aws

ウェブクライアント SDK デベロッパーガイド

Amazon DCV

Amazon DCV: ウェブクライアント SDK デベロッパーガイド

Table of Contents

Amazon DCV ウェブクライアント SDK とは

> **Note**
>
> Amazon DCV は以前は NICE DCV と呼ばれていました。

Amazon 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。Amazon 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。

Amazon DCV ウェブクライアント SDK とは、独自の Amazon 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 Amazon DCV セッションへの接続や操作を行うことができます。

Amazon 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、Amazon 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 Amazon 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

Amazon DCV ウェブクライアント SDK の使用を開始する前に、Amazon DCV と Amazon 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、「Amazon DCV 管理者ガイド」を参照してください。

Amazon DCV ウェブクライアント SDK では、Amazon 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 Amazon 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 Amazon DCV サーバーへの接続
- Linux Amazon 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)
- マルチモニターのサポート

これらの機能の詳細については、「Amazon 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

Amazon 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。

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

バージョニング規則

Amazon 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。バージョニング規則は、通常、セマンティックバージョ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。

Amazon DCV ウェブクライアント SDK の開始方法

Amazon 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、Amazon DCV 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。

Amazon DCV ウェブクライアント SDK を開始するには

1. Amazon 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. Amazon DCV ウェブクライアント SDK アーカイブと Amazon DCV ウェブクライアント SDK アーカイブ署名を Amazon DCV 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 Amazon 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-1.8.7-858.zip.sign nice-dcv-web-client-
   sdk-1.8.7-858.zip
   ```

4. 署名が正常に検証された場合は、Amazon 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**
>
> - Amazon 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、フォルダ構造を保持する必要があります。
> - Amazon 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、Amazon 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。

Amazon 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、Amazon DCV サーバーを認証して接続する方法、および Amazon 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

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、Amazon DCV ウェブクライアント SDK で Amazon 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、Amazon DCV サーバーを接続する方法、および Amazon 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 Amazon 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 Amazon 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、Amazon 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 Amazon 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、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。。

Amazon 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、Amazon 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 Amazon 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 Amazon 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 Amazon 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) と認可トークン (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 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);
}
```

Amazon DCV サーバーに接続します。`firstFrame` コールバック関数は、Amazon 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。Amazon 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));
}
```

Amazon DCV 機能の操作

Amazon DCV 機能の可用性は、Amazon 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。

Amazon 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 Amazon 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、「Amazon DCV 管理者ガイド」の「Amazon 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

Amazon 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、Amazon 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、拒否されます。

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

Amazon DCV ウェブ UI SDK の使用

以下のチュートリアルでは、Amazon DCV サーバーに対して認証を行い、接続して Amazon 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

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、Amazon 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、「Amazon 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。

Amazon 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 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、Amazon 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 Amazon DCV ウェブクライアント SDK、Amazon 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 Amazon DCV サーバーに対して認証を行い、Amazon 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>
  );
}

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。。

Amazon 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、Amazon 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 Amazon 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 Amazon 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 Amazon 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) と認可トークン (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、「Amazon DCV ウェブ UI SDK のリファレンス」を参照してください。

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、Amazon 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

DCV プロトコルのクライアント側を実装するモジュール。

エクスポーズ

方法

リスト

authenticate(authParams) → {Authentication}

指定した Amazon 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 Amazon 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` 。
`authenticationToken`	文字列	認証に使用する認証トークン。
`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。

タイプ

Authentication

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

指定した Amazon 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。

タイプ

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:

タイプ

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:

タイプ

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。

タイプ:

- AudioErrorCode

(定数) AuthenticationError :AuthenticationErrorCode

AuthenticationError コードの列挙型。

タイプ:

- AuthenticationErrorCode

(定数) ChannelError :ChannelErrorCode

ChannelError コードの列挙型。

タイプ:

- ChannelErrorCode

(定数) ClosingReasonError :ClosingReasonErrorCode

ClosingReasonError コードの列挙型。

タイプ:

- ClosingReasonErrorCode

(定数) ConnectionError :ConnectionErrorCode

ConnectionError コードの列挙型。

タイプ:

- ConnectionErrorCode

(定数) CustomChannelError :CustomChannelErrorCode

CustomChannelError コードの列挙型。

タイプ:

- CustomChannelErrorCode

(定数) DisplayConfigError :DisplayConfigErrorCode

DisplayConfigError コードの列挙型。

タイプ:

- DisplayConfigErrorCode

(定数) FileStorageError :FileStorageErrorCode

FileStorageError コードの列挙型。

タイプ:

- FileStorageErrorCode

(定数) LogLevel :LogLevel

使用可能な SDK ログレベル。

タイプ:

- LogLevel

(定数) MultiMonitorError :MultiMonitorErrorCode

MultiMonitorError コードの列挙型。

タイプ:

- MultiMonitorErrorCode

(定数) ResolutionError :ResolutionErrorCode

ResolutionError コードの列挙型。

タイプ:

- ResolutionErrorCode

(定数) TimezoneRedirectionError :TimezoneRedirectionErrorCode

TimeZoneRerectionError コードの列挙型。

タイプ:

- TimezoneRedirectionErrorCode

(定数) TimezoneRedirectionSetting :TimezoneRedirectionSettingCode

TimezoneRedirectionSetting コードの列挙型。

タイプ:

- TimezoneRedirectionSettingCode

(定数) TimezoneRedirectionStatus :TimezoneRedirectionStatusCode

TimezoneRedirectionStatus の列挙型。

タイプ:

- TimezoneRedirectionStatusCode

(定数) バージョン

Amazon 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	文字列	拡張文字列。
versionStr	文字列	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。

タイプ:

- ScreenshotErrorCode

(定数) WebcamError :WebcamErrorCode

WebcameError コードの列挙型。

タイプ:

- 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)
- Channel

- 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

- モニタリング
- 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
- httpExtraSearchParamsCallback(メソッド、URL、本文)
- httpExtraHeadersCallback(メソッド、URL、本文)

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`

タイプ:

- 数値

authenticationCallbacks

認証コールバック

タイプ:

- 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。
httpExtraSearchParamsCallback	httpExtraSearchParamsCallback	リクエストを開始する前にカスタムクエリパラメータを認証 URL に挿入するために authenticate メソッドで呼び出されるコールバック関数。また、 connectメソッドで使用して、DCV サーバーへの WebSocket 接続を確立するときに使用する URL をカスタマイズすることもできます。

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

タイプ:

- 数値

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。 名前 / 型 / 説明 code / AuthenticationErrorCode / エラーコードです。 message / 文字列 / エラーメッ

名前	型	説明
		名前 / 型 / 説明 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 `require` `redenti` `s` / Array.<O ject> / リクエストされた認証情報オブジェクトの配列。 名 前 / 型 / 説明 n / s / リクエ

名前	型	説明
		名前 型 説明
		説明 ストされた認証情報の名前。
		t リクエスト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>	Amazon 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 `sessior` / string / Amazon DCV セッション ID。 `authTok` / 文字列 / Amazon DCV セッションの認証トークン。

Channel

指定できる使用可能なチャンネル。

タイプ:

- "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`

タイプ:

- 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 | dataSilert | autoCone | newDailable / 常に存在します。イベントの名前。

名前	型	説明

名前	型	属性	説明
clipb Data	Object | string		クリップボード内のデータ。
autoC	boolea	<optio >	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

名前	型	説明
		名前 / 型 / 属性 / 説明 かを示します。 maxDaze / 数値 / <optio> / クリップボードに配置できるデータの最大量。 error / 文字列 / <optio> / 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
- EXTERNAL_PROTOCOL_CONNECTION_EVICTED
- DISCONNECTION_REQUESTED

タイプ:

- 数値

Colorspace

指定できる使用可能な色空間。

タイプ:

- "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。

タイプ:

- 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が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	Amazon 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。
filePrinted	filePrintedCallback	ファイルが Amazon 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	Amazon 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	Amazon 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	Amazon DCV サーバーからライセンス状態に関する通知が

名前	型	説明
		送信されたときに呼び出されるコールバック関数。
idleWarningNotification	idleWarningNotificationCallback	Amazon 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	Amazon 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 (Amazon 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。
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。

名前	型	説明
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。
httpExtraSearchParamsCallback	httpExtraSearchParamsCallback	Amazon DCV サーバーへの WebSocket 接続を確立するときに URL をカスタマイズするために呼び出されるコールバック関数。SDK がリクエストを送信する前に、このコールバックを authenticate メソッドで使用して、クエリパラメータを認証 URL に動的に追加することもできます。
httpExtraHeadersCallback	httpExtraHeadersCallback	接続の確立中に HTTP リクエストにカスタムヘッダーを追加するために呼び出されるコールバック関数。

ConnectionConfig

Amazon DCV 接続の設定。

タイプ:

- オブジェクト

プロパティ:

名前	型	説明
url	string	実行中の Amazon DCV サーバーのホスト名と

名前	型	説明
		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` 。
`sessionId`	文字列	Amazon DCV セッション ID。
`authToken`	文字列	セッションへの接続に使用する認証トークン。
`baseUrl`	文字列	SDK ファイルのロード元である絶対 URL または相対 URL。
`resourceBaseUrl`	文字列	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`	文字列	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。

名前	型	説明
volumeLevel	integer	希望するボリュームレベル。有効範囲は 0 ~ 100 です。
clipboardAutoSync	boolean	互換性のあるウェブブラウザで、Amazon 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、Amazon 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。

名前	型	説明
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)
- GATEWAY_BUSY
- UNSUPPORTED_CREDENTIAL (DCV Server 2022.2 以降)
- TRANSPORT_ERROR

タイプ:

- 数値

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`

タイプ:

- 数値

dataChannelCallback(info)

Amazon 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。 名前 / 型 / 説明 `name` / string / データチャン

名前	型	説明
		名前 / 型 / 説明 ネルの名前。 token / 文字列 / 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。

名前	型	説明
		名前 / 型 / 説明 code / 数値 / 理由コード。 message / 文字列 / 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。 名前 / 型 / 説明 enabled / ブール型 / 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

名前	型	説明
		名前 / 型 / 説明 かどうかを示します。
displayId	数値	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

タイプ:

- 数値

displayLayoutCallback(serverWidth, serverHeight, heads)

ディスプレイレイアウトまたは解像度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
serverWidth	数値	プライマリディスプレイの幅 (ピクセル)。
serverHeight	数値	プライマリディスプレイの高さ (ピクセル)。

名前	型	説明
heads	Array.<Monitor>	Amazon 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 (Amazon 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 - Amazon 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 Amazon 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 - Amazon 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 Amazon 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-redirection - タイムゾーンリダイレクト機能の可用性を示します (Amazon DCV ウェブクライアント SDK バージョン 1.3.0 以降)。

タイプ:

- 文字列

featuresUpdateCallback(featuresList)

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。

パラメータ :

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

fileDownloadCallback(fileResource)

Amazon 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 `id` / string / ファイルの識別子。 `url` / 文字列 / ファイルのダウンロードに使用する URL。

名前	型	説明
		名前 / 型 / 説明 domain / 文字列 / リソースドメイン。 token / 文字列 / 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 Amazon 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。

名前	型	説明
		名前 / 型 / 説明 id / string / 印刷されたファイルの識別子。 url / 文字列 / 印刷されたファイルのダウンロードに使用するURL。 domain / 文字列 / リソースドメイン。この場合は printer。 token / 文字列 / 印刷されたファイルのダウンロードに使用する認証トークン

名前	型	説明
		名前 / 型 / 説明 。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。

タイプ:

- 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。
`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。

名前	型	説明
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	ブール型	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

タイプ:

- 数値

firstFrameCallback(resizeEnabled, relativeMouseModeEnabled, displayId)

Amazon 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	ブール型	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	数値	ディスプレイの識別子。

idleWarningNotificationCallback(disconnectionDateTime)

Amazon 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)

Amazon 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オーナーであるか

名前	型	説明
		名前 / 型 / 説明 どうかを示します。 connectnId / 数値 / 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)

Amazon 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。 status / 文字列 / ライセンスの

名前	型	説明
		名前 型 説明 ステータス。 message 文字列 メッセージ。 leftDay 数値 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 数値 ライセンスを取得していない接続の数。 licensingMode 文字列 ライセンス取

名前	型	説明
		名前 / 型 / 説明 得モード。 documentationUrl / 文字列 / 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。

タイプ:

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

モニタリング

タイプ:

- 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。 名前 / 型 / 説明 x / 数値 / ディスプレイヘッドの最初の x 座標。 y / 数値 / ディスプレイヘッドの最初の y 座標。 width / 数値 / ディスプレイヘッドの幅 (ピクセル)。 height / 数値 / ディスプレイヘッドの高さ (ピ

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	数値	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

タイプ:

- 数値

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。 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。 changec / 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	文字列	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	文字列	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

タイプ:

- 数値

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)

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。

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`

タイプ:

- 数値

serverInfo

タイプ:

- 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。 名前 / 型 / 説明 major / 数値 / メジャーバージョン番号。 minor / 数値 / マイナーバージョン番号。 revisic / 数値 / リビジョンバージョン番号。
os	文字列	OS。
arch	文字列	アーキテクチャ。

名前	型	説明
hostname	文字列	ホスト名。

stats

タイプ:

- オブジェクト

プロパティ:

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

タイプ:

- 数値

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`

タイプ:

- 数値

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

タイプ:

- 数値

WebcamErrorCode

DCV モジュールで使用できる WebcameError コードの列挙型

- SETTING_WEBCAM_FAILED
- CHANNEL_NOT_AVAILABLE

タイプ:

- 数値

httpExtraSearchParamsCallback(メソッド、URL、本文)

認証と接続の確立中にカスタムクエリパラメータを URLs に挿入するために呼び出されるコールバック関数。これにより、カスタムクエリパラメータを追加し、 AWS 署名バージョン 4 (SigV4) 署名値を追加して、外部システムを介した接続を保護および認可する機能など、高度な統合シナリオが可能になります。

このコールバックは、Amazon DCV サーバーへの WebSocket 接続を確立するときに使用する URL をカスタマイズするためにも使用されます。

パラメータ :

名前	型	説明
`method`	string	リクエストに使用されている HTTP メソッド。
`url`	文字列	リクエストに使用される URL。
`body`	文字列	リクエストボディのコンテンツ。

戻り値:

`URLSearchParams` URL に追加するカスタムクエリパラメータを含む オブジェクト。

タイプ

URLSearchParams

httpExtraHeadersCallback(メソッド、URL、本文)

接続の確立中に HTTP リクエストにカスタムヘッダー (など`Authorization`) を挿入するため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
`method`	string	リクエストに使用されている HTTP メソッド。
`url`	文字列	リクエストに使用される URL。
`body`	文字列	リクエストボディのコンテンツ。

戻り値:

HTTP リクエストに追加するカスタムヘッダーを表すキーと値のペアを含むオブジェクト。

タイプ

オブジェクト

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。

エクスポーズ

- 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}
- enableHighPixelDensity(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}
- setMicrophone(enable, deviceId) → {Promise|Promise.<{code: AudioErrorCode, message: string}>}
- 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 :

<table>
<tr><th>名前</th><th>型</th><th>説明</th></tr>
<tr><td><code>win</code></td><td>オブジェクト</td><td>ディスプレイをアタッチする必要があるウィンドウ。</td></tr>
<tr><td><code>displayConf</code></td><td>オブジェクト</td><td>ディスプレイの設定。
<table>
<tr><th>名前</th><th>型</th><th>属性</th><th>説明</th></tr>
<tr><td><code>displ</code></td><td>数値</td><td><optio></td><td>ディスプレイの ID。</td></tr>
<tr><td><code>displ vName</code></td><td></td><td></td><td>ディスプレイ div の名前。</td></tr>
</table>
</td></tr>
</table>

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。

タイプ

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`	ブール型		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。
`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`	数値	<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:

タイプ

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`	数値	デタッチするディスプレイの ID。

戻り値:

タイプ

void

disconnect() → {void}

Amazon DCV サーバーから切断して接続を閉じます。

戻り値:

タイプ

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 (Amazon 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`	ブール型	切断される接続の ID。

戻り値:

タイプ

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	ブール型	表示品質の更新を有効にするには、true を指定します。表示品質の更新を無効にするには、false を指定します。

戻り値:

タイプ

void

enableHighPixelDensity(enable) → {void}

クライアントで高ピクセル密度を有効または無効にします。

パラメータ :

名前	型	説明
enable	ブール型	高ピクセル密度を有効にするかどうか。

戻り値:

タイプ

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
`enable`	ブール型	タイムゾーンリダイレクトを有効にするには、`true` 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、`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。

タイプ

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:

タイプ

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。

タイプ

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

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

Amazon 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。

タイプ

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

getServerInfo() → {serverInfo}

Amazon 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。

タイプ

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。

タイプ

Promise | Promise.<{code: ScreenshotErrorCode, message: string}>

getStats() → {stats}

Amazon 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。

タイプ

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/

名前	型	説明
		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。

タイプ

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 Amazon DCV サーバーで作成された場合にそのチャネルを接続で開きます。

パラメータ :

名前	型	説明
`name`	string	チャネルの名前。
`authToken`	文字列	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`	文字列	データチャネルの名前空間。Amazon DCV ウェブクライアント SDK 1.2.0 および

名前	型	説明
		Amazon 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。

タイプ

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 Amazon 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。

タイプ

{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。 名前 / 型 / 説明 sequenc / Array.<Object> / 登録するキーボードショートカット。 名前 / 型 / 説明 k / K v / KeyboardEvent.key が押したキーの値。詳細については、https:

名前	型	説明
		名前 型 説明
		名前 型 説明
		// developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key を参照してください。
		location 送信するキーの

名前	型	説明
		名前 型 説明
		名 型 説明
		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/

名前	型	説明
		名前 型 説明 API / KeyboardEvent/location 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る意図されたアクション。

名前	型	説明
		名前 型 説明
		名前 型 説明
		k k ユーザが押したキーの値。詳細については、https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/KeyboardEvent.key

名前	型	説明
		名前 型 説明
		key を参照してください。
		location 送信するキーの配列。キーボードにおけるキーの場

名前	型	説明
		名前 型 説明 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

名前	型	説明
		名前 型 説明 名前 型 説明さい。

戻り値:

タイプ

void

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

Amazon 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。Amazon DCV ウェブクライアント SDK 1.1.0 および Amazon 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`	ブール型	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。

タイプ

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。

タイプ

Promise | Promise.<{code: ResolutionErrorCode, message: string}>

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

Amazon 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`	数値	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`	数値	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。

タイプ

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。

タイプ

boolean

sendKeyboardShortcut(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 :

<table>
<tr><th>名前</th><th>型</th><th>説明</th></tr>
<tr><td>shortcut</td><td>Array.<Object></td><td>送信するキーの配列。
<table>
<tr><th>名前</th><th>型</th><th>説明</th></tr>
<tr><td>key</td><td>Keyboard
vent.key</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/Web/API/KeyboardEvent/key を参照してください。</td></tr>
<tr><td>locatic</td><td>Keyboard
vent.loca
tion</td><td>送信するキーの配列。キーボードにおけ</td></tr>
</table></td></tr>
</table>

名前	型	説明		
		名前	型	説明
				るキーの場所。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。

戻り値:

タイプ

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	数値		最低画質。
max	数値	<optional>	最高画質。

戻り値:

タイプ

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リングが行われたことを Amazon 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	使用するスケーリング比率。厳密に正の数である必要があります。
displayId	数値	スケー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。

タイプ

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 :

<table>
<tr><th>名前</th><th>型</th><th>説明</th></tr>
<tr><td>quirks</td><td>オブジェクト</td><td>有効または無効にするキーボード特異性。
<table>
<tr><th>名前</th><th>型</th><th>説明</th></tr>
<tr><td>macOptionToAlt</td><td>ブール型</td><td>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。</td></tr>
<tr><td>macCommandToCtrl</td><td>boolean</td><td>macOS の Command</td></tr>
</table>
</td></tr>
</table>

名前	型	説明
		名前 / 型 / 説明 キーを Ctrl にマップするには、`true` を指定します。それ以外の場合は、`false`を指定します。

戻り値:

タイプ

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`	数値	最大表示幅 (ピクセル)。許容される最小値は `0` です。

名前	型	説明
maxHeight	数値	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:

タイプ

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	ブール型	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。

タイプ

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`	数値	最小表示幅 (ピクセル)。許容される最小値は `0` です。
`minHeight`	数値	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:

タイプ

void

setUploadBandwidth(value) → {number}

Amazon 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`	数値	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`。

タイプ

数値

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	数値	使用するボリュームレベル。

戻り値:

タイプ

void

setMicrophone(enable, deviceId) → {Promise|Promise.<{code: AudioErrorCode, message: string}>}

(実験的 - 将来変更される可能性があります) 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	ブール型	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。
deviceId	文字列	マイクのデバイス ID。deviceId が指定されていない場合は、default deviceId が使用されます。

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。

タイプ

Promise | Promise.<{code: AudioErrorCode, message: string}>

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`	ブール型	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`	文字列	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。

タイプ

Promise.<string> | 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 Amazon 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。

タイプ

boolean

認証クラス

dcv 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。

エクスポーズ

- 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:

タイプ

void

sendCredentials(credentials) → {void}

クライアントから提供された認証情報を Amazon 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:

タイプ

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。

エクスポーズ

- 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
`urlParameters`	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:

タイプ

void

decline() → {void}

リソースを破棄します。

戻り値:

タイプ

void

Amazon DCV ウェブ UI SDK

JavaScript React コンポーネントライブラリ。現在は、Amazon 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。

エクスポーズ

- 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	はい	Amazon 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、Amazon 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
			名前 型 必須 説明 ses 文字列 はい Amazon DCV セッション ID。 aut Strir はい セッションへの接続に使用する認証トークン。 ser Strir はい 実行中の Amazon 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://dcv_host_address:port。例: https://my-dcv-server:8443

名前	型	必須	説明
			名前 / 型 / 必須 / 説明
			bas / Strir / はい / SDK ファイルのロード元である絶対 URL または相対 URL。
			res ase / Strir / No (defa "") / 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。 onDisconnect 関数 No (default: () => {}) コールバック関数は、Amazon 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 Logl No (def Log INF ビューアで使用するログレベル。 l.

UIConfig

名前	型	必須	説明
uiConfig	オブジェクト	No (default: {})	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。 名前 / 型 / 必須 / 説明 too / オブジェクト / No (def {}) / 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
			説明
			displayOnMobile (default: true) の表示/非表示を定義するオプション。
			displayOnFullscreen (default: true) の全画面表示ボ

名前	型	必須	説明
			名前 型 必須 説明
			説明 必須
			ボタンの表示/非表示を定義するオプション。
			必須 (default: true)
			画面のマルチモニターボタ

名前	型	必須	説明
			名前 型 必須 説明
			説明

ョン。ンの表示/非表示を定義するオプシ

Amazon 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、Amazon 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

Amazon 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、Amazon DCV ウェブクライアント SDK のリリースノートをリリース日別に示します。

トピック

- 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

1.8.7 — 2024 年 10 月 31 日

ビルド番号	変更とバグ修正	
• セマンティックバージョン: 1.8.7 • ビルド: 858	• Firefox 130 以降のレンダリングを修正しました	

1.8.4 — 2024 年 10 月 1 日

ビルド番号	新機能	変更とバグ修正
• セマンティックバージョン: 1.8.4 • ビルド: 840	以下の機能が追加されています。 • 名前を「Amazon DCV ウェブクライアント SDK」に変更 • 高 DPI ディスプレイ用の新しい API enableHighPixelDensity を追加 • 互換性のあるブラウザでマイクを選択するための実験的な API setMicrophone を追加 •	• ウェブカメラの取り扱いを改善しました。 • オーディオ再生処理を改善しました。 • WebCodecs の取り扱いを改善しました。 • マイクとウェブカメラのプラグとプラグの取り外しを改善しました。

ビルド番号	新機能	変更とバグ修正
	• 新しい接続エラー GATEWAY_BUSY、UNSUPPORTED_CREDENTIAL、および TRANSPORT_ERROR を追加 • 新しい終了理由 EXTERNAL_PROTOCOL_CONNECTION_EVICTED および DISCONNECTION_REQUESTED を追加	• マルチモニター時のリモートウィンドウのドラッグを改善しました。 • ファイルストレージのアップロードとダウンロードのアクセス許可が正しく伝播されるようになりました。 • レンダリングに関する軽微な修正

1.5.10 – 2023 年 12 月 19 日

バージョン	リリースノート
• セマンティックバージョン: 1.5.10 • ビルド: 684	変更とバグ修正 • ストリームのデコードエラーを修正

1.5.6 — 2023 年 11 月 9 日

バージョン	リリースノート
• セマンティックバージョン: 1.5.6 • ビルド: 659	変更とバグ修正 • 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。 •

バージョン	リリースノート
	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
• セマンティックバージョン: 1.4.4 • ビルド: 573	変更とバグ修正 • 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 `navigator.keyboard.lock` 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ボ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。 • 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。 • 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
• セマンティックバージョン: 1.4.0 • ビルド: 476	新機能 • 複数のファイルをアップロードする新しい `uploadFiles` メソッドを `FileStorage` 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。 • オーディオとウェブカメラにも WebCodecs ブラウザ API が使用されるようになりました。 変更とバグ修正 • 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。 • `setUploadBandwidth` で 1 Gbps までの値が許可されるようになりました。 • 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。

バージョン	リリースノート
	• Webcam API の堅牢性が向上しました。リクエストの進行中は設定を変更できず、`webcam.setEnabled` 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
• セマンティックバージョン: 1.3.1 • ビルド: 413	変更とバグ修正 • 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。 • 切断時に空白ページになる問題を修正しました。 • 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

バージョン	リリースノート
• セマンティックバージョン: 1.3.0 • ビルド: 407	新機能 • UI ビューアコンポーネントに 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。 変更とバグ修正 • 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、`setDisplayScale` 関数は不要になりました。 • `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 `disconnect()` 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
• セマンティックバージョン: 1.2.1 • ビルド: 358	変更とバグ修正 • 2019.1 以前の Amazon 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
• セマンティックバージョン: 1.2.0 • ビルド: 352	変更とバグ修正 • 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 (`fileDownload` および `filePrinted` 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 `accept` および `decline` 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。 • 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
• 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 • 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
• 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 • 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
• セマンティックバージョン: 1.1.1 • ビルド: 309	変更とバグ修正 • 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
• セマンティックバージョン: 1.1.0 • ビルド: 295	新機能 •

バージョン	リリースノート
	`DCVViewer` React コンポーネントを含む Amazon DCV ウェブ UI SDK ライブラリをリリースしました。 • Amazon 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。 変更とバグ修正 • WebCodecs のデコードのサポートを改善しました。 • キーボードのさまざまな改善を行いました。 • 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
• セマンティックバージョン: 1.0.4 •	新機能 •

バージョン	リリースノート
ビルド: 249	同じページから複数の接続を開けるようになりました。 • CDN から SDK をロードできるようになりました。

1.0.3 — 2021 年 9 月 1 日

バージョン	リリースノート
• セマンティックバージョン: 1.0.3 • ビルド: 202	新機能 • 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 `enableWebCodecs` を使用するオブジェクト `ConnectionConfig` 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 `image/png` に対するサポートを追加しました。 • 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 (Amazon DCV サーバー 2021.2 が必要)。 変更とバグ修正 • 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
• セマンティックバージョン: 1.0.2 • ビルド: 167	• 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
• セマンティックバージョン: 1.0.1 • ビルド: 141	• 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。 • ウェブカメラの取り扱いを改善しました。 • 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
• セマンティックバージョン: 1.0.0 • ビルド: 81	Amazon DCV ウェブクライアント SDK の初回リリース。

バージョン	リリースノート

ドキュメント履歴

次の表は、Amazon 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
Amazon DCV ウェブクライアント SDK バージョン 1.8.4	Amazon DCV ウェブクライアント SDK 1.8.4 が利用可能になりました。詳細については、SDK v.1.8.4 を参照してください。	2024 年 10 月 1 日
Amazon DCV ウェブクライアント SDK バージョン 1.5.6	Amazon 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
Amazon DCV ウェブクライアント SDK バージョン 1.4.4	Amazon 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
Amazon DCV ウェブクライアント SDK バージョン 1.4.0	Amazon 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
Amazon DCV ウェブクライアント SDK バージョン 1.3.1	Amazon 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	2022 年 12 月 9 日

変更	説明	日付
	は、SDK v.1.3.1 をご参照ください。	
Amazon DCV ウェブクライアント SDK バージョン 1.3.0	Amazon 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
Amazon DCV ウェブクライアント SDK バージョン 1.2.0	Amazon 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 1 月 29 日
Amazon DCV ウェブクライアント SDK バージョン 1.1.0	Amazon 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
Amazon 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。