
- [displayAvailabilityCallback\(status, displayId\)](#)
- [DisplayConfigErrorCode](#)
- [displayLayoutCallback\(serverWidth, serverHeight, heads\)](#)
- [機能](#)
- [featuresUpdateCallback\(featuresList\)](#)
- [fileDownloadCallback\(fileResource\)](#)
- [filePrintedCallback\(printResource\)](#)
- [filestorage](#)
- [filestorageEnabledCallback\(enabled\)](#)
- [FileStorageErrorCode](#)
- [firstFrameCallback\(resizeEnabled, relativeMouseModeEnabled, displayId\)](#)
- [idleWarningNotificationCallback\(disconnectionDateTime\)](#)
- [collaboratorListCallback\(collaborators\)](#)
- [licenseNotificationCallback\(notification\)](#)
- [list\(path\)](#)
- [LogLevel](#)
- [モニター](#)

- [MultiMonitorErrorCode](#)
- [qualityIndicatorStateCallback\(state\)](#)
- [renameDirectory\(src, dest\)](#)
- [renameFile\(src, dest\)](#)
- [ResolutionErrorCode](#)
- [ファイルの取得 \(パス\)](#)
- [screenshotCallback\(screenshot\)](#)
- [ScreenshotErrorCode](#)
- [serverInfo](#)
- [stats](#)
- [storeFile\(file, dir\)](#)
- [TimezoneRedirectionErrorCode](#)
- [TimezoneRedirectionSettingCode](#)
- [TimezoneRedirectionStatusCode](#)
- [WebcamErrorCode](#)

AudioErrorCode

DCV モジュールで使用できる AudioError コードの列挙型。

- SETTING_AUDIO_FAILED
- CHANNEL_NOT_AVAILABLE

型:

- number

authenticationCallbacks

認証コールバック

型:

- オブジェクト

プロパティ:

名前	型	説明
promptCredentials	authPromptCredentialsCallback	ユーザーが認証情報のチャレンジを受けたときに呼び出されるコールバック関数。
error	authErrorCallback	認証が失敗したときに呼び出されるコールバック関数。
success	authSuccessCallback	認証が成功したときに呼び出されるコールバック関数。

AuthenticationErrorCode

DCV モジュールで使用できる AuthenticationError コードの列挙型。

- 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

型:

- number

authErrorCallback(authentication, error)

認証が失敗したときに呼び出されるコールバック関数。

パラメータ:

名前	型	説明									
authentication	Authentication	Authentication オブジェクト。									
error	オブジェクト	認証プロセスによって発生したエラーオブジェクト。 <table border="1"> <thead> <tr> <th>名前</th> <th>型</th> <th>説明</th> </tr> </thead> <tbody> <tr> <td>code</td> <td>AuthenticationErrorCode</td> <td>エラーコードです。</td> </tr> <tr> <td>message</td> <td>string</td> <td>エラーメッセージです。</td> </tr> </tbody> </table>	名前	型	説明	code	AuthenticationErrorCode	エラーコードです。	message	string	エラーメッセージです。
名前	型	説明									
code	AuthenticationErrorCode	エラーコードです。									
message	string	エラーメッセージです。									

authPromptCredentialsCallback(authentication, challenge)

ユーザーが認証情報のチャレンジを受けたときに呼び出されるコールバック関数。ユーザーは、要求された認証情報を提供してチャレンジに応答する必要があります。

パラメータ:

名前	型	説明
authentication	Authentication	Authentication オブジェクト。
challenge	オブジェクト	チャレンジ。

名前	型	説明		
		名前	型	説明
		requiredAuthentication	Array.<Object>	<p>リクエストされた認証情報オブジェクトの配列。</p>
		ns	String	リクエストされた認証情報の名前。
		ts	String	リクエスト

名前	型	説明		
		名前	型	説明
				名前、型、説明された認証情報の型。

authSuccessCallback(authentication, authenticationData)

認証が成功したときに呼び出されるコールバック関数。

パラメータ:

名前	型	説明
authentication	Authentication	Authentication オブジェクト。
authenticationData	Array.<Object>	NICE DCV セッション ID と認証トークンを含むオブジェクトの配列。

名前	型	説明
session	string	NICE DCV セッション ID

名前	型	説明		
		名前	型	説明
				セッション ID。
		authTok	string	NICE DCV セッションの認証トークン。

チャンネル

指定できる使用可能なチャンネル。

型:

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

ChannelErrorCode

DCV モジュールで使用できる ChannelError コードの列挙型。

- ALREADY_OPEN
- INITIALIZATION_FAILED
- REJECTED

型:

- number

clipboardEventCallback(event)

clipboardEvent の生成時に呼び出されるコールバック関数。

パラメータ:

名前	型	説明	
event	オブジェクト	クリップボードイベントに関する情報。	
		名前	型
		属性	説明
		name	established copy dataSignal autoCopy newDataAvailable
			常に存在しません。イベントの名前。
		clipboardData	Object string
			クリップボード内のデータ。
		autoCopy	boolean <optional>
			セッションクリップボード

名前	型	説明			
		名前	型	属性	説明
					からローカルクライアントトリップボードへの自動コピーが有効かどうかを示します。
		maxDataSize	number	<optional>	トリップボードに配置できるデータの

名前	型	説明			
		名前	型	属性	説明
					最大量。
		error	string	<optional>	エラー情報 (該当する場合)。

ClosingReasonErrorCode

DCV モジュールで使用できる ClosingReasonError コードの列挙型。

- 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

型:

- number

Colorspace

指定できる使用可能な色空間。

型:

- "RGB" | "YUV_REC601" | "YUV_REC709"

connectionCallbacks

接続エラーが発生した場合に呼び出すことができるコールバック。

型:

- オブジェクト

プロパティ:

名前	型	説明
disconnect	disconnectCallback	接続の終了時に呼び出されるコールバック関数。
displayLayout	displayLayoutCallback	ディスプレイレイアウトまたは解像度に変更されたときに呼び出されるコールバック関数。
displayAvailability	displayAvailabilityCallback	ディスプレイの可用性が変更されたときに呼び出されるコールバック関数。
firstFrame	firstFrameCallback	NICE DCV サーバーから最初のフレームを受信したときに呼び出されるコールバック関数。

名前	型	説明
filePrinted	filePrintedCallback	ファイルが NICE DCV サーバーに出力されるときに呼び出されるコールバック関数。
fileDownload	fileDownloadCallback	NICE DCV サーバーからファイルをダウンロードする準備ができたときに呼び出されるコールバック関数。
dataChannel	dataChannelCallback	NICE DCV サーバーからデータチャンネルの可用性に関する通知が送信されたときに呼び出されるコールバック関数。
licenseNotification	licenseNotificationCallback	NICE DCV サーバーからライセンス状態に関する通知が送信されたときに呼び出されるコールバック関数。
idleWarningNotification	idleWarningNotificationCallback	NICE DCV サーバーがアイドルタイムアウト警告を送信したときに呼び出されるコールバック関数。
collaboratorList	collaboratorListCallback	NICE DCV サーバーがコラボレーターのリストを送信したときに呼び出されるコールバック関数 (NICE DCV ウェブクライアント SDK バージョン 1.1.0 以降)。
qualityIndicatorState	qualityIndicatorStateCallback	接続品質インジケータで状態が変化したときに呼び出されるコールバック関数。

名前	型	説明
filestorageEnabled	filestorageEnabledCallback	ファイルストレージが有効または無効になったときに呼び出されるコールバック関数。
featuresUpdate	featuresUpdateCallback	機能のステータスが変化したときに呼び出されるコールバック関数。
clipboardEvent	clipboardEventCallback	clipboardEvent の生成時に呼び出されるコールバック関数。
deviceChangeEvent	deviceChangeEventCallback	deviceChange イベントがトリガーされたときに呼び出されるコールバック関数。
screenshot	screenshotCallback	screenshot が使用可能であるときに呼び出されるコールバック関数。

ConnectionConfig

NICE DCV 接続の設定。

型:

- オブジェクト

プロパティ:

名前	型	説明
url	string	実行中の NICE DCV サーバーのホスト名とポートを次の形式で指定します: <code>https://dcv_host_address:po</code>

名前	型	説明
		rt 例: <code>https://my-dcv-server:8443</code> 。
sessionId	string	NICE DCV セッション ID。
authToken	string	セッションへの接続に使用する認証トークン。
baseUrl	string	SDK ファイルのロード元である絶対 URL または相対 URL。
resourceBaseUrl	string	DCV リソースのアクセス元である絶対 URL または相対 URL。
enabledChannels	Array.< Channel >	有効化できるチャンネルのリストを示します。指定しない場合、または空の配列を指定した場合、デフォルトによりすべての使用可能なチャンネルになります。
losslessColorspace	Colorspace	使用される色空間を示します。指定されない場合、デフォルトの「RGB」になります。
divId	string	SDK によりリモートストリームでキャンバスが作成される HTML DOM の div オブジェクトの ID。
volumeLevel	integer	希望するボリュームレベル。有効範囲は 0~100 です。

名前	型	説明
clipboardAutoSync	boolean	互換性のあるウェブブラウザで、NICE DCV セッションクリップボードからローカルクライアントクリップボードへの自動コピーが有効かどうかを示します。
dynamicAudioTuning	boolean	接続が確立されたときに、NICE DCV サーバーのオーディオ設定に基づいてオーディオの動的チューニングを行うかどうかを示します。
clientHiDpiScaling	boolean	クライアントの DPI に基づいてキャンバスのスケーリングを行うかどうかを示します。
highColorAccuracy	boolean	高い色精度が使用可能な場合に使用するかどうかを示します。指定されない場合、デフォルトは false です。
enableWebCodecs	ブール値	WebCodecs が使用可能な場合に使用するかどうかを示します。指定されない場合のデフォルト値は false です。
observers	connectionCallbacks	接続に関連するイベントを呼び出すためのコールバック関数。
callbacks	connectionCallbacks	observers プロパティと同じですが、各コールバックに Connection オブジェクトが最初のパラメータとして含まれます。

ConnectionErrorCode

DCV モジュールで使用できる ConnectionError コードの列挙型。

- ALREADY_OPEN
- INVALID_CONFIG
- INITIALIZATION_FAILED
- REJECTED
- MAIN_CHANNEL_ALREADY_OPEN
- GENERIC_ERROR (DCV Server 2021.0 以降)
- INTERNAL_SERVER_ERROR (DCV Server 2021.0 以降)
- AUTHENTICATION_FAILED (DCV Server 2021.0 以降)
- PROTOCOL_ERROR (DCV Server 2021.0 以降)
- INVALID_SESSION_ID (DCV Server 2021.0 以降)
- INVALID_CONNECTION_ID (DCV Server 2021.0 以降)
- CONNECTION_LIMIT_REACHED (DCV Server 2021.0 以降)
- SERVER_UNREACHABLE (DCV Server 2022.1 以降)

型:

- number

createDirectory(path)

パラメータ:

名前	型	説明
path	string	ディレクトリを作成するサーバーの絶対パス。ターゲットディレクトリの名前も含まれます。

CustomChannelErrorCode

DCV モジュールで使用できる CustomChannelError コードの列挙型。

- TRANSPORT_ERROR

型:

- number

dataChannelCallback(info)

NICE DCV サーバーからデータチャンネルの可用性に関する通知が送信されたときに呼び出されるコールバック関数。

パラメータ:

名前	型	説明									
info	オブジェクト	データチャンネルに関する情報。 <table><thead><tr><th>名前</th><th>型</th><th>説明</th></tr></thead><tbody><tr><td>name</td><td>string</td><td>データチャンネルの名前。</td></tr><tr><td>token</td><td>string</td><td>データチャンネルの認証トークン。</td></tr></tbody></table>	名前	型	説明	name	string	データチャンネルの名前。	token	string	データチャンネルの認証トークン。
名前	型	説明									
name	string	データチャンネルの名前。									
token	string	データチャンネルの認証トークン。									

deleteFile(path)

パラメータ:

名前	型	説明
path	string	削除するファイルを識別するサーバーの絶対パス。

deviceChangeEventCallback

deviceChange イベントがトリガーされたときに呼び出されるコールバック関数。

disconnectCallback(reason)

接続の終了時に呼び出されるコールバック関数。

パラメータ:

名前	型	説明									
reason	オブジェクト	切断の理由。 <table border="1" data-bbox="1068 1192 1507 1675"> <thead> <tr> <th>名前</th> <th>型</th> <th>説明</th> </tr> </thead> <tbody> <tr> <td>code</td> <td>number</td> <td>理由コード。</td> </tr> <tr> <td>message</td> <td>string</td> <td>理由メッセージ。</td> </tr> </tbody> </table>	名前	型	説明	code	number	理由コード。	message	string	理由メッセージ。
名前	型	説明									
code	number	理由コード。									
message	string	理由メッセージ。									

displayAvailabilityCallback(status, displayId)

ディスプレイの可用性が変更されたときに呼び出されるコールバック関数。

パラメータ:

名前	型	説明									
status	オブジェクト	ディスプレイのステータス。 <table border="1"> <thead> <tr> <th>名前</th> <th>型</th> <th>説明</th> </tr> </thead> <tbody> <tr> <td>enabled</td> <td>boolean</td> <td>ディスプレイが有効かどうかを示します。</td> </tr> <tr> <td>closed</td> <td>boolean</td> <td>ディスプレイが閉じているかどうかを示します。</td> </tr> </tbody> </table>	名前	型	説明	enabled	boolean	ディスプレイが有効かどうかを示します。	closed	boolean	ディスプレイが閉じているかどうかを示します。
名前	型	説明									
enabled	boolean	ディスプレイが有効かどうかを示します。									
closed	boolean	ディスプレイが閉じているかどうかを示します。									
displayId	number	ディスプレイの識別子。									

DisplayConfigErrorCode

DCV モジュールで使用できる DisplayConfigError コードの列挙型。

- INVALID_ARGUMENT
- UNSUPPORTED_OPERATION
- NO_CHANNEL

型:

- number

displayLayoutCallback(serverWidth, serverHeight, heads)

ディスプレイレイアウトまたは解像度に変更されたときに呼び出されるコールバック関数。

パラメータ:

名前	型	説明
serverWidth	number	プライマリディスプレイの幅 (ピクセル)。
serverHeight	number	プライマリディスプレイの高さ (ピクセル)。
heads	Array.< Monitor >	NICE DCV サーバーでサポートされるディスプレイヘッド。

機能

機能値。

- display - シングルディスプレイビデオストリームの可用性を示します。
- display-multi - マルチディスプレイビデオストリームの可用性を示します。
- high-color-accuracy - 高い色精度の可用性を示します (NICE DCV ウェブクライアント SDK バージョン 1.1.0 以降)。
- mouse - マウス機能の可用性を示します。
- keyboard - キーボード機能の可用性を示します。
- keyboard-sas - SAS シーケンス (Control + Alt + Delete) 機能の可用性を示します。
- relative-mouse - 相対マウスモードの可用性を示します。
- clipboard-copy - NICE DCV サーバーからクライアントへのクリップボードコピー機能の可用性を示します。
- clipboard-paste - クライアントから NICE DCV サーバーへのクリップボード貼り付け機能の可用性を示します。

- audio-in - マイクを使用したオーディオ入力機能の可用性を示します。
- audio-out - オーディオ再生機能の可用性を示します。
- webcam - ウェブカメラストリーミング機能の可用性を示します。
- file-download - NICE DCV サーバーからクライアントへのファイルダウンロード機能の可用性を示します。
- file-upload - クライアントから NICE DCV サーバーへのファイルアップロード機能の可用性を示します。
- timezone-redirect - タイムゾーンリダイレクト機能の可用性を示します (NICE DCV ウェブクライアント SDK バージョン 1.3.0 以降)。

型:

- string

featuresUpdateCallback(featuresList)

機能のステータスが変わったときに呼び出されるコールバック関数。

パラメータ:

名前	型	説明
featuresList	Array.< feature >	変化した一連の機能。

fileDownloadCallback(fileResource)

NICE DCV サーバーからファイルをダウンロードする準備ができたときに呼び出されるコールバック関数。

パラメータ:

名前	型	説明
fileResource	オブジェクト	ダウンロード可能になったファイルに関する情報。

名前	型	説明		
		名前	型	説明
		id	string	ファイルの識別子。
		url	string	ファイルのダウンロードに使用する URL。
		domain	string	リソースドメイン。
		token	string	ファイルのダウンロードに使用する認証トークン。このトークンは URL にも含まれます。

filePrintedCallback(printResource)

ファイルが NICE DCV サーバーに出力されるときに呼び出されるコールバック関数。

パラメータ:

名前	型	説明															
printResource	オブジェクト	印刷されたファイルに関する情報。 <table><thead><tr><th>名前</th><th>型</th><th>説明</th></tr></thead><tbody><tr><td>id</td><td>string</td><td>印刷されたファイルの識別子。</td></tr><tr><td>url</td><td>string</td><td>印刷されたファイルのダウンロードに使用する URL。</td></tr><tr><td>domain</td><td>string</td><td>リソースドメイン。この場合は printer。</td></tr><tr><td>token</td><td>string</td><td>印刷された</td></tr></tbody></table>	名前	型	説明	id	string	印刷されたファイルの識別子。	url	string	印刷されたファイルのダウンロードに使用する URL。	domain	string	リソースドメイン。この場合は printer。	token	string	印刷された
名前	型	説明															
id	string	印刷されたファイルの識別子。															
url	string	印刷されたファイルのダウンロードに使用する URL。															
domain	string	リソースドメイン。この場合は printer。															
token	string	印刷された															

名前	型	説明		
		名前	型	説明
				ファイルのダウンロードに使用する認証トークン。このトークンはURLにも含まれます。

filestorage

ファイルシステムにおけるアクションの調査と実行を可能にするオブジェクト。

型:

- オブジェクト

プロパティ:

名前	型	説明
list	list	サーバー上の提供されたパスに存在するアイテム (ファイルとディレクトリ) の一覧表示を許可する関数。

名前	型	説明
createDirectory	createDirectory	サーバー上の指定されたパスでのディレクトリ作成を許可する関数。
retrieveFile	retrieveFile	サーバー上の指定されたパスでファイルのローカルダウンロードを許可する関数。
deleteFile	deleteFile	サーバー上の指定されたパスでのファイル削除を許可する関数。
renameFile	renameFile	指定ソースパスから指定送信先パスへのファイル名変更を許可する関数。
renameDirectory	renameDirectory	指定ソースパスから絶対送信先パスへのディレクトリ名変更を許可する関数。
storeFile	storeFile	サーバー上の指定されたパスへのローカルファイルのアップロードを許可する関数。

filestorageEnabledCallback(enabled)

ファイルストレージが有効になったときに呼び出されるコールバック関数。Internet Explorer 11 のレイジーチャンネルのみ。

パラメータ:

名前	型	説明
enabled	boolean	ファイルストレージが有効かどうかを示します。

FileStorageErrorCode

DCV モジュールで使用できる FileStorageError コードの列挙型

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

型:

- number

firstFrameCallback(resizeEnabled, relativeMouseModeEnabled, displayId)

NICE DCV サーバーから最初のフレームを受信したときに呼び出されるコールバック関数。ディスプレイごとに放出されます。

パラメータ:

名前	型	説明
resizeEnabled	boolean	クライアントディスプレイのレイアウトのサイズ変更がサーバーでサポートされているかどうかを示します。
relativeMouseModeEnabled	boolean	相対マウスモードがサーバーでサポートされているかどうかを示します。
displayId	number	ディスプレイの識別子。

idleWarningNotificationCallback(disconnectionDateTime)

NICE DCV サーバーがアイドルタイムアウト警告を送信したときに呼び出されるコールバック関数。

パラメータ:

名前	型	説明
disconnectionDateTime	日付	切断の日時。

collaboratorListCallback(collaborators)

NICE DCV サーバーがコラボレーターのリストを送信したときに呼び出されるコールバック関数。

パラメータ:

名前	型	説明
collaborators	Array.<Object>	コラボレーターに関する情報を含むオブジェクトのリスト。

名前	型	説明
username	string	コラボレーターのユーザー名。
owner	boolean	コラボレーターがセッション

名前	型	説明		
		名前	型	説明
				ンオーナーであるかどうかを示します。
		connect number nId		サーバーによって接続に割り当てられた ID を示します。

licenseNotificationCallback(notification)

NICE DCV サーバーからライセンス状態に関する通知が送信されたときに呼び出されるコールバック関数。

パラメータ:

名前	型	説明		
notification	オブジェクト	通知。		
		名前	型	説明
		product string		DCV 生成物。

名前	型	説明		
		名前	型	説明
		status	string	ライセンスのステータス。
		message	string	メッセージ。
		leftDay	number	ライセンスの有効期限が切れるまでの日数。
		isDemo	boolean	ライセンスがデモライセンスかどうかを示します。
		numUnlicensed	number	ライセンスを取得していない接続の数。

名前	型	説明									
		<table border="1"> <thead> <tr> <th>名前</th> <th>型</th> <th>説明</th> </tr> </thead> <tbody> <tr> <td>licenseMode</td> <td>string</td> <td>ライセンス取得モード。</td> </tr> <tr> <td>documentationUrl</td> <td>string</td> <td>ドキュメントの URL。</td> </tr> </tbody> </table>	名前	型	説明	licenseMode	string	ライセンス取得モード。	documentationUrl	string	ドキュメントの URL。
名前	型	説明									
licenseMode	string	ライセンス取得モード。									
documentationUrl	string	ドキュメントの URL。									

list(path)

パラメータ:

名前	型	説明
path	string	コンテンツを一覧表示するサーバー上の絶対パス。

LogLevel

使用可能な SDK ログレベル。

型:

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

モニター

型:

- オブジェクト

プロパティ:

名前	型	説明															
name	string	ディスプレイヘッドの名前。															
rect	オブジェクト	ディスプレイヘッドに関する情報。 <table border="1" data-bbox="1068 548 1507 1879"> <thead> <tr> <th>名前</th> <th>型</th> <th>説明</th> </tr> </thead> <tbody> <tr> <td>x</td> <td>number</td> <td>ディスプレイヘッドの最初の x 座標。</td> </tr> <tr> <td>y</td> <td>number</td> <td>ディスプレイヘッドの最初の y 座標。</td> </tr> <tr> <td>width</td> <td>number</td> <td>ディスプレイヘッドの幅 (ピクセル)。</td> </tr> <tr> <td>height</td> <td>number</td> <td>ディスプレイヘッドの高さ (ピクセル)。</td> </tr> </tbody> </table>	名前	型	説明	x	number	ディスプレイヘッドの最初の x 座標。	y	number	ディスプレイヘッドの最初の y 座標。	width	number	ディスプレイヘッドの幅 (ピクセル)。	height	number	ディスプレイヘッドの高さ (ピクセル)。
名前	型	説明															
x	number	ディスプレイヘッドの最初の x 座標。															
y	number	ディスプレイヘッドの最初の y 座標。															
width	number	ディスプレイヘッドの幅 (ピクセル)。															
height	number	ディスプレイヘッドの高さ (ピクセル)。															

名前	型	説明		
		名前	型	説明
				クセル)。
primary	boolean	ディスプレイヘッドがプライマリディスプレイヘッドかどうかを示します。これは、リモートオペレーティングシステムがある場合はそこから決定されます。		
dpi	number	ディスプレイヘッドの DPI。		

MultiMonitorErrorCode

DCV モジュールで使用できる MultimonitorError コードの列挙型

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

型:

- number

qualityIndicatorStateCallback(state)

接続品質インジケータで状態が変化したときに呼び出されるコールバック関数。

パラメータ:

名前	型	説明												
state	Array.<Object>	接続品質に関する情報。 <table border="1" data-bbox="1068 422 1507 995"> <thead> <tr> <th>名前</th> <th>型</th> <th>説明</th> </tr> </thead> <tbody> <tr> <td>name</td> <td>string</td> <td>インジケータの名前。</td> </tr> <tr> <td>status</td> <td>NORMAL WARNIN CRITICA</td> <td>ステータスの説明。</td> </tr> <tr> <td>changed</td> <td>boolean</td> <td>ステータスが変化したかどうかを示します。</td> </tr> </tbody> </table>	名前	型	説明	name	string	インジケータの名前。	status	NORMAL WARNIN CRITICA	ステータスの説明。	changed	boolean	ステータスが変化したかどうかを示します。
名前	型	説明												
name	string	インジケータの名前。												
status	NORMAL WARNIN CRITICA	ステータスの説明。												
changed	boolean	ステータスが変化したかどうかを示します。												

renameDirectory(src, dest)

パラメータ:

名前	型	説明
src	string	名前を変更するディレクトリを識別するサーバー上の絶対ソースパス。

名前	型	説明
dest	string	ターゲットパスとディレクトリ名を指定するサーバー上の絶対送信先パス。

renameFile(src, dest)

パラメータ:

名前	型	説明
src	string	名前を変更するファイルを識別するサーバー上の絶対ソースパス。
dest	string	ターゲットパスとファイル名を指定するサーバー上の絶対送信先パス。

ResolutionErrorCode

DCV モジュールで使用できる ResolutionError コードの列挙型

- INVALID_ARGUMENT
- NO_CHANNEL
- NOT_IMPLEMENTED

型:

- number

ファイルの取得 (パス)

パラメータ:

名前	型	説明
path	string	ローカルでダウンロードするファイルを識別するサーバー上の絶対パス。

screenshotCallback(screenshot)

スクリーンショットが使用可能であるときに呼び出されるコールバック関数。

パラメータ:

名前	型	説明
screenshot	byte[]	PNG 形式のスクリーンショットバッファ。スクリーンショットの取得に失敗した場合は null。

ScreenshotErrorCode

DCV モジュールで使用できる ScreenShotError コードの列挙型。

- NO_CHANNEL
- GENERIC_ERROR

型:

- number

serverInfo

型:

- オブジェクト

プロパティ:

名前	型	説明												
name	string	ソフトウェアの名前。												
version	オブジェクト	ソフトウェアのバージョン番号。 <table border="1"> <thead> <tr> <th>名前</th> <th>型</th> <th>説明</th> </tr> </thead> <tbody> <tr> <td>major</td> <td>number</td> <td>メジャーバージョン番号。</td> </tr> <tr> <td>minor</td> <td>number</td> <td>マイナーバージョン番号。</td> </tr> <tr> <td>revisio</td> <td>number</td> <td>リビジョン番号。</td> </tr> </tbody> </table>	名前	型	説明	major	number	メジャーバージョン番号。	minor	number	マイナーバージョン番号。	revisio	number	リビジョン番号。
名前	型	説明												
major	number	メジャーバージョン番号。												
minor	number	マイナーバージョン番号。												
revisio	number	リビジョン番号。												
os	string	OS。												
arch	string	アーキテクチャ。												

名前	型	説明
hostname	string	ホスト名。

stats

型:

- オブジェクト

プロパティ:

名前	型	説明
fps	number	現在のフレーム/秒。
traffic	number	現在のトラフィック (ビット/秒)。
peakTraffic	number	接続確立以降のトラフィックのピーク (ビット/秒)。
latency	number	現在のレイテンシー (ミリ秒)。
currentChannels	number	接続確立後から開いているチャンネルの数。
openedChannels	number	現在開いているチャンネルの数。
channelErrors	number	エラーを報告したチャンネルの数。

storeFile(file, dir)

パラメータ:

名前	型	説明
file	File	サーバーにアップロードするファイルオブジェクト (詳細については https://developer.mozilla.org/en-US/docs/Web/API/File を参照)。
dir	string	ファイルをアップロードするサーバーの絶対パス。

TimezoneRedirectionErrorCode

DCV モジュールで使用できる TimezoneRedirectionError コードの列挙型

- INVALID_ARGUMENT
- NO_CHANNEL
- USER_CANNOT_CHANGE

型:

- number

TimezoneRedirectionSettingCode

DCV モジュールで使用できる TimezoneRedirectionSetting コードの列挙型

- ALWAYS_OFF
- ALWAYS_ON
- CLIENT_DECIDES

型:

- number

TimezoneRedirectionStatusCode

DCV モジュールで使用できる TimezoneRedirectionStatusCode コードの列挙型

- SUCCESS
- PERMISSION_ERROR
- GENERIC_ERROR

型:

- number

WebcamErrorCode

DCV モジュールで使用できる WebcamError コードの列挙型

- SETTING_WEBCAM_FAILED
- CHANNEL_NOT_AVAILABLE

型:

- number

接続クラス

dcv モジュールの [connect メソッド](#) を呼び出すと得られる接続クラス。使用方法を示した例については「[開始方法](#)」セクションを参照してください。

エクスポート

- [方法](#)

方法

リスト

- [attachDisplay\(win, displayConf\) → {Promise.<number>|Promise.<{code: MultiMonitorErrorCode, message: string}>}](#)
- [captureClipboardEvents\(enabled, win, displayId\) → {void}](#)
- [detachDisplay\(displayId\) → {void}](#)
- [disconnect\(\) → {void}](#)
- [disconnectCollaborator\(connectionId\) → {void}](#)
- [enableDisplayQualityUpdates\(enable\) → {void}](#)
- [enableTimezoneRedirection\(enable\) → {Promise|Promise.<{code: TimezoneRedirectionErrorCode, message: string}>}](#)
- [enterRelativeMouseMode\(\) → {void}](#)
- [getConnectedDevices\(\) → {Promise.<Array.<MediaDeviceInfo>>|Promise.<{message: string}>}](#)
- [getFileExplorer\(\) → {Promise.<filestorage>|Promise.<{code: ChannelErrorCode, message: string}>}](#)
- [getServerInfo\(\) → {serverInfo}](#)
- [getScreenshot\(\) → {Promise|Promise.<{code: ScreenshotErrorCode, message: string}>}](#)
- [getStats\(\) → {stats}](#)
- [latchModifierKey\(key, location, isDown\) → {boolean}](#)
- [openChannel\(name, authToken, callbacks, namespace\) → {Promise|Promise.<{code: ChannelErrorCode, message: string}>}](#)
- [queryFeature\(featureName\) → {Promise.<{enabled: boolean, remote?: string, autoCopy?: boolean, autoPaste?: boolean, serviceStatus?: string, available?: boolean}>|Promise.<{message: string}>}](#)
- [registerKeyboardShortcuts\(shortcuts\) → {void}](#)
- [requestDisplayConfig\(highColorAccuracy\) → {Promise|Promise.<{code: DisplayConfigErrorCode, message: string}>}](#)
- [requestDisplayLayout\(layout\) → {Promise|Promise.<{code: ResolutionErrorCode, message: string}>}](#)
- [requestResolution\(width, height\) → {Promise|Promise.<{code: ResolutionErrorCode, message: string}>}](#)
- [sendKeyboardEvent\(event\) → {boolean}](#)
- [sendKeyboardShortcut\(shortcut\) → {void}](#)

- [setDisplayQuality\(min, maxopt\) → {void}](#)
- [setDisplayScale\(scaleRatio, displayId\) → {Promise|Promise.<{code: ResolutionErrorCode, message: string}>}](#) (DEPRECATED)
- [setKeyboardQuirks\(quirks\) → {void}](#)
- [setMaxDisplayResolution\(maxWidth, maxHeight\) → {void}](#)
- [setMicrophone\(enable\) → {Promise|Promise.<{code: AudioErrorCode, message: string}>}](#)
- [setMinDisplayResolution\(minWidth, minHeight\) → {void}](#)
- [setUploadBandwidth\(value\) → {number}](#)
- [setVolume\(volume\) → {void}](#)
- [setWebcam\(enable, deviceId\) → {Promise|Promise.<{code: WebcamErrorCode, message: string}>}](#)
- [syncClipboards\(\) → {boolean}](#)

[attachDisplay\(win, displayConf\) → {Promise.<number>|Promise.<{code: MultiMonitorErrorCode, message: string}>}](#)

特定のディスプレイをウィンドウにアタッチします。メインディスプレイはアタッチできません。成功すると、関数が `displayId` を返します。

パラメータ:

名前	型	説明								
win	オブジェクト	ディスプレイをアタッチする必要があるウィンドウ。								
displayConf	オブジェクト	ディスプレイの設定。 <table border="1" data-bbox="1068 1535 1507 1879"> <thead> <tr> <th>名前</th> <th>型</th> <th>属性</th> <th>説明</th> </tr> </thead> <tbody> <tr> <td>displayId</td> <td>number</td> <td><optional></td> <td>ディスプレイの ID。</td> </tr> </tbody> </table>	名前	型	属性	説明	displayId	number	<optional>	ディスプレイの ID。
名前	型	属性	説明							
displayId	number	<optional>	ディスプレイの ID。							

名前	型	説明			
		名前	型	属性	説明
		displayName			ディスプレイ div の名前。

戻り値:

Promise。拒否された場合、プロミスがエラーオブジェクトを返します。

型

Promise.<number> | Promise.<{code: [MultiMonitorErrorCode](#), message: string}>

captureClipboardEvents(enabled, win, displayId) → {void}

コピー/貼り付けイベントのリッスンを開始または停止します。インタラクティブなクリップボード (貼り付けの場合は常時) の場合、コピー/貼り付けイベントのリッスンを開始する必要があります。例えばモーダルが表示されている場合など、必要なときにのみ、リッスンを開始および停止すると便利です。

パラメータ:

名前	型	属性	説明
enabled	boolean		イベントのリッスンを開始するには、true を指定します。イベントのリッスンを停止するには、false を指定します。

名前	型	属性	説明
win	オブジェクト	<optional>	イベントをリッスンするウィンドウ。省略した場合、デフォルトのウィンドウが使用されます。
displayId	number	<optional>	イベントをリッスンするディスプレイのID。省略した場合、デフォルトのウィンドウディスプレイが使用されます。

戻り値:

型

void

detachDisplay(displayId) → {void}

特定のディスプレイをデタッチします。メインディスプレイはデタッチできません。

パラメータ:

名前	型	説明
displayId	number	デタッチするディスプレイのID。

戻り値:

型

void

disconnect() → {void}

NICE DCV サーバーから切断して接続を閉じます。

戻り値:

型

void

disconnectCollaborator(connectionId) → {void}

指定された接続 ID で接続しているコラボレーターの切断を要求します (NICE DCV ウェブクライアント SDK バージョン 1.1.0 以降)。

パラメータ:

名前	型	説明
connectionId	boolean	切断される接続の ID。

戻り値:

型

void

enableDisplayQualityUpdates(enable) → {void}

更新を受けないストリーミングエリアの表示品質の更新を有効または無効にします。表示品質の更新を無効にすると、帯域幅の使用量は減少しますが、表示品質も低下します。

パラメータ:

名前	型	説明
enable	boolean	表示品質の更新を有効にするには、true を指定します。表

名前	型	説明
		示品質の更新を無効にするには、 <code>false</code> を指定します。

戻り値:

型

`void`

`enableTimezoneRedirection(enable) → {Promise|Promise.<{code: TimezoneRedirectionErrorCode, message: string}>}`

タイムゾーンリダイレクトを有効または無効にします。有効にすると、クライアントはサーバーのデスクトップタイムゾーンをクライアントのタイムゾーンと一致させるようにサーバーに要求します。

パラメータ:

名前	型	説明
<code>enable</code>	<code>boolean</code>	タイムゾーンリダイレクトを有効にするには、 <code>true</code> を指定します。タイムゾーンリダイレクトを無効にするには、 <code>false</code> を指定します。

戻り値:

Promise。拒否された場合、プロミスがエラーオブジェクトを返します。

型

`Promise.<number> | Promise.<{code: TimezoneRedirectionErrorCode, message: string}>`

`enterRelativeMouseMode() → {void}`

相対マウスモードを有効にします。

戻り値:

型

void

`getConnectedDevices() → {Promise.<Array.<MediaDeviceInfo>>|Promise.<{message: string}>}`

クライアントコンピュータに接続されているメディアデバイスのリストをリクエストします。

戻り値:

成功した場合、解決する Promise を `MediaDeviceInfo` オブジェクトの配列に返します。詳細については、<https://developer.mozilla.org/en-US/docs/Web/API/MediaDeviceInfo> を参照してください。拒否された場合、プロミスがエラーオブジェクトを返します。

型

`Promise.<Array.<MediaDeviceInfo>> | Promise.<{message: string}>`

`getFileExplorer() → {Promise.<filestorage>|Promise.<{code: ChannelErrorCode, message: string}>}`

NICE DCV サーバーのファイルストレージを管理するためにオブジェクトを取得します。

戻り値:

Promise。満たされた場合はファイルエクスプローラーオブジェクト、拒否された場合はエラーオブジェクトに対して、解決するプロミスを返します。

型

`Promise.<filestorage> | Promise.<{code: ChannelErrorCode, message: string}>`

`getServerInfo() → {serverInfo}`

NICE DCV サーバーに関する情報を取得します。

戻り値:

サーバーソフトウェアに関する情報。

型

[serverInfo](#)

getScreenshot() → {Promise|Promise.<{code: [ScreenshotErrorCode](#), message: string}>}

リモートデスクトップのスクリーンショットを PNG 形式で取得します。スクリーンショットは [ScreenshotCallback](#) オブザーバーに返されます。失敗した場合は代わりに null が返されます。

戻り値:

リクエストが処理されたら解決する Promise。拒否された場合、エラーオブジェクトが送られます。

型

Promise | Promise.<{code: [ScreenshotErrorCode](#), message: string}>

getStats() → {[stats](#)}

NICE DCV サーバーに関する統計を取得します。

戻り値:

ストリーミング統計に関する情報。

型

[stats](#)

latchModifierKey(key, location, isDown) → {boolean}

許可された修飾子に対する単一のキーボード keydown または keyup を送信します。

パラメータ:

名前	型	説明
key	Control Alt AltGraph Meta OS Shift	送信するキー。
location	KeyboardEvent.location	キーの場所。詳細については、 https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/location を参照してください。
isDown	boolean	挿入するキーイベントがキーダウン (true) またはキーアップ (false) である場合。

戻り値:

リクエストされた組み合わせが有効な場合、関数が true を返し、それ以外の場合は false を返します。

型

boolean

`openChannel(name, authToken, callbacks, namespace) → {Promise|Promise.<{code: ChannelErrorCode, message: string}>}`

カスタムデータチャンネルが NICE DCV サーバーで作成された場合にそのチャンネルを接続で開きます。

パラメータ:

名前	型	説明
name	string	チャンネルの名前。

名前	型	説明
authToken	string	チャンネルへの接続に使用する認証トークン。
callbacks	オブジェクト	呼び出す onMessage と onClose コールバック関数。
namespace	string	データチャンネルの名前空間。NICE DCV ウェブクライアント SDK 1.2.0 および NICE DCV サーバー 2022.1 以降で使用できます。

戻り値:

Promise。拒否された場合、エラーオブジェクトが送られます。

型

Promise | Promise.<{code: [ChannelErrorCode](#), message: string}>

queryFeature(featureName) → {Promise.<{enabled: boolean, remote?: string, autoCopy?: boolean, autoPaste?: boolean, serviceStatus?: string, available?: boolean}>|Promise.<{message: string}>}

特定の NICE DCV サーバー機能のステータスについてクエリを行います。

パラメータ:

名前	型	説明
featureName	機能	クエリを行う機能の名前。

戻り値:

Promise。解決すると、関数は常に enabled プロパティを含む status オブジェクトを返し、場合によっては他のプロパティも返します。拒否された場合、関数が error オブジェクトを返します。

型

```
{Promise.<{enabled: boolean, remote?: string, autoCopy?: boolean, autoPaste?: boolean,
serviceStatus?: string, available?: boolean}> | Promise.<{message: string}>
```

```
registerKeyboardShortcuts(shortcuts) → {void}
```

キーボードショートカットを登録します。

パラメータ:

名前	型	説明						
shortcuts	Array.<Object>	登録するキーとマッピングの配列。 <table border="1" data-bbox="1068 877 1507 1444"> <thead> <tr> <th>名前</th> <th>型</th> <th>説明</th> </tr> </thead> <tbody> <tr> <td>sequence</td> <td>Array.<Object></td> <td>登録するキーボードショートカット。</td> </tr> </tbody> </table>	名前	型	説明	sequence	Array.<Object>	登録するキーボードショートカット。
名前	型	説明						
sequence	Array.<Object>	登録するキーボードショートカット。						

名前	型	説明		
		名前	型	説明
				名前
				値。 詳細 につ つい て は、 https:// // develop er.mozill a.org/ en- US/ docs/ W eb/ API/ Ke yboardEve nt/ key を参 照し てく ださ

名前	型	説明		
		名前	型	説明
				説明
				い。
				1 k 送 boardE v 信 loca ti す る キー の 配 列 。 キー ボ ー ド に お け る キー の 場 所 。 詳 細 に つ い て は、htt ps://

名前	型	説明		
		名前	型	説明
				名前
				deve loper.moz illa.org/ en- US/ doc s/ Web/ API / Keyboard Event/ loc ation を 参 照 し て く だ さ い 。
		output	Array.<Object>	ショー トカッ トによ って実 行され

名前	型	説明		
		名前	型	説明
				<p>る意図されたアクション。</p>
				<p>説明</p>
				<p>KeyboardEvent.key が押したキーの値。詳細については、https://developer.mozilla.org/en-US/docs/</p>

名前	型	説明		
		名前	型	説明
				名前
				Web/API/KeyboardEvent/keyboardEvent/keyを参照してください。
				KeyboardEvent.locationするキーの配列。キーボードに

名前	型	説明		
		名前	型	説明
				説明
				おけるキーの場所。詳細については、https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/location

名前	型	説明		
		名前	型	説明

戻り値:

型

void

requestDisplayConfig(highColorAccuracy) → {Promise|Promise.<{code: [DisplayConfigErrorCode](#), message: string}>}

NICE DCV サーバーから更新されたディスプレイ設定をリクエストします。NICE DCV ウェブクライアント SDK 1.1.0 および NICE DCV サーバー 2022.0 以降で使用できます。

パラメータ:

名前	型	説明
highColorAccuracy	boolean	高い色精度を要求すべきかどうか。

戻り値:

Promise。拒否された場合、プロミスがエラーオブジェクトを返します。

型

Promise | Promise.<{code: [DisplayConfigErrorCode](#), message: string}>

requestDisplayLayout(layout) → {Promise|Promise.<{code: [ResolutionErrorCode](#), message: string}>}

接続の更新済みディスプレイレイアウトをリクエストします。

パラメータ:

名前	型	説明
layout	Array.< Monitor >	レイアウト内のリクエストされたディスプレイ。

戻り値:

Promise。拒否された場合、エラーオブジェクトが送られます。

型

Promise | Promise.<{code: [ResolutionErrorCode](#), message: string}>

requestResolution(width, height) → {Promise|Promise.<{code: [ResolutionErrorCode](#), message: string}>}

NICE DCV サーバーから更新されたディスプレイ解像度をリクエストします。

パラメータ:

名前	型	説明
width	number	リクエストする幅 (ピクセル)。許容される最小値は 0 です。
height	number	リクエストする高さ (ピクセル)。許容される最小値は 0 です。

戻り値:

Promise。拒否された場合、プロミスがエラーオブジェクトを返します。

型

Promise | Promise.<{code: [ResolutionErrorCode](#), message: string}>

sendKeyboardEvent(event) → {boolean}

キーボードショートカットイベントを送信します。キーボードイベントの詳細については、<https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent> を参照してください。有効なキーボードイベントには、keydown、keypress、keyup が含まれます。キーボードイベントの詳細については、<https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent#events> を参照してください。

パラメータ:

名前	型	説明
event	KeyboardEvent	送信するキーボードイベント。

戻り値:

イベントが有効でない場合、関数が `false` を返します。イベントが有効である場合、関数が `true` を返します。

型

`boolean`

sendKeysShortcut(shortcut) → {void}

キーボードショートカットを送信します。この関数を使用して、完全な `keydown` シーケンスまたは `keyup` シーケンスを送信します。例えば、`Ctrl+Alt + Del` を送信すると、全てのキーに対して `keydown` イベントが送信され、続いて `keyup` イベントが送信されます。単一のキーを送信したい場合であってもこの関数を使用してください。

パラメータ:

名前	型	説明						
<code>shortcut</code>	<code>Array.<Object></code>	送信するキーの配列。 <table border="1" data-bbox="1068 1123 1507 1879"> <thead> <tr> <th>名前</th> <th>型</th> <th>説明</th> </tr> </thead> <tbody> <tr> <td><code>key</code></td> <td><code>KeyboardEvent.key</code></td> <td>ユーザーが押したキーの値。詳細については、https://developer.mozilla.org/en-US/docs/</td> </tr> </tbody> </table>	名前	型	説明	<code>key</code>	<code>KeyboardEvent.key</code>	ユーザーが押したキーの値。詳細については、 https://developer.mozilla.org/en-US/docs/
名前	型	説明						
<code>key</code>	<code>KeyboardEvent.key</code>	ユーザーが押したキーの値。詳細については、 https://developer.mozilla.org/en-US/docs/						

名前	型	説明		
		名前	型	説明
				Web/API/KeyboardEvent/keyを参照してください。
		location	KeyboardEvent.location	送信するキーの配列。キーボードにおけるキーの場所。詳細については、 https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/location

名前	型	説明		
		名前	型	説明
				を参照してください。

戻り値:

型

void

setDisplayQuality(min, maxopt) → {void}

接続に使用する画質を設定します。有効範囲は 0~100 で、1 が最低画質、100 が最高画質になります。現在の値を維持するには 0 を指定します。

パラメータ:

名前	型	属性	説明
min	number		最低画質。
max	number	<optional>	最高画質。

戻り値:

型

void

`setDisplayScale(scaleRatio, displayId) → {Promise|Promise.<{code: ResolutionErrorCode, message: string}>}` (DEPRECATED)

バージョン 1.3.0 以降は非推奨です。もうディスプレイのスケールを設定する必要はなくなりました。マウス座標は内部で自動的に管理されます。

クライアント側でディスプレイのスケールリングが行われたを NICE DCV に通知します。これを使用して、クライアントの表示比率に合わせてマウスイベントのスケールリングを行う必要があることをサーバーに通知します。

パラメータ:

名前	型	説明
scaleRatio	フロート	使用するスケールリング比率。厳密に正の数である必要があります。
displayId	number	スケールリングを行うディスプレイの ID。

戻り値:

Promise。拒否された場合、プロミスがエラーオブジェクトを返します。

型

Promise | Promise.<{code: [ResolutionErrorCode](#), message: string}>

`setKeyboardQuirks(quirks) → {void}`

クライアントコンピュータのキーボード特異性を設定します。

パラメータ:

名前	型	説明
quirks	オブジェクト	有効または無効にするキーボード特異性。

名前	型	説明		
		名前	型	説明
		macOptiToAlt	boolean	macOSのOptionキーをAltにマップするには、trueを指定します。それ以外の場合は、falseを指定します。
		macCommandToControl	boolean	macOSのCommandキーをCtrlにマップするには、trueを指定します。それ以外の場合

名前	型	説明		
		名前	型	説明
				は、falseを指定します。

戻り値:

型

void

`setMaxDisplayResolution(maxWidth, maxHeight) → {void}`

接続に使用する最高表示解像度を設定します。

パラメータ:

名前	型	説明
maxWidth	number	最大表示幅 (ピクセル)。許容される最小値は 0 です。
maxHeight	number	最大表示高さ (ピクセル)。許容される最小値は 0 です。

戻り値:

型

void

`setMicrophone(enable)` → {Promise|Promise.<{code: [AudioErrorCode](#), message: string}>}

マイクを有効または無効にします。

パラメータ:

名前	型	説明
enable	boolean	マイクを有効にするには、true を指定します。マイクを無効にするには、false を指定します。

戻り値:

Promise。拒否された場合、プロミスがエラーオブジェクトを返します。

型

Promise | Promise.<{code: [AudioErrorCode](#), message: string}>

`setMinDisplayResolution(minWidth, minHeight)` → {void}

接続に使用する最低表示解像度を設定します。アプリケーションによっては、最低表示解像度が必要になる場合があります。必要な最低解像度がクライアントでサポートされている最高解像度よりも大きい場合は、サイズ変更戦略が使用されます。この関数は慎重に使用してください。サイズ変更戦略により、マウスとタッチ入力システムの精度が低下する可能性があります。

パラメータ:

名前	型	説明
minWidth	number	最小表示幅 (ピクセル)。許容される最小値は 0 です。
minHeight	number	最小表示高さ (ピクセル)。許容される最小値は 0 です。

戻り値:

型

void

setUpUploadBandwidth(value) → {number}

NICE DCV サーバーへのファイルのアップロードに使用する最大帯域幅を設定します。

パラメータ:

名前	型	説明
value	number	アップストリーム帯域幅の上限 (kbps)。有効範囲は 1024 ~ 102400 kbps です。

戻り値:

- 設定された帯域幅限界。サーバーでファイルストレージ機能が無効になっている場合は null。

型

number

setVolume(volume) → {void}

オーディオに使用するボリュームレベルを設定します。有効範囲は 0 ~ 100 で、0 が最小ボリューム、100 が最高ボリュームです。

パラメータ:

名前	型	説明
volume	number	使用するボリュームレベル。

戻り値:

型

void

setWebcam(enable, deviceId) → {Promise|Promise.<{code: [WebcamErrorCode](#), message: string}>}

ウェブカメラを有効または無効にします。

パラメータ:

名前	型	説明
enable	boolean	ウェブカメラを有効にするには true を指定します。ウェブカメラを無効にするには false を指定します。
deviceId	string	ウェブカメラのデバイス ID。

戻り値:

Promise は、成功した場合、アタッチ/デタッチされたウェブカメラのデバイス ID に対する解決策を見つけます。拒否された場合、プロミスがエラーオブジェクトを返します。

型

Promise | Promise.<{code: [WebcamErrorCode](#), message: string}>

syncClipboards() → {boolean}

ローカルクライアントクリップボードとリモート NICE DCV サーバークリップボードを同期します。自動コピーがブラウザでサポートされている必要があります。

戻り値:

クリップボードが同期されている場合、関数が `true` を返します。クリップボードが同期されていない場合、またはブラウザで自動コピーがサポートされていない場合、関数が `false` を返します。

型

`boolean`

認証クラス

`dcm` モジュールの [authenticate メソッド](#) を呼び出して認証トークンを取得する場合は認証クラスを使用する必要があります。使用方法を示した例については「[開始方法](#)」セクションを参照してください。

エクスポート

- [方法](#)

方法

リスト

- [retry\(\) → {void}](#)
- [sendCredentials\(credentials\) → {void}](#)

`retry() → {void}`

認証プロセスを再試行します。

戻り値:

型

`void`

`sendCredentials(credentials) → {void}`

クライアントから提供された認証情報を NICE DCV サーバに送信します。

パラメータ:

名前	型	説明
credentials	オブジェクト	提供された認証情報を含むオブジェクト。認証情報の名前と型は、チャレンジで指定された名前と型と同じでなければなりません。

戻り値:

型

void

リソースクラス

リソースクラスは、印刷またはダウンロードされたばかりの対応ファイルを取得または破棄できます。これらのアクションを実行すると、対応するオブザーバー関数 [filePrinted](#) と [fileDownload](#) がリソースオブジェクトを唯一の引数としてそれぞれ呼び出されます。これらのリソースを許可または拒否して、参照するファイルを取得または破棄できます。

エクスポート

- [方法](#)

方法

リスト

- [accept\(urlParameters\) → {void}](#)
- [decline\(\) → {void}](#)

`accept(urlParameters) → {void}`

リソースをローカルにダウンロードします。

パラメータ:

名前	型	説明
<code>urlParameters</code>	オブジェクト	リソースを取得するリクエストに渡された URL 検索パラメータのキーと値のペアを含むオプションのオブジェクト。

戻り値:

型

`void`

`decline()` → `{void}`

リソースを破棄します。

戻り値:

型

`void`

NICE DCV ウェブ UI SDK

JavaScript React コンポーネントライブラリ。現在は、NICE DCV サーバーに接続し、ツールバーをレンダリングしてリモートストリームと通信する `DCVViewer` という 1 つの React コンポーネントをエクスポートしています。

エクスポート

- [コンポーネント](#)

コンポーネント

リスト

- [DCVViewer](#)

DCVViewer

リモートストリームとの通信に役立つ機能を備えたツールバーをレンダリングする React コンポーネント。

プロパティ:

リスト

- [dcv](#)
- [UIConfig](#)

dcv

名前	型	必須	説明
dcv	オブジェクト	Yes	NICE DCV サーバーへの接続を確立するために必要なプロパティを定義し、ログレベルと、NICE DCV ウェブクライアント SDK アセットをロードして DCV リソースにアクセスする URL を設定するオブジェクト。

名前	型	必須	説明																
			<table border="1"> <thead> <tr> <th>名前</th> <th>型</th> <th>必須</th> <th>説明</th> </tr> </thead> <tbody> <tr> <td>ses</td> <td>文字列</td> <td>Yes</td> <td>NICE DCV セッション ID。</td> </tr> <tr> <td>aut</td> <td>文字列</td> <td>Yes</td> <td>セッションへの接続に使用する認証トークン。</td> </tr> <tr> <td>ser</td> <td>文字列</td> <td>Yes</td> <td>実行中の NICE DCV サー</td> </tr> </tbody> </table>	名前	型	必須	説明	ses	文字列	Yes	NICE DCV セッション ID。	aut	文字列	Yes	セッションへの接続に使用する認証トークン。	ser	文字列	Yes	実行中の NICE DCV サー
名前	型	必須	説明																
ses	文字列	Yes	NICE DCV セッション ID。																
aut	文字列	Yes	セッションへの接続に使用する認証トークン。																
ser	文字列	Yes	実行中の NICE DCV サー																

名前	型	必須	説明			
			名前	型	必須	説明
						バー の ホ ス ト 名 と ポ ー ト を 次 の 形 式 で 指 定 し ま す: https:// d cv_host_a ddress:po rt。 例: https:// m y- dcv- ser ver:8443

名前	型	必須	説明			
			名前	型	必須	説明
			bas	文字列	Yes	SDK ファイルのロード元である絶対 URL または相対 URL。
			resource	文字列	No (default: "")	DCV リソースのアクセス元で

名前	型	必須	説明			
			名前	型	必須	説明
						ある絶対URLまたは相対URL。
			onD ect	func	No (def: () => {})	コー ル バッ ク 関 数 は 、NICE DCV サー バー との 接 続 を 切 断 し た

名前	型	必須	説明			
			名前	型	必須	説明
						ときに呼び出され、接続が閉じられます。
log	Log!	No				ビューア 1. で使用するログレベル。 (def: Log INF)

UIConfig

名前	型	必須	説明								
uiConfig	オブジェクト	No (default: {})	<p>ツールバーを表示するかどうか、および全画面表示ボタンとマルチモニターボタンをツールバーに表示するかどうかを設定するプロパティを定義するオブジェクト。</p> <table border="1"> <thead> <tr> <th>名前</th> <th>型</th> <th>必須</th> <th>説明</th> </tr> </thead> <tbody> <tr> <td>too</td> <td>オブジェクト</td> <td>No (default: {})</td> <td>ツールバーの設定オプションを定義するオブジェクト。</td> </tr> </tbody> </table>	名前	型	必須	説明	too	オブジェクト	No (default: {})	ツールバーの設定オプションを定義するオブジェクト。
名前	型	必須	説明								
too	オブジェクト	No (default: {})	ツールバーの設定オプションを定義するオブジェクト。								

名前	型	必須	説明			
			名前	型	必須	説明
						<p>visible (default: true) の表示/非表示を定義するオプション。</p> <p>full-screen (default: true) の全画面表示ボ</p>

名前	型	必須	説明			
			名前	型	必須	説明
						<p>説明欄 タンの表示/非表示を定義するオプション。</p> <p>Default (Default) 直ぐ) のマルチモニターボタ</p>

名前	型	必須	説明			
			名前	型	必須	説明
						説明欄の表示/非表示を定義するオプション。

NICE DCV ウェブクライアント SDK のリリースノートとドキュメント履歴

このページには、NICE DCV ウェブクライアント SDK のリリースノートとドキュメント履歴が掲載されています。

トピック

- [NICE DCV ウェブクライアント SDK リリースノート](#)
- [ドキュメント履歴](#)

NICE DCV ウェブクライアント SDK リリースノート

このセクションでは、NICE DCV ウェブクライアント SDK のリリースノートを一覧で示します。

トピック

- [1.5.6 — 2023 年 11 月 9 日](#)
- [1.4.4 — 2022 年 6 月 29 日](#)
- [1.4.0 — 2023 年 3 月 28 日](#)
- [1.3.1 — 2022 年 12 月 9 日](#)
- [1.3.0 — 2022 年 11 月 11 日](#)
- [1.2.1 — 2022 年 7 月 21 日](#)
- [1.2.0 — 2022 年 6 月 29 日](#)
- [1.1.3 — 2022 年 5 月 23 日](#)
- [1.1.2 — 2022 年 5 月 19 日](#)
- [1.1.1 — 2022 年 3 月 23 日](#)
- [1.1.0 — 2022 年 2 月 23 日](#)
- [1.0.4 — 2021 年 12 月 20 日](#)
- [1.0.3 — 2021 年 9 月 1 日](#)
- [1.0.2 — 2021 年 7 月 30 日](#)
- [1.0.1 — 2021 年 5 月 31 日](#)
- [1.0.0 — 2021 年 3 月 24 日](#)

1.5.6 — 2023 年 11 月 9 日

バージョン	リリースノート
<ul style="list-style-type: none">セマンティックバージョン: 1.5.6ビルド: 659	<p>変更とバグ修正</p> <ul style="list-style-type: none">ストリームのデコードとレンダリングのパフォーマンスが向上しました。Internet Explorer 11 のサポートが除外されました。

1.4.4 — 2022 年 6 月 29 日

バージョン	リリースノート
<ul style="list-style-type: none">セマンティックバージョン: 1.4.4ビルド: 573	<p>変更とバグ修正</p> <ul style="list-style-type: none">ビューアの UI コンポーネントは、サポートされるブラウザで <code>navigator.keyboard.lock</code> API を使用して、全画面で特殊キーを処理するようになりました。Chrome 114 以降を使用すると色が不適切になる問題を修正しました。WebCodecs の検出が改善されました。ウィンドウに入るときのマウスポタンの状態に関する問題を修正しました。macOS で修飾キーが押されたままになる問題を修正しました。

バージョン	リリースノート
	<p>ネットワーク状態が低下したときのオーディオの堅牢性が向上しました。</p> <ul style="list-style-type: none"> メモリリークを修正しました。 時間とレベルを含むようにログを改善しました。

1.4.0 — 2023 年 3 月 28 日

バージョン	リリースノート
<ul style="list-style-type: none"> セマンティックバージョン: 1.4.0 ビルド: 476 	<p>新機能</p> <ul style="list-style-type: none"> 複数のファイルをアップロードする新しい <code>uploadFiles</code> メソッドを <code>FileStorage</code> オブジェクトに追加しました。 Viewer UI コンポーネントは、ドラッグアンドドロップでファイルのアップロードを開始できるようになりました。 がオーディオとウェブカメラにも <code>WebCodecs</code> ブラウザ API が使用されるようになりました。 <p>変更とバグ修正</p> <ul style="list-style-type: none"> 同じページから繰り返し接続することによるメモリリークを修正しました。

バージョン	リリースノート
	<p><code>setUploadBandwidth</code> で 1 Gbps までの値が許可されるようになりました。</p> <ul style="list-style-type: none"> UI コンポーネントのレンダリングを最適化しました。 Windows でのアニメーションカーソルのサポートが修正されました。 同じ操作でテキストデータと画像データの両方が存在する場合のクリップボードサポートの問題を修正しました。 Webcam API の堅牢性を向上しました。リクエストが進行中の間は設定を変更できず、<code>webcam.setEnabled</code> は進行中のリクエストのデバイス ID を追跡し、Promise を返すようになりました。Viewer UI コンポーネントは、エラーが発生した場合に通知を表示するようになりました。

1.3.1 — 2022 年 12 月 9 日

バージョン	リリースノート
<ul style="list-style-type: none"> セマンティックバージョン: 1.3.1 ビルド: 413 	<p>変更とバグ修正</p> <ul style="list-style-type: none"> タイムゾーンリダイレクト UI がサーバーと同期しなくなる問題を修正しました。 複数回の再接続後のメモリリークを修正しました。

バージョン	リリースノート
	<p>切断時に空白ページになる問題を修正しました。</p> <ul style="list-style-type: none">オーディオデコーダーを終了する際にコンソールに警告が表示されるバグを修正しました。

1.3.0 — 2022 年 11 月 11 日

バージョン	リリースノート
<ul style="list-style-type: none">セマンティックバージョン: 1.3.0ビルド: 407	<p>新機能</p> <ul style="list-style-type: none">UI ビューアーコンポーネントに Cloudscape (https://cloudscape.design) を導入しました。タイムゾーンリダイレクトのサポートを追加しました。 <p>変更とバグ修正</p> <ul style="list-style-type: none">DCV ビューアにフォーカスがあると、非同期クリップボードで更新されない問題を修正しました。クライアント側でディスプレイをスケールする場合、<code>setDisplayScale</code> 関数は不要になりました。

バージョン	リリースノート
	DCVViewer コンポーネントは、アンマウントされたときに自動的に <code>disconnect()</code> を呼び出すようになりました。

1.2.1 — 2022 年 7 月 21 日

バージョン	リリースノート
<ul style="list-style-type: none"> セマンティックバージョン: 1.2.1 ビルド: 358 	<p>変更とバグ修正</p> <ul style="list-style-type: none"> 2019.1 以前の NICE DCV サーバーへの接続に失敗する問題を修正しました。

1.2.0 — 2022 年 6 月 29 日

バージョン	リリースノート
<ul style="list-style-type: none"> セマンティックバージョン: 1.2.0 ビルド: 352 	<p>変更とバグ修正</p> <ul style="list-style-type: none"> 受信したフレームがサポートされている最大解像度 (4096x2160) よりも大きい場合にクラッシュするバグを修正しました。 リソースオブジェクト (<code>fileDownload</code> および <code>filePrinted</code> オブザーバーに引数として渡される) には、オブジェクト上で呼び出すことができる <code>accept</code> および <code>decline</code> メソッドが導入され、それぞれリソースのダウンロードと破棄を実行できるようになりました。

バージョン	リリースノート
	<ul style="list-style-type: none"> 切断時の自動クリップボード同期に関するマイナーバグを修正しました。

1.1.3 — 2022 年 5 月 23 日

バージョン	リリースノート
<ul style="list-style-type: none"> セマンティックバージョン: 1.1.3 ビルド: 329 	変更とバグ修正 <ul style="list-style-type: none"> web-url-path オプションを指定したときに正常に接続できない問題を修正しました。

1.1.2 — 2022 年 5 月 19 日

バージョン	リリースノート
<ul style="list-style-type: none"> セマンティックバージョン: 1.1.2 ビルド: 322 	変更とバグ修正 <ul style="list-style-type: none"> 接続後に入力が正しく機能しなくなる問題を修正しました。 スケール比が 1 より大きい場合のマウス座標を修正しました。

1.1.1 — 2022 年 3 月 23 日

バージョン	リリースノート
<ul style="list-style-type: none"> 	変更とバグ修正

バージョン	リリースノート
<ul style="list-style-type: none"> セマンティックバージョン: 1.1.1 ビルド: 309 	<ul style="list-style-type: none"> サーバーとの通信がタイムアウトになった時に Transport Error を報告します。 高解像度のストリーミング時に繰り返し発生するデコードエラーを修正しました。

1.1.0 — 2022 年 2 月 23 日

バージョン	リリースノート
<ul style="list-style-type: none"> セマンティックバージョン: 1.1.0 ビルド: 295 	<p>新機能</p> <ul style="list-style-type: none"> DCVViewer React コンポーネントを含む NICE DCV ウェブ UI SDK ライブラリをリリースしました。 NICE DCV ウェブクライアント SDK を UMD モジュールと ES モジュールの両方としてエクスポートできます。 高い色精度のサポートを追加しました。 セッションに接続しているクライアントを一覧表示して操作する機能を追加しました。接続と切断の通知を追加しました。 <p>変更とバグ修正</p> <ul style="list-style-type: none"> WebCodecs のデコードのサポートを改善しました。 様々なキーボードを改善しました。

バージョン	リリースノート
	<ul style="list-style-type: none"> クリップボードが無効になっているときにセカンドスクリーンを開けないバグを修正しました。

1.0.4 — 2021 年 12 月 20 日

バージョン	リリースノート
<ul style="list-style-type: none"> セマンティックバージョン: 1.0.4 ビルド: 249 	<p>新機能</p> <ul style="list-style-type: none"> 同じページから複数の接続を開くことをサポートしました。 CDN からの SDK をロードすることをサポートしました。

1.0.3 — 2021 年 9 月 1 日

バージョン	リリースノート
<ul style="list-style-type: none"> セマンティックバージョン: 1.0.3 ビルド: 202 	<p>新機能</p> <ul style="list-style-type: none"> WebCodecs の実験的なサポート。これはデフォルトでは無効になっているため、新しいプロパティ <code>enableWebCodecs</code> を使用するオブジェクト <code>ConnectionConfig</code> を通じて有効にする必要があります。 クリップボード: Chromium ベースのブラウザのデータ型 <code>image/png</code> に対するサポートを追加しました。

バージョン	リリースノート
	<ul style="list-style-type: none"> サーバーのスクリーンショットを PNG イメージとして取得するためのオブザーバー/コールバックを追加しました (NICE DCV サーバー 2021.2 が必要)。 <p>変更とバグ修正</p> <ul style="list-style-type: none"> キーボード修飾子の処理を改良しました。

1.0.2 — 2021 年 7 月 30 日

バージョン	リリースノート
<ul style="list-style-type: none"> セマンティックバージョン: 1.0.2 ビルド: 167 	<ul style="list-style-type: none"> スタイラスイベントの圧力検出を修正しました。 Chrome の韓国語キーボードレイアウトのサポートを改善しました。

1.0.1 — 2021 年 5 月 31 日

バージョン	リリースノート
<ul style="list-style-type: none"> セマンティックバージョン: 1.0.1 ビルド: 141 	<ul style="list-style-type: none"> 接続エラーとクローズの理由の伝播を修正しました。 ファイルストレージチャンクの進行状況の更新を修正しました。

バージョン	リリースノート
	<p>ウェブカメラの取り扱いを改善しました。</p> <ul style="list-style-type: none"> オーディオイン処理を改善しました。

1.0.0 — 2021 年 3 月 24 日

バージョン	リリースノート
<ul style="list-style-type: none"> セマンティックバージョン: 1.0.0 ビルド: 81 	NICE DCV ウェブクライアント SDK の初回リリース。

ドキュメント履歴

次の表は、NICE DCV ウェブクライアント SDK の今回のリリースの内容をまとめたものです。

変更	説明	日付
NICE DCV ウェブクライアント SDK バージョン 1.5.6	NICE DCV ウェブクライアント SDK 1.5.6 が利用可能になりました。詳細については、 SDK v.1.5.6 をご参照ください。	2023 年 11 月 9 日
NICE DCV ウェブクライアント SDK バージョン 1.4.4	NICE DCV ウェブクライアント SDK 1.4.4 が利用可能になりました。詳細については、 SDK v.1.4.4 をご参照ください。	2023 年 6 月 29 日

変更	説明	日付
NICE DCV ウェブクライアント SDK バージョン 1.4.0	NICE DCV ウェブクライアント SDK 1.4.0 が利用可能になりました。詳細については、 SDK v.1.4.0 をご参照ください。	2023 年 3 月 28 日
NICE DCV ウェブクライアント SDK バージョン 1.3.1	NICE DCV ウェブクライアント SDK 1.3.1 が利用可能になりました。詳細については、 SDK v.1.3.1 をご参照ください。	2022 年 12 月 9 日
NICE DCV ウェブクライアント SDK バージョン 1.3.0	NICE DCV ウェブクライアント SDK 1.3.0 が利用可能になりました。詳細については、 SDK v.1.3.0 をご参照ください。	2022 年 11 月 11 日
NICE DCV ウェブクライアント SDK バージョン 1.2.0	NICE DCV ウェブクライアント SDK 1.2.0 が利用可能になりました。詳細については、 SDK v.1.2.0 をご参照ください。	2022 年 6 月 29 日
NICE DCV ウェブクライアント SDK バージョン 1.1.0	NICE DCV ウェブクライアント SDK 1.1.0 が利用可能になりました。詳細については、 SDK v.1.1.0 をご参照ください。	2022 年 2 月 23 日
NICE DCV ウェブクライアント SDK バージョン 1.0.1	タイプミスを修正しました。軽微なバグを修正しました。 SDK v.1.0.1 を参照してください。	2021 年 5 月 31 日
初回リリース	このコンテンツの初版です。	2021 年 3 月 24 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。