



# NICE DCV

## Web Client SDK Developer Guide



## **NICE DCV: Web Client SDK Developer Guide**

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

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

---

## Table of Contents

What NICE DCV Web Client SDK? .....	1
Prerequisites .....	1
Supported features .....	1
Browser support .....	2
Versioning convention .....	2
Getting started .....	3
Connect to a NICE DCV server and get the first frame .....	3
Step 1: Prepare your HTML page .....	4
Step 2: Authenticate, connect, and get the first frame .....	4
Bonus: Automatically create an HTML login form .....	6
Working with NICE DCV features .....	6
Understanding the featuresUpdate callback function .....	7
Handling feature updates .....	7
SDK reference .....	8
DCV module .....	8
Methods .....	8
Members .....	10
Type and callback definitions .....	12
Connection Class .....	35
Methods .....	8
Authentication Class .....	51
Methods .....	8
Release Notes and Document History .....	52
Release Notes .....	52
1.0.3 — September 01, 2021 .....	52
1.0.2 — July 30, 2021 .....	52
1.0.1 — May 31, 2021 .....	53
1.0.0 — March 24, 2021 .....	53
Document History .....	53

# What Is the NICE DCV Web Client SDK?

NICE DCV is a high-performance remote display protocol. It lets you securely deliver remote desktops and application streaming from any cloud or data center to any device, over varying network conditions. By using NICE DCV with Amazon EC2, you can run graphics-intensive applications remotely on Amazon EC2 instances. You can then stream the results to more modest client machines, which eliminates the need for expensive dedicated workstations.

The NICE DCV Web Client SDK is a JavaScript library that you can use to develop your own NICE DCV web browser client applications. Your end users can use these applications to connect to and interact with a running NICE DCV session.

Using the NICE DCV Web Client SDK as a building block, you can build customized web applications that provide users with instant access to their desktop or applications from anywhere, with a responsive and fluid performance that is almost indistinguishable from a natively installed application.

This guide explains how to use the NICE DCV Web Client SDK to build your custom web browser client applications to interact with NICE DCV sessions within your workflows.

## Topics

- [Prerequisites \(p. 1\)](#)
- [Supported features \(p. 1\)](#)
- [Browser support \(p. 2\)](#)
- [Versioning convention \(p. 2\)](#)

## Prerequisites

Before you start working with the NICE DCV Web Client SDK, ensure that you're familiar with NICE DCV and NICE DCV sessions. For more information, see the [NICE DCV Administrator Guide](#).

The NICE DCV Web Client SDK supports NICE DCV server version 2020 and later.

## Supported features

You can build custom web browser client applications that support the following NICE DCV features:

- Connect to Windows NICE DCV servers
- Connect to Linux NICE DCV servers
- Manage streaming modes
- Transfer files
- Print from sessions
- Copy and paste
- Stereo 2.0 audio playback
- Stereo 2.0 audio recording (on Windows servers)

- Touchscreen
- Stylus (on Linux, Windows 10, and Windows Server 2019 servers)
- Multiple monitor support

For more information about these features, see [Supported features](#) in the *NICE DCV User Guide*.

## Browser support

The NICE DCV Web Client SDK supports JavaScript (ES6) and it can be used from JavaScript or TypeScript applications.

The NICE DCV Web Client SDK supports the following web browsers:

Browser	Version
Google Chrome	Latest three major versions
Mozilla Firefox	Latest three major versions
Microsoft Edge	Latest three major versions
Apple Safari for macOS	Latest three major versions

## Versioning convention

The NICE DCV Web Client SDK version is defined in the following format: *major.minor.patch*. The versioning convention generally adheres to the [semantic versioning model](#). A change in the major version, such as from 1.x.x to 2.x.x, indicates that breaking changes that might require code changes and a planned deployment have been introduced. A change in the minor version, such as from 1.1.x to 1.2.x, is backwards compatible, but might include deprecated elements.

# Getting started with the NICE DCV Web Client SDK

The NICE DCV Web Client SDK comprises of a main `dcv.js` file and some auxiliary components. All the files are distributed inside a compressed archive that can be downloaded from the [NICE website](#).

## To get started with the NICE DCV Web Client SDK

1. The NICE DCV Web Client SDK archive is digitally signed with a secure GPG signature. To verify the archive's signature, you must import the NICE GPG key. To do so, open a terminal window and import the NICE GPG key.

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

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

2. Download the **NICE DCV Web Client SDK archive** and the **NICE DCV Web Client SDK archive signature** from the [NICE website](#).
3. Verify the signature of the NICE DCV Web Client SDK archive using the signature.

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

For example:

```
$ gpg --verify nice-dcv-web-client-sdk-1.0.1-141.zip.sign nice-dcv-web-client-  
sdk-1.0.1-141.zip
```

4. If the signature verifies successfully, extract the contents of the NICE DCV Web Client SDK archive and place the extracted directory on your web server. For example:

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

### Important

You must retain the folder structure when deploying the NICE DCV Web Client SDK on your web server.

## Connect to a NICE DCV server and get the first frame

The following tutorial shows you how to prepare your HTML page for your custom web client, how to authenticate and connect to a NICE DCV server, and how to receive the first frame of streamed content from the NICE DCV session.

### Topics

- [Step 1: Prepare your HTML page \(p. 4\)](#)
- [Step 2: Authenticate, connect, and get the first frame \(p. 4\)](#)
- [Bonus: Automatically create an HTML login form \(p. 6\)](#)

## Step 1: Prepare your HTML page

In your web page, you must load the needed JavaScript modules and you must add a `<div>` HTML element with a valid `id` where you want the NICE DCV Web Client SDK to draw the content stream from the remote NICE DCV server.

For example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>DCV first connection</title>
  </head>
  <body>
    <script type="module" src="main.js"></script>
    <div id="dcv-display"></div>
  </body>
</html>
```

## Step 2: Authenticate, connect, and get the first frame

This section shows how to complete the user authentication process, how to connect the NICE DCV server, and how to receive the first frame of content from the NICE DCV server.

First, import the NICE DCV Web Client SDK as a Universal Module Definition (UMD) module.

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

Define the variables to use to store the Authentication object, Connection object, and the NICE DCV server URL.

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

On script load, log the NICE DCV Web Client SDK version, and on page load, call the main function.

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

The main function sets the log level and starts the authentication process.

```
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
    }
  );
}
```

```
    }  
  );  
}
```

The `promptCredentials`, `error`, and `success` functions are mandatory callback functions that must be defined in the authentication process.

If the NICE DCV server prompts for credentials, the `promptCredentials` callback function receives the requested credential challenge from the NICE DCV server. If the NICE DCV server is configured to use system authentication, then the credentials must be provided in the form of a user name and a password. The following code samples assume that the user name is `my_dcv_user` and that the password is `my_password`.

If authentication fails, the `error` callback function receives an error object from the NICE DCV server.

If the authentication succeeds, the `success` callback function receives an array of couples that includes the session id (`sessionId`) and authorization tokens (`authToken`) for each session that the `my_dcv_user` user is allowed to connect to on the NICE DCV server. The following code sample calls the `connect` function and connects to the first session returned in the array.

```
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);  
}
```

Connect to the NICE DCV server. The `firstFrame` callback method is called when the first frame is received from the NICE DCV server.

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

## Bonus: Automatically create an HTML login form

The challenge object is returned when the `promptCredentials` callback function is called. It includes a property named `requiredCredentials` that is an array of objects - one object per credential that is requested by the NICE DCV server. Each object includes the name and the type of the requested credential. You can use the challenge and `requiredCredentials` objects to automatically create an HTML login form.

The following code sample shows you how to do this.

```
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));  
}
```

## Working with NICE DCV features

The availability of NICE DCV features depends on the permissions configured for the NICE DCV session and the capabilities of the client's web browser.

The features that are available in a NICE DCV session are managed by the permissions that have been specified for the session. This means that even if a feature is supported by the NICE DCV Web Client SDK, access to that feature might be prevented based on the permissions defined by the session administrator. For more information, see [Configuring NICE DCV Authorization](#) in the *NICE DCV Administrator Guide*.

## Understanding the featuresUpdate callback function

When the availability of a feature in a NICE DCV session changes, the NICE DCV Web Client SDK notifies you using the `featuresUpdate` callback function that you specify at the time of establishing the connection. For example:

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

The callback function notifies you only of the features for which the availability has changed. The `list` parameter is an array of strings, and it includes only the names of the updated features. For example, if the availability of the audio input feature changes for the session, the parameter includes only `["audio-in"]`. If at a later point, the availability of the clipboard copy and paste features change for the session, the parameter includes only `["clipboard-copy", "clipboard-paste"]`.

## Handling feature updates

The `featuresUpdate` callback function only notifies you that the availability of one or more features has changed. To know which features were updated, you must query the feature using the `connection.queryFeature` method. This can be done at any time after the notification of change has been received. This method returns a `Promise` that resolves to the requested feature's updated status. The status value is always associated and it has a Boolean (`true|false`) property called `enabled`. Some features might have additional properties in the status value. If the feature's availability has not been updated, it's rejected.

The following example code shows how to do this.

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

# SDK reference

This section provides descriptions, syntax, and usage examples for each of the NICE DCV Web Client SDK.

## Topics

- [DCV module \(p. 8\)](#)
- [Connection Class \(p. 35\)](#)
- [Authentication Class \(p. 51\)](#)

## DCV module

A module that implements the client side of the DCV protocol.

## Exposes

- [Methods \(p. 8\)](#)
- [Members \(p. 10\)](#)
- [Type and callback definitions \(p. 12\)](#)

## Methods

### List

- [authenticate\(url, callbacks\) → {Authentication} \(p. 8\)](#)
- [connect\(config\) → {Promise.<Connection>|Promise.<{code: ConnectionErrorCode, message: string}>} \(p. 9\)](#)
- [setLogHandler\(handler\) → {void} \(p. 9\)](#)
- [setLogLevel\(level\) → {void} \(p. 10\)](#)

## [authenticate\(url, callbacks\) → {Authentication \(p. 51\)}](#)

Starts the authentication process for the specified NICE DCV server endpoint.

### Parameters:

Name	Type	Description
<code>url</code>	string	The host name and port of the running NICE DCV server in the following format: <code>https://dcv_host_address:port</code> . For example: <code>https://my-dcv-server:8443</code> .
<code>callbacks</code>	<a href="#">authenticationCallbacks (p. 14)</a>	The callbacks that are available to be called during the authentication process.

### Returns:

- The Authentication object.

Type

[Authentication \(p. 51\)](#)

**connect(config) → {Promise.<Connection (p. 35)> | Promise.<{code: ConnectionErrorCode (p. 21), message: string}>}**

Connects to the specified NICE DCV server endpoint. If connection succeeds, it returns a Connection object. If connection fails, it returns an error object.

### Parameters:

Name	Type	Description
config	<a href="#">ConnectionConfig (p. 20)</a>	The ConnectionConfig object.

### Returns:

- A Connection object, or an error object.

Type

Promise.<[Connection \(p. 35\)](#)> | Promise.<{code: [ConnectionErrorCode \(p. 21\)](#), message: string}>

**setLogHandler(handler) → {void}**

Sets a custom log handler function. When overriding the default log handler, the original log entry position will be lost when debugging with the browser console.

### Parameters:

Name	Type	Description
handler	function	The custom log handler function. The handler function contains level (number), levelName (string), domain (string), and message (string).

### Returns:

Type

void

## setLogLevel(level) → {void}

Sets the log level. This is required only if the default log handler is used.

### Parameters:

Name	Type	Description
level	<a href="#">LogLevel</a> (p. 30)	The log level to use.

### Returns:

Type

void

## Members

### List

- [\(constant\) AudioError :AudioErrorCode](#) (p. 10)
- [\(constant\) AuthenticationError :AuthenticationErrorCode](#) (p. 10)
- [\(constant\) ChannelError :ChannelErrorCode](#) (p. 11)
- [\(constant\) ClosingReasonError :ClosingReasonErrorCode](#) (p. 11)
- [\(constant\) ConnectionError :ConnectionErrorCode](#) (p. 11)
- [\(constant\) CustomChannelError :CustomChannelErrorCode](#) (p. 11)
- [\(constant\) FileStorageError :FileStorageErrorCode](#) (p. 11)
- [\(constant\) MultiMonitorError :MultiMonitorErrorCode](#) (p. 11)
- [\(constant\) ResolutionError :ResolutionErrorCode](#) (p. 12)
- [\(constant\) version](#) (p. 12)
- [\(constant\) WebcamError :WebcamErrorCode](#) (p. 12)

## [\(constant\) AudioError :AudioErrorCode](#) (p. 13)

The AudioError codes enum.

### Type:

- [AudioErrorCode](#) (p. 13)

## [\(constant\) AuthenticationError :AuthenticationErrorCode](#) (p. 14)

The AuthenticationError codes enum.

### Type:

- [AuthenticationErrorCode](#) (p. 14)

## (constant) [ChannelError :ChannelErrorCode \(p. 17\)](#)

The ChannelError codes enum.

Type:

- [ChannelErrorCode \(p. 17\)](#)

## (constant) [ClosingReasonError :ClosingReasonErrorCode \(p. 18\)](#)

The ClosingReasonError codes enum.

Type:

- [ClosingReasonErrorCode \(p. 18\)](#)

## (constant) [ConnectionError :ConnectionErrorCode \(p. 21\)](#)

The ConnectionError codes enum.

Type:

- [ConnectionErrorCode \(p. 21\)](#)

## (constant) [CustomChannelError :CustomChannelErrorCode \(p. 22\)](#)

The CustomChannelError codes enum.

Type:

- [CustomChannelErrorCode \(p. 22\)](#)

## (constant) [FileStorageError :FileStorageErrorCode \(p. 28\)](#)

The FileStorageError codes enum.

Type:

- [FileStorageErrorCode \(p. 28\)](#)

## (constant) [MultiMonitorError :MultiMonitorErrorCode \(p. 31\)](#)

The MultiMonitorError codes enum.

Type:

- [MultiMonitorErrorCode \(p. 31\)](#)

## (constant) `ResolutionError :ResolutionErrorCode` (p. 32)

The `ResolutionError` codes enum.

Type:

- [ResolutionErrorCode](#) (p. 32)

## (constant) `version`

The NICE DCV version with major, minor, patch, revision, extended, and versionStr.

Properties:

Name	Type	Description
<code>major</code>	integer	The major version number.
<code>minor</code>	integer	The minor version number.
<code>patch</code>	integer	The patch version number.
<code>revision</code>	integer	The revision number.
<code>extended</code>	string	The extended string.
<code>versionStr</code>	string	A concatenation of the major, minor, patch, and revision numbers in the form <code>major.minor.patch+build.revision</code> .

## (constant) `WebcamError :WebcamErrorCode` (p. 35)

The `WebcamError` codes enum.

Type:

- [WebcamErrorCode](#) (p. 35)

## Type and callback definitions

List

- [AudioErrorCode](#) (p. 13)
- [authenticationCallbacks](#) (p. 14)
- [AuthenticationErrorCode](#) (p. 14)
- [authErrorCallback\(authentication, error\)](#) (p. 15)
- [authPromptCredentialsCallback\(authentication, challenge\)](#) (p. 15)
- [authSuccessCallback\(authentication, authenticationData\)](#) (p. 16)
- [Channel](#) (p. 16)
- [ChannelErrorCode](#) (p. 17)

- [clipboardEventCallback\(event\)](#) (p. 17)
- [ClosingReasonErrorCode](#) (p. 18)
- [Colorspace](#) (p. 18)
- [connectionCallbacks](#) (p. 18)
- [ConnectionConfig](#) (p. 20)
- [ConnectionErrorCode](#) (p. 21)
- [createDirectory\(path\)](#) (p. 22)
- [CustomChannelErrorCode](#) (p. 22)
- [dataChannelCallback\(info\)](#) (p. 22)
- [deleteFile\(path\)](#) (p. 23)
- [deviceChangeEventCallback\(\)](#) (p. 23)
- [disconnectCallback\(reason\)](#) (p. 23)
- [displayAvailabilityCallback\(status, displayId\)](#) (p. 23)
- [displayLayoutCallback\(serverWidth, serverHeight, heads\)](#) (p. 24)
- [feature](#) (p. 24)
- [featuresUpdateCallback\(featuresList\)](#) (p. 25)
- [fileDownloadCallback\(fileResource\)](#) (p. 25)
- [filePrintedCallback\(printResource\)](#) (p. 26)
- [filestorage](#) (p. 27)
- [filestorageEnabledCallback\(enabled\)](#) (p. 27)
- [FileStorageErrorCode](#) (p. 28)
- [firstFrameCallback\(resizeEnabled, relativeMouseModeEnabled, displayId\)](#) (p. 28)
- [idleWarningNotificationCallback\(disconnectionDateTime\)](#) (p. 28)
- [licenseNotificationCallback\(notification\)](#) (p. 28)
- [list\(path\)](#) (p. 30)
- [LogLevel](#) (p. 30)
- [Monitor](#) (p. 30)
- [MultiMonitorErrorCode](#) (p. 31)
- [qualityIndicatorStateCallback\(state\)](#) (p. 31)
- [renameDirectory\(src, dest\)](#) (p. 32)
- [renameFile\(src, dest\)](#) (p. 32)
- [ResolutionErrorCode](#) (p. 32)
- [retrieveFile\(path\)](#) (p. 33)
- [screenshotCallback\(screenshot\)](#) (p. 33)
- [serverInfo](#) (p. 33)
- [stats](#) (p. 34)
- [storeFile\(file, dir\)](#) (p. 34)
- [WebcamErrorCode](#) (p. 35)

## AudioErrorCode

The AudioError code enums available in the DCV module

- `SETTING_AUDIO_FAILED`



- CHANNEL\_NOT\_AVAILABLE

### Type:

- number

## authenticationCallbacks

Authentication callbacks

### Type:

- Object

### Properties:

Name	Type	Description
promptCredentials	<a href="#">authPromptCredentialsCallback (p. 15)</a>	The callback function to be called when the user is challenged for credentials.
error	<a href="#">authErrorCallback (p. 15)</a>	The callback function to be called when authentication fails.
success	<a href="#">authSuccessCallback (p. 16)</a>	The callback function to be called when authentication succeeds.

## AuthenticationErrorCode

The AuthenticationError code enums available in the DCV module

- 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

### Type:

- number

## authErrorCallback(authentication, error)

The callback function to be called when authentication fails.

### Parameters:

Name	Type	Description									
authentication	<a href="#">Authentication (p. 51)</a>	The Authentication object.									
error	Object	The error object raised by the authentication process. <table border="1" data-bbox="1101 632 1479 905"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>code</td> <td><a href="#">AuthenticationErrorCode (p. 14)</a></td> <td>The error code.</td> </tr> <tr> <td>message</td> <td>string</td> <td>The error message.</td> </tr> </tbody> </table>	Name	Type	Description	code	<a href="#">AuthenticationErrorCode (p. 14)</a>	The error code.	message	string	The error message.
Name	Type	Description									
code	<a href="#">AuthenticationErrorCode (p. 14)</a>	The error code.									
message	string	The error message.									

## authPromptCredentialsCallback(authentication, challenge)

The callback function to be called when the user is challenged for credentials. The user must answer the challenge by providing the requested credentials.

### Parameters:

Name	Type	Description												
authentication	<a href="#">Authentication (p. 51)</a>	The Authentication object.												
challenge	Object	The challenge. <table border="1" data-bbox="1101 1388 1479 1883"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>required</td> <td>Array&lt;Object&gt;</td> <td>array of requested credential objects.               <table border="1" data-bbox="1357 1640 1568 1883"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>name</td> <td>string</td> <td>The name of the requested credential.</td> </tr> </tbody> </table> </td> </tr> </tbody> </table>	Name	Type	Description	required	Array<Object>	array of requested credential objects. <table border="1" data-bbox="1357 1640 1568 1883"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>name</td> <td>string</td> <td>The name of the requested credential.</td> </tr> </tbody> </table>	Name	Type	Description	name	string	The name of the requested credential.
Name	Type	Description												
required	Array<Object>	array of requested credential objects. <table border="1" data-bbox="1357 1640 1568 1883"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>name</td> <td>string</td> <td>The name of the requested credential.</td> </tr> </tbody> </table>	Name	Type	Description	name	string	The name of the requested credential.						
Name	Type	Description												
name	string	The name of the requested credential.												

Name	Type	Description		
		Name	Type	Description

## authSuccessCallback(authentication, authenticationData)

The callback function to be called when authentication succeeds.

### Parameters:

Name	Type	Description
authentication	<a href="#">Authentication (p. 51)</a>	The Authentication object.
authenticationData	Array.<Object>	An array of objects that include NICE DCV session IDs and authentication tokens.

Name	Type	Description
sessionId	string	The NICE DCV session ID.
authToken	string	The authentication token for the NICE DCV session.

## Channel

The available channels that can be specified.

### Type:

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

## ChannelErrorCode

The ChannelError code enums available in the DCV module

- ALREADY\_OPEN
- INITIALIZATION\_FAILED
- REJECTED

### Type:

- number

## clipboardEventCallback(event)

The callback function to call when a clipboardEvent is generated.

### Parameters:

Name	Type	Description																
event	Object	Information about the clipboard event. <table border="1" data-bbox="1101 1018 1537 1881"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Attribute</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>name</td> <td>established   copy   paste   dataSizeAlert   autoCopyDone   newDataAvailable</td> <td></td> <td>Always present. The name of the event.</td> </tr> <tr> <td>clipboardData</td> <td>Object   string</td> <td></td> <td>The data in the clipboard.</td> </tr> <tr> <td>autoCopy</td> <td>boolean &lt;optional&gt;</td> <td></td> <td>Indicates whether automatic copying from the session clipboard to the local client clipboard</td> </tr> </tbody> </table>	Name	Type	Attribute	Description	name	established   copy   paste   dataSizeAlert   autoCopyDone   newDataAvailable		Always present. The name of the event.	clipboardData	Object   string		The data in the clipboard.	autoCopy	boolean <optional>		Indicates whether automatic copying from the session clipboard to the local client clipboard
Name	Type	Attribute	Description															
name	established   copy   paste   dataSizeAlert   autoCopyDone   newDataAvailable		Always present. The name of the event.															
clipboardData	Object   string		The data in the clipboard.															
autoCopy	boolean <optional>		Indicates whether automatic copying from the session clipboard to the local client clipboard															

Name	Type	Description			
		Name	Type	Attribute	Description
					is enabled.
		maxDataSize	number	<optional>	The maximum amount of data that can be placed in the clipboard.
		error	string	<optional>	Error information if applicable.

## ClosingReasonErrorCode

The ClosingReasonError code enums available in the DCV module

- TRANSPORT\_ERROR
- NO\_ERROR
- GENERIC\_ERROR
- INTERNAL\_SERVER\_ERROR
- PROTOCOL\_ERROR
- AUTHORIZATION\_DENIED
- AUTHORIZATION\_REVOKED
- ACCESS\_REJECTED
- IDLE\_TIMEOUT\_EXPIRED

### Type:

- number

## Colorspace

The available colorspace that can be specified.

### Type:

- "RGB" | "YUV\_REC601" | "YUV\_REC709"

## connectionCallbacks

The callbacks that are available to be called in the event of a connection error.

## Type:

- Object

## Properties:

Name	Type	Description
disconnect	<a href="#">disconnectCallback (p. 23)</a>	The callback function to call when the connection ends.
displayLayout	<a href="#">displayLayoutCallback (p. 24)</a>	The callback function to call when the display layout or resolution is changed.
displayAvailability	<a href="#">displayAvailabilityCallback (p. 23)</a>	The callback function to call when a display's availability changes.
firstFrame	<a href="#">firstFrameCallback (p. 28)</a>	The callback function to call when the first frame is received from the NICE DCV server.
filePrinted	<a href="#">filePrintedCallback (p. 26)</a>	The callback function to call when a file is printed on the NICE DCV server.
fileDownload	<a href="#">fileDownloadCallback (p. 25)</a>	The callback function to call when a file is ready to be downloaded from the NICE DCV server.
dataChannel	<a href="#">dataChannelCallback (p. 22)</a>	The callback function to call when the NICE DCV server sends a notification about the availability of a data channel.
licenseNotification	<a href="#">licenseNotificationCallback (p. 28)</a>	The callback function to call when the NICE DCV server sends a notification about the license state.
idleWarningNotification	<a href="#">idleWarningNotificationCallback (p. 28)</a>	The callback function to call when the NICE DCV server sends an idle timeout warning.
qualityIndicatorState	<a href="#">qualityIndicatorStateCallback (p. 31)</a>	The callback function to call when the connection quality indicator changes state.
filestorageEnabled	<a href="#">filestorageEnabledCallback (p. 27)</a>	The callback function to call when file storage is enabled or disabled.
featuresUpdate	<a href="#">featuresUpdateCallback (p. 25)</a>	The callback function to call when a feature's status changes.

Name	Type	Description
<code>clipboardEvent</code>	<a href="#">clipboardEventCallback (p. 17)</a>	The callback function to call when a <code>clipboardEvent</code> is generated.
<code>deviceChangeEvent</code>	<a href="#">deviceChangeEventCallback (p. 23)</a>	The callback function to call when an <code>deviceChange</code> event is triggered.
<code>screenshot</code>	<a href="#">screenshotCallback (p. 33)</a>	The callback function to call when a <code>screenshot</code> is available.

## ConnectionConfig

The configuration for a NICE DCV connection.

### Type:

- Object

### Properties:

Name	Type	Description
<code>url</code>	string	The host name and port of the running NICE DCV server in the following format: <code>https://dcv_host_address:port</code> . For example: <code>https://my-dcv-server:8443</code> .
<code>sessionId</code>	string	The NICE DCV session ID.
<code>authToken</code>	string	The authentication token to use when connecting to the server.
<code>baseUrl</code>	string	The absolute or relative URL from which to load SDK files.
<code>resourceBaseUrl</code>	string	The absolute or relative URL from which to access DCV resources.
<code>enabledChannels</code>	Array.< <a href="#">Channel (p. 16)</a> >	Indicates the list of channels that can be enabled. If not specified or an empty array is provided, it defaults to all the available channels.
<code>losslessColorspace</code>	<a href="#">Colorspace (p. 18)</a>	Indicates the colorspace that will be used. If not specified, it defaults to "RGB".

Name	Type	Description
divId	string	The ID of the <code>div</code> object in the HTML DOM where SDK should create the canvas with the remote stream.
volumeLevel	integer	The preferred volume level. The valid range is 0 to 100.
clipboardAutoSync	boolean	Indicates whether automatic copying from the NICE DCV session clipboard to the local client clipboard is enabled for compatible web browsers.
dynamicAudioTuning	boolean	Indicates whether to dynamically tune the audio based on the NICE DCV server audio settings when a connection is established.
clientHiDpiScaling	boolean	Indicates whether to scale the canvas based on the client's DPI.
enableWebCodecs	Boolean	Indicates whether WebCodecs should be used if available. Defaults to <code>false</code> if not specified.
observers	<a href="#">connectionCallbacks (p. 18)</a>	The callback functions to call for events that are related to the connection.
callbacks	<a href="#">connectionCallbacks (p. 18)</a>	The same as the observers, but each callback adds the <a href="#">Connection (p. 35)</a> object as the first parameter.

## ConnectionErrorCode

The ConnectionError code enums available in the DCV module

- `ALREADY_OPEN`
- `INVALID_CONFIG`
- `INITIALIZATION_FAILED`
- `REJECTED`
- `MAIN_CHANNEL_ALREADY_OPEN`
- `GENERIC_ERROR` (since DCV Server 2021.0)
- `INTERNAL_SERVER_ERROR` (since DCV Server 2021.0)
- `AUTHENTICATION_FAILED` (since DCV Server 2021.0)
- `PROTOCOL_ERROR` (since DCV Server 2021.0)
- `INVALID_SESSION_ID` (since DCV Server 2021.0)
- `INVALID_CONNECTION_ID` (since DCV Server 2021.0)
- `CONNECTION_LIMIT_REACHED` (since DCV Server 2021.0)



Type:

- number

## createDirectory(path)

Parameters:

Name	Type	Description
path	string	The absolute path on the server where we want to create a directory. It should also include the name of the target directory.

## CustomChannelErrorCode

The CustomChannelError code enums available in the DCV module

- TRANSPORT\_ERROR

Type:

- number

## dataChannelCallback(info)

The callback function to call when the NICE DCV server sends a notification about the availability of a data channel.

Parameters:

Name	Type	Description									
info	Object	Information about the data channel. <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>name</td><td>string</td><td>The name of the data channel.</td></tr><tr><td>token</td><td>string</td><td>The authentication token for the data channel.</td></tr></tbody></table>	Name	Type	Description	name	string	The name of the data channel.	token	string	The authentication token for the data channel.
Name	Type	Description									
name	string	The name of the data channel.									
token	string	The authentication token for the data channel.									

## deleteFile(path)

Parameters:

Name	Type	Description
path	string	The absolute path on the server identifying the file we want to delete.

## deviceChangeEventCallback()

The callback function to call when an `deviceChange` event is triggered.

## disconnectCallback(reason)

The callback function to call when the connection ends.

Parameters:

Name	Type	Description									
reason	Object	The reason for the disconnection. <table border="1" data-bbox="1101 1066 1481 1339"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>code</td><td>number</td><td>The reason code.</td></tr><tr><td>message</td><td>string</td><td>The reason message.</td></tr></tbody></table>	Name	Type	Description	code	number	The reason code.	message	string	The reason message.
Name	Type	Description									
code	number	The reason code.									
message	string	The reason message.									

## displayAvailabilityCallback(status, displayId)

The callback function to call when a display's availability changes.

Parameters:

Name	Type	Description						
status	Object	The status of the display. <table border="1" data-bbox="1101 1738 1481 1883"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>enabled</td><td>boolean</td><td>Indicates if the display</td></tr></tbody></table>	Name	Type	Description	enabled	boolean	Indicates if the display
Name	Type	Description						
enabled	boolean	Indicates if the display						

Name	Type	Description		
		Name	Type	Description
				is enabled.
		closed	boolean	Indicates if the display is closed.
displayId	number	The identifier for the display.		

## displayLayoutCallback(serverWidth, serverHeight, heads)

The callback function to call when the display layout or resolution is changed.

### Parameters:

Name	Type	Description
serverWidth	number	The width (in pixels) of the primary display.
serverHeight	number	The height (in pixels) of the primary display.
heads	Array.< <a href="#">Monitor (p. 30)</a> >	The display heads supported by the NICE DCV server.

## feature

The feature values.

- `display` - Indicates the availability of a single-display video stream.
- `display-multi` - Indicates the availability of a multi-display video stream.
- `mouse` - Indicates the availability of mouse functionality.
- `keyboard` - Indicates the availability of keyboard functionality.
- `keyboard-sas` - Indicates the availability of SAS sequence (Control + Alt + Delete) functionality.
- `relative-mouse` - Indicates the availability of relative mouse mode.
- `clipboard-copy` - Indicates the availability of clipboard copy functionality from NICE DCV server to the client.
- `clipboard-paste` - Indicates the availability of clipboard paste functionality from the client to the NICE DCV server.
- `audio-in` - Indicates the availability of audio input functionality using the microphone.
- `audio-out` - Indicates the availability of audio playback functionality.
- `webcam` - Indicates the availability of webcam streaming functionality.
- `file-download` - Indicates availability of file download functionality from the NICE DCV server to the client.
- `file-upload` - Indicates availability of file upload functionality from the client to the NICE DCV server.

### Type:

- string

## featuresUpdateCallback(featuresList)

The callback function to call when a feature's status changes.

### Parameters:

Name	Type	Description
featuresList	Array.<feature (p. 24)>	An array of features that have changed.

## fileDownloadCallback(fileResource)

The callback function to call when a file is ready to be downloaded from the NICE DCV server.

### Parameters:

Name	Type	Description															
fileResource	Object	Information about the file that is ready for download. <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>id</td><td>string</td><td>The identifier for the file.</td></tr><tr><td>url</td><td>string</td><td>The URL to use to download the file.</td></tr><tr><td>domain</td><td>string</td><td>The resource domain.</td></tr><tr><td>token</td><td>string</td><td>The authentication token to use to download the file. The token is also included</td></tr></tbody></table>	Name	Type	Description	id	string	The identifier for the file.	url	string	The URL to use to download the file.	domain	string	The resource domain.	token	string	The authentication token to use to download the file. The token is also included
Name	Type	Description															
id	string	The identifier for the file.															
url	string	The URL to use to download the file.															
domain	string	The resource domain.															
token	string	The authentication token to use to download the file. The token is also included															

Name	Type	Description		
		<b>Name</b>	<b>Type</b>	<b>Description</b>
				in the URL.

## filePrintedCallback(printResource)

The callback function to call when a file is printed on the NICE DCV server.

### Parameters:

Name	Type	Description		
printResource	Object	Information about the printed file.		
		<b>Name</b>	<b>Type</b>	<b>Description</b>
		id	string	The identifier for the printed file.
		url	string	The URL to use to download the printed file.
		domain	string	The resource domain. In this case, printer.
		token	string	The authentication token to use to download the printed file. The token is also included in the URL.

## filestorage

Object that allows for exploring and performing actions on the file system.

### Type:

- Object

### Properties:

Name	Type	Description
<code>list</code>	<a href="#">list (p. 30)</a>	Function that allows to list the items (files and directories) present at the supplied path on the server.
<code>createDirectory</code>	<a href="#">createDirectory (p. 22)</a>	Function that allows to create a directory at the specified path on the server.
<code>retrieveFile</code>	<a href="#">retrieveFile (p. 33)</a>	Function that allows to locally download a file at the specified path on the server.
<code>deleteFile</code>	<a href="#">deleteFile (p. 23)</a>	Function that allows to delete a file at the specified path on the server.
<code>renameFile</code>	<a href="#">renameFile (p. 32)</a>	Function that allows to rename a file from the specified source path to the specified destination path.
<code>renameDirectory</code>	<a href="#">renameDirectory (p. 32)</a>	Function that allows to rename a directory from the specified source path to the absolute destination path.
<code>storeFile</code>	<a href="#">storeFile (p. 34)</a>	Function that allows to upload a local file to the supplied path on the server.

## filestorageEnabledCallback(enabled)

The callback function to call when file storage is enabled. Lazy channel on Internet Explorer 11 only.

### Parameters:

Name	Type	Description
<code>enabled</code>	boolean	Indicates whether file storage is enabled.

## FileStorageErrorCode

The FileStorageError code enums available in the DCV module

- CANCELLED
- ABORTED
- INVALID\_ARGUMENT
- NOT\_IMPLEMENTED
- ERROR
- ALREADY\_EXIST
- NOT\_FOUND

### Type:

- number

## firstFrameCallback(resizeEnabled, relativeMouseModeEnabled, displayId)

The callback function to call when the first frame is received from the NICE DCV server. Emitted for each display.

### Parameters:

Name	Type	Description
resizeEnabled	boolean	Indicates whether the server supports resizing the client display layout.
relativeMouseModeEnabled	boolean	Indicates whether the server supports relative mouse mode.
displayId	number	The identifier for the display.

## idleWarningNotificationCallback(disconnectionDateTime)

The callback function to call when the NICE DCV server sends an idle timeout warning.

### Parameters:

Name	Type	Description
disconnectionDateTime	Date	The date and time of the disconnection.

## licenseNotificationCallback(notification)

The callback function to call when the NICE DCV server sends a notification about the license state.

Parameters:

Name	Type	Description																											
notification	Object	The notification. <table border="1" data-bbox="1102 436 1479 1581"> <thead> <tr> <th data-bbox="1102 436 1224 487">Name</th> <th data-bbox="1224 436 1346 487">Type</th> <th data-bbox="1346 436 1479 487">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="1102 487 1224 569">product</td> <td data-bbox="1224 487 1346 569">string</td> <td data-bbox="1346 487 1479 569">The DCV product.</td> </tr> <tr> <td data-bbox="1102 569 1224 709">status</td> <td data-bbox="1224 569 1346 709">string</td> <td data-bbox="1346 569 1479 709">The status of the license.</td> </tr> <tr> <td data-bbox="1102 709 1224 791">message</td> <td data-bbox="1224 709 1346 791">string</td> <td data-bbox="1346 709 1479 791">A message.</td> </tr> <tr> <td data-bbox="1102 791 1224 1016">leftDays</td> <td data-bbox="1224 791 1346 1016">number</td> <td data-bbox="1346 791 1479 1016">The number of days before the license expires.</td> </tr> <tr> <td data-bbox="1102 1016 1224 1184">isDemo</td> <td data-bbox="1224 1016 1346 1184">boolean</td> <td data-bbox="1346 1016 1479 1184">Indicates if the license is a demo license.</td> </tr> <tr> <td data-bbox="1102 1184 1224 1352">numUnlicensed</td> <td data-bbox="1224 1184 1346 1352">number</td> <td data-bbox="1346 1184 1479 1352">The number of unlicensed connections.</td> </tr> <tr> <td data-bbox="1102 1352 1224 1465">licensingMode</td> <td data-bbox="1224 1352 1346 1465">string</td> <td data-bbox="1346 1352 1479 1465">The licensing mode.</td> </tr> <tr> <td data-bbox="1102 1465 1224 1581">documentationUrl</td> <td data-bbox="1224 1465 1346 1581">string</td> <td data-bbox="1346 1465 1479 1581">The URL for the documentation.</td> </tr> </tbody> </table>	Name	Type	Description	product	string	The DCV product.	status	string	The status of the license.	message	string	A message.	leftDays	number	The number of days before the license expires.	isDemo	boolean	Indicates if the license is a demo license.	numUnlicensed	number	The number of unlicensed connections.	licensingMode	string	The licensing mode.	documentationUrl	string	The URL for the documentation.
Name	Type	Description																											
product	string	The DCV product.																											
status	string	The status of the license.																											
message	string	A message.																											
leftDays	number	The number of days before the license expires.																											
isDemo	boolean	Indicates if the license is a demo license.																											
numUnlicensed	number	The number of unlicensed connections.																											
licensingMode	string	The licensing mode.																											
documentationUrl	string	The URL for the documentation.																											



## list(path)

### Parameters:

Name	Type	Description
path	string	The absolute path on the server of which we want to list the content.

## LogLevel

The available SDK log levels.

### Type:

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

## Monitor

### Type:

- Object

### Properties:

Name	Type	Description												
name	string	The name of the display head.												
rect	Object	Information about the display head. <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>x</td><td>number</td><td>The initial x coordinate for the display head.</td></tr><tr><td>y</td><td>number</td><td>The initial y coordinate for the display head.</td></tr><tr><td>width</td><td>number</td><td>The width (in pixels) of the</td></tr></tbody></table>	Name	Type	Description	x	number	The initial x coordinate for the display head.	y	number	The initial y coordinate for the display head.	width	number	The width (in pixels) of the
Name	Type	Description												
x	number	The initial x coordinate for the display head.												
y	number	The initial y coordinate for the display head.												
width	number	The width (in pixels) of the												

Name	Type	Description		
		Name	Type	Description
				display head.
		height	number	The height (in pixels) of the display head.
primary	boolean	Indicates whether the display head is the primary display head. This is determined from the remote operating system if available.		
dpi	number	The DPI of the display head.		

## MultiMonitorErrorCode

The MultiMonitorError code enums available in the DCV module

- NO\_DISPLAY\_CHANNEL
- MAX\_DISPLAY\_NUMBER\_REACHED
- INVALID\_ARGUMENT
- DISPLAY\_NOT\_OPENED\_BY\_SERVER
- REQUEST\_TIMEOUT
- GENERIC\_ERROR
- NO\_ERROR

### Type:

- number

## qualityIndicatorStateCallback(state)

The callback function to call when the connection quality indicator changes state.

### Parameters:

Name	Type	Description		
state	Array.<Object>	Information about the connection quality.		
		<b>Name</b>	<b>Type</b>	<b>Description</b>
		name	string	The name

Name	Type	Description		
		<b>Name</b>	<b>Type</b>	<b>Description</b>
				of the indicator.
		status	NORMAL   WARNING   CRITICAL	Description of the status.
		changed	boolean	Indicates whether the status changed.

## renameDirectory(src, dest)

Parameters:

Name	Type	Description
src	string	The absolute source path on the server identifying the directory we want to rename.
dest	string	The absolute destination path on the server specifying the target path and directory name.

## renameFile(src, dest)

Parameters:

Name	Type	Description
src	string	The absolute source path on the server identifying the file we want to rename.
dest	string	The absolute destination path on the server specifying the target path and file name.

## ResolutionErrorCode

The ResolutionError code enums available in the DCV module

- INVALID\_ARGUMENT
- NO\_CHANNEL

### Type:

- number

## retrieveFile(path)

### Parameters:

Name	Type	Description
path	string	The absolute path on the server identifying the file we want to download locally.

## screenshotCallback(screenshot)

The callback function to call when a screenshot is available.

### Parameters:

Name	Type	Description
screenshot	byte[]	Screenshot buffer in PNG format, or null if screenshot retrieval failed.

## serverInfo

### Type:

- Object

### Properties:

Name	Type	Description									
name	string	The name of the software.									
version	Object	The software version number. <table border="1" data-bbox="1101 1627 1481 1881"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>major</td><td>number</td><td>The major version number.</td></tr><tr><td>minor</td><td>number</td><td>The minor</td></tr></tbody></table>	Name	Type	Description	major	number	The major version number.	minor	number	The minor
Name	Type	Description									
major	number	The major version number.									
minor	number	The minor									

Name	Type	Description		
		Name	Type	Description
				version number.
		revision number		The revision version number.
os	string	The OS.		
arch	string	The architecture.		
hostname	string	The hostname.		

## stats

Type:

- Object

Properties:

Name	Type	Description
fps	number	The current frames per second.
traffic	number	The current traffic in bit/s.
peakTraffic	number	The peak of traffic in bit/s since the connection was established.
latency	number	The current latency in ms.
currentChannels	number	The number of channels that have been opened since the connection was established.
openedChannels	number	The number of currently opened channels.
channelErrors	number	The number of channels which have reported an error.

## storeFile(file, dir)

Parameters:

Name	Type	Description
file	File	The file object (for more information see <a href="https://">https://</a>

Name	Type	Description
		developer.mozilla.org/en-US/docs/Web/API/File) we want to upload to the server.
dir	string	The absolute path on the server where we want to upload the file.

## WebcamErrorCode

The WebcamError code enums available in the DCV module

- SETTING\_WEBCAM\_FAILED
- CHANNEL\_NOT\_AVAILABLE

Type:

- number

## Connection Class

The Connection Class obtained by calling the [connect method \(p. 9\)](#) of the `dcv` module. For an example showing how to use it, see the [Getting started \(p. 4\)](#) section.

### Exposes

- [Methods \(p. 8\)](#)

## Methods

### List

- [attachDisplay\(window, displayConf\) → {Promise.<number>|Promise.<{code: MultiMonitorErrorCode, message: string}>}](#) (p. 36)
- [captureClipboardEvents\(enabled, win, displayId\) → {void}](#) (p. 37)
- [detachDisplay\(displayId\) → {void}](#) (p. 37)
- [disconnect\(\) → {void}](#) (p. 38)
- [enableDisplayQualityUpdates\(enable\) → {void}](#) (p. 38)
- [enterRelativeMouseMode\(\) → {void}](#) (p. 38)
- [getConnectedDevices\(\) → {Promise.<Array.<MediaDeviceInfo>>|Promise.<{message: string}>}](#) (p. 38)
- [getFileExplorer\(\) → {Promise.<filestorage>|Promise.<{code: ChannelErrorCode, message: string}>}](#) (p. 39)
- [getServerInfo\(\) → {serverInfo}](#) (p. 39)
- [getStats\(\) → {stats}](#) (p. 39)
- [latchModifierKey\(key, location, isDown\) → {boolean}](#) (p. 39)
- [openChannel\(name, authToken, callbacks\) → {Promise|Promise.<{code: ChannelErrorCode, message: string}>}](#) (p. 40)

- `queryFeature(featureName) → {Promise.<{enabled: boolean}>|Promise.<{message: string}>}` (p. 41)
- `registerKeyboardShortcuts(shortcuts) → {void}` (p. 41)
- `requestDisplayLayout(layout) → {Promise|Promise.<{code: ResolutionErrorCode, message: string}>}` (p. 44)
- `requestResolution(width, height) → {Promise|Promise.<{code: ResolutionErrorCode, message: string}>}` (p. 44)
- `sendKeyboardEvent(event) → {boolean}` (p. 45)
- `sendKeyboardShortcut(shortcut) → {void}` (p. 45)
- `setDisplayQuality(min, maxopt) → {void}` (p. 46)
- `setDisplayScale(scaleRatio, displayId) → {Promise|Promise.<{code: ResolutionErrorCode, message: string}>}` (p. 47)
- `setKeyboardQuirks(quirks) → {void}` (p. 47)
- `setMaxDisplayResolution(maxWidth, maxHeight) → {void}` (p. 48)
- `setMicrophone(enable) → {Promise|Promise.<{code: AudioErrorCode, message: string}>}` (p. 48)
- `setMinDisplayResolution(minWidth, minHeight) → {void}` (p. 49)
- `setUpUploadBandwidth(value) → {number}` (p. 49)
- `setVolume(volume) → {void}` (p. 50)
- `setWebcam(enable, deviceId) → {Promise|Promise.<{code: WebcamErrorCode, message: string}>}` (p. 50)
- `syncClipboards() → {boolean}` (p. 50)

## `attachDisplay(window, displayConf) → {Promise.<number>|Promise.<{code: MultiMonitorErrorCode (p. 31), message: string}>}`

Attaches a specific display to a window. You can't attach the main display. If successful, the function returns the `displayId`.

### Parameters:

Name	Type	Description												
<code>window</code>	Object	The window to which the display must be attached.												
<code>displayConf</code>	Object	The configuration of the display. <table border="1" data-bbox="1101 1518 1474 1881"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Attribute</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>displayId</code></td> <td>Number</td> <td>&lt;optional&gt;</td> <td>The ID of the display.</td> </tr> <tr> <td><code>displayDivName</code></td> <td></td> <td></td> <td>The name of the display div.</td> </tr> </tbody> </table>	Name	Type	Attribute	Description	<code>displayId</code>	Number	<optional>	The ID of the display.	<code>displayDivName</code>			The name of the display div.
Name	Type	Attribute	Description											
<code>displayId</code>	Number	<optional>	The ID of the display.											
<code>displayDivName</code>			The name of the display div.											

## Returns:

- Promise. If rejected, the promise returns an error object.

Type

Promise.<number> | Promise.<{code: [MultiMonitorErrorCode](#) (p. 31), message: string}>

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

Starts or stops listening to copy-paste events. In the case of interactive clipboards (always in the case of paste) we need to start listening to the copy/paste events. It could be useful to start and stop listening only when it is needed, for example, when a modal is shown.

## Parameters:

Name	Type	Attributes	Description
enabled	boolean		To start listening to events, specify <code>true</code> . To stop listening to events, specify <code>false</code> .
win	Object	<optional>	The window in which to listen for events. If omitted, the default window is used.
displayId	number	<optional>	The ID of the display that should listen the events. If omitted, the default display of the window is used.

## Returns:

Type

void

## detachDisplay(displayId) → {void}

Detaches a specific display. The main display cannot be detached.

## Parameters:

Name	Type	Description
displayId	number	The ID of the display to detach.



### Returns:

Type

void

## disconnect() → {void}

Disconnects from the NICE DCV server and closes the connection.

### Returns:

Type

void

## enableDisplayQualityUpdates(enable) → {void}

Enables or disables display quality updates for streaming areas that do not receive updates. Disabling display quality updates reduces bandwidth usage, but it also decreases the display quality.

### Parameters:

Name	Type	Description
enable	boolean	To enable display quality updates, specify <code>true</code> . To disable display quality updates, specify <code>false</code> .

### Returns:

Type

void

## enterRelativeMouseMode() → {void}

Enables relative mouse mode.

### Returns:

Type

void

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

Requests a list of the media devices connected to the client computer.

### Returns:

If successful, it returns a Promise that resolves to an array of `MediaDeviceInfo` objects. For more information, see <https://developer.mozilla.org/en-US/docs/Web/API/MediaDeviceInfo>. If rejected, the promise returns an error object.

Type

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

**`getFileExplorer()` → {Promise.<filestorage (p. 27)> | Promise.<{code: ChannelErrorCode (p. 17), message: string}>}**

Gets an object to manage the NICE DCV server's file storage.

### Returns:

- Promise. Resolves to the file explorer object if fulfilled, or an error object if rejected.

Type

Promise.<filestorage (p. 27)> | Promise.<{code: ChannelErrorCode (p. 17), message: string}>

**`getServerInfo()` → {serverInfo (p. 33)}**

Gets information about the NICE DCV server.

### Returns:

Information about the server software.

Type

serverInfo (p. 33)

**`getStats()` → {stats (p. 34)}**

Gets statistics about the NICE DCV server.

### Returns:

Information about the streaming statistics.

Type

stats (p. 34)

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

Sends a single keyboard `keydown` or `keyup` event for an allowed modifier.

### Parameters:

Name	Type	Description
key	Control   Alt   AltGraph   Meta   OS   Shift	The key to send.
location	KeyboardEvent.location	The key's location. For more information, see <a href="https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/location">https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/location</a> .
isDown	boolean	If the key event to inject is a keydown ( <code>true</code> ) or a keyup ( <code>false</code> ).

### Returns:

If the requested combination is valid, the function returns `true`, otherwise it returns `false`.

Type

boolean

**`openChannel(name, authToken, callbacks) → {Promise Promise.<{code: ChannelErrorCode (p. 17), message: string}>}`**

Opens a custom data channel on the connection if it was created on the NICE DCV Server.

### Parameters:

Name	Type	Description
name	string	The name of the channel.
authToken	string	The authentication token to use to connect to the channel.
callbacks	Object	The <code>onMessage</code> and <code>onClose</code> callbacks functions to call.

### Returns:

Promise. If rejected we receive an error object.

Type

Promise | Promise.<{code: ChannelErrorCode (p. 17), message: string}>

## queryFeature(featureName) → {Promise.<{enabled: boolean}> | Promise.<{message: string}>}

Queries the status of a specific NICE DCV server feature.

### Parameters:

Name	Type	Description
featureName	<a href="#">feature</a> (p. 24)	The name of the feature to query.

### Returns:

Promise. If resolved, the function returns a `status` object with an `enabled` property. If rejected, the function returns an `error` object.

Type

Promise.<{enabled: boolean}> | Promise.<{message: string}>

## registerKeyboardShortcuts(shortcuts) → {void}

Registers keyboard shortcuts.

### Parameters:

Name	Type	Description												
shortcuts	Array.<Object>	The array of keys and mappings to register. <table border="1" data-bbox="1101 1276 1490 1491"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>sequence</td> <td>Array.&lt;Object&gt;</td> <td>The keyboard shortcut to register.</td> </tr> </tbody> </table> <table border="1" data-bbox="1360 1533 1619 1881"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>key</td> <td>KeyboardEvent.key</td> <td>The value of the key pressed by the user. For</td> </tr> </tbody> </table>	Name	Type	Description	sequence	Array.<Object>	The keyboard shortcut to register.	Name	Type	Description	key	KeyboardEvent.key	The value of the key pressed by the user. For
Name	Type	Description												
sequence	Array.<Object>	The keyboard shortcut to register.												
Name	Type	Description												
key	KeyboardEvent.key	The value of the key pressed by the user. For												

Name	Type	Description		
		Name	Type	Description
				<p>more information, see <a href="https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key">https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key</a>.</p> <p><b>KeyboardEvent.location</b></p> <p>array of keys to send. The location of the key on the keyboard. For more information, see <a href="https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/location">https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/location</a>.</p>
		output	Array.<Object>	<p>The intended action to be performed by the shortcut.</p>

Name	Type	Description		
		Name	Type	Description
				<p><b>key</b> The KeyboardEvent.key value of the key pressed by the user. For more information, see <a href="https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key">https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key</a>.</p>
				<p><b>location</b> KeyboardEvent.location array of keys to send. The location of the key on the keyboard. For more information, see <a href="https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/location">https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/location</a>.</p>

### Returns:

Type

void

**requestDisplayLayout(layout) → {Promise|Promise.<{code: ResolutionErrorCode (p. 32), message: string}>}**

Requests an updated display layout for the connection.

### Parameters:

Name	Type	Description
layout	Array.<Monitor (p. 30)>	The requested displays in the layout.

### Returns:

- Promise. If rejected we receive an error object.

Type

Promise | Promise.<{code: ResolutionErrorCode (p. 32), message: string}>

**requestResolution(width, height) → {Promise|Promise.<{code: ResolutionErrorCode (p. 32), message: string}>}**

Requests an updated display resolution from the NICE DCV server.

### Parameters:

Name	Type	Description
width	number	The width to request in pixels. The minimum allowed value is 0.
height	number	The height to request in pixels. The minimum allowed value is 0.

### Returns:

- Promise. If rejected, the promise returns an error object.

Type

Promise | Promise.<{code: ResolutionErrorCode (p. 32), message: string}>

## sendKeysKeyboardEvent(event) → {boolean}

Sends a keyboard shortcut event. For more information about keyboard events, see <https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent>. Valid Keyboard events include: `keydown`, `keypress`, and `keyup`. For more information about these events, see <https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent#events>.

### Parameters:

Name	Type	Description
event	KeyboardEvent	The keyboard event to send.

### Returns:

If the event is not valid, the function returns `false`. If the event is valid, the function returns `true`.

Type

boolean

## sendKeysKeyboardShortcut(shortcut) → {void}

Sends a keyboard shortcut. Use this function to send a full `keydown` or `keyup` sequence. For example, sending `Ctrl + Alt + Del` sends the `keydown` events for all the keys followed by the `keyup` events. Use this function even if you want to send a single key.

### Parameters:

Name	Type	Description						
shortcut	Array.<Object>	The array of keys to send. <table border="1"><thead><tr><th>Name</th><th>Type</th><th>Description</th></tr></thead><tbody><tr><td>key</td><td>KeyboardEvent</td><td>The key value of the key pressed by the user. For more information, see <a href="https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key">https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key</a>.</td></tr></tbody></table>	Name	Type	Description	key	KeyboardEvent	The key value of the key pressed by the user. For more information, see <a href="https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key">https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key</a> .
Name	Type	Description						
key	KeyboardEvent	The key value of the key pressed by the user. For more information, see <a href="https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key">https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key</a> .						



Name	Type	Description		
		<b>Name</b>	<b>Type</b>	<b>Description</b>
		location	KeyboardEvent	The location array of keys to send. The location of the key on the keyboard. For more information, see <a href="https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/location">https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/location</a> .

### Returns:

Type

void

### setDisplayQuality(min, maxopt) → {void}

Sets the image quality to use for the connection. Valid range is 0 to 100, with 1 being the lowest image quality and 100 being the highest image quality. Specify 0 to retain the current value.

### Parameters:

Name	Type	Attributes	Description
min	number		The minimum image quality.
max	number	<optional>	The maximum image quality.

### Returns:

Type

void

## setDisplayScale(scaleRatio, displayId) → {Promise| Promise.<{code: ResolutionErrorCode (p. 32), message: string}>}

Notifies the NICE DCV that the display is scaled on the client side. Use this to notify the server that it needs to scale mouse events to match the client's display ratio.

### Parameters:

Name	Type	Description
scaleRatio	float	The scaling ratio to use (allowed range (0, 1]).
displayId	number	The ID of the display to scale.

### Returns:

Promise. If rejected, the promise returns an error object.

Type

Promise | Promise.<{code: ResolutionErrorCode (p. 32), message: string}>

## setKeyboardQuirks(quirks) → {void}

Sets keyboard quirks for the client computer.

### Parameters:

Name	Type	Description									
quirks	Object	The keyboard quirks to enable or disable. <table border="1" data-bbox="1101 1360 1481 1883"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>macOptionKey</td> <td>boolean</td> <td>To map the Option key to Alt for macOS, specify true. Otherwise, specify false.</td> </tr> <tr> <td>macCommandKey</td> <td>boolean</td> <td>To map the Command key to</td> </tr> </tbody> </table>	Name	Type	Description	macOptionKey	boolean	To map the Option key to Alt for macOS, specify true. Otherwise, specify false.	macCommandKey	boolean	To map the Command key to
Name	Type	Description									
macOptionKey	boolean	To map the Option key to Alt for macOS, specify true. Otherwise, specify false.									
macCommandKey	boolean	To map the Command key to									

Name	Type	Description		
		<b>Name</b>	<b>Type</b>	<b>Description</b>
				Ctrl for macOS, specify <code>true</code> . Otherwise, specify <code>false</code> .

### Returns:

Type

`void`

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

Sets the maximum display resolution to use for the connection.

### Parameters:

Name	Type	Description
<code>maxWidth</code>	number	The maximum display width in pixels. The minimum allowed value is 0.
<code>maxHeight</code>	number	The maximum display height in pixels. The minimum allowed value is 0.

### Returns:

Type

`void`

## `setMicrophone(enable) → {Promise|Promise.<{code: AudioErrorCode (p. 13), message: string}>}`

Enables or disables the microphone.

### Parameters:

Name	Type	Description
<code>enable</code>	boolean	To enable the microphone, specify <code>true</code> . To disable the microphone, specify <code>false</code> .

### Returns:

- Promise. If rejected, the promise returns an error object.

Type

Promise | Promise.<{code: [AudioErrorCode](#) (p. 13), message: string}>

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

Sets the minimum display resolution to use for the connection. Some applications might require a minimum display resolution. If the minimum required resolution is larger than the maximum resolution supported by the client, a resize strategy is used. Use this function carefully. The resize strategy could cause a less precise mouse and touch input system.

### Parameters:

Name	Type	Description
minWidth	number	The minimum display width in pixels. The minimum allowed value is 0.
minHeight	number	The minimum display height in pixels. The minimum allowed value is 0.

### Returns:

Type

void

## setUploadBandwidth(value) → {number}

Sets the maximum bandwidth to use for uploading files to the NICE DCV server.

### Parameters:

Name	Type	Description
value	number	The maximum bandwidth limit in kbps. Valid range is 1024 kbps to 102400 kbps.

### Returns:

- The set bandwidth limit. null if the file storage feature is disabled on the server.

Type

number

## setVolume(volume) → {void}

Sets the volume level to use for audio. Valid range is 0 to 100, with 0 being the lowest volume and 100 being the highest volume.

### Parameters:

Name	Type	Description
volume	number	The volume level to use.

### Returns:

Type

void

## setWebcam(enable, deviceId) → {Promise|Promise.<{code: WebcamErrorCode (p. 35), message: string}>}

Enables or disables the webcam.

### Parameters:

Name	Type	Description
enable	boolean	To enable the webcam, specify <code>true</code> . To disable the webcam, specify <code>false</code> .
deviceId	string	The device ID of the webcam.

### Returns:

- Promise that, if successful, resolves to the attached/detached webcam deviceId. If rejected, the promise returns an error object.

Type

Promise | Promise.<{code: [WebcamErrorCode \(p. 35\)](#), message: string}>

## syncClipboards() → {boolean}

Synchronizes the local client clipboard with the remote NICE DCV server clipboard. Autocopy must be supported by the browser.

### Returns:

If the clipboards have been synchronized, the function returns `true`. If the clipboards have not been synchronized, or if the browser does not support autocopy, the function returns `false`.

Type

boolean

## Authentication Class

The Authentication Class must be used to obtain an authentication token by calling the [authenticate method \(p. 8\)](#) of the `dcv` module. For an example showing how to use it, see the [Getting started \(p. 4\)](#) section.

### Exposes

- [Methods \(p. 8\)](#)

## Methods

### List

- [retry\(\) → {void} \(p. 51\)](#)
- [sendCredentials\(credentials\) → {void} \(p. 51\)](#)

### `retry() → {void}`

Retries the authentication process.

### Returns:

Type

void

### `sendCredentials(credentials) → {void}`

Sends the authentication credentials provided by the client to the NICE DCV server.

### Parameters:

Name	Type	Description
<code>credentials</code>	object	The object containing the supplied credentials. The credentials must have the same name and be of the same type that is specified in the challenge.

### Returns:

Type

void

# Release Notes and Document History for NICE DCV Web Client SDK

This page provides the release notes and document history for NICE DCV Web Client SDK.

## Contents

- [NICE DCV Web Client SDK Release Notes \(p. 52\)](#)
- [Document History \(p. 53\)](#)

## NICE DCV Web Client SDK Release Notes

This section provides release notes for the NICE DCV Web Client SDK by release date.

### Releases

- [1.0.3 — September 01, 2021 \(p. 52\)](#)
- [1.0.2 — July 30, 2021 \(p. 52\)](#)
- [1.0.1 — May 31, 2021 \(p. 53\)](#)
- [1.0.0 — March 24, 2021 \(p. 53\)](#)

### 1.0.3 — September 01, 2021

Version	Release notes
<ul style="list-style-type: none"><li>• Semantic version: 1.0.3</li><li>• Build: 202</li></ul>	<p>New features</p> <ul style="list-style-type: none"><li>• Experimental support for WebCodecs. This is disabled by default and must be enabled via the <code>ConnectionConfig</code> object using the new property <code>enableWebCodecs</code>.</li><li>• Clipboard: added support for <code>image/png</code> data type on Chromium based browsers.</li><li>• Added observer/callback to get the server's screenshot as a PNG image (requires NICE DCV server 2021.2).</li></ul> <p>Changes and bug fixes</p> <ul style="list-style-type: none"><li>• Improved handling of keyboard modifiers.</li></ul>

### 1.0.2 — July 30, 2021

Version	Release notes
<ul style="list-style-type: none"><li>• Semantic version: 1.0.2</li></ul>	<ul style="list-style-type: none"><li>• Fixed pressure detection for stylus events.</li></ul>

Version	Release notes
<ul style="list-style-type: none"><li>Build: 167</li></ul>	<ul style="list-style-type: none"><li>Improved support for Korean keyboard layout on Chrome.</li></ul>

## 1.0.1 — May 31, 2021

Version	Release notes
<ul style="list-style-type: none"><li>Semantic version: 1.0.1</li><li>Build: 141</li></ul>	<ul style="list-style-type: none"><li>Fixed propagation of connection errors and close reasons</li><li>Fixed filestorage chunk progress update</li><li>Improved webcam handling</li><li>Improved audio-in processing</li></ul>

## 1.0.0 — March 24, 2021

Version	Release notes
<ul style="list-style-type: none"><li>Semantic version: 1.0.0</li><li>Build: 81</li></ul>	Initial release of the NICE DCV Web Client SDK.

## Document History

The following table describes the documentation for this release of NICE DCV Web Client SDK.

Change	Description	Date
NICE DCV Web Client SDK version 1.0.1	Fixed some typos. Minor bugs fixed, see <a href="#">SDK v.1.0.1 (p. 53)</a> .	May 31, 2021
Initial release	First publication of this content.	March 24, 2021