Latest

- Note: Visit the Chroma Animation Guide to find the latest supported plugin for Chroma RGB.

CChromaEditor - C++ Dynamic Library for playing and editing Chroma animations

Table of Contents

- See Also
- User Privacy
- Dependencies
- About
- Security
- Windows PC
- Windows Cloud
- SDK Integration
- Chroma Design
- Revisions
- Sample Project
- Tools
- Integration
- Testing
- Haptic Design
- Modding
- General
- Chroma Sensa
- Synesthesia
- Initialize SDK
- Is Active
- Is Connected
- Play Chroma Animation
- Set Event Name
- Use Forward Chroma Events
- Microsoft Dynamic Lighting
- Frameworks supported
- Assets
- Dialog
- Streaming
- Streaming Logic Flow
- File Format
- Extras
- Full API

See Also

Docs:

- Chroma Animation Guide - Visual examples of the Chroma animation API methods

Plugins:

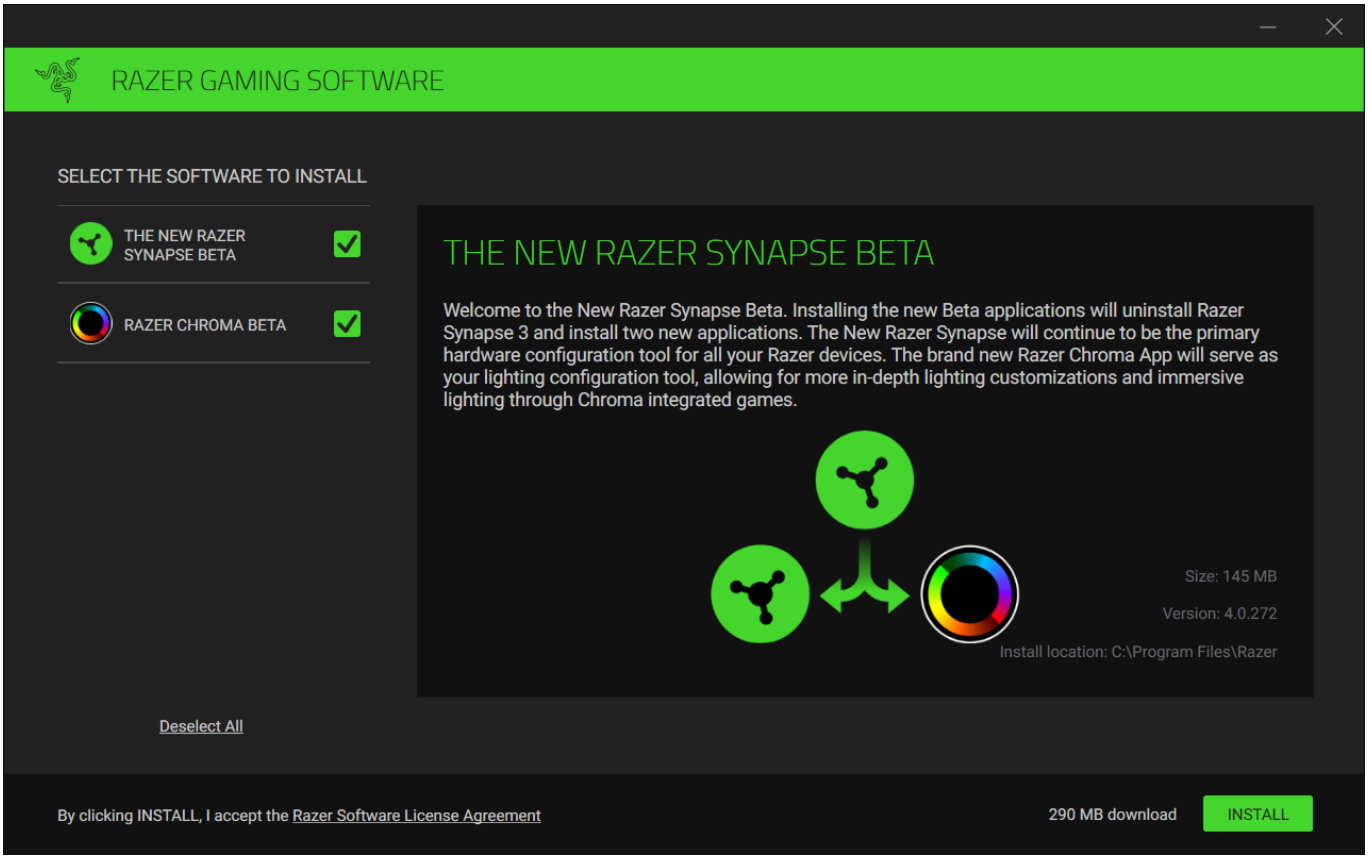
- CChromaEditor - C++ library for playing and editing Chroma animations

User Privacy

Note: The Chroma SDK requires only the minimum amount of information necessary to operate during initialization, including the title of the application or game, description of the application or game, application or game author, and application or game support contact information. This information is displayed in the Chroma app. The Chroma SDK does not monitor or collect any personal data related to users.

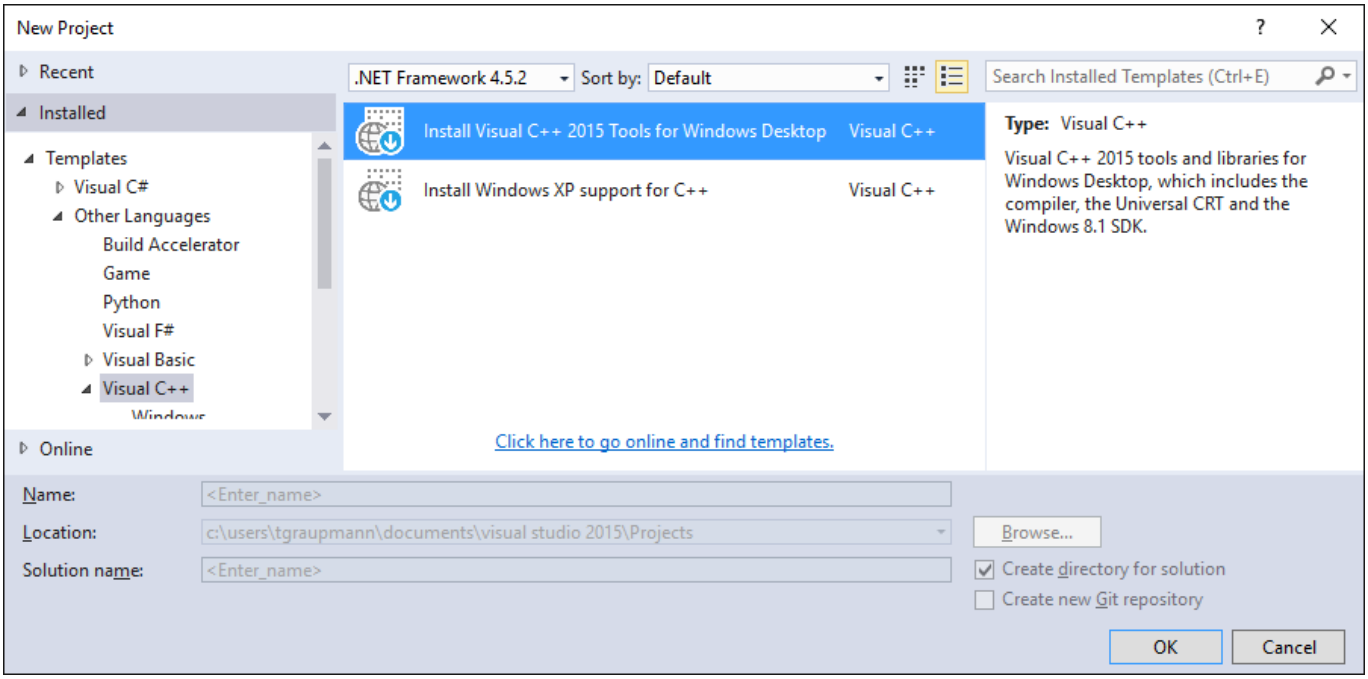
Dependencies

To use the Chroma SDK first install the new Razer Synapse and Chroma App.



- If you don't have Chroma hardware, you can see Chroma effects with the Chroma Emulator
- The CChromaEditor library is full source and can build with VS 2015/2017/2019/2022 so make sure you include the corresponding Microsoft Visual C++ Redistributable for Visual Studio with your application for the versions that you build with. The core ChromaSDK is built with VS 2015, so you'll want that redistributable at a minimum.
- To compile: Install Visual Studio

- To compile: Install Windows 10 SDK
- To compile: Install `Templates->Other Languages->Visual C++->Visual C++ 2015 Tools for Windows Desktop` which can be installed through the `Visual Studio New Project Dialog`



- Use retarget solution to select an available Windows SDK.

About

The CChromaEditorLibrary is a DLL that is shared by Chroma RGB plugins to provide a common API and universal animation format with playback and editing capabilities.

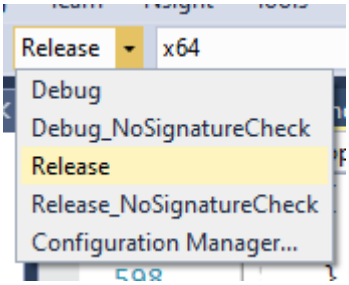
Security

The C++ Chroma Editor Library loads the core Razer DLL `RzChromatic.dll` and the Razer stream library `RzChromaStreamPlugin.dll`. To avoid a 3rd party injecting malicious code, the C++ Chroma Editor Library checks for a valid signature on the Razer libraries. The DLL issuer is validated to be `Razer USA Ltd.` Init and InitSDK will return `RZRESULT_DLL_INVALID_SIGNATURE` if the signature check fails.

The sample apps use the `CHECK_CHROMA_LIBRARY_SIGNATURE` preprocessor definition to enable signature checking on the Chroma Editor Library. Signature checking can be used on the Razer libraries downloaded from Github releases.

```
#ifdef CHECK_CHROMA_LIBRARY_SIGNATURE
    // verify the library has a valid signature
    _sInvalidSignature = !VerifyLibrarySignature::VerifyModule(path);
#endif
```

The project has `NoSignatureCheck` configurations to debug with unsigned builds for testing feature updates. The `NoSignatureCheck` configurations do not set the `CHECK_CHROMA_LIBRARY_SIGNATURE` preprocessor definition.



Chroma Editor Library

The `Chroma Editor Library` is a helper library for Chroma animation playback and realtime manipulation of Chroma animations.

The latest versions of the `Chroma Editor Library` can be found in Releases for `Windows-PC` and `Windows-Cloud`.

Windows PC

For `Windows PC` builds the `RzChromatic.dll` and `RzChromaStreamPlugin.dll` are not packaged with the build. These libraries are automatically updated and managed by Synapse and the Chroma Connect module. Avoid including these files in your build folder for `Windows PC` builds.

32-bit libraries

```
Win32BuildFolder\CChromaEditorLibrary.dll
```

64-bit libraries

```
Win64BuildFolder\CChromaEditorLibrary64.dll
```

Windows Cloud

`Windows Cloud` builds run on cloud platforms using `Windows` such as `Amazon Luna`, `Microsoft Game Pass`, and `NVidia GeForce Now`. Game instances run in the cloud without direct access to Chroma hardware. Chroma effects stream across the Internet to reach your local machine and connected hardware. No extra code is required to add Cloud support. In the case with `NVidia GeForce Now`, the cloud runs the same Epic Games and Steam builds as the desktop version and support Chroma streaming. Viewers can watch the cloud stream via the Razer Stream Portal.

SDK Integration

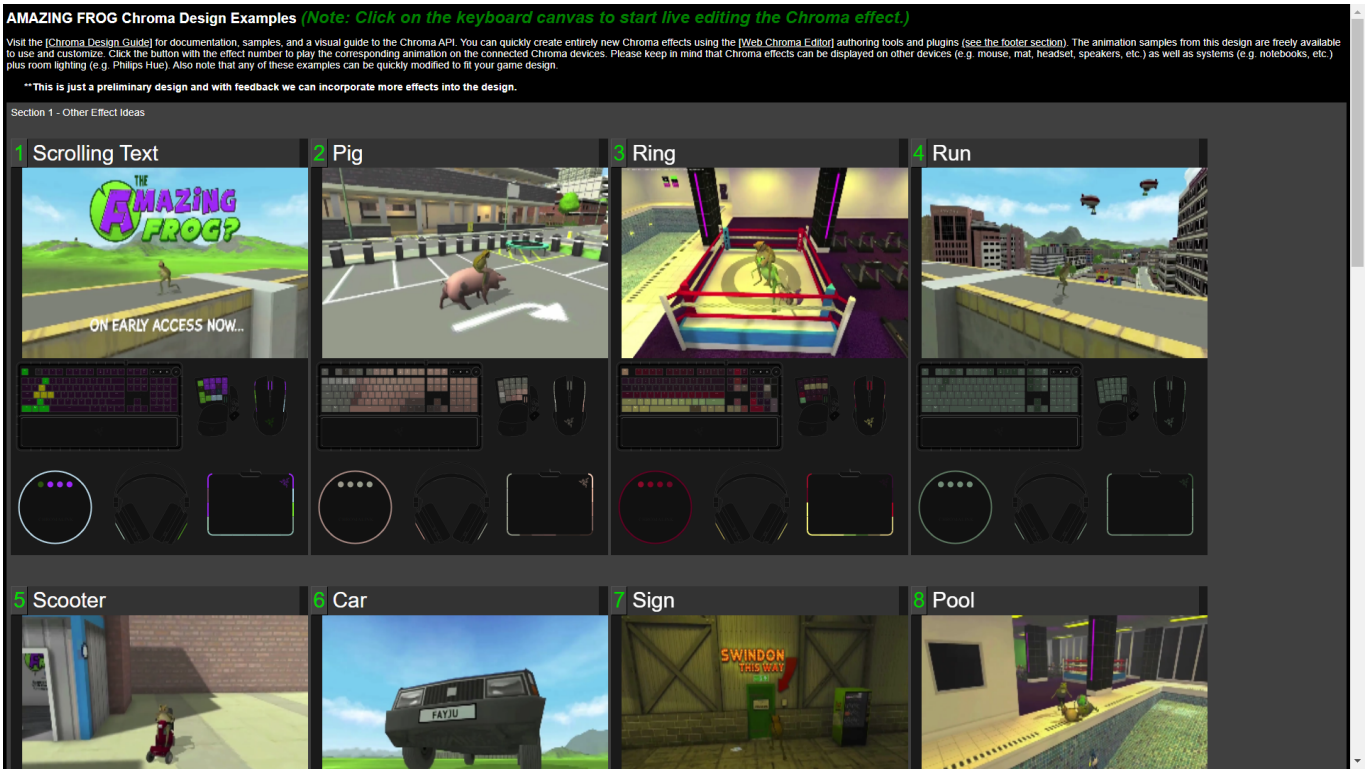
The SDK integration process involves the following:

1. Chroma Design
2. Revisions

3. Sample Project
4. Tools
5. Integration
6. Testing
7. Haptic Design
8. Modding

Chroma Design

The Chroma Design is the starting point. The team provides 15 sample effects that play on an animated web page. The sample effects correspond to short gameplay video clips and give an idea to the type of animation that could play for a set of game events. The samples are available to use for the specified effect or can be used for any other effect which is completely up to the developer. The developer may ask for effect revisions or additional sample effects. If gameplay video is not available, the developer can provide a description or reference art to conceptualize the desired effect.



Revisions

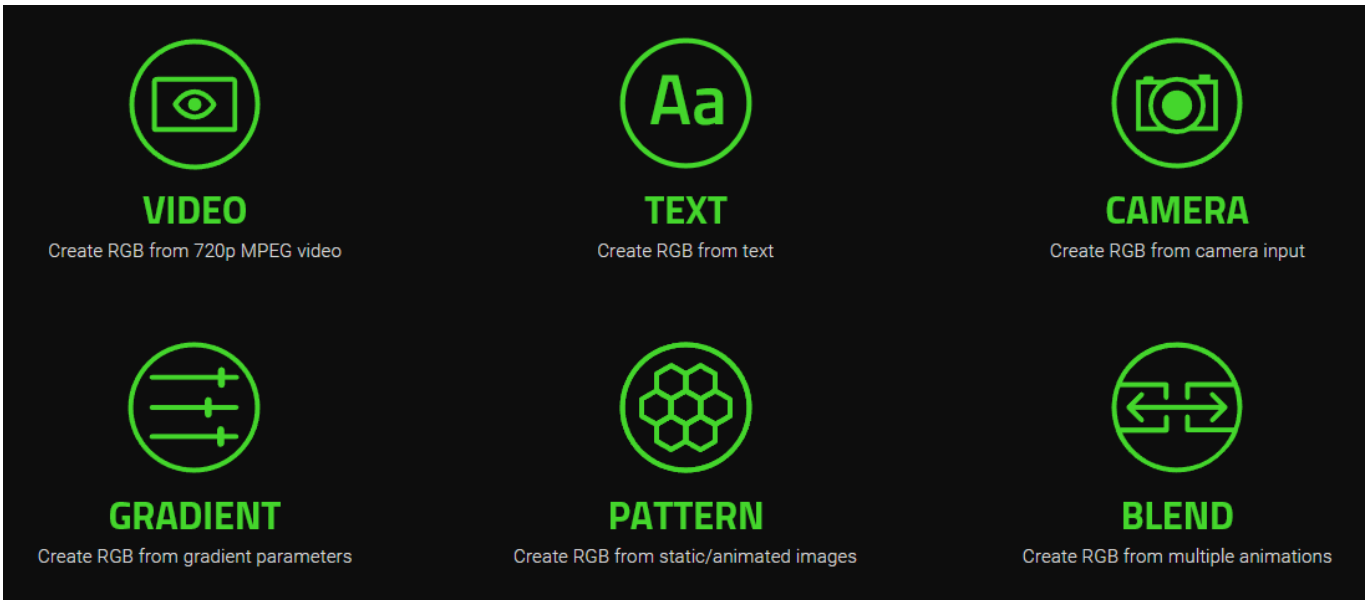
Some Chroma Designs require revisions to add more requested effects or to make changes through the feedback of reviewing the Chroma Design. Revisions can be requested which result in a subset of alterations from the previous design or add completely new game events. **Fill out the Chroma_Sensa_Template_Developers.xlsx Template which provides all the necessary fields for making design requests and revisions.**

Sample Project

The developer specifies which game engine is used by the game so that a sample project can be shared with sample code for the specified engine. The sample project will have the same effects that were defined in the Chroma Design and ported to the target language/game engine. The sample project will include a plugin to add the Chroma SDK to the specified game engine, and the ported sample code and sample animations from the `Chroma Design`.

Tools

- The Web Chroma Editor creates Chroma animations and code snippets from several input sources. Designers can create Chroma animations without writing any code. The toolset can use input sources as video, text, camera, web cam, desktop capture, gradients, patterns, images, and blended animations.



- The Chroma Design Converter can automatically port a web based Chroma Design to several languages and game engines.
- The Synesthesia Console can generate haptic configurations automatically for your Chroma integration.

Integration

The integration process can be as easy as copy and paste from the sample project into the game code. Most likely, it's a matter of finding game triggers in the game code to find the optimal place to add a call to `PlayAnimation()`. The typical Chroma integration process lasts 3 - 5 days for a single developer. Haptics integration can take 0 days by using automatic mode. Manually adding haptics can take about the same amount of work as Chroma to add the calls to `SetEventName()` in the right places. Chroma and haptics are independent meaning sometimes they play together and sometimes they play separately, which is completely up to the designer. **In most cases for game engines after the game build completes, the Chroma animations need to be copied to the animation folder within the game's content folder.**

Testing

The team can provide QA on the game build when integration has completed. Steam beta keys and Epic Store beta keys make testing possible before a game launches. This can be a good way to provide design revisions by testing and giving feedback on the build. To support the QA process, it will be important to include a level selector and potentially console commands that make it easy to navigate the build to test the game triggers

at the right moments to validate the visuals work as expected. Beta key access is limited to the engineering and QA review team.

Haptic Design

Just like Chroma Designs, the Haptic Design can be provided by the team. Adding haptic support does not require adding assets to the game. Haptics can be added to a game without code changes and after the game has released. Haptics can be added through creation of a haptic configuration file. Developers can use the Synesthesia Console which automates creation of the haptic configuration file within `HapticFolders` and will add some mockup haptic files (simple haptic effect which can be edited with Haptic Composer) when event names follow a naming convention. Haptic configuration files are automatically distributed by the team through `Chroma App` updates.

Modding

The decision to add Chroma mod support for a title is completely up to the developer. If the developer decides to block modding, Chroma animations can be loaded from a byte array which sandboxes and protects against any modifications to the Chroma animation assets. If the developer wants to use modding, Chroma animation assets are placed within the installation directory. Modders can modify the Chroma animations assets that are loaded by the title. The API provides `CloseAnimation` which reloads the Chroma animation from disk. This allows Chroma animations to be modified externally without needing to relaunch the title. Chroma animation playback also supports relative paths from the content folder. Relative paths can be used to organize several mods within the content folder. The title can have a configuration menu that switches between mod subfolder names which changes the relative path for loading the Chroma animations. The C++ Chroma Mod Sample shows how relative paths can be used to detect and use mods, which is applicable for any game or custom engine.

General

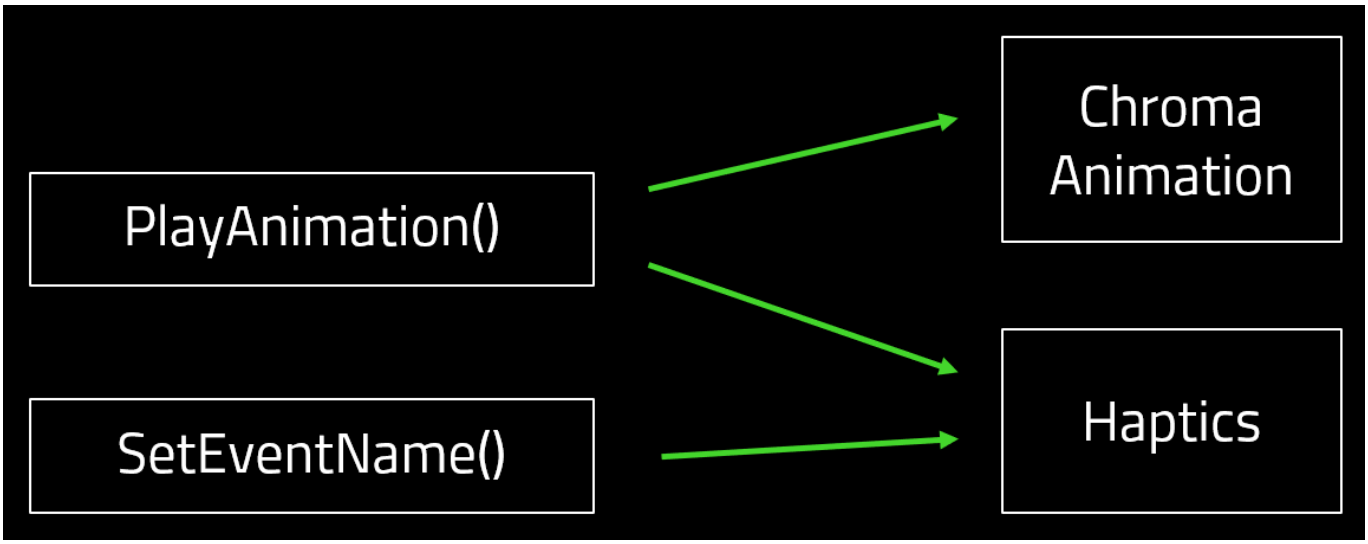
- The Chroma SDK allows an application or game to set the details in the Chroma Apps list within the `Chroma App`.

This document provides a guide to integrating Chroma RGB using the Chroma C++ SDK. Chroma can be included through premade Chroma animations or APIs. Here is the list of available methods:

- Initialize SDK: Initialize the Chroma SDK to use the library.
- Is Active: Check if the app/game has Chroma focus.
- Is Connected: Check if Chroma hardware is connected.
- Play Chroma Animation: Playback a Chroma animation asset.
- Set Event Name: Name a game event or game trigger in order to also add Haptics to the Chroma event.
- Use Forward Chroma Events: Enable or disable automatic invocation of `SetEventName()` when invoking `PlayAnimation()` using the animation name.

Chroma Sensa

Chroma Sensa is the combination of Chroma and Razer Sensa HD Haptics in a single SDK. By integrating RGB lighting and haptics into game environments and events, players can enjoy a truly immersive gaming experience. The `Chroma SDK` is capable of playing Chroma animations and haptics on the Razer Sensa HD Haptics devices. The default mode allows automatic triggering of haptics effects when Chroma animations are played with `PlayAnimation()`. Manual mode is set by `UseForwardChromaEvents(false)` and haptics can be triggered independently of Chroma animations with SetEventName().



Event names can follow a naming convention which assists with the generation of the haptics configuration for your title. Event names are specified with the `SetEventName()` method. The event name suffix can be left off or used to prepopulate common settings for `_ON`, `_OFF`, and `_MERGE`.

- "Jump" - (without a suffix) Existing haptics stop, the named haptic plays to completion and then ends
- "Attack_ON" - Existing haptics continue to play, the named haptic plays as a continuous looping haptic
- "Attack_OFF" - Existing haptics continue to play, the named looping haptic stops
- "Punch_MERGE" - Existing haptics continue to play, the named haptic plays to completion and ends
- "Block_MERGE" - Existing haptics continue to play, the named haptic plays to completion and ends

Upon completion of Chroma and haptic implementation, the list of Chroma events and game triggers should be shared with the team to be add to the game's Chroma Workshop entry.

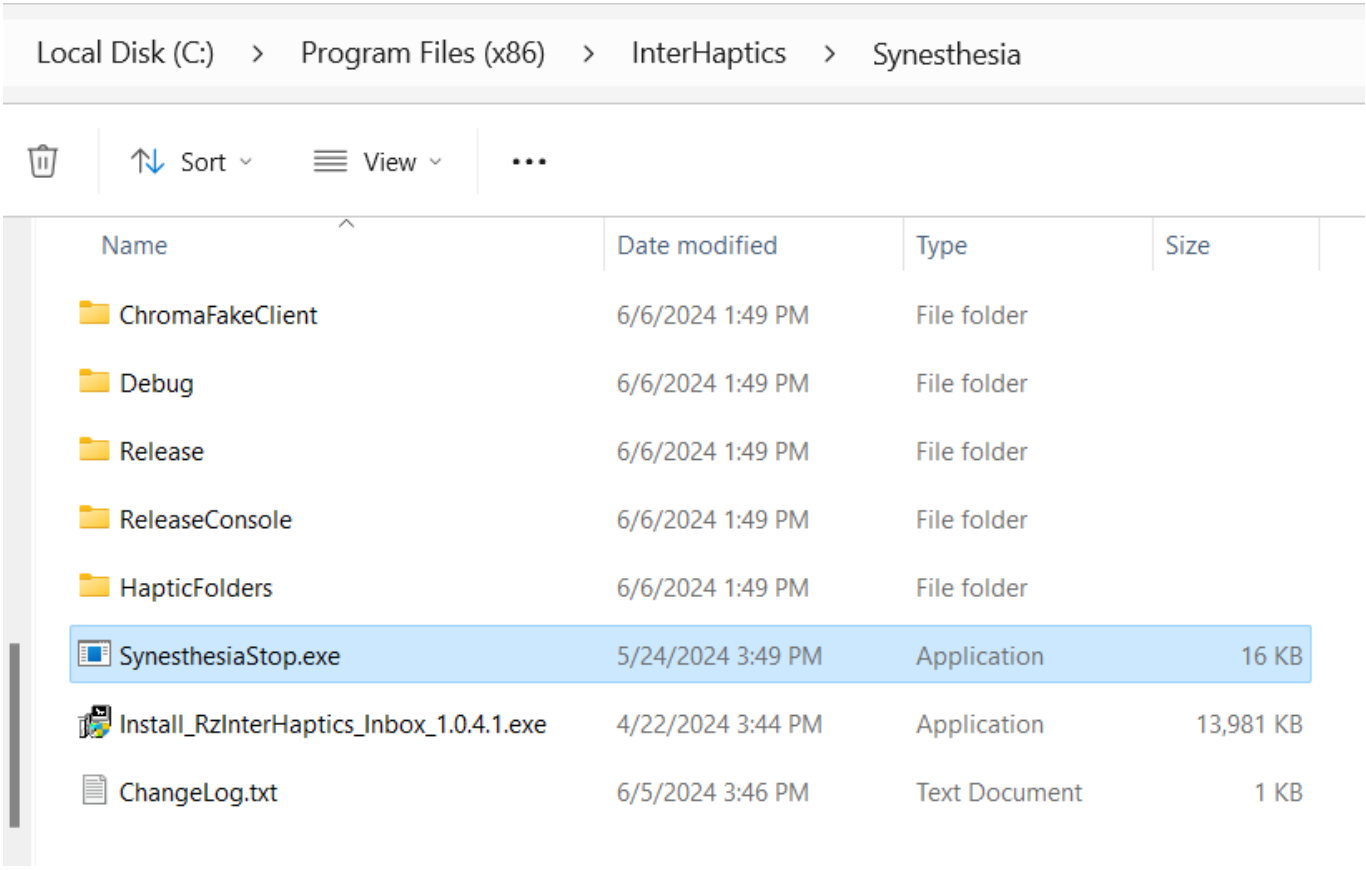
Targeting features can be **optionally** described for each haptics effect.

- "Target" defaults to `"All"`. GroupID options can be found at https://www.interhaptics.com/doc/interhaptics-engine/#groupid
- "Spatialization" defaults to `"Global"`. Other LateralFlag options can be found at https://www.interhaptics.com/doc/interhaptics-engine/#lateralflag
- "Gain" defaults to 1.0.

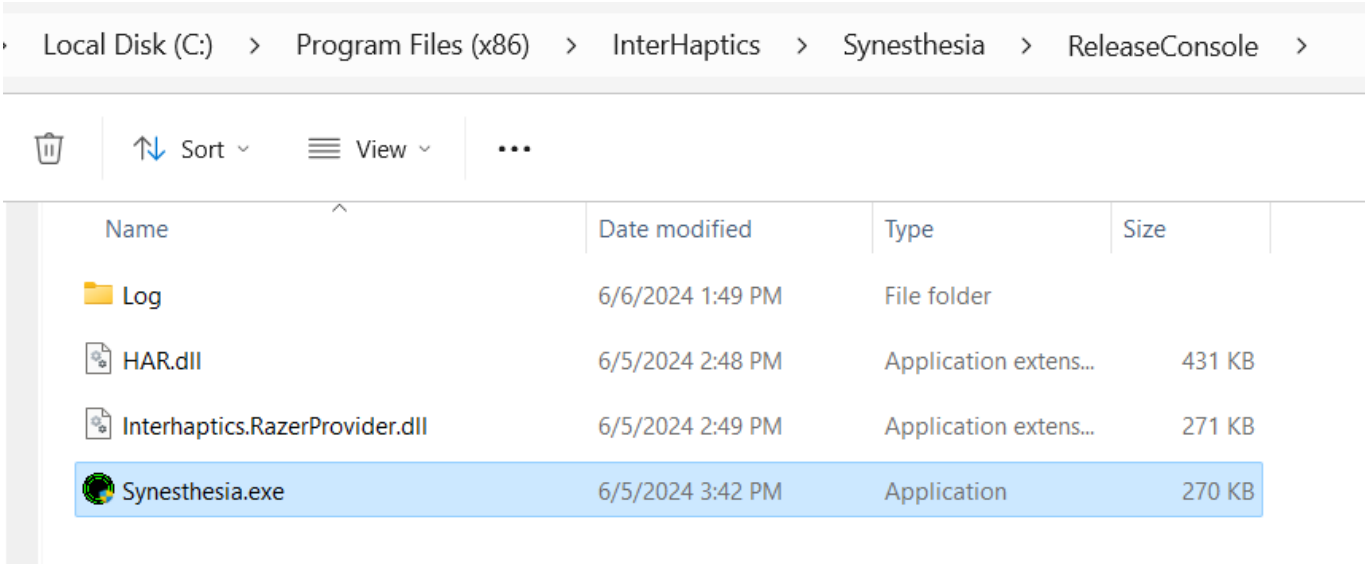
Synesthesia

The Synesthesia Console makes creating the haptics configuration for game integration super easy. Download and run the installer to get started creating a haptics config.

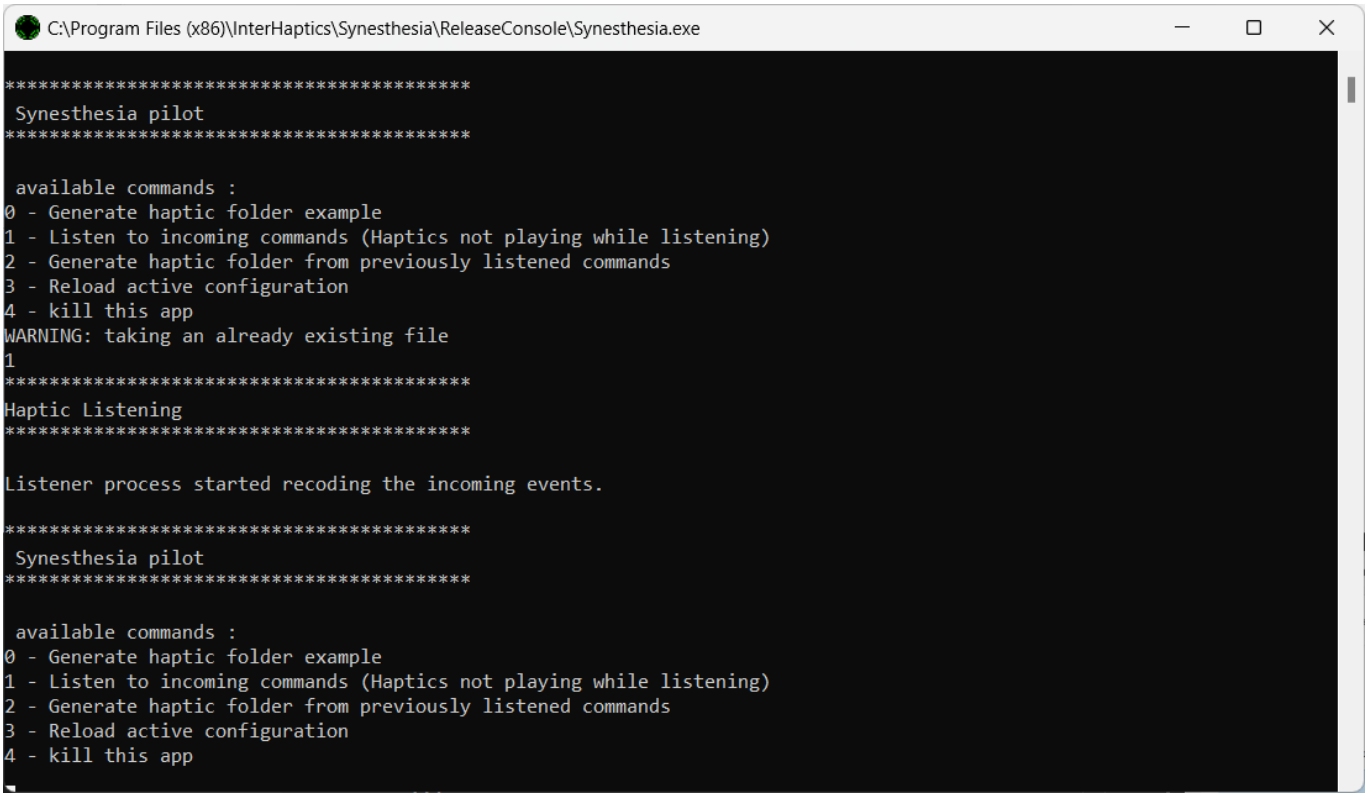
1. Run `SynesthesiaStop.exe` to stop any existing background or haptic consoles



2. Run the `Synesthesia Console` for the interactive prompt



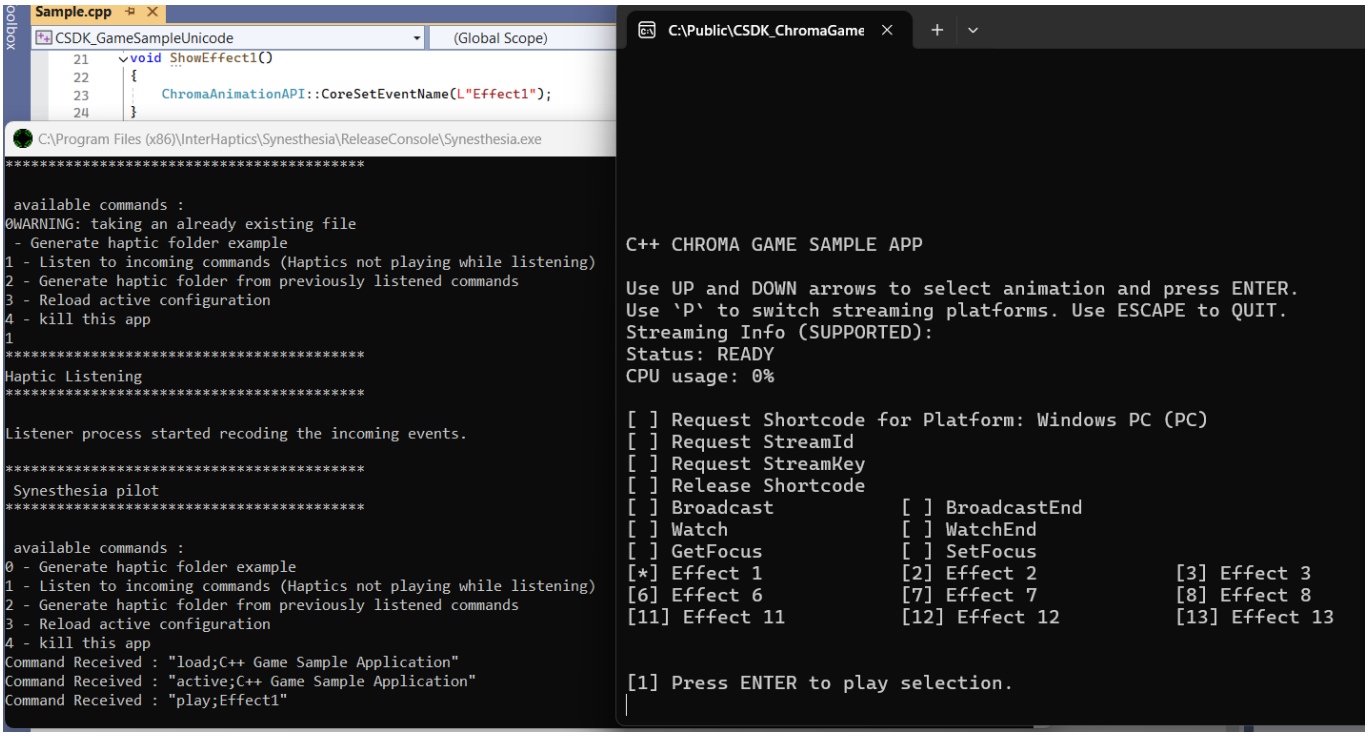
3. Enter option `1` and press `Enter` to listen for incoming commands



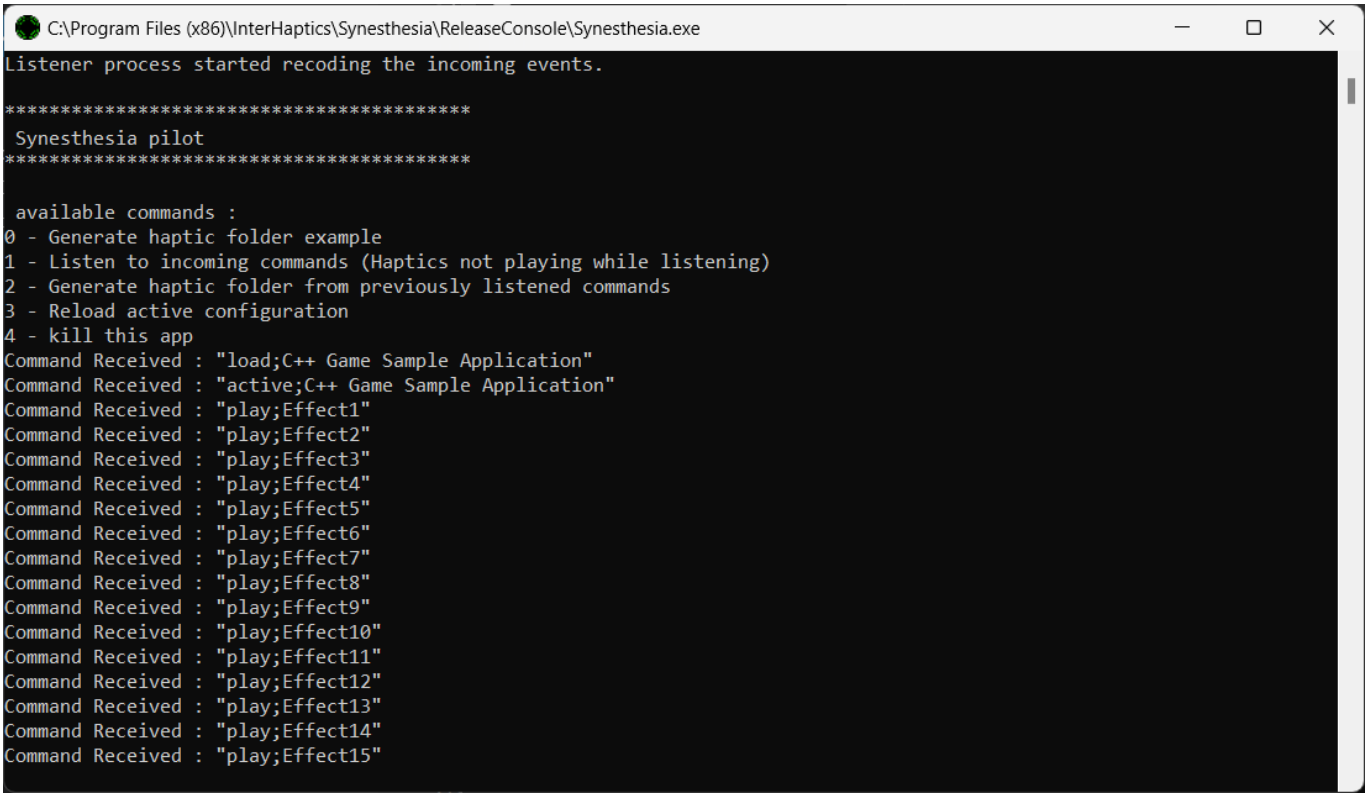
4. Launch your game that uses `PlayAnimation` or `SetEvent` directly to trigger haptic commands.

When the application launches and initializes Chroma, the command to `load` the haptic configuration file is sent. When the application receives Chroma focus, the `active` command is sent. When `PlayAnimation` or `SetEvent` is called, the `play` command is sent.

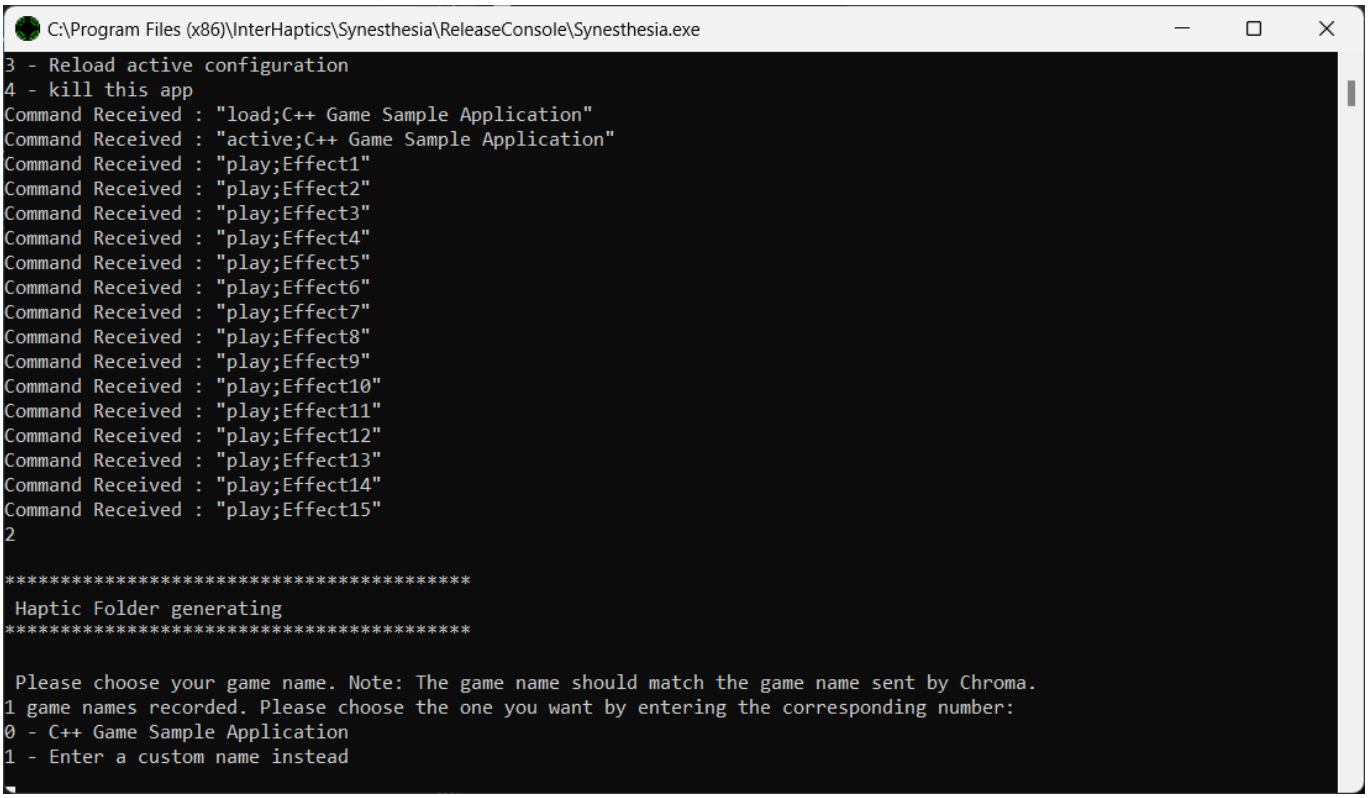
```
Command Received : "load;C++ Game Sample Application"
Command Received : "active;C++ Game Sample Application"
Command Received : "play;Effect1"
```



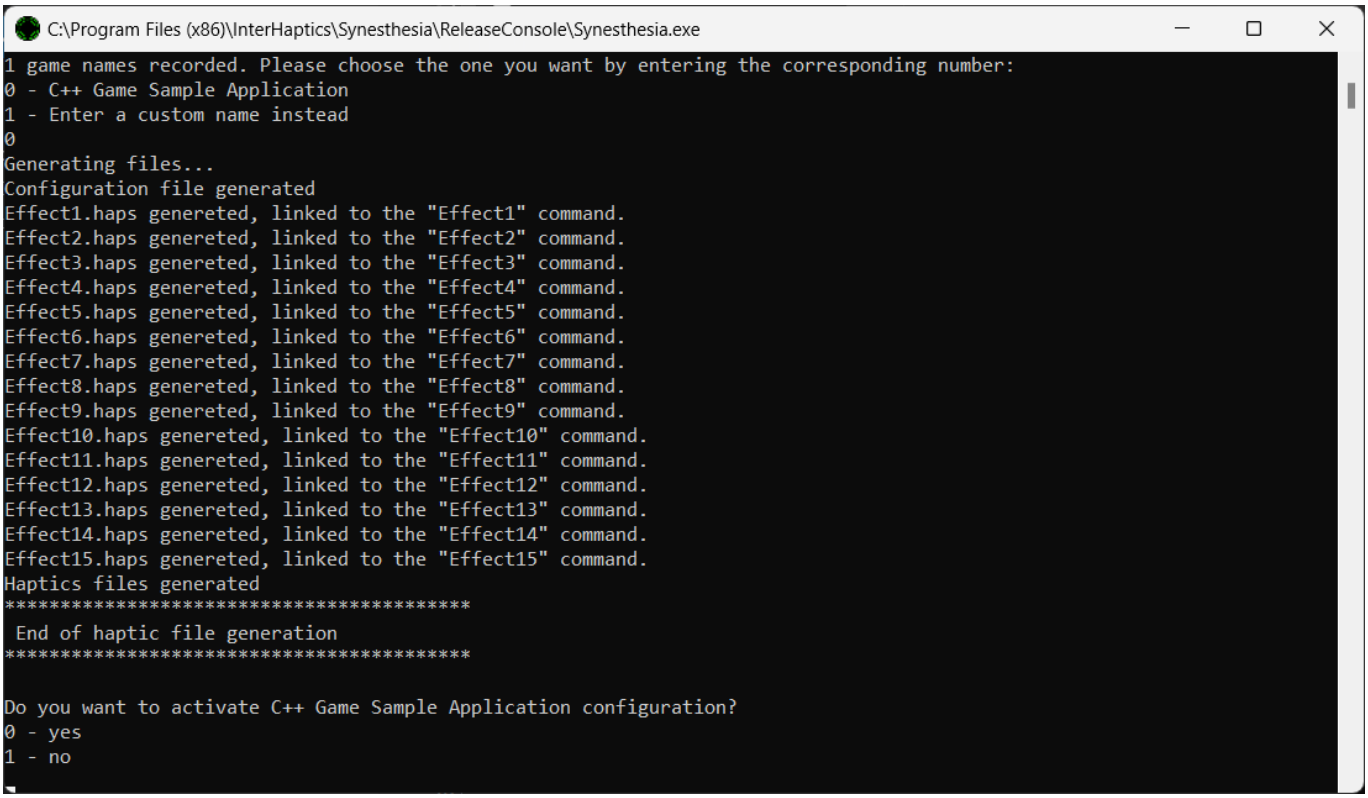
5. Play through all the game triggers to send any possible commands the game might use. This will be useful for generating the haptic configuration next.



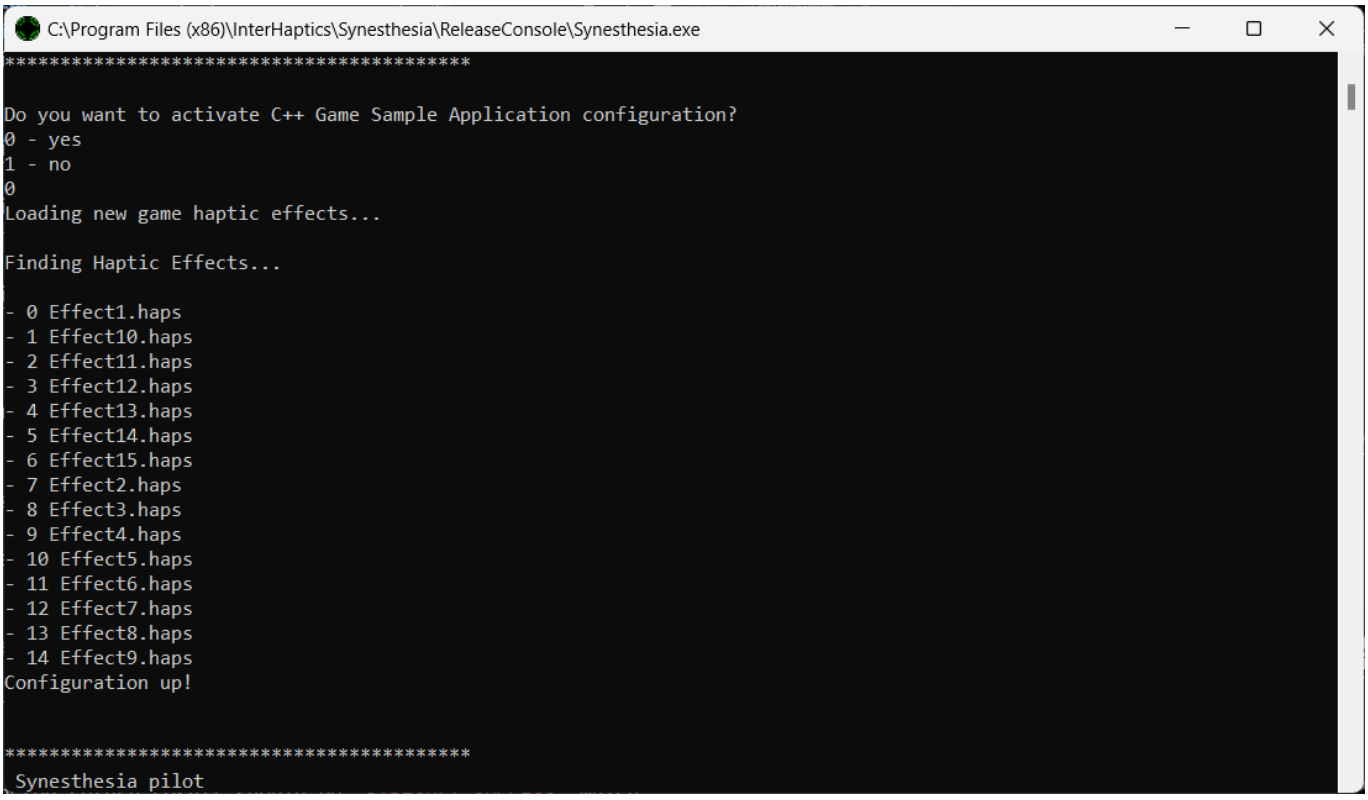
6. Enter option `2` and press `Enter` to generate the haptics configuration



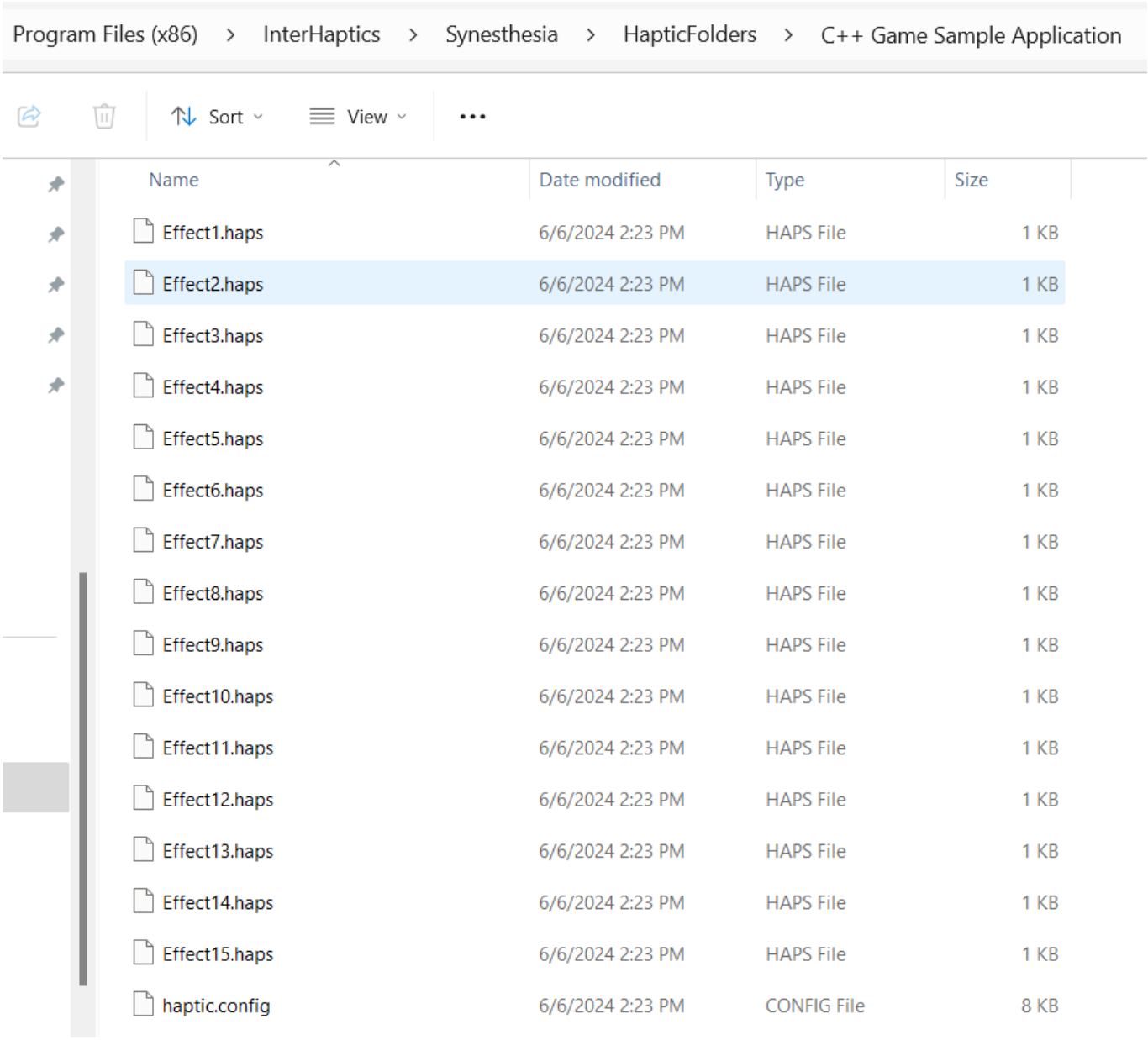
7. Enter option `0` and press `Enter` to use the detected application name used by the Chroma initialization



8. Enter option `0` and press `Enter` to use activate the new haptic configuration file. Now when the game triggers haptic events, the configured haptic events will play.



The `haptic.config` and `haps` default haptics effects were generated in the `HapticFolders` by the console.



The `haptic.config` contains default targeting for the generated entries for each detected command.

```
{
    "ExternalCommands": [
        {
            "External_Command_ID": "Effect1",
            "Haptic_Events": [
                {
                    "Haptic_Effect": "Effect1",
                    "Loop": 1,
                    "Mixing": "Override",
                    "Targeting": [
                        {
                            "Gain": 1.0,
                            "Spatialization": "Global",
                            "Target": "All"
                        }
                    ]
                }
```

```
            ]
        },
        ...
    ]
}
```

Namespace

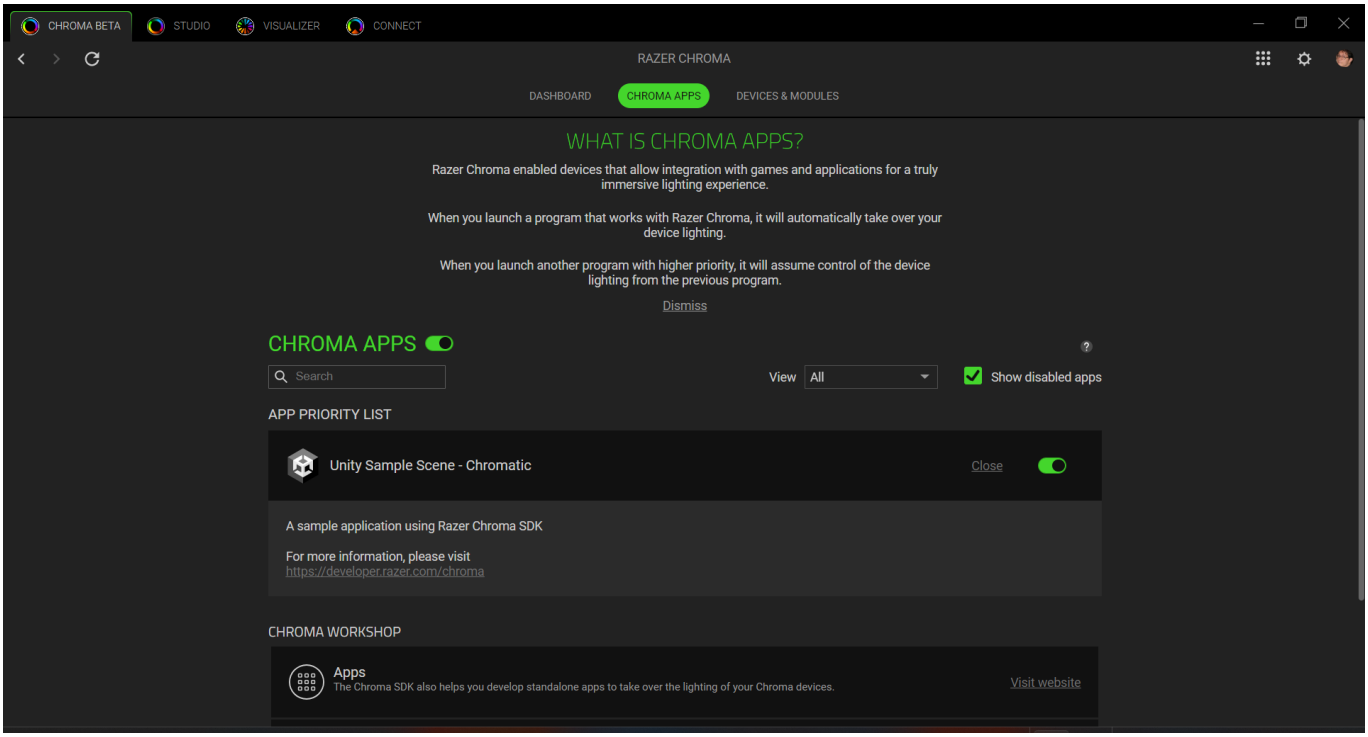
Add the Chroma SDK namespace to use the API.

```
#include "Razer\ChromaAnimationAPI.h"

using namespace ChromaSDK;
```

Initialize SDK

Initialize the Chroma SDK in order to utilize the API. The `InitSDK` method takes an `AppInfo` parameter which defines the application or game details that will appear in the `Chroma App` within the `Chroma Apps` tab. The expected return result should be `RZRESULT_SUCCESS` which indicates the API is ready for use. If a non-success result is returned, the Chroma implementation should be disabled until the next time the application or game is launched. Reasons for failure are likely to be the user does not have the `Synapse` or the `Chroma App` installed. After successfully initializing the Chroma SDK, wait approximately 100 ms before playing Chroma animations.




```
APPINFOTYPE appInfo = {};

_tcscpy_s(appInfo.Title, 256, _T("Sample Game Title"));
_tcscpy_s(appInfo.Description, 1024, _T("Sample Game Description"));
```

```
    _tcscpy_s(appInfo.Author.Name, 256, _T("Company Name"));
    _tcscpy_s(appInfo.Author.Contact, 256, _T("Company Website or Email"));

    //appInfo.SupportedDevice =
    //    0x01 | // Keyboards
    //    0x02 | // Mice
    //    0x04 | // Headset
    //    0x08 | // Mousepads
    //    0x10 | // Keypads
    //    0x20   // ChromaLink devices
    appInfo.SupportedDevice = (0x01 | 0x02 | 0x04 | 0x08 | 0x10 | 0x20);
    //    0x01 | // Utility. (To specifiy this is an utility application)
    //    0x02   // Game. (To specifiy this is a game);
    appInfo.Category = 0x02;

    RZRESULT result = ChromaAnimationAPI::InitSDK(&appInfo);
   if (result == RZRESULT_SUCCESS)
    {
       // Init Success! Ready to use the Chroma SDK!
   }
   else
    {
       // Init Failed! Stop using the Chroma SDK until the next game launch!";
   }
```

Applications should uninitialize the Chroma SDK with Uninit() for a clean exit. Uninitialization is only needed if the Chroma SDK was successfully initialized.

```
int result = ChromaAnimationAPI::Uninit();
if (result == RZRESULT_SUCCESS)
{
    // Chroma has been uninitialized!
}
else
{
    // Uninitialization was unsuccessful!
}
```

Is Active

Many applications and games can use the Chroma SDK at the same time, yet only one can have the Chroma focus. The `APP PRIORITY LIST` defines the priority order and the highest on the list receives the Chroma focus when more than one are actively using the Chroma SDK. Users can adjust the priority order by dragging and dropping or toggling the app completely off. The IsActive() method allows an application or game to check if it has Chroma focus. This allows the title to free up overhead when Chroma is not in use. If a title uses this to check for focus, the state should be periodically checked to turn Chroma back on when focus is returned. When active returns false, the title can stop playing Chroma animations, disable idle animations, and inactivate dynamic Chroma to free up some overhead. Keep in mind that some apps use Chroma notifications so they will only briefly take Chroma focus and then return it typically over a 5 second period.

```
bool isActive;
int result = ChromaAnimationAPI::IsActive(isActive);
if (result == RZRESULT_SUCCESS)
{
    if (isActive)
    {
        // The game currently has the Chroma focus!
    }
    else
    {
        // The game does not currently have the Chroma focus!"
    }
}
else
{
    // Unable to check for Chroma focus. Unexpected result!
}
```

Is Connected

To further reduce overhead, a title can check if supported devices are connected before showing Chroma effects. The IsConnected() method can indicate if supported devices are in use to help determine if Chroma should be active. Games often will include a menu settings option to toggle Chroma RGB support, with being on by default as an additional way that users can minimize overhead.

```
DEVICE_INFO_TYPE deviceInfo = { DEVICE_INFO_TYPE::DEVICE_ALL };
int result = ChromaAnimationAPI::CoreIsConnected(deviceInfo);
if (result == RZRESULT_SUCCESS)
{
    if (deviceInfo.Connected > 0)
    {
        // Chroma devices are connected!
    }
    else
    {
        // "No Chroma devices are connected!";
    }
}
else
{
    // "Unable to check for Chroma devices. Unexpected result!";
}
```

Play Chroma Animation

The Chroma SDK supports playing premade Chroma animations which are placed in the `StreamingAssets` folder or subfolders within. Chroma animations can be created in the web authoring tools, or dynamically created and modified using the API. Call PlayAnimation() to play Chroma animations with or without looping.

Animations have a device category, and playing an animation will stop an existing animation from playing before playing the new animation for the given device category. The animation name is file path of the Chroma animation relative to the `StreamingAssets` folder.

```
bool loop = false;
vector<string> devices =
{
    "ChromaLink",
    "Headset",
    "Keyboard",
    "Keypad",
    "Mouse",
    "Mousepad"
};
for (int i = 0; i < devices.size(); ++i)
{
    string animationName = "Animations/Spiral_" + devices[i] + ".chroma";
    ChromaAnimationAPI::PlayAnimationName(animationName.c_str(), loop);
}
```

Set Event Name

Chroma events can be named to add supplemental technology to your lighting experience. By naming game events and game triggers, the event name can be used as a lookup to play things like haptics effects. `SetEventName(L"Jump")` could be used when playing a Chroma animation of a jump effect. Using `L"Jump"` a corresponding haptic effect can be added with the Chroma effect to enhance emersion for the title. No other APIs are required to add haptics effects other than to invoke SetEventName(). To stop haptics playback use `SetEventName(L"")` with an empty string. A Chroma animation does not need to be playing in order to trigger haptics manually with SetEventName().

```
// Trigger haptic effect
int result = ChromaAnimationAPI::CoreSetEventName(L"Jump");
if (result == RZRESULT_SUCCESS)
{
    // Chroma event named successfully!"
}
else
 {
    // Unable to set event name. Unexpected result!"
 }

// Stop haptic playback
result = ChromaAnimationAPI::CoreSetEventName(L"");
if (result == RZRESULT_SUCCESS)
{
    // Haptics stopped successfully!"
}
else
{
```

```
    // Unable to stop haptics. Unexpected result!"
}
```

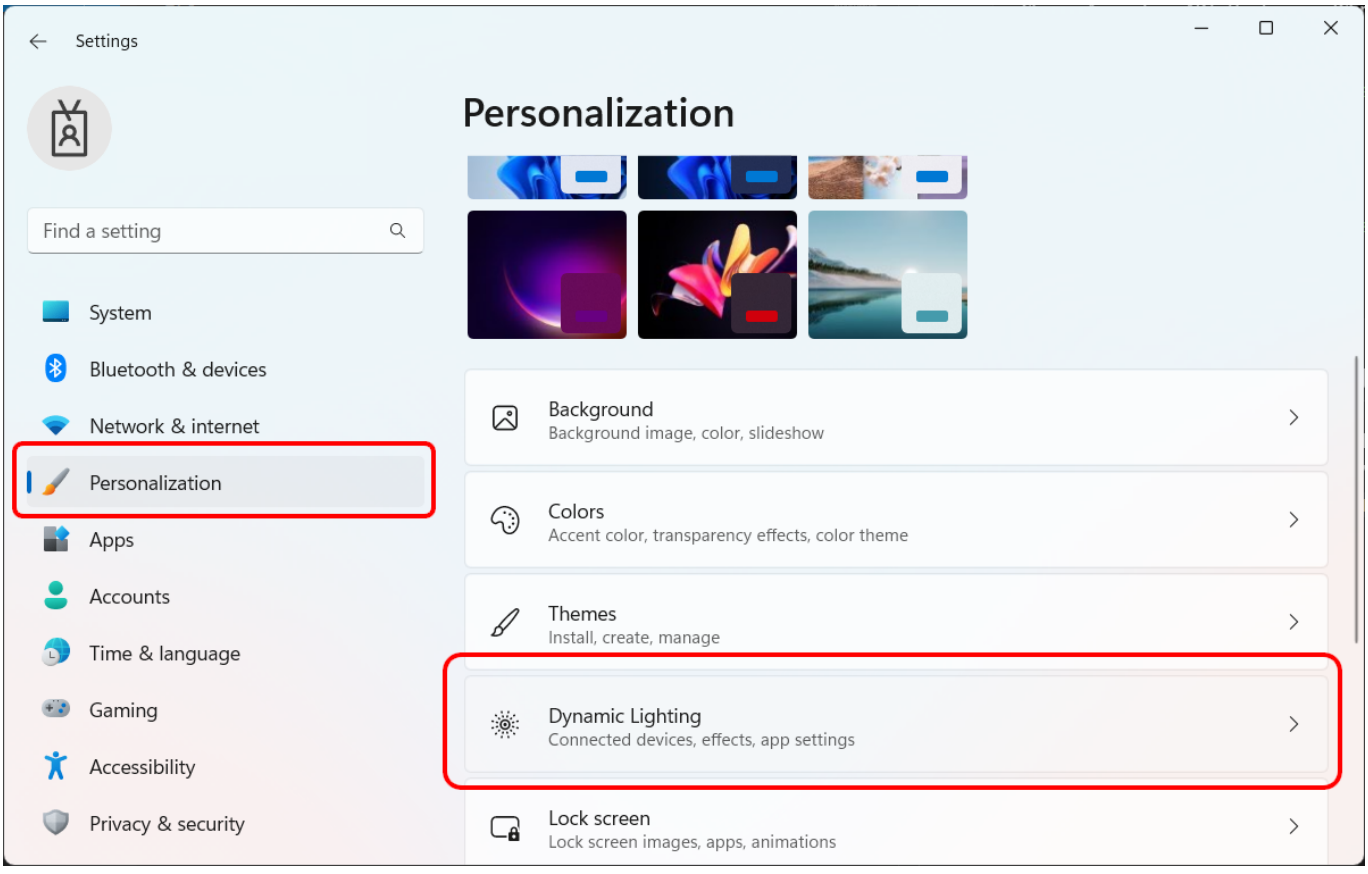
Use Forward Chroma Events

By default when PlayAnimation is called, the animation name is automatically sent to SetEventName(). In order to disable the default behaviour set the toggle to false. PlayAnimation() as shown above is called for each device category. It will be more efficent to use SetEventName() once for the Chroma animation set. Manual mode gives the title explicit control over when SetEventName() is called.

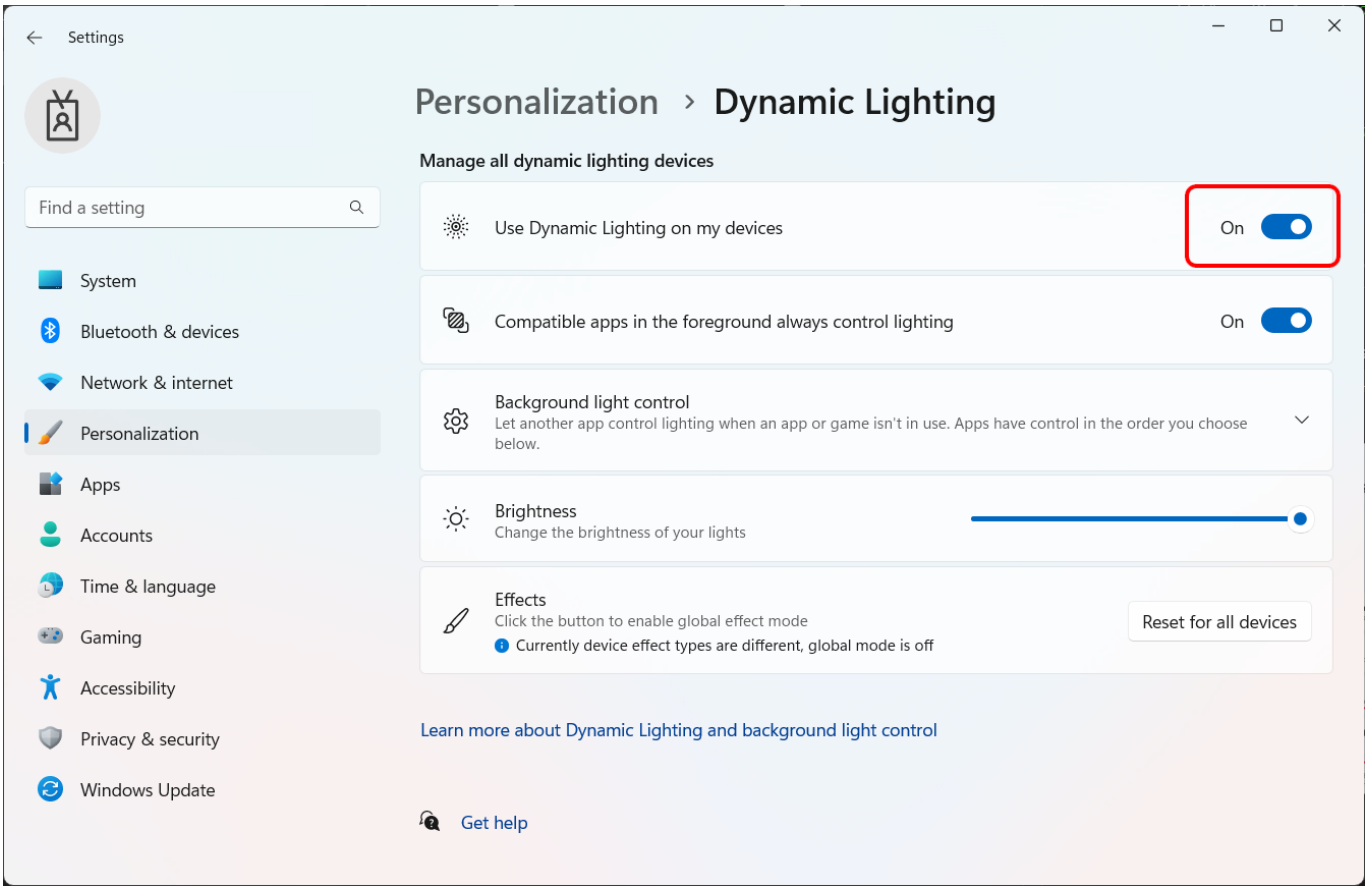
```
bool toggle = false; // manual mode
ChromaAnimationAPI::UseForwardChromaEvents(toggle);
if (toggle)
{
    // When PlayAnimation is used, the name is sent to SetEventName().
 }
else
{
    // The PlayAnimation name is not forwarded.
}
```

Microsoft Dynamic Lighting

Windows 11 launched Microsoft Dynamic Lighting which is built-in to the Windows Settings Personalization on Windows. Microsoft DL became generally available in `Windows 11 22H2`. See the list of supported devices.



For HID compatible devices, with `Dynamic Lighting` set to `ON` and `Chroma App` set as the ambient controller, Chroma effects will display on DL compatible hardware. No extra coding is required to add this compatibility. `Chroma App` handles Chroma compatibility with DL and it is completely automatic.



API Class

The `ChromaAnimationAPI` class provides a wrapper for the Chroma Editor Library. The wrapper for the API can be found at CConsoleEditor/ChromaAnimationAPI.h and CConsoleEditor/ChromaAnimationAPI.cpp.

Getting Started

Running the editor application

1 Run the Chroma Editor Installer to associate `.chroma` animations with the editor.

2 Double-click a `.chroma` animation file to open in the editor

Building the editor

1 Open `CChromaEditor.sln` in Visual Studio

2 The `CChromaEditorLibrary` project builds the native C++ DLL for `x64` and `x86` platforms

3 The `CConsoleEditor` project is a console project that uses the `DLL` and provides a command-line interface. The only parameter is the file path to a `Chroma` animation file. When no parameter is provided, the editor opens `temp.chroma` on the desktop.

Frameworks supported

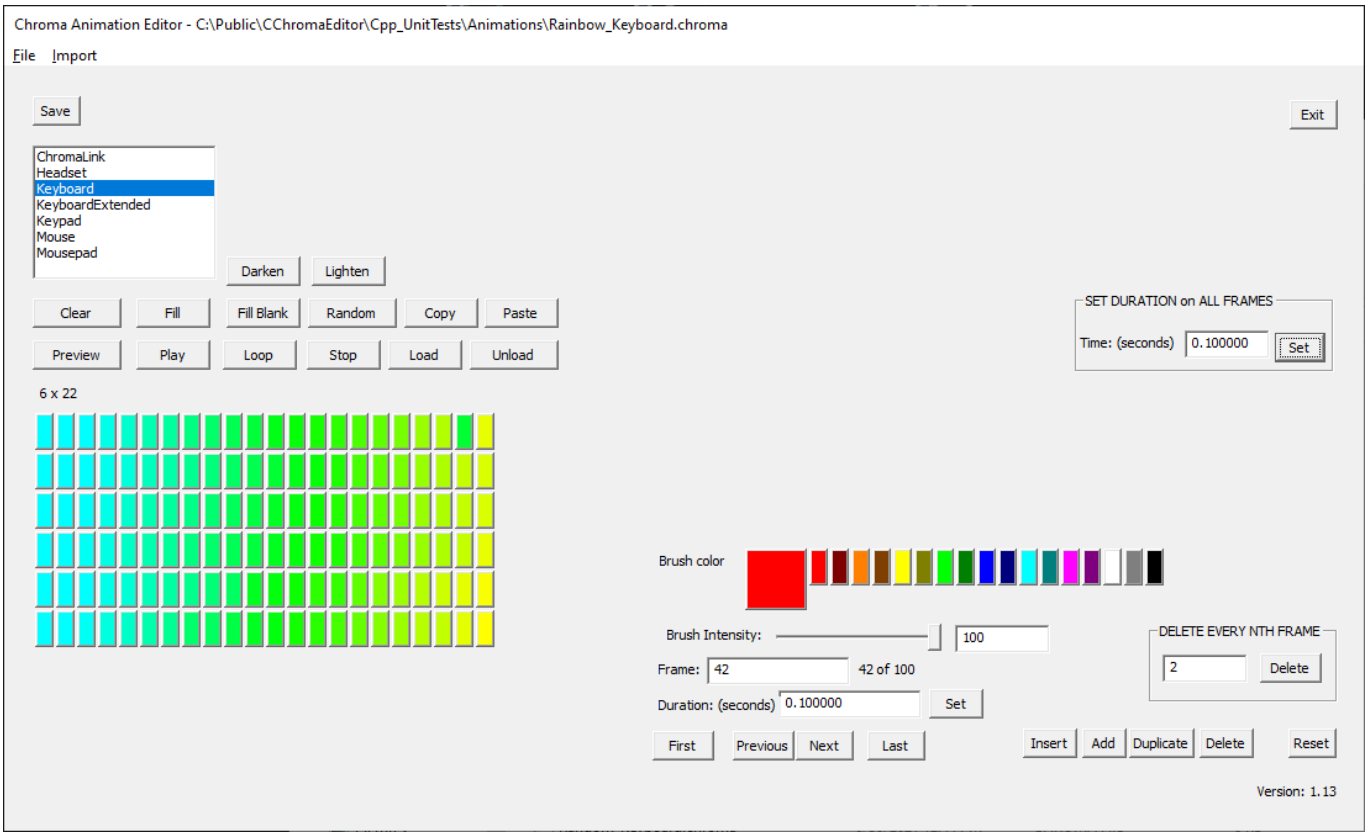
- Windows ChromaSDK (32-bit)
- Windows ChromaSDK (64-bit)

Assets

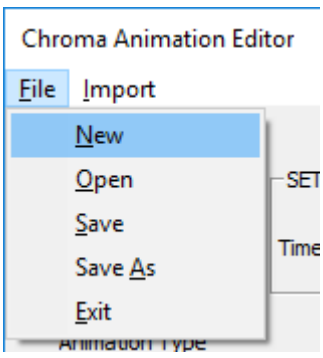
The `chroma` binary file format is supported in multiple engines and can even play on websites. Authoring tools like the Web Chroma Editor can easily record animations for the full device set which can be downloaded in a zip file. The `Web Chroma Editor` is also able to embed `chroma` animation files into code for specific languages.

Dialog

The standalone Chroma editor has a MFC dialog that can be used to create Chroma animations for the set of devices.



File Menu



New

Start with a new animationName

Open

Open a file dialog and open a `Chroma` animation

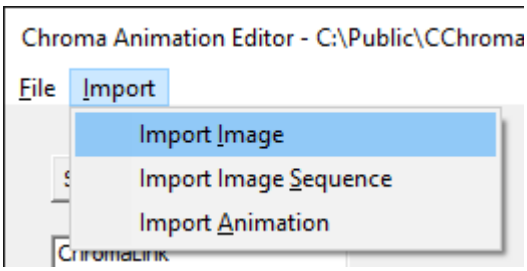
Save

Save the open animation

Save As

Open a file dialog and save the open animation

Import Menu



Import Image

Import a BMP, JPG, or PNG texture into the grid layout. The images will be stretched to fit the grid.

Import Image Sequence

Import a numbered set of images as an animation.

Import Animation

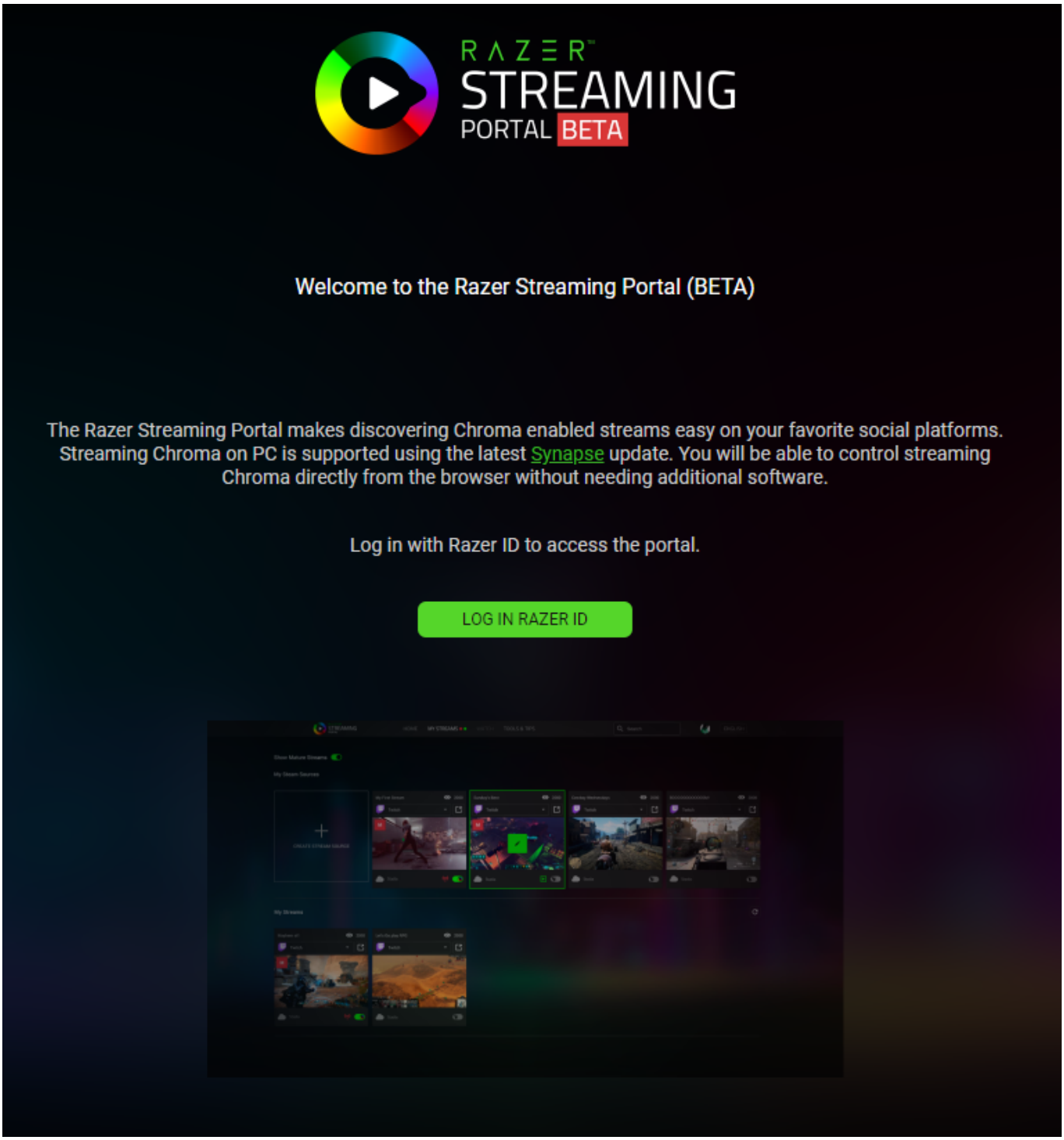
Import a GIF animation into the grid layout. Multiple frames will be added if they exist in the GIF. The image will be stretched to fit the grid.

Streaming

The Razer Streaming Portal supports streaming Chroma RGB on video streaming platforms and can be embedded on any 3rd party site. In order to stream Chroma RGB, you'll need the latest Synapse and Chroma Connect module. No other software is required to control streaming from the browser or the API below.

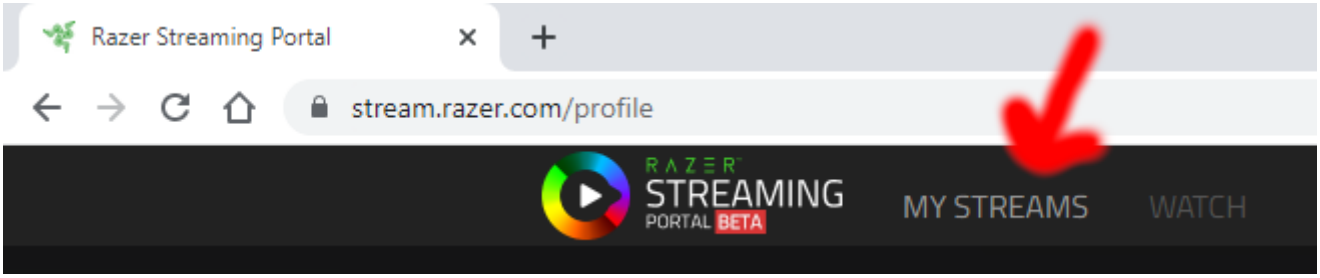
Login

The first step is to login into the Razer Streaming Portal with Razer ID.



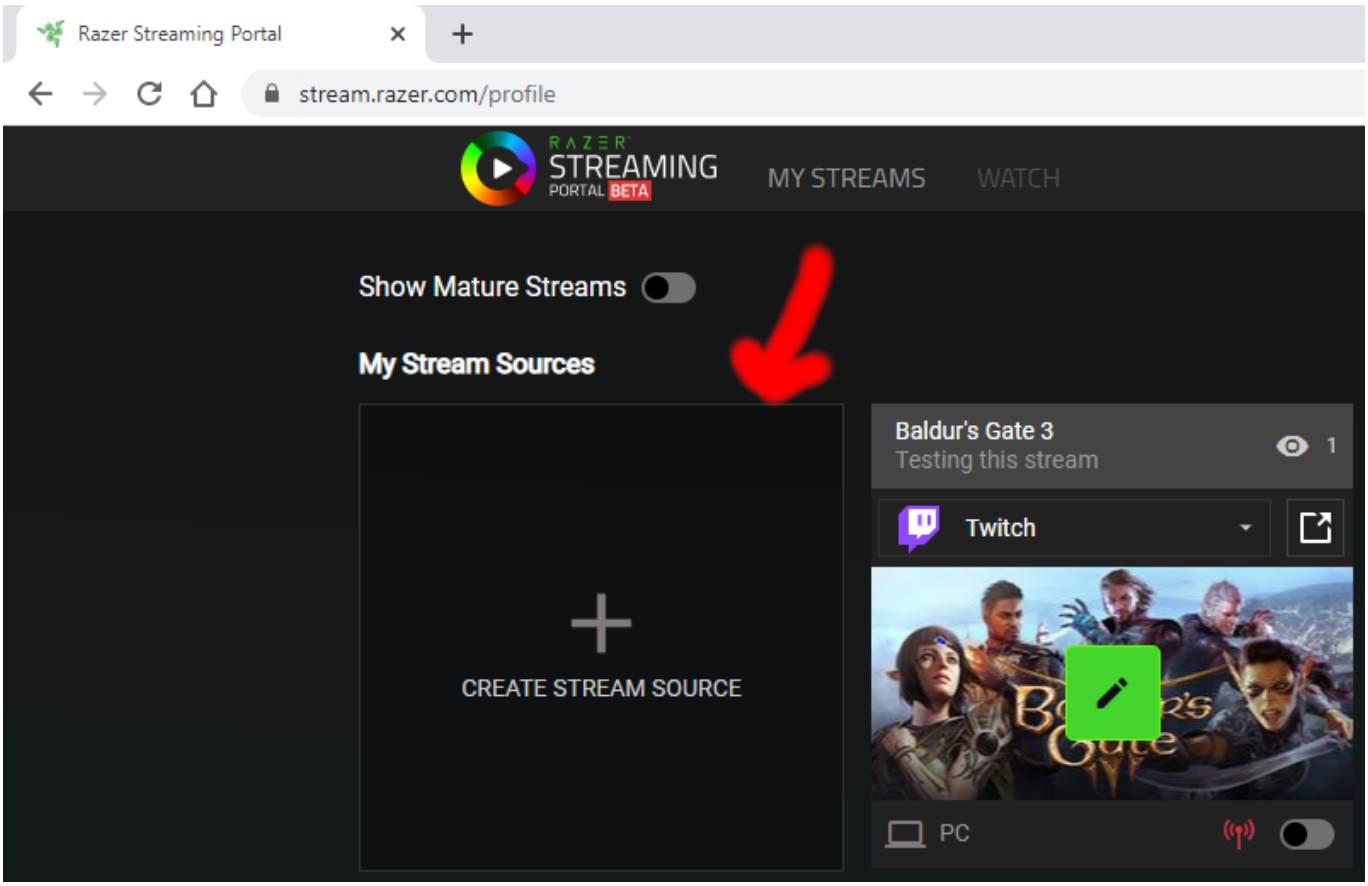
Profile

Navigate to the user profile by clicking `MY STREAMS`.



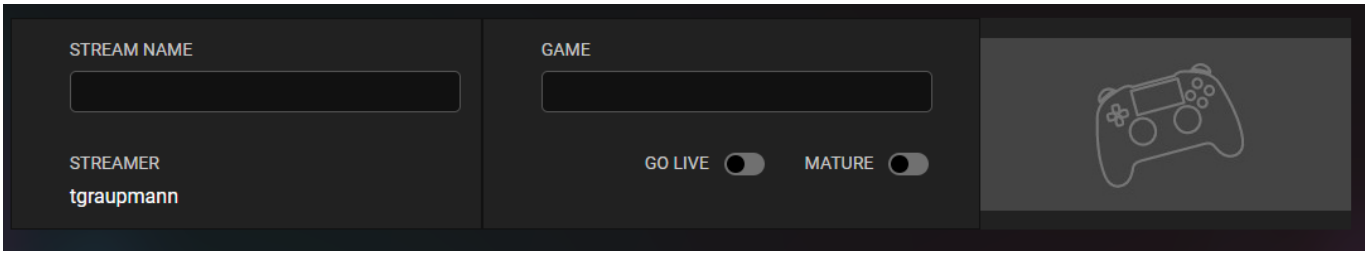
Stream Source

Create a `Stream Source` to broadcast `Chroma RGB` events.



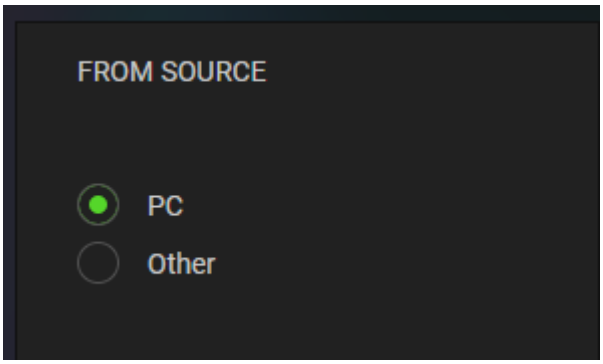
Name Your Stream Source

Enter the name of your stream source and enter name of the game to generate the stream thumbnail. Make sure `GO LIVE` is turned on. The `Mature` toggle is optional and can be filtered.



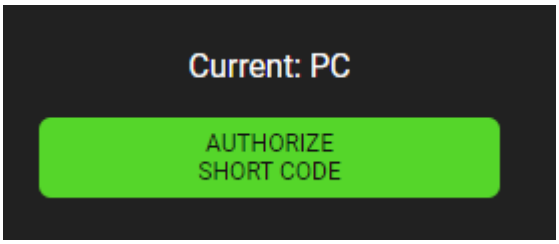
Source

PC games set the source to `PC`. `Cloud Gaming`, `Console`, and `Mobile` games set the `Source` to `Other`.



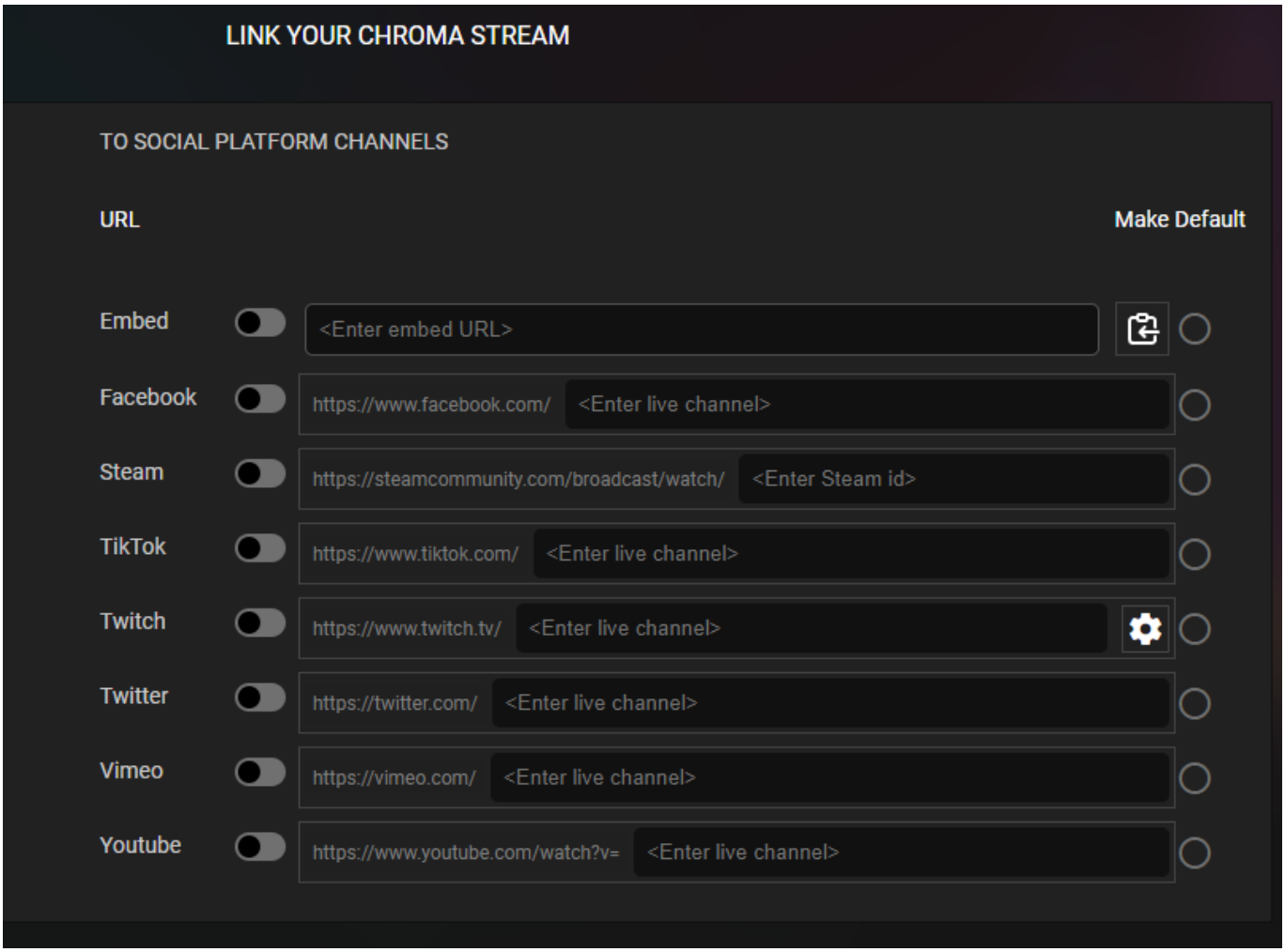
Authorize

Authorizing a shortcode is used to allow the broadcast to be toggled on sites outside the `Razer Streaming Portal` or by games using the API to authorize a `ShortCode`. Once the `Shortcode` is authorized, the `Stream Id` and `Stream Key` can be obtained before releasing the `ShortCode`.



Link Stream Source

The `Stream Source` can be linked to several social platforms. Enable the toggle for social platforms to appear on your `Stream Source` tile. Pick a `default` social platform for yuour `Stream Source` tile.

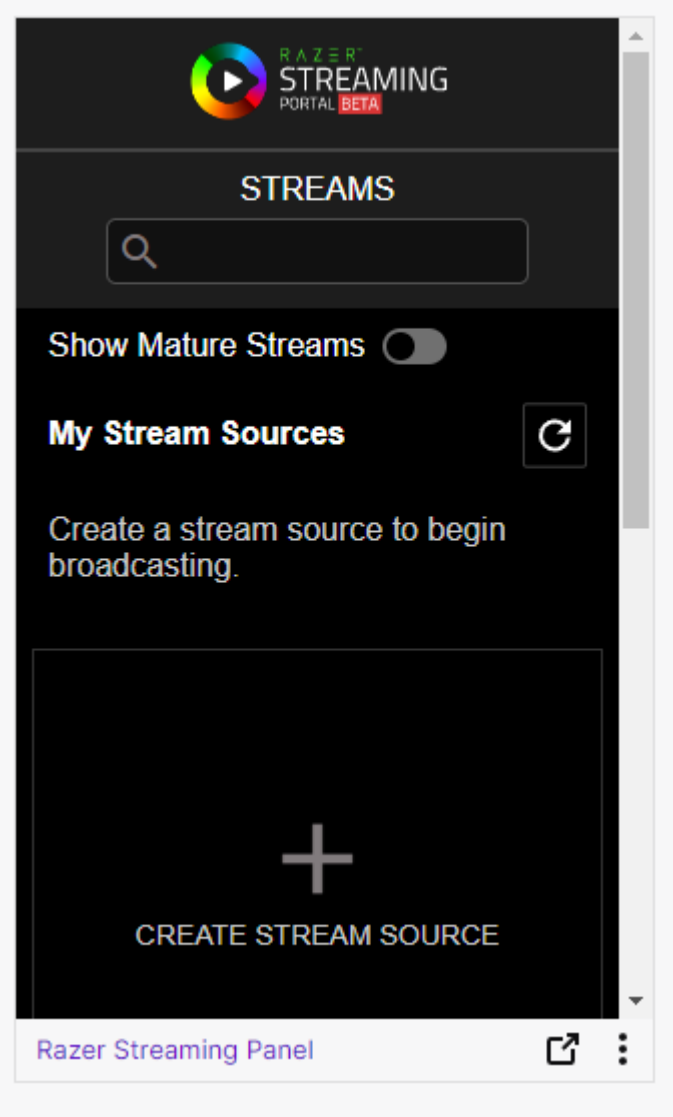


Embed

`Chroma RGB` streaming can be an `Embed` on a 3rd party webpage. Click the `Embed` clipboard icon to add the HTML5 iframe snippet to the clipboard which can be pasted into a 3rd party webpage. The `Embed` URL must match the URL of the 3rd party webpage address.

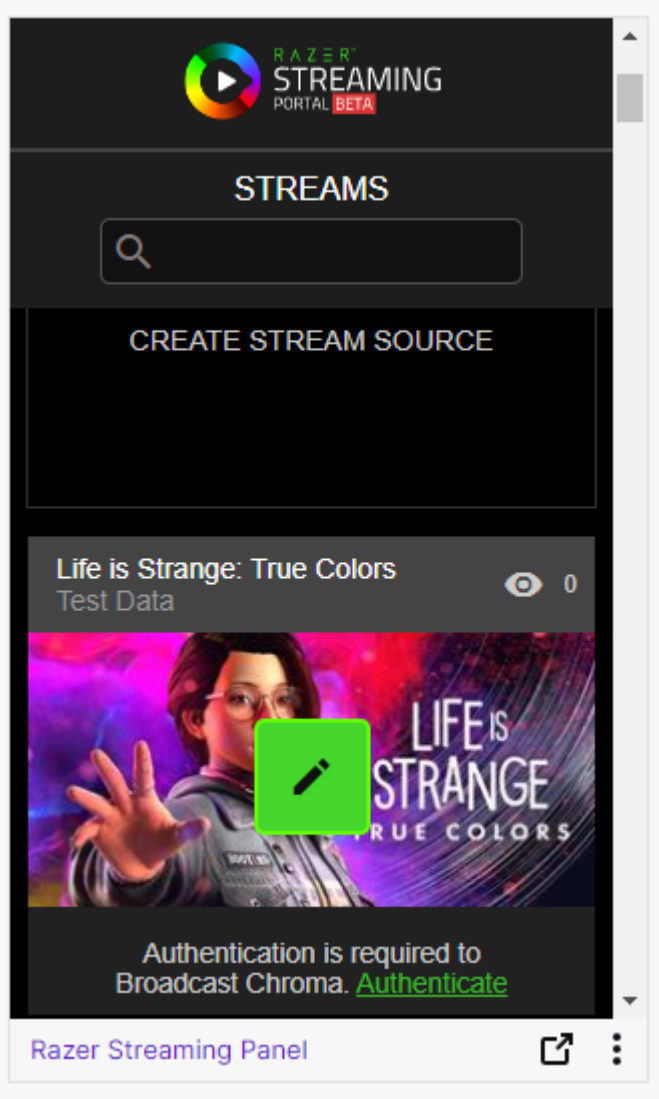
Twitch

Add the Razer Streaming Panel Twitch Extension to stream Chroam RGB on Twitch. There's a easily link by clicking the Twitch gear icon.

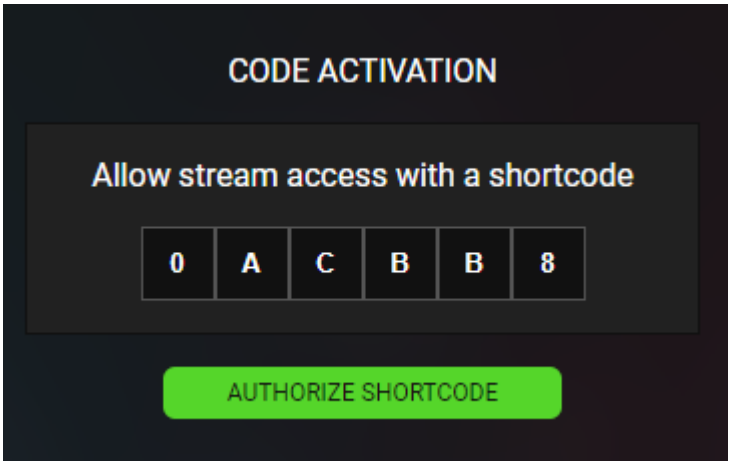


Twitch Panel

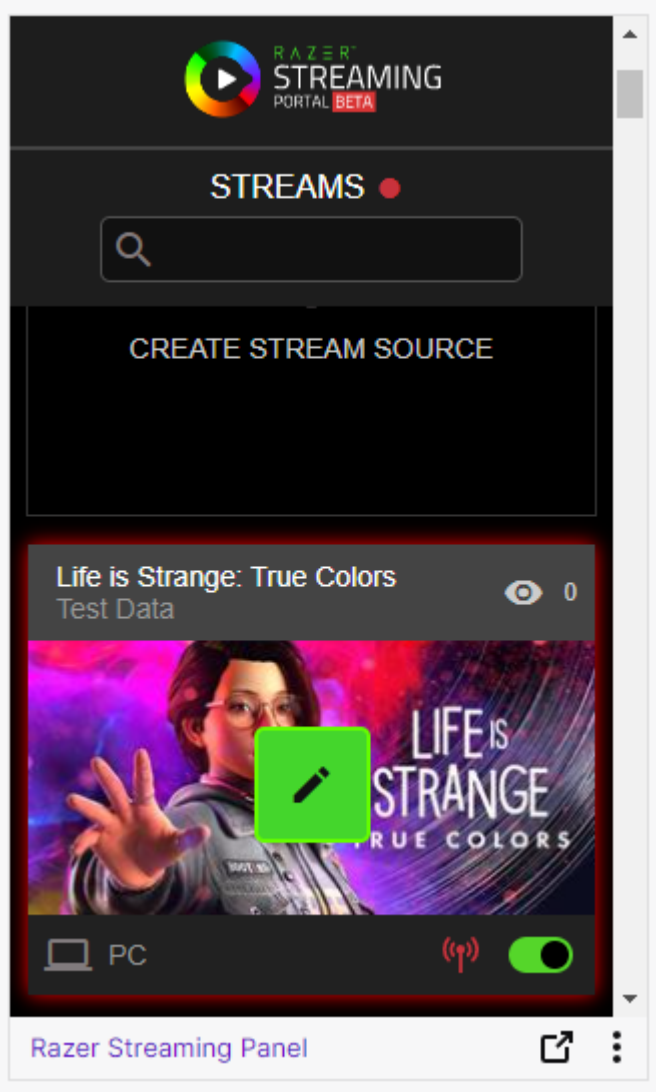
In order to broadcast via the Twitch panel, use the `Authenticate` link to authorize the automatic shortcode.



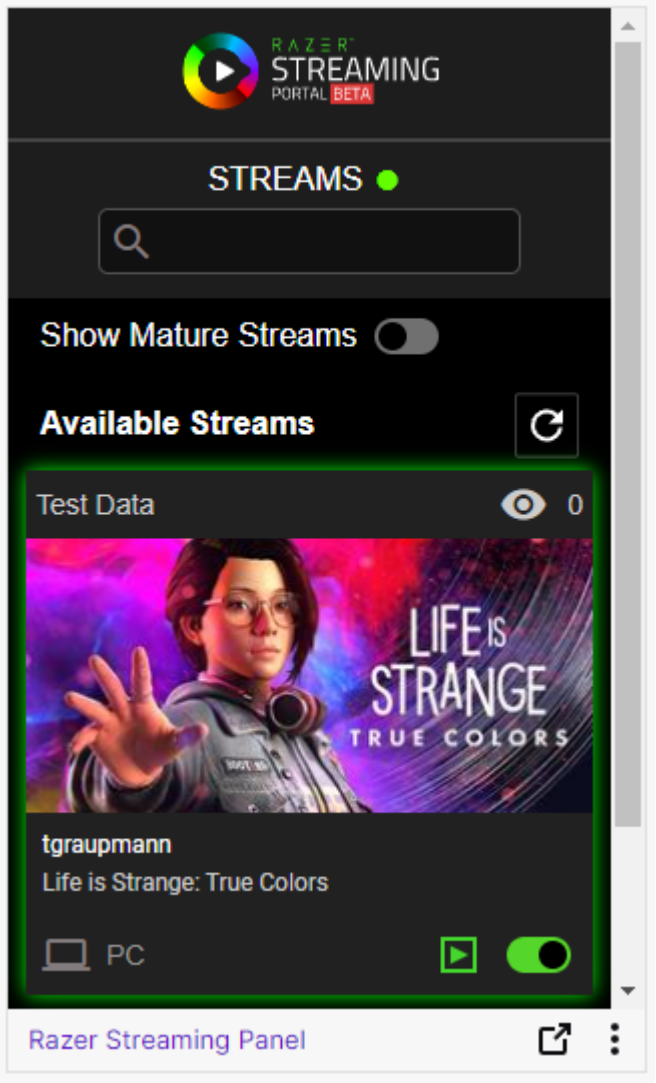
The shortcode will appear in the code activation popup and then click the `AUTHORIZE SHORTCODE` button. After the `Stream Source` has been authorized, the window can be closed.



If the `Stream Source` is associated with the logged in `Razer ID`, authorization will give access to Twitch to allow the broadcast toggle to be used from the Twitch panel.



Viewers of a live Twitch channel will automatically tune into the `Chroma RGB` from the broadcast. The streamer and viewers' hardware will be in sync.

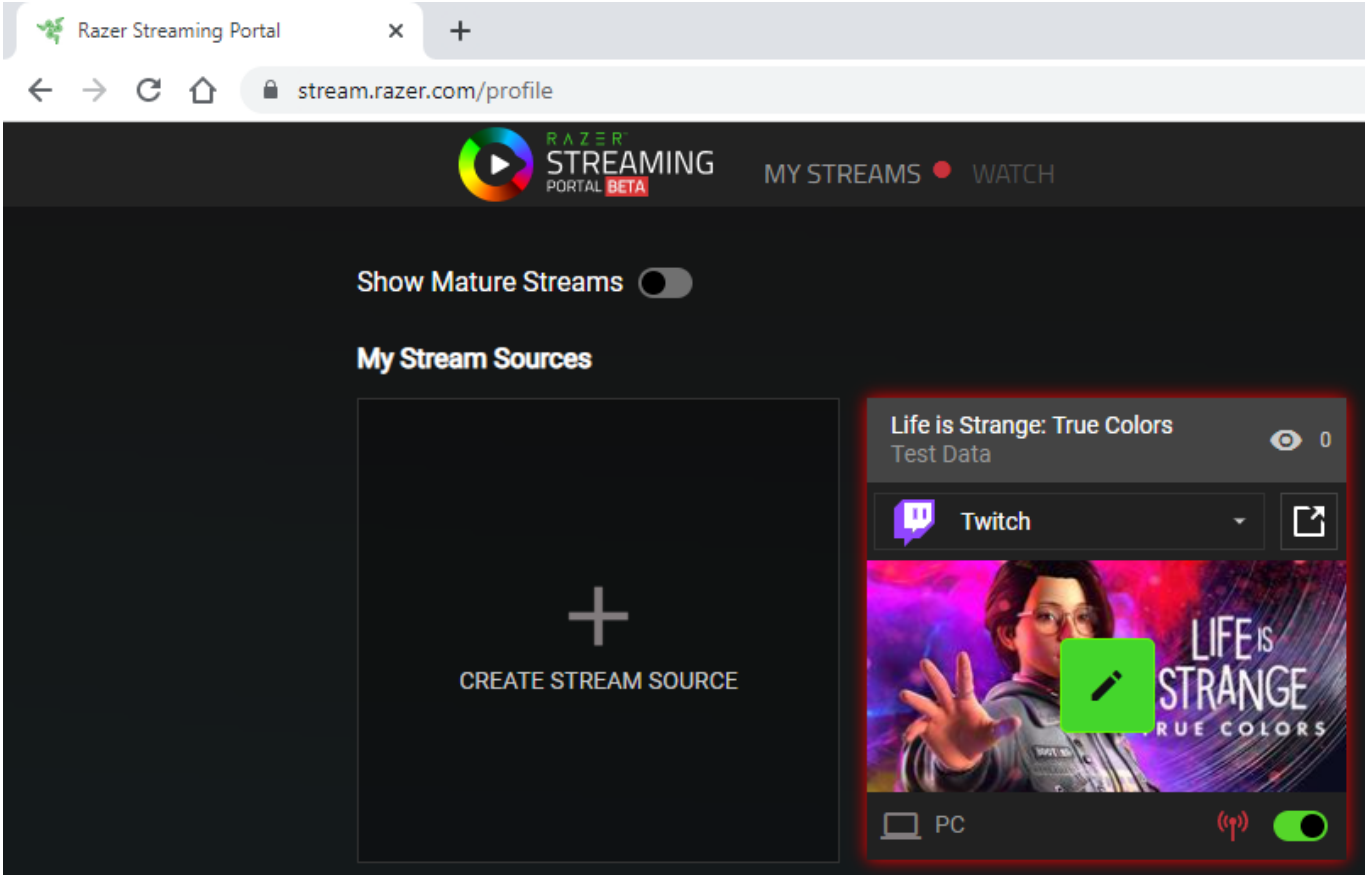


Save

Save the `Stream Source` settings when finished. You can rename your `Stream Source` and save the changes. You can reuse the same `Stream Source` for multiple games by changing the game title / thumbnail. You can create multiple `Stream Sources` for mulitple games. You can always delete a `Stream Source` to keep things tidy. If you've accidentally streamed your `Stream Key` to viewers, you can always delete the `Stream Source` and create a new `Stream Source` to generate a new `Stream Id` and `Stream Key`.

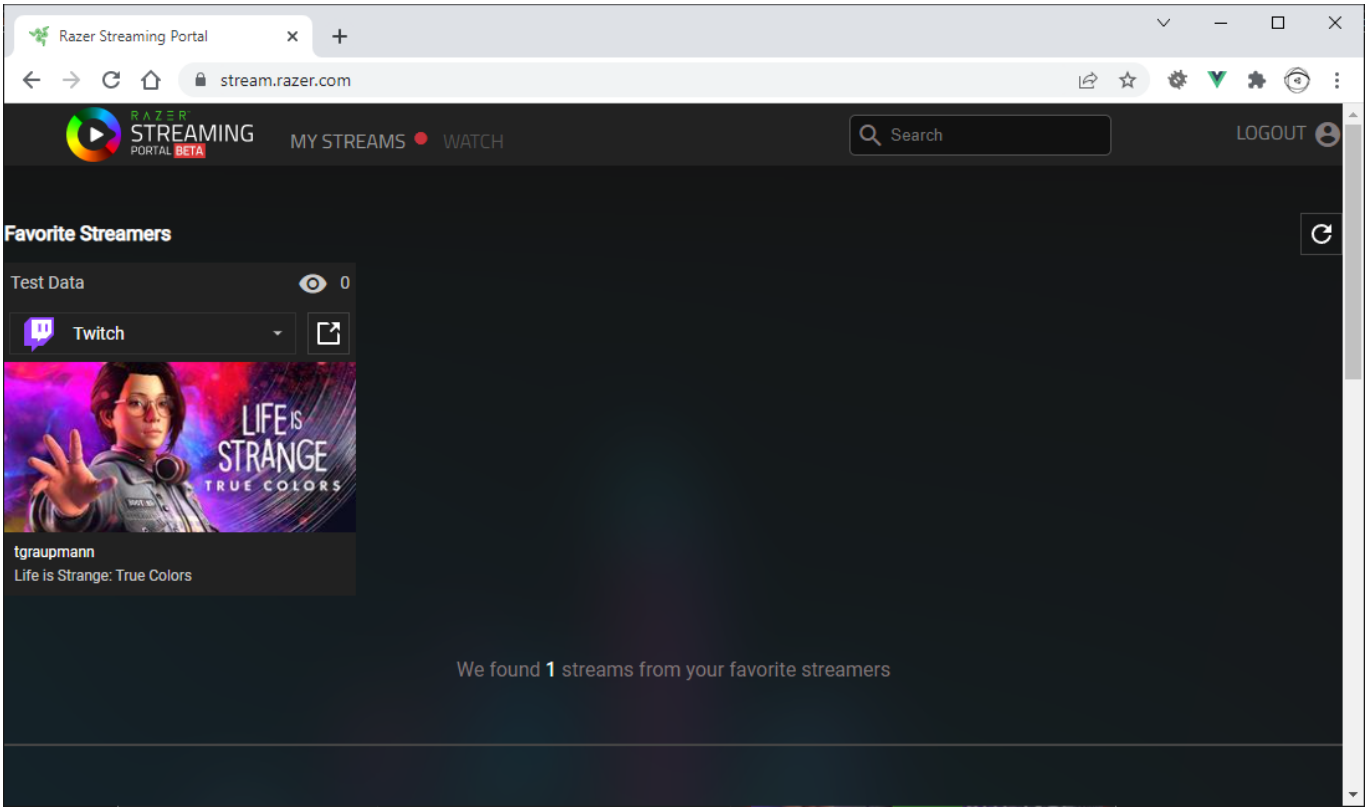
Broadcast

Once a `Stream Source` has been created, the toggle can be used to `Broadcast` to the social platforms that have been linked. Browse the Chroma Workshop to find games that have `Chroma RGB` integration. The broadcast toggle for `Chroma RGB` can be used before launching a game or after, either will work.



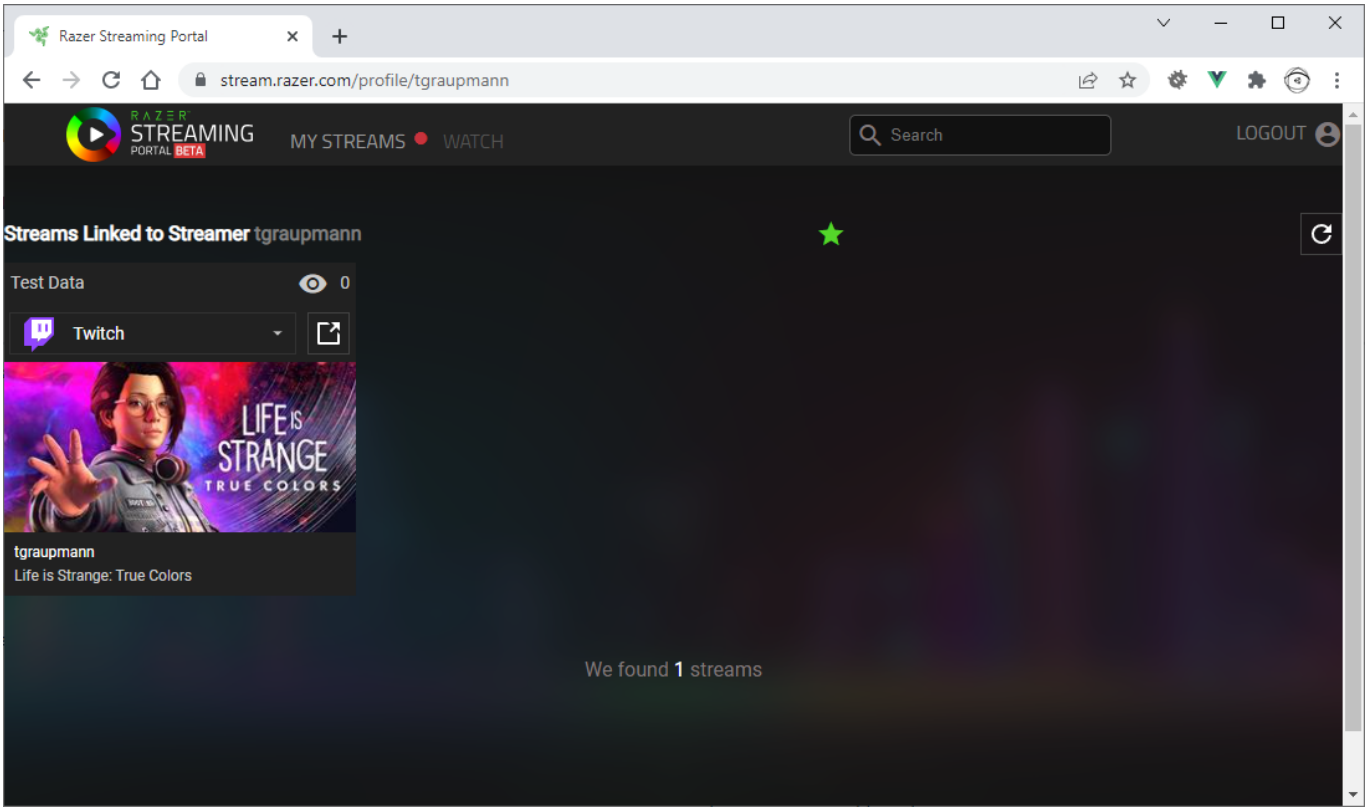
Streams

When the `Chroma RGB` broadcast is active and game `Chroma RGB` events are detected, the `Stream Source` tile will appear on the main page of the portal. The `Stream Source` can appear in the `Favorite Streamers` section if the associated streamer has been favorited. The `Stream Source` can appear in the `Favorite Games` section if the associated game has been favorited. The `Stream Source` appears in the `Available Stream` section sorted by view count and can be found through pagination.



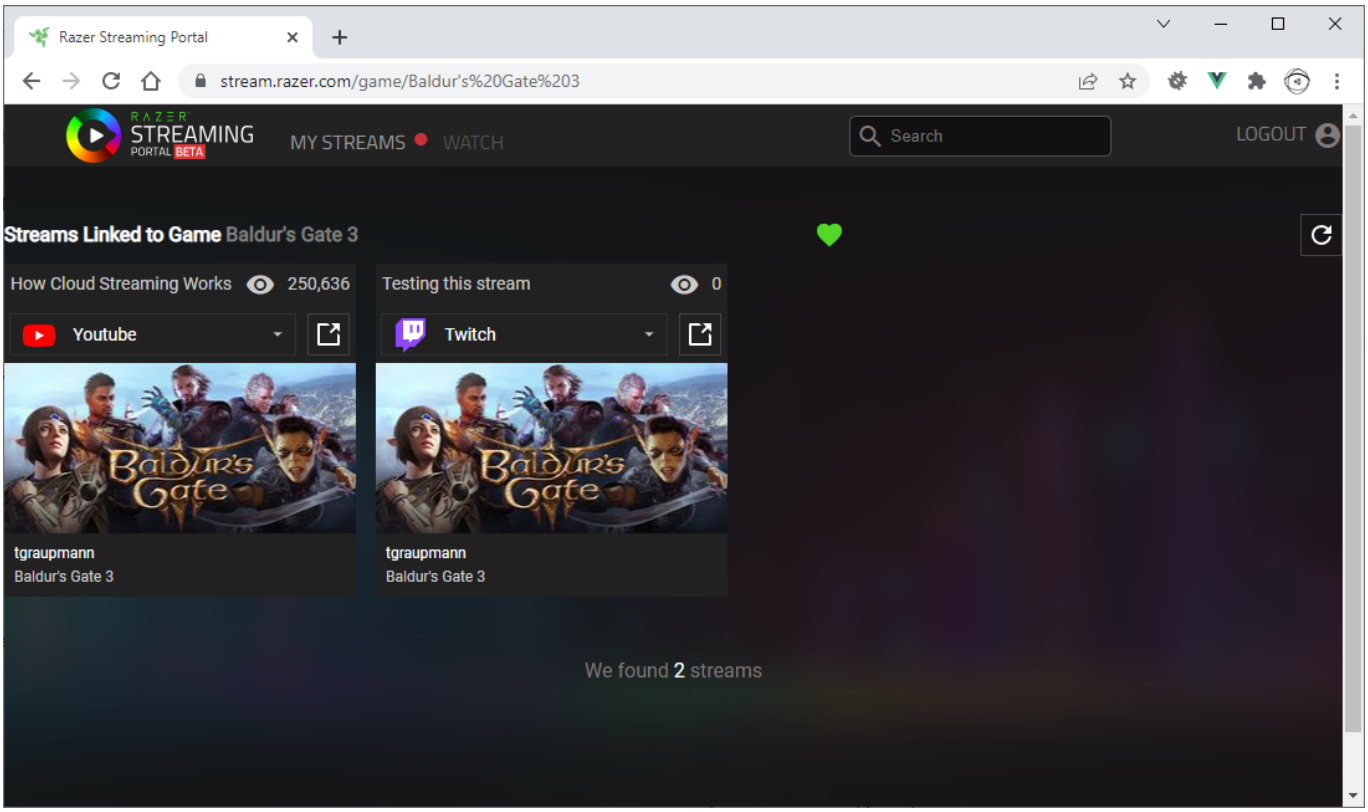
Favorite Streamers

Clicking the streamer link on the `Stream Source` navigates to a page where favorites can be toggled via the star icon.



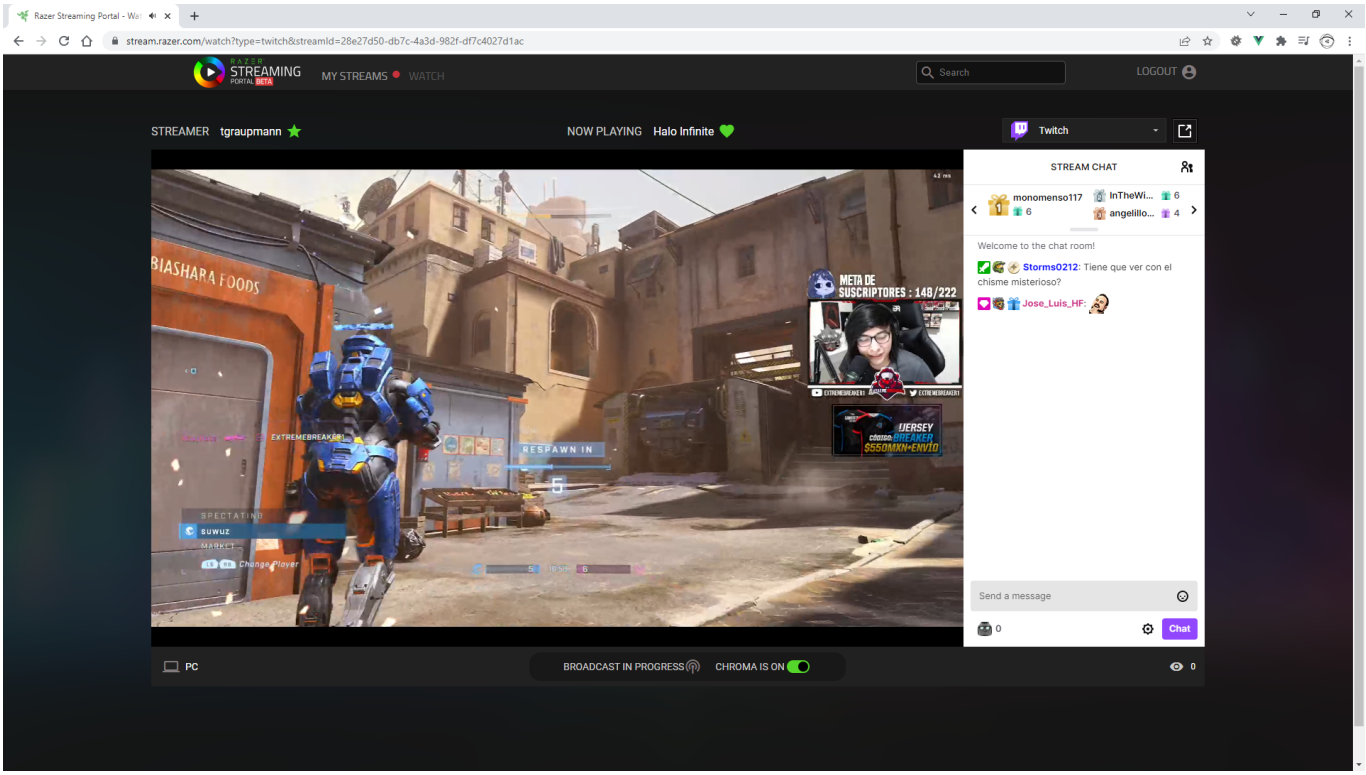
Favorite Games

Clicking the game link on the `Stream Source` navigates to a page where favorites can be toggled via the heart icon.



Watch

Clicking a `Stream Source` tile opens the watch page. The watch page streams `Chroma RGB` including audio and video with live chat (if available) when linked to the `Stream Source`. `Chroma` is initialized when the page receives focus, and uninitializes `Chroma` when the page loses focus. The watch page also has a drop down to switch between social channels that are linked with the stream.



Streaming Logic Flow

By default, any game that implements `Chroma RGB` on PC is compatible with streaming that can be controlled from the Razer Streaming Portal. Games are also able to control streaming via the API. Using the streaming API is most useful when the instance of the game is running within `Cloud Gaming`. Streaming can either broadcast or watch a `Chroma RGB` stream. The API supports broadcasting or watching but not both at the same time. It's recommended that games have a `Chroma RGB` toggle in the game settings that is on by default. For streaming `Chroma RGB`, games should be able to display a `shortcode` for stream authorization and a `broadcast toggle`.

Step 1. At any time, the game can use `StreamGetStatus` to get the streaming current status to display to the user.

Step 2. To prepare for broadcasting, the streamer creates a `Stream Source` on the profile page.

Step 3. The game invokes `StreamGetAuthShortcode` to display a six digit alpha-numeric `shortcode` to the streamer. The `shortcode` will expire within 5 minutes.

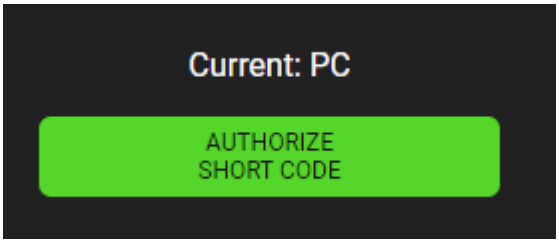
On PC, the only streaming platform is `"PC"`.

Cloud gaming platforms are specified below.

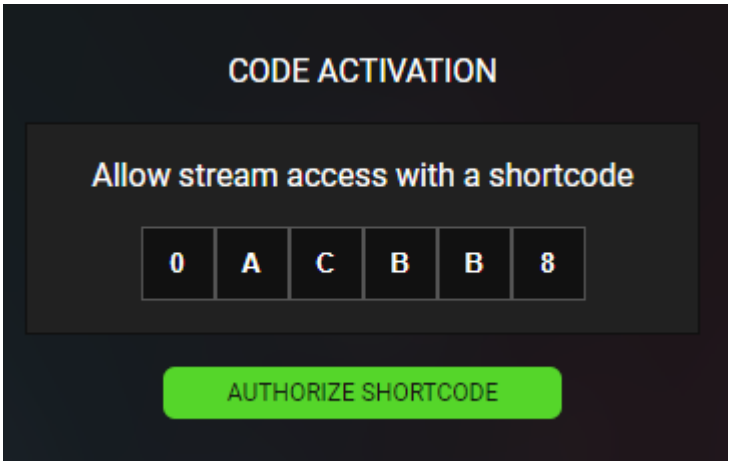
```
"GEFORCE_NOW"
"LUNA"
"STADIA"
"GAME_PASS"
```

Step 4. The game invokes `StreamGetId` and `StreamGetKey` on a 3-second interval while waiting on user authorization.

Step 5. The streamer can edit any `Stream Source` on the profile page and click `AUTHORIZE SHORT CODE`.



Step 6. The streamer enters the shortcode and clicks `AUTHORIZE SHORT CODE`.



Step 7. At this point `StreamGetId` and `StreamGetKey` return valid data to the game. The `Stream Id` and `Stream Key` should be saved in the user settings for feature streaming. To complete the authorization, invoke `StreamReleaseShortcode` which will prevent any other users from obtaining the auth details corresponding to the shortcode.

Step 8. The game can now invoke `StreamBroadcast` to begin broadcasting `Chroma RGB`. Both the `Stream Id` and `Stream Key` are needed to broadcast. `StreamBroadcastEnd` will end the broadcast.

Step 9. The game can invoke `StreamWatch` to watch the `Chroma RGB` stream for a given `Stream Id`. Watching a stream will end by invoking `StreamWatchEnd`.

File Format

Version: (int)

EChromaSDKDeviceTypeEnum: (byte)

```
enum EChromaSDKDeviceTypeEnum
{
    DE_1D = 0,
    DE_2D,
};
```

- Depending on the `EChromaSDKDeviceTypeEnum`, the file will use either the `1D` or `2D` file format.

1D File Format

EChromaSDKDevice1DEnum: (byte)

```
enum EChromaSDKDevice1DEnum
{
    DE_ChromaLink = 0,
    DE_Headset,
    DE_Mousepad,
};
```

Frame Count: (unsigned int)

Frames: (FChromaSDKColorFrame1D[])

```
struct FChromaSDKColorFrame1D
{
    float Duration;
    std::vector<COLORREF> Colors;
};
```

Duration: (float)

Color Array: (int[])

2D File Format

EChromaSDKDevice2DEnum: (byte)

```
enum EChromaSDKDevice2DEnum
{
    DE_Keyboard = 0,
    DE_Keypad,
    DE_Mouse,
    DE_KeyboardExtended,
};
```

Frame Count: (unsigned int)

Frames: (FChromaSDKColorFrame2D[])

```
struct FChromaSDKColors
{
    std::vector<COLORREF> Colors;
};
```

```
struct FChromaSDKColorFrame2D
{
    float Duration;
    std::vector<FChromaSDKColors> Colors;
};
```

Duration: (float)

Color Array: (int[][])

Extras

PluginPlayComposite

`PluginPlayComposite` automatically handles initializing the `ChromaSDK`. The named animation files for the `.chroma` set will be automatically opened. The set of animations will play with looping `on` or `off`.

```
EXPORT_API void PluginPlayComposite(const char* name, bool loop);

// Given "Random" this will invoke:
// PluginPlayAnimationName("Random_ChromaLink.chroma", loop);
// PluginPlayAnimationName("Random_Headset.chroma", loop);
// PluginPlayAnimationName("Random_Keyboard.chroma", loop);
// PluginPlayAnimationName("Random_Keypad.chroma", loop);
// PluginPlayAnimationName("Random_Mouse.chroma", loop);
// PluginPlayAnimationName("Random_Mousepad.chroma", loop);
```

PluginStopComposite

`PluginStopComposite` automatically handles initializing the `ChromaSDK`. The named animation files for the `.chroma` set will be automatically opened. The set of animations will be stopped if playing.

```
EXPORT_API void PluginStopComposite(const char* name);

// Given "Random" this will invoke:
// PluginStopAnimationName("Random_ChromaLink.chroma");
// PluginStopAnimationName("Random_Headset.chroma");
// PluginStopAnimationName("Random_Keyboard.chroma");
// PluginStopAnimationName("Random_Keypad.chroma");
```

```
// PluginStopAnimationName("Random_Mouse.chroma");
// PluginStopAnimationName("Random_Mousepad.chroma");
```

PluginCloseComposite

`PluginCloseComposite` closes a set of animations so they can be reloaded from disk. The set of animations will be stopped if playing.

```
EXPORT_API void PluginCloseComposite(const char* name);

// Given "Random" this will invoke:
// PluginCloseAnimationName("Random_ChromaLink.chroma");
// PluginCloseAnimationName("Random_Headset.chroma");
// PluginCloseAnimationName("Random_Keyboard.chroma");
// PluginCloseAnimationName("Random_Keypad.chroma");
// PluginCloseAnimationName("Random_Mouse.chroma");
// PluginCloseAnimationName("Random_Mousepad.chroma");
```

EChromaSDKDeviceTypeEnum

The supported device types are `1D` and `2D`.

```
enum EChromaSDKDeviceTypeEnum
{
    DE_1D = 0,
    DE_2D,
};
```

EChromaSDKDevice1DEnum

`1D` devices are `ChromaLink`, `Headset`, and `Mousepad`.

```
enum EChromaSDKDevice1DEnum
{
    DE_ChromaLink = 0,
    DE_Headset,
    DE_Mousepad,
};
```

EChromaSDKDevice2DEnum

`2D` devices are `Keyboard`, `Keypad`, and `Mouse`.

```
enum EChromaSDKDevice2DEnum
{
```

```
    DE_Keyboard = 0,
    DE_Keypad,
    DE_Mouse,
};
```

(Auto-documentation needs sample snippet section)

PluginSetKeysColorName

Set an array of animation keys to a static color for the given frame.

```
EXPORT_API void PluginSetKeysColorName(const char* path, int frameId, const int*
rzkeys, int keyCount, int color);
```

Usage:

```
const char* animationName = "Blank_Keyboard.chroma";
int frameCount = _gMethodGetFrameCountName(animationName);

int wasdKeys[4] =
{
  (int)Keyboard::RZKEY::RZKEY_W,
  (int)Keyboard::RZKEY::RZKEY_A,
  (int)Keyboard::RZKEY::RZKEY_S,
  (int)Keyboard::RZKEY::RZKEY_D,
};
for (int i = 0; i < frameCount; ++i)
{
  _gMethodSetKeysColorName(animationName, i, wasdKeys, size(wasdKeys), 0xFF);
}
_gMethodPlayAnimationName(animationName, false);
```

PluginSetKeysNonZeroColorName

Set an array of animation keys to a static color for the given frame if the existing color is not already black.

```
EXPORT_API void PluginSetKeysNonZeroColorName(const char* path, int frameId,
  const int* rzkeys, int keyCount, int color);
```

Usage:

```
const char* animationName = "Blank_Keyboard.chroma";
int frameCount = _gMethodGetFrameCountName(animationName);

int wasdKeys[4] =
```

```
{
  (int)Keyboard::RZKEY::RZKEY_W,
  (int)Keyboard::RZKEY::RZKEY_A,
  (int)Keyboard::RZKEY::RZKEY_S,
  (int)Keyboard::RZKEY::RZKEY_D,
};
for (int i = 0; i < frameCount; ++i)
{
  _gMethodSetKeysNonZeroColorName(animationName, i, wasdKeys, size(wasdKeys),
0xFF);
}
_gMethodPlayAnimationName(animationName, false);
```

Full API

- Note: See the Chroma Animation Guide for visual examples of the API methods.

The API has various methods with the `D` suffix where `double` return-type/parameters were used. This is to support engines like `GameMaker` which have a limited number of data-types.

(Start of automation)

Methods:

- PluginAddColor
- PluginAddFrame
- PluginAddNonZeroAllKeys
- PluginAddNonZeroAllKeysAllFrames
- PluginAddNonZeroAllKeysAllFramesName
- PluginAddNonZeroAllKeysAllFramesNameD
- PluginAddNonZeroAllKeysAllFramesOffset
- PluginAddNonZeroAllKeysAllFramesOffsetName
- PluginAddNonZeroAllKeysAllFramesOffsetNameD
- PluginAddNonZeroAllKeysName
- PluginAddNonZeroAllKeysOffset
- PluginAddNonZeroAllKeysOffsetName
- PluginAddNonZeroAllKeysOffsetNameD
- PluginAddNonZeroTargetAllKeysAllFrames
- PluginAddNonZeroTargetAllKeysAllFramesName
- PluginAddNonZeroTargetAllKeysAllFramesNameD
- PluginAddNonZeroTargetAllKeysAllFramesOffset
- PluginAddNonZeroTargetAllKeysAllFramesOffsetName
- PluginAddNonZeroTargetAllKeysAllFramesOffsetNameD
- PluginAddNonZeroTargetAllKeysOffset
- PluginAddNonZeroTargetAllKeysOffsetName
- PluginAddNonZeroTargetAllKeysOffsetNameD
- PluginAppendAllFrames
- PluginAppendAllFramesName
- PluginAppendAllFramesNameD

- PluginClearAll
- PluginClearAnimationType
- PluginCloseAll
- PluginCloseAnimation
- PluginCloseAnimationD
- PluginCloseAnimationName
- PluginCloseAnimationNameD
- PluginCloseComposite
- PluginCloseCompositeD
- PluginCopyAllKeys
- PluginCopyAllKeysName
- PluginCopyAnimation
- PluginCopyAnimationName
- PluginCopyAnimationNameD
- PluginCopyBlueChannelAllFrames
- PluginCopyBlueChannelAllFramesName
- PluginCopyBlueChannelAllFramesNameD
- PluginCopyGreenChannelAllFrames
- PluginCopyGreenChannelAllFramesName
- PluginCopyGreenChannelAllFramesNameD
- PluginCopyKeyColor
- PluginCopyKeyColorAllFrames
- PluginCopyKeyColorAllFramesName
- PluginCopyKeyColorAllFramesNameD
- PluginCopyKeyColorAllFramesOffset
- PluginCopyKeyColorAllFramesOffsetName
- PluginCopyKeyColorAllFramesOffsetNameD
- PluginCopyKeyColorName
- PluginCopyKeyColorNameD
- PluginCopyKeysColor
- PluginCopyKeysColorAllFrames
- PluginCopyKeysColorAllFramesName
- PluginCopyKeysColorName
- PluginCopyKeysColorOffset
- PluginCopyKeysColorOffsetName
- PluginCopyNonZeroAllKeys
- PluginCopyNonZeroAllKeysAllFrames
- PluginCopyNonZeroAllKeysAllFramesName
- PluginCopyNonZeroAllKeysAllFramesNameD
- PluginCopyNonZeroAllKeysAllFramesOffset
- PluginCopyNonZeroAllKeysAllFramesOffsetName
- PluginCopyNonZeroAllKeysAllFramesOffsetNameD
- PluginCopyNonZeroAllKeysName
- PluginCopyNonZeroAllKeysNameD
- PluginCopyNonZeroAllKeysOffset
- PluginCopyNonZeroAllKeysOffsetName

- PluginCopyNonZeroAllKeysOffsetNameD
- PluginCopyNonZeroKeyColor
- PluginCopyNonZeroKeyColorName
- PluginCopyNonZeroKeyColorNameD
- PluginCopyNonZeroTargetAllKeys
- PluginCopyNonZeroTargetAllKeysAllFrames
- PluginCopyNonZeroTargetAllKeysAllFramesName
- PluginCopyNonZeroTargetAllKeysAllFramesNameD
- PluginCopyNonZeroTargetAllKeysAllFramesOffset
- PluginCopyNonZeroTargetAllKeysAllFramesOffsetName
- PluginCopyNonZeroTargetAllKeysAllFramesOffsetNameD
- PluginCopyNonZeroTargetAllKeysName
- PluginCopyNonZeroTargetAllKeysNameD
- PluginCopyNonZeroTargetAllKeysOffset
- PluginCopyNonZeroTargetAllKeysOffsetName
- PluginCopyNonZeroTargetAllKeysOffsetNameD
- PluginCopyNonZeroTargetZeroAllKeysAllFrames
- PluginCopyNonZeroTargetZeroAllKeysAllFramesName
- PluginCopyNonZeroTargetZeroAllKeysAllFramesNameD
- PluginCopyRedChannelAllFrames
- PluginCopyRedChannelAllFramesName
- PluginCopyRedChannelAllFramesNameD
- PluginCopyZeroAllKeys
- PluginCopyZeroAllKeysAllFrames
- PluginCopyZeroAllKeysAllFramesName
- PluginCopyZeroAllKeysAllFramesNameD
- PluginCopyZeroAllKeysAllFramesOffset
- PluginCopyZeroAllKeysAllFramesOffsetName
- PluginCopyZeroAllKeysAllFramesOffsetNameD
- PluginCopyZeroAllKeysName
- PluginCopyZeroAllKeysOffset
- PluginCopyZeroAllKeysOffsetName
- PluginCopyZeroKeyColor
- PluginCopyZeroKeyColorName
- PluginCopyZeroKeyColorNameD
- PluginCopyZeroTargetAllKeys
- PluginCopyZeroTargetAllKeysAllFrames
- PluginCopyZeroTargetAllKeysAllFramesName
- PluginCopyZeroTargetAllKeysAllFramesNameD
- PluginCopyZeroTargetAllKeysName
- PluginCoreCreateChromaLinkEffect
- PluginCoreCreateEffect
- PluginCoreCreateHeadsetEffect
- PluginCoreCreateKeyboardEffect
- PluginCoreCreateKeypadEffect
- PluginCoreCreateMouseEffect

- PluginCoreCreateMousepadEffect
- PluginCoreDeleteEffect
- PluginCoreInit
- PluginCoreInitSDK
- PluginCoreIsActive
- PluginCoreIsConnected
- PluginCoreQueryDevice
- PluginCoreSetEffect
- PluginCoreSetEventName
- PluginCoreStreamBroadcast
- PluginCoreStreamBroadcastEnd
- PluginCoreStreamGetAuthShortcode
- PluginCoreStreamGetFocus
- PluginCoreStreamGetId
- PluginCoreStreamGetKey
- PluginCoreStreamGetStatus
- PluginCoreStreamGetStatusString
- PluginCoreStreamReleaseShortcode
- PluginCoreStreamSetFocus
- PluginCoreStreamSupportsStreaming
- PluginCoreStreamWatch
- PluginCoreStreamWatchEnd
- PluginCoreUnInit
- PluginCreateAnimation
- PluginCreateAnimationInMemory
- PluginCreateEffect
- PluginDeleteEffect
- PluginDuplicateFirstFrame
- PluginDuplicateFirstFrameName
- PluginDuplicateFirstFrameNameD
- PluginDuplicateFrames
- PluginDuplicateFramesName
- PluginDuplicateFramesNameD
- PluginDuplicateMirrorFrames
- PluginDuplicateMirrorFramesName
- PluginDuplicateMirrorFramesNameD
- PluginFadeEndFrames
- PluginFadeEndFramesName
- PluginFadeEndFramesNameD
- PluginFadeStartFrames
- PluginFadeStartFramesName
- PluginFadeStartFramesNameD
- PluginFillColor
- PluginFillColorAllFrames
- PluginFillColorAllFramesName
- PluginFillColorAllFramesNameD

- PluginFillColorAllFramesRGB
- PluginFillColorAllFramesRGBName
- PluginFillColorAllFramesRGBNameD
- PluginFillColorName
- PluginFillColorNameD
- PluginFillColorRGB
- PluginFillColorRGBName
- PluginFillColorRGBNameD
- PluginFillNonZeroColor
- PluginFillNonZeroColorAllFrames
- PluginFillNonZeroColorAllFramesName
- PluginFillNonZeroColorAllFramesNameD
- PluginFillNonZeroColorAllFramesRGB
- PluginFillNonZeroColorAllFramesRGBName
- PluginFillNonZeroColorAllFramesRGBNameD
- PluginFillNonZeroColorName
- PluginFillNonZeroColorNameD
- PluginFillNonZeroColorRGB
- PluginFillNonZeroColorRGBName
- PluginFillNonZeroColorRGBNameD
- PluginFillRandomColors
- PluginFillRandomColorsAllFrames
- PluginFillRandomColorsAllFramesName
- PluginFillRandomColorsAllFramesNameD
- PluginFillRandomColorsBlackAndWhite
- PluginFillRandomColorsBlackAndWhiteAllFrames
- PluginFillRandomColorsBlackAndWhiteAllFramesName
- PluginFillRandomColorsBlackAndWhiteAllFramesNameD
- PluginFillRandomColorsBlackAndWhiteName
- PluginFillRandomColorsBlackAndWhiteNameD
- PluginFillRandomColorsName
- PluginFillRandomColorsNameD
- PluginFillThresholdColors
- PluginFillThresholdColorsAllFrames
- PluginFillThresholdColorsAllFramesName
- PluginFillThresholdColorsAllFramesNameD
- PluginFillThresholdColorsAllFramesRGB
- PluginFillThresholdColorsAllFramesRGBName
- PluginFillThresholdColorsAllFramesRGBNameD
- PluginFillThresholdColorsMinMaxAllFramesRGB
- PluginFillThresholdColorsMinMaxAllFramesRGBName
- PluginFillThresholdColorsMinMaxAllFramesRGBNameD
- PluginFillThresholdColorsMinMaxRGB
- PluginFillThresholdColorsMinMaxRGBName
- PluginFillThresholdColorsMinMaxRGBNameD
- PluginFillThresholdColorsName

- PluginFillThresholdColorsNameD
- PluginFillThresholdColorsRGB
- PluginFillThresholdColorsRGBName
- PluginFillThresholdColorsRGBNameD
- PluginFillThresholdRGBColorsAllFramesRGB
- PluginFillThresholdRGBColorsAllFramesRGBName
- PluginFillThresholdRGBColorsAllFramesRGBNameD
- PluginFillThresholdRGBColorsRGB
- PluginFillThresholdRGBColorsRGBName
- PluginFillThresholdRGBColorsRGBNameD
- PluginFillZeroColor
- PluginFillZeroColorAllFrames
- PluginFillZeroColorAllFramesName
- PluginFillZeroColorAllFramesNameD
- PluginFillZeroColorAllFramesRGB
- PluginFillZeroColorAllFramesRGBName
- PluginFillZeroColorAllFramesRGBNameD
- PluginFillZeroColorName
- PluginFillZeroColorNameD
- PluginFillZeroColorRGB
- PluginFillZeroColorRGBName
- PluginFillZeroColorRGBNameD
- PluginGet1DColor
- PluginGet1DColorName
- PluginGet1DColorNameD
- PluginGet2DColor
- PluginGet2DColorName
- PluginGet2DColorNameD
- PluginGetAnimation
- PluginGetAnimationCount
- PluginGetAnimationD
- PluginGetAnimationId
- PluginGetAnimationName
- PluginGetCurrentFrame
- PluginGetCurrentFrameName
- PluginGetCurrentFrameNameD
- PluginGetDevice
- PluginGetDeviceName
- PluginGetDeviceNameD
- PluginGetDeviceType
- PluginGetDeviceTypeName
- PluginGetDeviceTypeNameD
- PluginGetFrame
- PluginGetFrameCount
- PluginGetFrameCountName
- PluginGetFrameCountNameD

- PluginGetFrameDuration
- PluginGetFrameDurationName
- PluginGetFrameName
- PluginGetKeyColor
- PluginGetKeyColorD
- PluginGetKeyColorName
- PluginGetLibraryLoadedState
- PluginGetLibraryLoadedStateD
- PluginGetMaxColumn
- PluginGetMaxColumnD
- PluginGetMaxLeds
- PluginGetMaxLedsD
- PluginGetMaxRow
- PluginGetMaxRowD
- PluginGetPlayingAnimationCount
- PluginGetPlayingAnimationId
- PluginGetRGB
- PluginGetRGBD
- PluginGetTotalDuration
- PluginGetTotalDurationName
- PluginHasAnimationLoop
- PluginHasAnimationLoopName
- PluginHasAnimationLoopNameD
- PluginInit
- PluginInitD
- PluginInitSDK
- PluginInsertDelay
- PluginInsertDelayName
- PluginInsertDelayNameD
- PluginInsertFrame
- PluginInsertFrameName
- PluginInsertFrameNameD
- PluginInvertColors
- PluginInvertColorsAllFrames
- PluginInvertColorsAllFramesName
- PluginInvertColorsAllFramesNameD
- PluginInvertColorsName
- PluginInvertColorsNameD
- PluginIsAnimationPaused
- PluginIsAnimationPausedName
- PluginIsAnimationPausedNameD
- PluginIsDialogOpen
- PluginIsDialogOpenD
- PluginIsInitialized
- PluginIsInitializedD
- PluginIsPlatformSupported

- PluginIsPlatformSupportedD
- PluginIsPlaying
- PluginIsPlayingD
- PluginIsPlayingName
- PluginIsPlayingNameD
- PluginIsPlayingType
- PluginIsPlayingTypeD
- PluginLerp
- PluginLerpColor
- PluginLoadAnimation
- PluginLoadAnimationD
- PluginLoadAnimationName
- PluginLoadComposite
- PluginMakeBlankFrames
- PluginMakeBlankFramesName
- PluginMakeBlankFramesNameD
- PluginMakeBlankFramesRandom
- PluginMakeBlankFramesRandomBlackAndWhite
- PluginMakeBlankFramesRandomBlackAndWhiteName
- PluginMakeBlankFramesRandomBlackAndWhiteNameD
- PluginMakeBlankFramesRandomName
- PluginMakeBlankFramesRandomNameD
- PluginMakeBlankFramesRGB
- PluginMakeBlankFramesRGBName
- PluginMakeBlankFramesRGBNameD
- PluginMirrorHorizontally
- PluginMirrorVertically
- PluginMultiplyColorLerpAllFrames
- PluginMultiplyColorLerpAllFramesName
- PluginMultiplyColorLerpAllFramesNameD
- PluginMultiplyIntensity
- PluginMultiplyIntensityAllFrames
- PluginMultiplyIntensityAllFramesName
- PluginMultiplyIntensityAllFramesNameD
- PluginMultiplyIntensityAllFramesRGB
- PluginMultiplyIntensityAllFramesRGBName
- PluginMultiplyIntensityAllFramesRGBNameD
- PluginMultiplyIntensityColor
- PluginMultiplyIntensityColorAllFrames
- PluginMultiplyIntensityColorAllFramesName
- PluginMultiplyIntensityColorAllFramesNameD
- PluginMultiplyIntensityColorName
- PluginMultiplyIntensityColorNameD
- PluginMultiplyIntensityName
- PluginMultiplyIntensityNameD
- PluginMultiplyIntensityRGB

- PluginMultiplyIntensityRGBName
- PluginMultiplyIntensityRGBNameD
- PluginMultiplyNonZeroTargetColorLerp
- PluginMultiplyNonZeroTargetColorLerpAllFrames
- PluginMultiplyNonZeroTargetColorLerpAllFramesName
- PluginMultiplyNonZeroTargetColorLerpAllFramesNameD
- PluginMultiplyNonZeroTargetColorLerpAllFramesRGB
- PluginMultiplyNonZeroTargetColorLerpAllFramesRGBName
- PluginMultiplyNonZeroTargetColorLerpAllFramesRGBNameD
- PluginMultiplyTargetColorLerp
- PluginMultiplyTargetColorLerpAllFrames
- PluginMultiplyTargetColorLerpAllFramesName
- PluginMultiplyTargetColorLerpAllFramesNameD
- PluginMultiplyTargetColorLerpAllFramesRGB
- PluginMultiplyTargetColorLerpAllFramesRGBName
- PluginMultiplyTargetColorLerpAllFramesRGBNameD
- PluginMultiplyTargetColorLerpName
- PluginOffsetColors
- PluginOffsetColorsAllFrames
- PluginOffsetColorsAllFramesName
- PluginOffsetColorsAllFramesNameD
- PluginOffsetColorsName
- PluginOffsetColorsNameD
- PluginOffsetNonZeroColors
- PluginOffsetNonZeroColorsAllFrames
- PluginOffsetNonZeroColorsAllFramesName
- PluginOffsetNonZeroColorsAllFramesNameD
- PluginOffsetNonZeroColorsName
- PluginOffsetNonZeroColorsNameD
- PluginOpenAnimation
- PluginOpenAnimationD
- PluginOpenAnimationFromMemory
- PluginOpenEditorDialog
- PluginOpenEditorDialogAndPlay
- PluginOpenEditorDialogAndPlayD
- PluginOpenEditorDialogD
- PluginOverrideFrameDuration
- PluginOverrideFrameDurationD
- PluginOverrideFrameDurationName
- PluginPauseAnimation
- PluginPauseAnimationName
- PluginPauseAnimationNameD
- PluginPlayAnimation
- PluginPlayAnimationD
- PluginPlayAnimationFrame
- PluginPlayAnimationFrameName

- PluginPlayAnimationFrameNameD
- PluginPlayAnimationLoop
- PluginPlayAnimationName
- PluginPlayAnimationNameD
- PluginPlayComposite
- PluginPlayCompositeD
- PluginPreviewFrame
- PluginPreviewFrameD
- PluginPreviewFrameName
- PluginReduceFrames
- PluginReduceFramesName
- PluginReduceFramesNameD
- PluginResetAnimation
- PluginResumeAnimation
- PluginResumeAnimationName
- PluginResumeAnimationNameD
- PluginReverse
- PluginReverseAllFrames
- PluginReverseAllFramesName
- PluginReverseAllFramesNameD
- PluginSaveAnimation
- PluginSaveAnimationName
- PluginSet1DColor
- PluginSet1DColorName
- PluginSet1DColorNameD
- PluginSet2DColor
- PluginSet2DColorName
- PluginSet2DColorNameD
- PluginSetChromaCustomColorAllFrames
- PluginSetChromaCustomColorAllFramesName
- PluginSetChromaCustomColorAllFramesNameD
- PluginSetChromaCustomFlag
- PluginSetChromaCustomFlagName
- PluginSetChromaCustomFlagNameD
- PluginSetCurrentFrame
- PluginSetCurrentFrameName
- PluginSetCurrentFrameNameD
- PluginSetCustomColorFlag2D
- PluginSetDevice
- PluginSetEffect
- PluginSetEffectCustom1D
- PluginSetEffectCustom2D
- PluginSetEffectKeyboardCustom2D
- PluginSetIdleAnimation
- PluginSetIdleAnimationName
- PluginSetKeyColor

- PluginSetKeyColorAllFrames
- PluginSetKeyColorAllFramesName
- PluginSetKeyColorAllFramesNameD
- PluginSetKeyColorAllFramesRGB
- PluginSetKeyColorAllFramesRGBName
- PluginSetKeyColorAllFramesRGBNameD
- PluginSetKeyColorName
- PluginSetKeyColorNameD
- PluginSetKeyColorRGB
- PluginSetKeyColorRGBName
- PluginSetKeyColorRGBNameD
- PluginSetKeyNonZeroColor
- PluginSetKeyNonZeroColorName
- PluginSetKeyNonZeroColorNameD
- PluginSetKeyNonZeroColorRGB
- PluginSetKeyNonZeroColorRGBName
- PluginSetKeyNonZeroColorRGBNameD
- PluginSetKeyRowColumnColorName
- PluginSetKeysColor
- PluginSetKeysColorAllFrames
- PluginSetKeysColorAllFramesName
- PluginSetKeysColorAllFramesRGB
- PluginSetKeysColorAllFramesRGBName
- PluginSetKeysColorName
- PluginSetKeysColorRGB
- PluginSetKeysColorRGBName
- PluginSetKeysNonZeroColor
- PluginSetKeysNonZeroColorAllFrames
- PluginSetKeysNonZeroColorAllFramesName
- PluginSetKeysNonZeroColorName
- PluginSetKeysNonZeroColorRGB
- PluginSetKeysNonZeroColorRGBName
- PluginSetKeysZeroColor
- PluginSetKeysZeroColorAllFrames
- PluginSetKeysZeroColorAllFramesName
- PluginSetKeysZeroColorAllFramesRGB
- PluginSetKeysZeroColorAllFramesRGBName
- PluginSetKeysZeroColorName
- PluginSetKeysZeroColorRGB
- PluginSetKeysZeroColorRGBName
- PluginSetKeyZeroColor
- PluginSetKeyZeroColorName
- PluginSetKeyZeroColorNameD
- PluginSetKeyZeroColorRGB
- PluginSetKeyZeroColorRGBName
- PluginSetKeyZeroColorRGBNameD

- PluginSetLogDelegate
- PluginSetStaticColor
- PluginSetStaticColorAll
- PluginStaticColor
- PluginStaticColorAll
- PluginStaticColorD
- PluginStopAll
- PluginStopAnimation
- PluginStopAnimationD
- PluginStopAnimationName
- PluginStopAnimationNameD
- PluginStopAnimationType
- PluginStopAnimationTypeD
- PluginStopComposite
- PluginStopCompositeD
- PluginSubtractColor
- PluginSubtractNonZeroAllKeys
- PluginSubtractNonZeroAllKeysAllFrames
- PluginSubtractNonZeroAllKeysAllFramesName
- PluginSubtractNonZeroAllKeysAllFramesNameD
- PluginSubtractNonZeroAllKeysAllFramesOffset
- PluginSubtractNonZeroAllKeysAllFramesOffsetName
- PluginSubtractNonZeroAllKeysAllFramesOffsetNameD
- PluginSubtractNonZeroAllKeysName
- PluginSubtractNonZeroAllKeysOffset
- PluginSubtractNonZeroAllKeysOffsetName
- PluginSubtractNonZeroAllKeysOffsetNameD
- PluginSubtractNonZeroTargetAllKeysAllFrames
- PluginSubtractNonZeroTargetAllKeysAllFramesName
- PluginSubtractNonZeroTargetAllKeysAllFramesNameD
- PluginSubtractNonZeroTargetAllKeysAllFramesOffset
- PluginSubtractNonZeroTargetAllKeysAllFramesOffsetName
- PluginSubtractNonZeroTargetAllKeysAllFramesOffsetNameD
- PluginSubtractNonZeroTargetAllKeysOffset
- PluginSubtractNonZeroTargetAllKeysOffsetName
- PluginSubtractNonZeroTargetAllKeysOffsetNameD
- PluginSubtractThresholdColorsMinMaxAllFramesRGB
- PluginSubtractThresholdColorsMinMaxAllFramesRGBName
- PluginSubtractThresholdColorsMinMaxAllFramesRGBNameD
- PluginSubtractThresholdColorsMinMaxRGB
- PluginSubtractThresholdColorsMinMaxRGBName
- PluginSubtractThresholdColorsMinMaxRGBNameD
- PluginTrimEndFrames
- PluginTrimEndFramesName
- PluginTrimEndFramesNameD
- PluginTrimFrame

- PluginTrimFrameName
- PluginTrimFrameNameD
- PluginTrimStartFrames
- PluginTrimStartFramesName
- PluginTrimStartFramesNameD
- PluginUninit
- PluginUninitD
- PluginUnloadAnimation
- PluginUnloadAnimationD
- PluginUnloadAnimationName
- PluginUnloadComposite
- PluginUnloadLibrarySDK
- PluginUnloadLibraryStreamingPlugin
- PluginUpdateFrame
- PluginUpdateFrameName
- PluginUseForwardChromaEvents
- PluginUseIdleAnimation
- PluginUseIdleAnimations
- PluginUsePreloading
- PluginUsePreloadingName

PluginAddColor

Return the sum of colors

```
// DLL Interface
EXPORT_API int PluginAddColor(
    const int color1, const int color2);

// Class Plugin
int result = ChromaAnimationAPI::AddColor(
    const int color1, const int color2);
```

PluginAddFrame

Adds a frame to the `Chroma` animation and sets the `duration` (in seconds). The `color` is expected to be an array of the dimensions for the `deviceType/device`. The `length` parameter is the size of the `color` array. For `EChromaSDKDevice1DEnum` the array size should be `MAX LEDS`. For `EChromaSDKDevice2DEnum` the array size should be `MAX ROW` times `MAX COLUMN`. Returns the animation id upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginAddFrame(
    int animationId, float duration, int* colors, int length);
```

```
// Class Plugin
int result = ChromaAnimationAPI::AddFrame(
    int animationId, float duration, int* colors, int length);
```

PluginAddNonZeroAllKeys

Add source color to target where color is not black for frame id, reference source and target by id.

```
// DLL Interface
EXPORT_API void PluginAddNonZeroAllKeys(
    int sourceAnimationId, int targetAnimationId, int frameId);

// Class Plugin
ChromaAnimationAPI::AddNonZeroAllKeys(
    int sourceAnimationId, int targetAnimationId, int frameId);
```

PluginAddNonZeroAllKeysAllFrames

Add source color to target where color is not black for all frames, reference source and target by id.

```
// DLL Interface
EXPORT_API void PluginAddNonZeroAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);

// Class Plugin
ChromaAnimationAPI::AddNonZeroAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);
```

PluginAddNonZeroAllKeysAllFramesName

Add source color to target where color is not black for all frames, reference source and target by name.

```
// DLL Interface
EXPORT_API void PluginAddNonZeroAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
ChromaAnimationAPI::AddNonZeroAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginAddNonZeroAllKeysAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginAddNonZeroAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
double result = ChromaAnimationAPI::AddNonZeroAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginAddNonZeroAllKeysAllFramesOffset

Add source color to target where color is not black for all frames starting at offset for the length of the source, reference source and target by id.

```
// DLL Interface
EXPORT_API void PluginAddNonZeroAllKeysAllFramesOffset(
    int sourceAnimationId, int targetAnimationId, int offset);

// Class Plugin
ChromaAnimationAPI::AddNonZeroAllKeysAllFramesOffset(
    int sourceAnimationId, int targetAnimationId, int offset);
```

PluginAddNonZeroAllKeysAllFramesOffsetName

Add source color to target where color is not black for all frames starting at offset for the length of the source, reference source and target by name.

```
// DLL Interface
EXPORT_API void PluginAddNonZeroAllKeysAllFramesOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int offset);

// Class Plugin
ChromaAnimationAPI::AddNonZeroAllKeysAllFramesOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int offset);
```

PluginAddNonZeroAllKeysAllFramesOffsetNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginAddNonZeroAllKeysAllFramesOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double offset);

// Class Plugin
```

```
double result = ChromaAnimationAPI::AddNonZeroAllKeysAllFramesOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double offset);
```

PluginAddNonZeroAllKeysName

Add source color to target where color is not black for frame id, reference source and target by name.

```
// DLL Interface
EXPORT_API void PluginAddNonZeroAllKeysName(
    const char* sourceAnimation, const char* targetAnimation, int frameId);

// Class Plugin
ChromaAnimationAPI::AddNonZeroAllKeysName(
    const char* sourceAnimation, const char* targetAnimation, int frameId);
```

PluginAddNonZeroAllKeysOffset

Add source color to target where color is not black for the source frame and target offset frame, reference source and target by id.

```
// DLL Interface
EXPORT_API void PluginAddNonZeroAllKeysOffset(
    int sourceAnimationId, int targetAnimationId, int frameId, int offset);

// Class Plugin
ChromaAnimationAPI::AddNonZeroAllKeysOffset(
    int sourceAnimationId, int targetAnimationId, int frameId, int offset);
```

PluginAddNonZeroAllKeysOffsetName

Add source color to target where color is not black for the source frame and target offset frame, reference source and target by name.

```
// DLL Interface
EXPORT_API void PluginAddNonZeroAllKeysOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
offset);

// Class Plugin
ChromaAnimationAPI::AddNonZeroAllKeysOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
offset);
```

PluginAddNonZeroAllKeysOffsetNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginAddNonZeroAllKeysOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double offset);

// Class Plugin
double result = ChromaAnimationAPI::AddNonZeroAllKeysOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double offset);
```

PluginAddNonZeroTargetAllKeysAllFrames

Add source color to target where the target color is not black for all frames, reference source and target by id.

```
// DLL Interface
EXPORT_API void PluginAddNonZeroTargetAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);

// Class Plugin
ChromaAnimationAPI::AddNonZeroTargetAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);
```

PluginAddNonZeroTargetAllKeysAllFramesName

Add source color to target where the target color is not black for all frames, reference source and target by name.

```
// DLL Interface
EXPORT_API void PluginAddNonZeroTargetAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
ChromaAnimationAPI::AddNonZeroTargetAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginAddNonZeroTargetAllKeysAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginAddNonZeroTargetAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
double result = ChromaAnimationAPI::AddNonZeroTargetAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginAddNonZeroTargetAllKeysAllFramesOffset

Add source color to target where the target color is not black for all frames starting at offset for the length of the source, reference source and target by id.

```
// DLL Interface
EXPORT_API void PluginAddNonZeroTargetAllKeysAllFramesOffset(
    int sourceAnimationId, int targetAnimationId, int offset);

// Class Plugin
ChromaAnimationAPI::AddNonZeroTargetAllKeysAllFramesOffset(
    int sourceAnimationId, int targetAnimationId, int offset);
```

PluginAddNonZeroTargetAllKeysAllFramesOffsetName

Add source color to target where the target color is not black for all frames starting at offset for the length of the source, reference source and target by name.

```
// DLL Interface
EXPORT_API void PluginAddNonZeroTargetAllKeysAllFramesOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int offset);

// Class Plugin
ChromaAnimationAPI::AddNonZeroTargetAllKeysAllFramesOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int offset);
```

PluginAddNonZeroTargetAllKeysAllFramesOffsetNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginAddNonZeroTargetAllKeysAllFramesOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double offset);

// Class Plugin
```

```
double result = ChromaAnimationAPI::AddNonZeroTargetAllKeysAllFramesOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double offset);
```

PluginAddNonZeroTargetAllKeysOffset

Add source color to target where target color is not blank from the source frame to the target offset frame, reference source and target by id.

```
// DLL Interface
EXPORT_API void PluginAddNonZeroTargetAllKeysOffset(
    int sourceAnimationId, int targetAnimationId, int frameId, int offset);

// Class Plugin
ChromaAnimationAPI::AddNonZeroTargetAllKeysOffset(
    int sourceAnimationId, int targetAnimationId, int frameId, int offset);
```

PluginAddNonZeroTargetAllKeysOffsetName

Add source color to target where target color is not blank from the source frame to the target offset frame, reference source and target by name.

```
// DLL Interface
EXPORT_API void PluginAddNonZeroTargetAllKeysOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
offset);

// Class Plugin
ChromaAnimationAPI::AddNonZeroTargetAllKeysOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
offset);
```

PluginAddNonZeroTargetAllKeysOffsetNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginAddNonZeroTargetAllKeysOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double offset);

// Class Plugin
double result = ChromaAnimationAPI::AddNonZeroTargetAllKeysOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double offset);
```

PluginAppendAllFrames

Append all source frames to the target animation, reference source and target by id.

```
// DLL Interface
EXPORT_API void PluginAppendAllFrames(
    int sourceAnimationId, int targetAnimationId);

// Class Plugin
ChromaAnimationAPI::AppendAllFrames(
    int sourceAnimationId, int targetAnimationId);
```

PluginAppendAllFramesName

Append all source frames to the target animation, reference source and target by name.

```
// DLL Interface
EXPORT_API void PluginAppendAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
ChromaAnimationAPI::AppendAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginAppendAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginAppendAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
double result = ChromaAnimationAPI::AppendAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginClearAll

`PluginClearAll` will issue a `CLEAR` effect for all devices.

```
// DLL Interface
EXPORT_API void PluginClearAll();
```

```
// Class Plugin
ChromaAnimationAPI::ClearAll();
```

PluginClearAnimationType

`PluginClearAnimationType` will issue a `CLEAR` effect for the given device.

```
// DLL Interface
EXPORT_API void PluginClearAnimationType(
    int deviceType, int device);

// Class Plugin
ChromaAnimationAPI::ClearAnimationType(
    int deviceType, int device);
```

PluginCloseAll

`PluginCloseAll` closes all open animations so they can be reloaded from disk. The set of animations will be stopped if playing.

```
// DLL Interface
EXPORT_API void PluginCloseAll();

// Class Plugin
ChromaAnimationAPI::CloseAll();
```

PluginCloseAnimation

Closes the `Chroma` animation to free up resources referenced by id. Returns the animation id upon success. Returns negative one upon failure. This might be used while authoring effects if there was a change necessitating re-opening the animation. The animation id can no longer be used once closed.

```
// DLL Interface
EXPORT_API int PluginCloseAnimation(int animationId);

// Class Plugin
int result = ChromaAnimationAPI::CloseAnimation(int animationId);
```

PluginCloseAnimationD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCloseAnimationD(double animationId);

// Class Plugin
double result = ChromaAnimationAPI::CloseAnimationD(double animationId);
```

PluginCloseAnimationName

Closes the `Chroma` animation referenced by name so that the animation can be reloaded from disk.

```
// DLL Interface
EXPORT_API void PluginCloseAnimationName(const char* path);

// Class Plugin
ChromaAnimationAPI::CloseAnimationName(const char* path);
```

PluginCloseAnimationNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCloseAnimationNameD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::CloseAnimationNameD(const char* path);
```

PluginCloseComposite

`PluginCloseComposite` closes a set of animations so they can be reloaded from disk. The set of animations will be stopped if playing.

```
// DLL Interface
EXPORT_API void PluginCloseComposite(const char* name);

// Class Plugin
ChromaAnimationAPI::CloseComposite(const char* name);
```

PluginCloseCompositeD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCloseCompositeD(const char* name);

// Class Plugin
double result = ChromaAnimationAPI::CloseCompositeD(const char* name);
```

PluginCopyAllKeys

Copy source animation to target animation for the given frame. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyAllKeys(
    int sourceAnimationId, int targetAnimationId, int frameId);

// Class Plugin
ChromaAnimationAPI::CopyAllKeys(
    int sourceAnimationId, int targetAnimationId, int frameId);
```

PluginCopyAllKeysName

Copy source animation to target animation for the given frame. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyAllKeysName(
    const char* sourceAnimation, const char* targetAnimation, int frameId);

// Class Plugin
ChromaAnimationAPI::CopyAllKeysName(
    const char* sourceAnimation, const char* targetAnimation, int frameId);
```

PluginCopyAnimation

Copy animation to named target animation in memory. If target animation exists, close first. Source is referenced by id.

```
// DLL Interface
EXPORT_API int PluginCopyAnimation(
    int sourceAnimationId, const char* targetAnimation);

// Class Plugin
int result = ChromaAnimationAPI::CopyAnimation(
    int sourceAnimationId, const char* targetAnimation);
```

PluginCopyAnimationName

Copy animation to named target animation in memory. If target animation exists, close first. Source is referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyAnimationName(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
ChromaAnimationAPI::CopyAnimationName(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginCopyAnimationNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyAnimationNameD(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
double result = ChromaAnimationAPI::CopyAnimationNameD(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginCopyBlueChannelAllFrames

Copy blue channel to other channels for all frames. Intensity range is 0.0 to 1.0. Reference the animation by id.

```
// DLL Interface
EXPORT_API void PluginCopyBlueChannelAllFrames(
    int animationId, float redIntensity, float greenIntensity);

// Class Plugin
ChromaAnimationAPI::CopyBlueChannelAllFrames(
    int animationId, float redIntensity, float greenIntensity);
```

PluginCopyBlueChannelAllFramesName

Copy blue channel to other channels for all frames. Intensity range is 0.0 to 1.0. Reference the animation by name.

```
// DLL Interface
EXPORT_API void PluginCopyBlueChannelAllFramesName(
    const char* path, float redIntensity, float greenIntensity);

// Class Plugin
ChromaAnimationAPI::CopyBlueChannelAllFramesName(
    const char* path, float redIntensity, float greenIntensity);
```

PluginCopyBlueChannelAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyBlueChannelAllFramesNameD(
    const char* path, double redIntensity, double greenIntensity);

// Class Plugin
double result = ChromaAnimationAPI::CopyBlueChannelAllFramesNameD(
    const char* path, double redIntensity, double greenIntensity);
```

PluginCopyGreenChannelAllFrames

Copy green channel to other channels for all frames. Intensity range is 0.0 to 1.0. Reference the animation by id.

```
// DLL Interface
EXPORT_API void PluginCopyGreenChannelAllFrames(
    int animationId, float redIntensity, float blueIntensity);

// Class Plugin
ChromaAnimationAPI::CopyGreenChannelAllFrames(
    int animationId, float redIntensity, float blueIntensity);
```

PluginCopyGreenChannelAllFramesName

Copy green channel to other channels for all frames. Intensity range is 0.0 to 1.0. Reference the animation by name.

```
// DLL Interface
EXPORT_API void PluginCopyGreenChannelAllFramesName(
    const char* path, float redIntensity, float blueIntensity);

// Class Plugin
```

```
ChromaAnimationAPI::CopyGreenChannelAllFramesName(
    const char* path, float redIntensity, float blueIntensity);
```

PluginCopyGreenChannelAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyGreenChannelAllFramesNameD(
    const char* path, double redIntensity, double blueIntensity);

// Class Plugin
double result = ChromaAnimationAPI::CopyGreenChannelAllFramesNameD(
    const char* path, double redIntensity, double blueIntensity);
```

PluginCopyKeyColor

Copy animation key color from the source animation to the target animation for the given frame. Reference the source and target by id.

```
// DLL Interface
EXPORT_API void PluginCopyKeyColor(
    int sourceAnimationId, int targetAnimationId, int frameId, int rzkey);

// Class Plugin
ChromaAnimationAPI::CopyKeyColor(
    int sourceAnimationId, int targetAnimationId, int frameId, int rzkey);
```

PluginCopyKeyColorAllFrames

Copy animation key color from the source animation to the target animation for all frames. Reference the source and target by id.

```
// DLL Interface
EXPORT_API void PluginCopyKeyColorAllFrames(
    int sourceAnimationId, int targetAnimationId, int rzkey);

// Class Plugin
ChromaAnimationAPI::CopyKeyColorAllFrames(
    int sourceAnimationId, int targetAnimationId, int rzkey);
```

PluginCopyKeyColorAllFramesName

Copy animation key color from the source animation to the target animation for all frames. Reference the source and target by name.

```
// DLL Interface
EXPORT_API void PluginCopyKeyColorAllFramesName(
    const char* sourceAnimation, const char* targetAnimation, int rzkey);

// Class Plugin
ChromaAnimationAPI::CopyKeyColorAllFramesName(
    const char* sourceAnimation, const char* targetAnimation, int rzkey);
```

PluginCopyKeyColorAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyKeyColorAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation, double rzkey);

// Class Plugin
double result = ChromaAnimationAPI::CopyKeyColorAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation, double rzkey);
```

PluginCopyKeyColorAllFramesOffset

Copy animation key color from the source animation to the target animation for all frames, starting at the offset for the length of the source animation. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyKeyColorAllFramesOffset(
    int sourceAnimationId, int targetAnimationId, int rzkey, int offset);

// Class Plugin
ChromaAnimationAPI::CopyKeyColorAllFramesOffset(
    int sourceAnimationId, int targetAnimationId, int rzkey, int offset);
```

PluginCopyKeyColorAllFramesOffsetName

Copy animation key color from the source animation to the target animation for all frames, starting at the offset for the length of the source animation. Source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyKeyColorAllFramesOffsetName(
```

```
    const char* sourceAnimation, const char* targetAnimation, int rzkey, int
offset);

// Class Plugin
ChromaAnimationAPI::CopyKeyColorAllFramesOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int rzkey, int
offset);
```

PluginCopyKeyColorAllFramesOffsetNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyKeyColorAllFramesOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double rzkey, double
offset);

// Class Plugin
double result = ChromaAnimationAPI::CopyKeyColorAllFramesOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double rzkey, double
offset);
```

PluginCopyKeyColorName

Copy animation key color from the source animation to the target animation for the given frame.

```
// DLL Interface
EXPORT_API void PluginCopyKeyColorName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
rzkey);

// Class Plugin
ChromaAnimationAPI::CopyKeyColorName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
rzkey);
```

PluginCopyKeyColorNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyKeyColorNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double rzkey);
```

```
// Class Plugin
double result = ChromaAnimationAPI::CopyKeyColorNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double rzkey);
```

PluginCopyKeysColor

Copy animation color for a set of keys from the source animation to the target animation for the given frame. Reference the source and target by id.

```
// DLL Interface
EXPORT_API void PluginCopyKeysColor(
    int sourceAnimationId, int targetAnimationId, int frameId, const int* keys,
    int size);

// Class Plugin
ChromaAnimationAPI::CopyKeysColor(
    int sourceAnimationId, int targetAnimationId, int frameId, const int* keys,
    int size);
```

PluginCopyKeysColorAllFrames

Copy animation color for a set of keys from the source animation to the target animation for all frames. Reference the source and target by id.

```
// DLL Interface
EXPORT_API void PluginCopyKeysColorAllFrames(
    int sourceAnimationId, int targetAnimationId, const int* keys, int size);

// Class Plugin
ChromaAnimationAPI::CopyKeysColorAllFrames(
    int sourceAnimationId, int targetAnimationId, const int* keys, int size);
```

PluginCopyKeysColorAllFramesName

Copy animation color for a set of keys from the source animation to the target animation for all frames. Reference the source and target by name.

```
// DLL Interface
EXPORT_API void PluginCopyKeysColorAllFramesName(
    const char* sourceAnimation, const char* targetAnimation, const int* keys,
    int size);

// Class Plugin
```

```
ChromaAnimationAPI::CopyKeysColorAllFramesName(
    const char* sourceAnimation, const char* targetAnimation, const int* keys,
    int size);
```

PluginCopyKeysColorName

Copy animation color for a set of keys from the source animation to the target animation for the given frame. Reference the source and target by name.

```
// DLL Interface
EXPORT_API void PluginCopyKeysColorName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, const
int* keys,
    int size);

// Class Plugin
ChromaAnimationAPI::CopyKeysColorName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, const
int* keys,
    int size);
```

PluginCopyKeysColorOffset

Copy animation color for a set of keys from the source animation to the target animation from the source frame to the target frame. Reference the source and target by id.

```
// DLL Interface
EXPORT_API void PluginCopyKeysColorOffset(
    int sourceAnimationId, int targetAnimationId, int sourceFrameId, int
targetFrameId,
    const int* keys, int size);

// Class Plugin
ChromaAnimationAPI::CopyKeysColorOffset(
    int sourceAnimationId, int targetAnimationId, int sourceFrameId, int
targetFrameId,
    const int* keys, int size);
```

PluginCopyKeysColorOffsetName

Copy animation color for a set of keys from the source animation to the target animation from the source frame to the target frame. Reference the source and target by name.

```
// DLL Interface
EXPORT_API void PluginCopyKeysColorOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int sourceFrameId,
    int targetFrameId, const int* keys, int size);

// Class Plugin
ChromaAnimationAPI::CopyKeysColorOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int sourceFrameId,
    int targetFrameId, const int* keys, int size);
```

PluginCopyNonZeroAllKeys

Copy source animation to target animation for the given frame. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroAllKeys(
    int sourceAnimationId, int targetAnimationId, int frameId);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroAllKeys(
    int sourceAnimationId, int targetAnimationId, int frameId);
```

PluginCopyNonZeroAllKeysAllFrames

Copy nonzero colors from a source animation to a target animation for all frames. Reference source and target by id.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);
```

PluginCopyNonZeroAllKeysAllFramesName

Copy nonzero colors from a source animation to a target animation for all frames. Reference source and target by name.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);
```

```
// Class Plugin
ChromaAnimationAPI::CopyNonZeroAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginCopyNonZeroAllKeysAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyNonZeroAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
double result = ChromaAnimationAPI::CopyNonZeroAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginCopyNonZeroAllKeysAllFramesOffset

Copy nonzero colors from a source animation to a target animation for all frames starting at the offset for the length of the source animation. The source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroAllKeysAllFramesOffset(
    int sourceAnimationId, int targetAnimationId, int offset);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroAllKeysAllFramesOffset(
    int sourceAnimationId, int targetAnimationId, int offset);
```

PluginCopyNonZeroAllKeysAllFramesOffsetName

Copy nonzero colors from a source animation to a target animation for all frames starting at the offset for the length of the source animation. The source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroAllKeysAllFramesOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int offset);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroAllKeysAllFramesOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int offset);
```

PluginCopyNonZeroAllKeysAllFramesOffsetNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyNonZeroAllKeysAllFramesOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double offset);

// Class Plugin
double result = ChromaAnimationAPI::CopyNonZeroAllKeysAllFramesOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double offset);
```

PluginCopyNonZeroAllKeysName

Copy nonzero colors from source animation to target animation for the specified frame. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroAllKeysName(
    const char* sourceAnimation, const char* targetAnimation, int frameId);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroAllKeysName(
    const char* sourceAnimation, const char* targetAnimation, int frameId);
```

PluginCopyNonZeroAllKeysNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyNonZeroAllKeysNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId);

// Class Plugin
double result = ChromaAnimationAPI::CopyNonZeroAllKeysNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId);
```

PluginCopyNonZeroAllKeysOffset

Copy nonzero colors from the source animation to the target animation from the source frame to the target offset frame. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroAllKeysOffset(
    int sourceAnimationId, int targetAnimationId, int frameId, int offset);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroAllKeysOffset(
    int sourceAnimationId, int targetAnimationId, int frameId, int offset);
```

PluginCopyNonZeroAllKeysOffsetName

Copy nonzero colors from the source animation to the target animation from the source frame to the target offset frame. Source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroAllKeysOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
offset);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroAllKeysOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
offset);
```

PluginCopyNonZeroAllKeysOffsetNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyNonZeroAllKeysOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double offset);

// Class Plugin
double result = ChromaAnimationAPI::CopyNonZeroAllKeysOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double offset);
```

PluginCopyNonZeroKeyColor

Copy animation key color from the source animation to the target animation for the given frame where color is not zero.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroKeyColor(
```

```
    int sourceAnimationId, int targetAnimationId, int frameId, int rzkey);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroKeyColor(
    int sourceAnimationId, int targetAnimationId, int frameId, int rzkey);
```

PluginCopyNonZeroKeyColorName

Copy animation key color from the source animation to the target animation for the given frame where color is not zero.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroKeyColorName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
rzkey);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroKeyColorName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
rzkey);
```

PluginCopyNonZeroKeyColorNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyNonZeroKeyColorNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double rzkey);

// Class Plugin
double result = ChromaAnimationAPI::CopyNonZeroKeyColorNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double rzkey);
```

PluginCopyNonZeroTargetAllKeys

Copy nonzero colors from the source animation to the target animation where the target color is nonzero for the specified frame. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroTargetAllKeys(
    int sourceAnimationId, int targetAnimationId, int frameId);
```

```
// Class Plugin
ChromaAnimationAPI::CopyNonZeroTargetAllKeys(
    int sourceAnimationId, int targetAnimationId, int frameId);
```

PluginCopyNonZeroTargetAllKeysAllFrames

Copy nonzero colors from the source animation to the target animation where the target color is nonzero for all frames. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroTargetAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroTargetAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);
```

PluginCopyNonZeroTargetAllKeysAllFramesName

Copy nonzero colors from the source animation to the target animation where the target color is nonzero for all frames. Source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroTargetAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroTargetAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginCopyNonZeroTargetAllKeysAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyNonZeroTargetAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
double result = ChromaAnimationAPI::CopyNonZeroTargetAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginCopyNonZeroTargetAllKeysAllFramesOffset

Copy nonzero colors from the source animation to the target animation where the target color is nonzero for all frames. Source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroTargetAllKeysAllFramesOffset(
    int sourceAnimationId, int targetAnimationId, int offset);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroTargetAllKeysAllFramesOffset(
    int sourceAnimationId, int targetAnimationId, int offset);
```

PluginCopyNonZeroTargetAllKeysAllFramesOffsetName

Copy nonzero colors from the source animation to the target animation where the target color is nonzero for all frames starting at the target offset for the length of the source animation. Source and target animations are referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroTargetAllKeysAllFramesOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int offset);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroTargetAllKeysAllFramesOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int offset);
```

PluginCopyNonZeroTargetAllKeysAllFramesOffsetNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyNonZeroTargetAllKeysAllFramesOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double offset);

// Class Plugin
double result = ChromaAnimationAPI::CopyNonZeroTargetAllKeysAllFramesOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double offset);
```

PluginCopyNonZeroTargetAllKeysName

Copy nonzero colors from the source animation to the target animation where the target color is nonzero for the specified frame. The source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroTargetAllKeysName(
    const char* sourceAnimation, const char* targetAnimation, int frameId);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroTargetAllKeysName(
    const char* sourceAnimation, const char* targetAnimation, int frameId);
```

PluginCopyNonZeroTargetAllKeysNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyNonZeroTargetAllKeysNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId);

// Class Plugin
double result = ChromaAnimationAPI::CopyNonZeroTargetAllKeysNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId);
```

PluginCopyNonZeroTargetAllKeysOffset

Copy nonzero colors from the source animation to the target animation where the target color is nonzero for the specified source frame and target offset frame. The source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroTargetAllKeysOffset(
    int sourceAnimationId, int targetAnimationId, int frameId, int offset);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroTargetAllKeysOffset(
    int sourceAnimationId, int targetAnimationId, int frameId, int offset);
```

PluginCopyNonZeroTargetAllKeysOffsetName

Copy nonzero colors from the source animation to the target animation where the target color is nonzero for the specified source frame and target offset frame. The source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroTargetAllKeysOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
offset);

// Class Plugin
```

```
ChromaAnimationAPI::CopyNonZeroTargetAllKeysOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
offset);
```

PluginCopyNonZeroTargetAllKeysOffsetNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyNonZeroTargetAllKeysOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double offset);

// Class Plugin
double result = ChromaAnimationAPI::CopyNonZeroTargetAllKeysOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double offset);
```

PluginCopyNonZeroTargetZeroAllKeysAllFrames

Copy nonzero colors from the source animation to the target animation where the target color is zero for all frames. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroTargetZeroAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroTargetZeroAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);
```

PluginCopyNonZeroTargetZeroAllKeysAllFramesName

Copy nonzero colors from the source animation to the target animation where the target color is zero for all frames. Source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyNonZeroTargetZeroAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
ChromaAnimationAPI::CopyNonZeroTargetZeroAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginCopyNonZeroTargetZeroAllKeysAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyNonZeroTargetZeroAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
double result = ChromaAnimationAPI::CopyNonZeroTargetZeroAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginCopyRedChannelAllFrames

Copy red channel to other channels for all frames. Intensity range is 0.0 to 1.0. Reference the animation by id.

```
// DLL Interface
EXPORT_API void PluginCopyRedChannelAllFrames(
    int animationId, float greenIntensity, float blueIntensity);

// Class Plugin
ChromaAnimationAPI::CopyRedChannelAllFrames(
    int animationId, float greenIntensity, float blueIntensity);
```

PluginCopyRedChannelAllFramesName

Copy green channel to other channels for all frames. Intensity range is 0.0 to 1.0. Reference the animation by name.

```
// DLL Interface
EXPORT_API void PluginCopyRedChannelAllFramesName(
    const char* path, float greenIntensity, float blueIntensity);

// Class Plugin
ChromaAnimationAPI::CopyRedChannelAllFramesName(
    const char* path, float greenIntensity, float blueIntensity);
```

PluginCopyRedChannelAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyRedChannelAllFramesNameD(
```

```
    const char* path, double greenIntensity, double blueIntensity);

// Class Plugin
double result = ChromaAnimationAPI::CopyRedChannelAllFramesNameD(
    const char* path, double greenIntensity, double blueIntensity);
```

PluginCopyZeroAllKeys

Copy zero colors from source animation to target animation for the frame. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyZeroAllKeys(
    int sourceAnimationId, int targetAnimationId, int frameId);

// Class Plugin
ChromaAnimationAPI::CopyZeroAllKeys(
    int sourceAnimationId, int targetAnimationId, int frameId);
```

PluginCopyZeroAllKeysAllFrames

Copy zero colors from source animation to target animation for all frames. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyZeroAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);

// Class Plugin
ChromaAnimationAPI::CopyZeroAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);
```

PluginCopyZeroAllKeysAllFramesName

Copy zero colors from source animation to target animation for all frames. Source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyZeroAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
ChromaAnimationAPI::CopyZeroAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginCopyZeroAllKeysAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyZeroAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
double result = ChromaAnimationAPI::CopyZeroAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginCopyZeroAllKeysAllFramesOffset

Copy zero colors from source animation to target animation for all frames starting at the target offset for the length of the source animation. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyZeroAllKeysAllFramesOffset(
    int sourceAnimationId, int targetAnimationId, int offset);

// Class Plugin
ChromaAnimationAPI::CopyZeroAllKeysAllFramesOffset(
    int sourceAnimationId, int targetAnimationId, int offset);
```

PluginCopyZeroAllKeysAllFramesOffsetName

Copy zero colors from source animation to target animation for all frames starting at the target offset for the length of the source animation. Source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyZeroAllKeysAllFramesOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int offset);

// Class Plugin
ChromaAnimationAPI::CopyZeroAllKeysAllFramesOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int offset);
```

PluginCopyZeroAllKeysAllFramesOffsetNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyZeroAllKeysAllFramesOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double offset);

// Class Plugin
double result = ChromaAnimationAPI::CopyZeroAllKeysAllFramesOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double offset);
```

PluginCopyZeroAllKeysName

Copy zero colors from source animation to target animation for the frame. Source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyZeroAllKeysName(
    const char* sourceAnimation, const char* targetAnimation, int frameId);

// Class Plugin
ChromaAnimationAPI::CopyZeroAllKeysName(
    const char* sourceAnimation, const char* targetAnimation, int frameId);
```

PluginCopyZeroAllKeysOffset

Copy zero colors from source animation to target animation for the frame id starting at the target offset for the length of the source animation. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyZeroAllKeysOffset(
    int sourceAnimationId, int targetAnimationId, int frameId, int offset);

// Class Plugin
ChromaAnimationAPI::CopyZeroAllKeysOffset(
    int sourceAnimationId, int targetAnimationId, int frameId, int offset);
```

PluginCopyZeroAllKeysOffsetName

Copy zero colors from source animation to target animation for the frame id starting at the target offset for the length of the source animation. Source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyZeroAllKeysOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
offset);
```

```
// Class Plugin
ChromaAnimationAPI::CopyZeroAllKeysOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
offset);
```

PluginCopyZeroKeyColor

Copy zero key color from source animation to target animation for the specified frame. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyZeroKeyColor(
    int sourceAnimationId, int targetAnimationId, int frameId, int rzkey);

// Class Plugin
ChromaAnimationAPI::CopyZeroKeyColor(
    int sourceAnimationId, int targetAnimationId, int frameId, int rzkey);
```

PluginCopyZeroKeyColorName

Copy zero key color from source animation to target animation for the specified frame. Source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyZeroKeyColorName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
rzkey);

// Class Plugin
ChromaAnimationAPI::CopyZeroKeyColorName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
rzkey);
```

PluginCopyZeroKeyColorNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyZeroKeyColorNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double rzkey);

// Class Plugin
```

```
double result = ChromaAnimationAPI::CopyZeroKeyColorNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double rzkey);
```

PluginCopyZeroTargetAllKeys

Copy nonzero color from source animation to target animation where target is zero for the frame. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyZeroTargetAllKeys(
    int sourceAnimationId, int targetAnimationId, int frameId);

// Class Plugin
ChromaAnimationAPI::CopyZeroTargetAllKeys(
    int sourceAnimationId, int targetAnimationId, int frameId);
```

PluginCopyZeroTargetAllKeysAllFrames

Copy nonzero color from source animation to target animation where target is zero for all frames. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginCopyZeroTargetAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);

// Class Plugin
ChromaAnimationAPI::CopyZeroTargetAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);
```

PluginCopyZeroTargetAllKeysAllFramesName

Copy nonzero color from source animation to target animation where target is zero for all frames. Source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyZeroTargetAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
ChromaAnimationAPI::CopyZeroTargetAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginCopyZeroTargetAllKeysAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginCopyZeroTargetAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
double result = ChromaAnimationAPI::CopyZeroTargetAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginCopyZeroTargetAllKeysName

Copy nonzero color from source animation to target animation where target is zero for the frame. Source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginCopyZeroTargetAllKeysName(
    const char* sourceAnimation, const char* targetAnimation, int frameId);

// Class Plugin
ChromaAnimationAPI::CopyZeroTargetAllKeysName(
    const char* sourceAnimation, const char* targetAnimation, int frameId);
```

PluginCoreCreateChromaLinkEffect

Direct access to low level API.

```
// DLL Interface
EXPORT_API RZRESULT PluginCoreCreateChromaLinkEffect(
    ChromaSDK::ChromaLink::EFFECT_TYPE Effect, PRZPARAM pParam, RZEFFECTID*
pEffectId);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CoreCreateChromaLinkEffect(
    ChromaSDK::ChromaLink::EFFECT_TYPE Effect, PRZPARAM pParam, RZEFFECTID*
pEffectId);
```

PluginCoreCreateEffect

Direct access to low level API.

```
```

```
// DLL Interface
EXPORT_API RZRESULT PluginCoreCreateEffect(
    RZDEVICEID DeviceId, ChromaSDK::EFFECT_TYPE Effect, PRZPARAM pParam,
RZEFFECTID* pEffectId);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CoreCreateEffect(
    RZDEVICEID DeviceId, ChromaSDK::EFFECT_TYPE Effect, PRZPARAM pParam,
RZEFFECTID* pEffectId);
```

PluginCoreCreateHeadsetEffect

Direct access to low level API.

```
// DLL Interface
EXPORT_API RZRESULT PluginCoreCreateHeadsetEffect(
    ChromaSDK::Headset::EFFECT_TYPE Effect, PRZPARAM pParam, RZEFFECTID*
pEffectId);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CoreCreateHeadsetEffect(
    ChromaSDK::Headset::EFFECT_TYPE Effect, PRZPARAM pParam, RZEFFECTID*
pEffectId);
```

PluginCoreCreateKeyboardEffect

Direct access to low level API.

```
// DLL Interface
EXPORT_API RZRESULT PluginCoreCreateKeyboardEffect(
    ChromaSDK::Keyboard::EFFECT_TYPE Effect, PRZPARAM pParam, RZEFFECTID*
pEffectId);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CoreCreateKeyboardEffect(
    ChromaSDK::Keyboard::EFFECT_TYPE Effect, PRZPARAM pParam, RZEFFECTID*
pEffectId);
```

PluginCoreCreateKeypadEffect

Direct access to low level API.

```
// DLL Interface
EXPORT_API RZRESULT PluginCoreCreateKeypadEffect(
```

```
    ChromaSDK::Keypad::EFFECT_TYPE Effect, PRZPARAM pParam, RZEFFECTID*
pEffectId);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CoreCreateKeypadEffect(
    ChromaSDK::Keypad::EFFECT_TYPE Effect, PRZPARAM pParam, RZEFFECTID*
pEffectId);
```

PluginCoreCreateMouseEffect

Direct access to low level API.

```
// DLL Interface
EXPORT_API RZRESULT PluginCoreCreateMouseEffect(
    ChromaSDK::Mouse::EFFECT_TYPE Effect, PRZPARAM pParam, RZEFFECTID* pEffectId);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CoreCreateMouseEffect(
    ChromaSDK::Mouse::EFFECT_TYPE Effect, PRZPARAM pParam, RZEFFECTID* pEffectId);
```

PluginCoreCreateMousepadEffect

Direct access to low level API.

```
// DLL Interface
EXPORT_API RZRESULT PluginCoreCreateMousepadEffect(
    ChromaSDK::Mousepad::EFFECT_TYPE Effect, PRZPARAM pParam, RZEFFECTID*
pEffectId);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CoreCreateMousepadEffect(
    ChromaSDK::Mousepad::EFFECT_TYPE Effect, PRZPARAM pParam, RZEFFECTID*
pEffectId);
```

PluginCoreDeleteEffect

Direct access to low level API.

```
// DLL Interface
EXPORT_API RZRESULT PluginCoreDeleteEffect(RZEFFECTID EffectId);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CoreDeleteEffect(RZEFFECTID EffectId);
```

PluginCoreInit

Direct access to low level API.

```
// DLL Interface
EXPORT_API RZRESULT PluginCoreInit();

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CoreInit();
```

PluginCoreInitSDK

Direct access to low level API.

```
// DLL Interface
EXPORT_API RZRESULT PluginCoreInitSDK(
    ChromaSDK::APPINFOTYPE* AppInfo);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CoreInitSDK(
    ChromaSDK::APPINFOTYPE* AppInfo);
```

PluginCoreIsActive

Direct access to low level API.

```
// DLL Interface
EXPORT_API RZRESULT PluginCoreIsActive(BOOL& Active);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CoreIsActive(BOOL& Active);
```

PluginCoreIsConnected

Direct access to low level API.

```
// DLL Interface
EXPORT_API RZRESULT PluginCoreIsConnected(
    ChromaSDK::DEVICE_INFO_TYPE& DeviceInfo);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CoreIsConnected(
    ChromaSDK::DEVICE_INFO_TYPE& DeviceInfo);
```

```

```

PluginCoreQueryDevice

Direct access to low level API.

```
// DLL Interface
EXPORT_API RZRESULT PluginCoreQueryDevice(
    RZDEVICEID DeviceId, ChromaSDK::DEVICE_INFO_TYPE& DeviceInfo);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CoreQueryDevice(
    RZDEVICEID DeviceId, ChromaSDK::DEVICE_INFO_TYPE& DeviceInfo);
```

PluginCoreSetEffect

Direct access to low level API.

```
// DLL Interface
EXPORT_API RZRESULT PluginCoreSetEffect(RZEFFECTID EffectId);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CoreSetEffect(RZEFFECTID EffectId);
```

PluginCoreSetEventName

Direct access to low level API.

```
// DLL Interface
EXPORT_API RZRESULT PluginCoreSetEventName(LPCTSTR Name);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CoreSetEventName(LPCTSTR Name);
```

PluginCoreStreamBroadcast

Begin broadcasting Chroma RGB data using the stored stream key as the endpoint. Intended for Cloud Gaming Platforms, restore the streaming key when the game instance is launched to continue streaming. streamId is a null terminated string streamKey is a null terminated string StreamGetStatus() should return the READY status to use this method.

```
// DLL Interface
EXPORT_API bool PluginCoreStreamBroadcast(
```

```
    const char* streamId, const char* streamKey);

// Class Plugin
bool result = ChromaAnimationAPI::CoreStreamBroadcast(
    const char* streamId, const char* streamKey);
```

PluginCoreStreamBroadcastEnd

End broadcasting Chroma RGB data. StreamGetStatus() should return the BROADCASTING status to use this method.

```
// DLL Interface
EXPORT_API bool PluginCoreStreamBroadcastEnd();

// Class Plugin
bool result = ChromaAnimationAPI::CoreStreamBroadcastEnd();
```

PluginCoreStreamGetAuthShortcode

shortcode: Pass the address of a preallocated character buffer to get the streaming auth code. The buffer should have a minimum length of 6. length: Length will return as zero if the streaming auth code could not be obtained. If length is greater than zero, it will be the length of the returned streaming auth code. Once you have the shortcode, it should be shown to the user so they can associate the stream with their Razer ID StreamGetStatus() should return the READY status before invoking this method. platform: is the null terminated string that identifies the source of the stream: { GEFORCE_NOW, LUNA, STADIA, GAME_PASS } title: is the null terminated string that identifies the application or game.

```
// DLL Interface
EXPORT_API void PluginCoreStreamGetAuthShortcode(
    char* shortcode, unsigned char* length, const wchar_t* platform, const
wchar_t* title);

// Class Plugin
ChromaAnimationAPI::CoreStreamGetAuthShortcode(
    char* shortcode, unsigned char* length, const wchar_t* platform, const
wchar_t* title);
```

PluginCoreStreamGetFocus

focus: Pass the address of a preallocated character buffer to get the stream focus. The buffer should have a length of 48 length: Length will return as zero if the stream focus could not be obtained. If length is greater than zero, it will be the length of the returned stream focus.

```
// DLL Interface
EXPORT_API bool PluginCoreStreamGetFocus(
    char* focus, unsigned char* length);

// Class Plugin
bool result = ChromaAnimationAPI::CoreStreamGetFocus(
    char* focus, unsigned char* length);
```

PluginCoreStreamGetId

Intended for Cloud Gaming Platforms, store the stream id to persist in user preferences to continue streaming if the game is suspended or closed. shortcode: The shortcode is a null terminated string. Use the shortcode that authorized the stream to obtain the stream id. streamId should be a preallocated buffer to get the stream key. The buffer should have a length of 48. length: Length will return zero if the key could not be obtained. If the length is greater than zero, it will be the length of the returned streaming id. Retrieve the stream id after authorizing the shortcode. The authorization window will expire in 5 minutes. Be sure to save the stream key before the window expires. StreamGetStatus() should return the READY status to use this method.

```
// DLL Interface
EXPORT_API void PluginCoreStreamGetId(
    const char* shortcode, char* streamId, unsigned char* length);

// Class Plugin
ChromaAnimationAPI::CoreStreamGetId(
    const char* shortcode, char* streamId, unsigned char* length);
```

PluginCoreStreamGetKey

Intended for Cloud Gaming Platforms, store the streaming key to persist in user preferences to continue streaming if the game is suspended or closed. shortcode: The shortcode is a null terminated string. Use the shortcode that authorized the stream to obtain the stream key. If the status is in the BROADCASTING or WATCHING state, passing a NULL shortcode will return the active streamId. streamKey should be a preallocated buffer to get the stream key. The buffer should have a length of 48. length: Length will return zero if the key could not be obtained. If the length is greater than zero, it will be the length of the returned streaming key. Retrieve the stream key after authorizing the shortcode. The authorization window will expire in 5 minutes. Be sure to save the stream key before the window expires. StreamGetStatus() should return the READY status to use this method.

```
// DLL Interface
EXPORT_API void PluginCoreStreamGetKey(
    const char* shortcode, char* streamKey, unsigned char* length);

// Class Plugin
ChromaAnimationAPI::CoreStreamGetKey(
    const char* shortcode, char* streamKey, unsigned char* length);
```

```

```

PluginCoreStreamGetStatus

Returns StreamStatus, the current status of the service

```
// DLL Interface
EXPORT_API ChromaSDK::Stream::StreamStatusType PluginCoreStreamGetStatus();

// Class Plugin
ChromaSDK::Stream::StreamStatusType result =
ChromaAnimationAPI::CoreStreamGetStatus();
```

PluginCoreStreamGetStatusString

Convert StreamStatusType to a printable string

```
// DLL Interface
EXPORT_API const char* PluginCoreStreamGetStatusString(
    ChromaSDK::Stream::StreamStatusType status);

// Class Plugin
const char* result = ChromaAnimationAPI::CoreStreamGetStatusString(
    ChromaSDK::Stream::StreamStatusType status);
```

PluginCoreStreamReleaseShortcode

This prevents the stream id and stream key from being obtained through the shortcode. This closes the auth window. shortcode is a null terminated string. StreamGetStatus() should return the READY status to use this method. returns success when shortcode has been released

```
// DLL Interface
EXPORT_API bool PluginCoreStreamReleaseShortcode(
    const char* shortcode);

// Class Plugin
bool result = ChromaAnimationAPI::CoreStreamReleaseShortcode(
    const char* shortcode);
```

PluginCoreStreamSetFocus

The focus is a null terminated string. Set the focus identifer for the application designated to automatically change the streaming state. Returns true on success.

```
// DLL Interface
EXPORT_API bool PluginCoreStreamSetFocus(const char* focus);

// Class Plugin
bool result = ChromaAnimationAPI::CoreStreamSetFocus(const char* focus);
```

PluginCoreStreamSupportsStreaming

Returns true if the Chroma streaming is supported. If false is returned, avoid calling stream methods.

```
// DLL Interface
EXPORT_API bool PluginCoreStreamSupportsStreaming();

// Class Plugin
bool result = ChromaAnimationAPI::CoreStreamSupportsStreaming();
```

PluginCoreStreamWatch

Begin watching the Chroma RGB data using streamID parameter. streamId is a null terminated string. StreamGetStatus() should return the READY status to use this method.

```
// DLL Interface
EXPORT_API bool PluginCoreStreamWatch(
    const char* streamId, unsigned long long timestamp);

// Class Plugin
bool result = ChromaAnimationAPI::CoreStreamWatch(
    const char* streamId, unsigned long long timestamp);
```

PluginCoreStreamWatchEnd

End watching Chroma RGB data stream. StreamGetStatus() should return the WATCHING status to use this method.

```
// DLL Interface
EXPORT_API bool PluginCoreStreamWatchEnd();

// Class Plugin
bool result = ChromaAnimationAPI::CoreStreamWatchEnd();
```

PluginCoreUnInit

Direct access to low level API.

```
// DLL Interface
EXPORT_API RZRESULT PluginCoreUnInit();

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CoreUnInit();
```

PluginCreateAnimation

Creates a `Chroma` animation at the given path. The `deviceType` parameter uses `EChromaSDKDeviceTypeEnum` as an integer. The `device` parameter uses `EChromaSDKDevice1DEnum` or `EChromaSDKDevice2DEnum` as an integer, respective to the `deviceType`. Returns the animation id upon success. Returns negative one upon failure. Saves a `Chroma` animation file with the `.chroma` extension at the given path. Returns the animation id upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginCreateAnimation(
    const char* path, int deviceType, int device);

// Class Plugin
int result = ChromaAnimationAPI::CreateAnimation(
    const char* path, int deviceType, int device);
```

PluginCreateAnimationInMemory

Creates a `Chroma` animation in memory without creating a file. The `deviceType` parameter uses `EChromaSDKDeviceTypeEnum` as an integer. The `device` parameter uses `EChromaSDKDevice1DEnum` or `EChromaSDKDevice2DEnum` as an integer, respective to the `deviceType`. Returns the animation id upon success. Returns negative one upon failure. Returns the animation id upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginCreateAnimationInMemory(
    int deviceType, int device);

// Class Plugin
int result = ChromaAnimationAPI::CreateAnimationInMemory(
    int deviceType, int device);
```

PluginCreateEffect

Create a device specific effect.

```
// DLL Interface
EXPORT_API RZRESULT PluginCreateEffect(
    RZDEVICEID deviceId, ChromaSDK::EFFECT_TYPE effect, int* colors, int size,
    ChromaSDK::FChromaSDKGuid* effectId);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::CreateEffect(
    RZDEVICEID deviceId, ChromaSDK::EFFECT_TYPE effect, int* colors, int size,
    ChromaSDK::FChromaSDKGuid* effectId);
```

PluginDeleteEffect

Delete an effect given the effect id.

```
// DLL Interface
EXPORT_API RZRESULT PluginDeleteEffect(
    const ChromaSDK::FChromaSDKGuid& effectId);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::DeleteEffect(
    const ChromaSDK::FChromaSDKGuid& effectId);
```

PluginDuplicateFirstFrame

Duplicate the first animation frame so that the animation length matches the frame count. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginDuplicateFirstFrame(
    int animationId, int frameCount);

// Class Plugin
ChromaAnimationAPI::DuplicateFirstFrame(
    int animationId, int frameCount);
```

PluginDuplicateFirstFrameName

Duplicate the first animation frame so that the animation length matches the frame count. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginDuplicateFirstFrameName(
    const char* path, int frameCount);
```

```
// Class Plugin
ChromaAnimationAPI::DuplicateFirstFrameName(
    const char* path, int frameCount);
```

PluginDuplicateFirstFrameNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginDuplicateFirstFrameNameD(
    const char* path, double frameCount);

// Class Plugin
double result = ChromaAnimationAPI::DuplicateFirstFrameNameD(
    const char* path, double frameCount);
```

PluginDuplicateFrames

Duplicate all the frames of the animation to double the animation length. Frame 1 becomes frame 1 and 2. Frame 2 becomes frame 3 and 4. And so on. The animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginDuplicateFrames(int animationId);

// Class Plugin
ChromaAnimationAPI::DuplicateFrames(int animationId);
```

PluginDuplicateFramesName

Duplicate all the frames of the animation to double the animation length. Frame 1 becomes frame 1 and 2. Frame 2 becomes frame 3 and 4. And so on. The animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginDuplicateFramesName(const char* path);

// Class Plugin
ChromaAnimationAPI::DuplicateFramesName(const char* path);
```

PluginDuplicateFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginDuplicateFramesNameD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::DuplicateFramesNameD(const char* path);
```

PluginDuplicateMirrorFrames

Duplicate all the animation frames in reverse so that the animation plays forwards and backwards. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginDuplicateMirrorFrames(int animationId);

// Class Plugin
ChromaAnimationAPI::DuplicateMirrorFrames(int animationId);
```

PluginDuplicateMirrorFramesName

Duplicate all the animation frames in reverse so that the animation plays forwards and backwards. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginDuplicateMirrorFramesName(const char* path);

// Class Plugin
ChromaAnimationAPI::DuplicateMirrorFramesName(const char* path);
```

PluginDuplicateMirrorFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginDuplicateMirrorFramesNameD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::DuplicateMirrorFramesNameD(const char* path);
```

PluginFadeEndFrames

Fade the animation to black starting at the fade frame index to the end of the animation. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFadeEndFrames(
    int animationId, int fade);

// Class Plugin
ChromaAnimationAPI::FadeEndFrames(
    int animationId, int fade);
```

PluginFadeEndFramesName

Fade the animation to black starting at the fade frame index to the end of the animation. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFadeEndFramesName(
    const char* path, int fade);

// Class Plugin
ChromaAnimationAPI::FadeEndFramesName(
    const char* path, int fade);
```

PluginFadeEndFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFadeEndFramesNameD(
    const char* path, double fade);

// Class Plugin
double result = ChromaAnimationAPI::FadeEndFramesNameD(
    const char* path, double fade);
```

PluginFadeStartFrames

Fade the animation from black to full color starting at 0 to the fade frame index. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFadeStartFrames(
    int animationId, int fade);
```

```
// Class Plugin
ChromaAnimationAPI::FadeStartFrames(
    int animationId, int fade);
```

PluginFadeStartFramesName

Fade the animation from black to full color starting at 0 to the fade frame index. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFadeStartFramesName(
    const char* path, int fade);

// Class Plugin
ChromaAnimationAPI::FadeStartFramesName(
    const char* path, int fade);
```

PluginFadeStartFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFadeStartFramesNameD(
    const char* path, double fade);

// Class Plugin
double result = ChromaAnimationAPI::FadeStartFramesNameD(
    const char* path, double fade);
```

PluginFillColor

Set the RGB value for all colors in the specified frame. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillColor(
    int animationId, int frameId, int color);

// Class Plugin
ChromaAnimationAPI::FillColor(
    int animationId, int frameId, int color);
```

PluginFillColorAllFrames

Set the RGB value for all colors for all frames. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillColorAllFrames(
    int animationId, int color);

// Class Plugin
ChromaAnimationAPI::FillColorAllFrames(
    int animationId, int color);
```

PluginFillColorAllFramesName

Set the RGB value for all colors for all frames. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillColorAllFramesName(
    const char* path, int color);

// Class Plugin
ChromaAnimationAPI::FillColorAllFramesName(
    const char* path, int color);
```

PluginFillColorAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillColorAllFramesNameD(
    const char* path, double color);

// Class Plugin
double result = ChromaAnimationAPI::FillColorAllFramesNameD(
    const char* path, double color);
```

PluginFillColorAllFramesRGB

Set the RGB value for all colors for all frames. Use the range of 0 to 255 for red, green, and blue parameters. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillColorAllFramesRGB(
```

```
    int animationId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillColorAllFramesRGB(
    int animationId, int red, int green, int blue);
```

PluginFillColorAllFramesRGBName

Set the RGB value for all colors for all frames. Use the range of 0 to 255 for red, green, and blue parameters. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillColorAllFramesRGBName(
    const char* path, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillColorAllFramesRGBName(
    const char* path, int red, int green, int blue);
```

PluginFillColorAllFramesRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillColorAllFramesRGBNameD(
    const char* path, double red, double green, double blue);

// Class Plugin
double result = ChromaAnimationAPI::FillColorAllFramesRGBNameD(
    const char* path, double red, double green, double blue);
```

PluginFillColorName

Set the RGB value for all colors in the specified frame. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillColorName(
    const char* path, int frameId, int color);

// Class Plugin
ChromaAnimationAPI::FillColorName(
    const char* path, int frameId, int color);
```

PluginFillColorNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillColorNameD(
    const char* path, double frameId, double color);

// Class Plugin
double result = ChromaAnimationAPI::FillColorNameD(
    const char* path, double frameId, double color);
```

PluginFillColorRGB

Set the RGB value for all colors in the specified frame. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillColorRGB(
    int animationId, int frameId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillColorRGB(
    int animationId, int frameId, int red, int green, int blue);
```

PluginFillColorRGBName

Set the RGB value for all colors in the specified frame. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillColorRGBName(
    const char* path, int frameId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillColorRGBName(
    const char* path, int frameId, int red, int green, int blue);
```

PluginFillColorRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillColorRGBNameD(
    const char* path, double frameId, double red, double green, double blue);
```

```
// Class Plugin
double result = ChromaAnimationAPI::FillColorRGBNameD(
    const char* path, double frameId, double red, double green, double blue);
```

PluginFillNonZeroColor

This method will only update colors in the animation that are not already set to black. Set the RGB value for a subset of colors in the specified frame. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillNonZeroColor(
    int animationId, int frameId, int color);

// Class Plugin
ChromaAnimationAPI::FillNonZeroColor(
    int animationId, int frameId, int color);
```

PluginFillNonZeroColorAllFrames

This method will only update colors in the animation that are not already set to black. Set the RGB value for a subset of colors for all frames. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillNonZeroColorAllFrames(
    int animationId, int color);

// Class Plugin
ChromaAnimationAPI::FillNonZeroColorAllFrames(
    int animationId, int color);
```

PluginFillNonZeroColorAllFramesName

This method will only update colors in the animation that are not already set to black. Set the RGB value for a subset of colors for all frames. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillNonZeroColorAllFramesName(
    const char* path, int color);

// Class Plugin
ChromaAnimationAPI::FillNonZeroColorAllFramesName(
    const char* path, int color);
```

PluginFillNonZeroColorAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillNonZeroColorAllFramesNameD(
    const char* path, double color);

// Class Plugin
double result = ChromaAnimationAPI::FillNonZeroColorAllFramesNameD(
    const char* path, double color);
```

PluginFillNonZeroColorAllFramesRGB

This method will only update colors in the animation that are not already set to black. Set the RGB value for a subset of colors for all frames. Use the range of 0 to 255 for red, green, and blue parameters. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillNonZeroColorAllFramesRGB(
    int animationId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillNonZeroColorAllFramesRGB(
    int animationId, int red, int green, int blue);
```

PluginFillNonZeroColorAllFramesRGBName

This method will only update colors in the animation that are not already set to black. Set the RGB value for a subset of colors for all frames. Use the range of 0 to 255 for red, green, and blue parameters. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillNonZeroColorAllFramesRGBName(
    const char* path, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillNonZeroColorAllFramesRGBName(
    const char* path, int red, int green, int blue);
```

PluginFillNonZeroColorAllFramesRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillNonZeroColorAllFramesRGBNameD(
    const char* path, double red, double green, double blue);

// Class Plugin
double result = ChromaAnimationAPI::FillNonZeroColorAllFramesRGBNameD(
    const char* path, double red, double green, double blue);
```

PluginFillNonZeroColorName

This method will only update colors in the animation that are not already set to black. Set the RGB value for a subset of colors in the specified frame. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillNonZeroColorName(
    const char* path, int frameId, int color);

// Class Plugin
ChromaAnimationAPI::FillNonZeroColorName(
    const char* path, int frameId, int color);
```

PluginFillNonZeroColorNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillNonZeroColorNameD(
    const char* path, double frameId, double color);

// Class Plugin
double result = ChromaAnimationAPI::FillNonZeroColorNameD(
    const char* path, double frameId, double color);
```

PluginFillNonZeroColorRGB

This method will only update colors in the animation that are not already set to black. Set the RGB value for a subset of colors in the specified frame. Use the range of 0 to 255 for red, green, and blue parameters. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillNonZeroColorRGB(
    int animationId, int frameId, int red, int green, int blue);
```

```
// Class Plugin
ChromaAnimationAPI::FillNonZeroColorRGB(
    int animationId, int frameId, int red, int green, int blue);
```

PluginFillNonZeroColorRGBName

This method will only update colors in the animation that are not already set to black. Set the RGB value for a subset of colors in the specified frame. Use the range of 0 to 255 for red, green, and blue parameters. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillNonZeroColorRGBName(
    const char* path, int frameId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillNonZeroColorRGBName(
    const char* path, int frameId, int red, int green, int blue);
```

PluginFillNonZeroColorRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillNonZeroColorRGBNameD(
    const char* path, double frameId, double red, double green, double blue);

// Class Plugin
double result = ChromaAnimationAPI::FillNonZeroColorRGBNameD(
    const char* path, double frameId, double red, double green, double blue);
```

PluginFillRandomColors

Fill the frame with random RGB values for the given frame. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillRandomColors(
    int animationId, int frameId);

// Class Plugin
ChromaAnimationAPI::FillRandomColors(
    int animationId, int frameId);
```

PluginFillRandomColorsAllFrames

Fill the frame with random RGB values for all frames. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillRandomColorsAllFrames(int animationId);

// Class Plugin
ChromaAnimationAPI::FillRandomColorsAllFrames(int animationId);
```

PluginFillRandomColorsAllFramesName

Fill the frame with random RGB values for all frames. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillRandomColorsAllFramesName(const char* path);

// Class Plugin
ChromaAnimationAPI::FillRandomColorsAllFramesName(const char* path);
```

PluginFillRandomColorsAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillRandomColorsAllFramesNameD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::FillRandomColorsAllFramesNameD(const char*
path);
```

PluginFillRandomColorsBlackAndWhite

Fill the frame with random black and white values for the specified frame. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillRandomColorsBlackAndWhite(
    int animationId, int frameId);

// Class Plugin
ChromaAnimationAPI::FillRandomColorsBlackAndWhite(
    int animationId, int frameId);
```

PluginFillRandomColorsBlackAndWhiteAllFrames

Fill the frame with random black and white values for all frames. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillRandomColorsBlackAndWhiteAllFrames(int animationId);

// Class Plugin
ChromaAnimationAPI::FillRandomColorsBlackAndWhiteAllFrames(int animationId);
```

PluginFillRandomColorsBlackAndWhiteAllFramesName

Fill the frame with random black and white values for all frames. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillRandomColorsBlackAndWhiteAllFramesName(const char*
path);

// Class Plugin
ChromaAnimationAPI::FillRandomColorsBlackAndWhiteAllFramesName(const char* path);
```

PluginFillRandomColorsBlackAndWhiteAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillRandomColorsBlackAndWhiteAllFramesNameD(const char*
path);

// Class Plugin
double result =
ChromaAnimationAPI::FillRandomColorsBlackAndWhiteAllFramesNameD(const char* path);
```

PluginFillRandomColorsBlackAndWhiteName

Fill the frame with random black and white values for the specified frame. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillRandomColorsBlackAndWhiteName(
    const char* path, int frameId);

// Class Plugin
```

```
ChromaAnimationAPI::FillRandomColorsBlackAndWhiteName(
    const char* path, int frameId);
```

PluginFillRandomColorsBlackAndWhiteNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillRandomColorsBlackAndWhiteNameD(
    const char* path, double frameId);

// Class Plugin
double result = ChromaAnimationAPI::FillRandomColorsBlackAndWhiteNameD(
    const char* path, double frameId);
```

PluginFillRandomColorsName

Fill the frame with random RGB values for the given frame. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillRandomColorsName(
    const char* path, int frameId);

// Class Plugin
ChromaAnimationAPI::FillRandomColorsName(
    const char* path, int frameId);
```

PluginFillRandomColorsNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillRandomColorsNameD(
    const char* path, double frameId);

// Class Plugin
double result = ChromaAnimationAPI::FillRandomColorsNameD(
    const char* path, double frameId);
```

PluginFillThresholdColors

Fill the specified frame with RGB color where the animation color is less than the RGB threshold. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillThresholdColors(
    int animationId, int frameId, int threshold, int color);

// Class Plugin
ChromaAnimationAPI::FillThresholdColors(
    int animationId, int frameId, int threshold, int color);
```

PluginFillThresholdColorsAllFrames

Fill all frames with RGB color where the animation color is less than the RGB threshold. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillThresholdColorsAllFrames(
    int animationId, int threshold, int color);

// Class Plugin
ChromaAnimationAPI::FillThresholdColorsAllFrames(
    int animationId, int threshold, int color);
```

PluginFillThresholdColorsAllFramesName

Fill all frames with RGB color where the animation color is less than the RGB threshold. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillThresholdColorsAllFramesName(
    const char* path, int threshold, int color);

// Class Plugin
ChromaAnimationAPI::FillThresholdColorsAllFramesName(
    const char* path, int threshold, int color);
```

PluginFillThresholdColorsAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillThresholdColorsAllFramesNameD(
    const char* path, double threshold, double color);

// Class Plugin
```

```
double result = ChromaAnimationAPI::FillThresholdColorsAllFramesNameD(
    const char* path, double threshold, double color);
```

PluginFillThresholdColorsAllFramesRGB

Fill all frames with RGB color where the animation color is less than the threshold. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillThresholdColorsAllFramesRGB(
    int animationId, int threshold, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillThresholdColorsAllFramesRGB(
    int animationId, int threshold, int red, int green, int blue);
```

PluginFillThresholdColorsAllFramesRGBName

Fill all frames with RGB color where the animation color is less than the threshold. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillThresholdColorsAllFramesRGBName(
    const char* path, int threshold, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillThresholdColorsAllFramesRGBName(
    const char* path, int threshold, int red, int green, int blue);
```

PluginFillThresholdColorsAllFramesRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillThresholdColorsAllFramesRGBNameD(
    const char* path, double threshold, double red, double green, double blue);

// Class Plugin
double result = ChromaAnimationAPI::FillThresholdColorsAllFramesRGBNameD(
    const char* path, double threshold, double red, double green, double blue);
```

PluginFillThresholdColorsMinMaxAllFramesRGB

Fill all frames with the min RGB color where the animation color is less than the min threshold AND with the max RGB color where the animation is more than the max threshold. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillThresholdColorsMinMaxAllFramesRGB(
    int animationId, int minThreshold, int minRed, int minGreen, int minBlue,
    int maxThreshold, int maxRed, int maxGreen, int maxBlue);

// Class Plugin
ChromaAnimationAPI::FillThresholdColorsMinMaxAllFramesRGB(
    int animationId, int minThreshold, int minRed, int minGreen, int minBlue,
    int maxThreshold, int maxRed, int maxGreen, int maxBlue);
```

PluginFillThresholdColorsMinMaxAllFramesRGBName

Fill all frames with the min RGB color where the animation color is less than the min threshold AND with the max RGB color where the animation is more than the max threshold. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillThresholdColorsMinMaxAllFramesRGBName(
    const char* path, int minThreshold, int minRed, int minGreen, int minBlue,
    int maxThreshold, int maxRed, int maxGreen, int maxBlue);

// Class Plugin
ChromaAnimationAPI::FillThresholdColorsMinMaxAllFramesRGBName(
    const char* path, int minThreshold, int minRed, int minGreen, int minBlue,
    int maxThreshold, int maxRed, int maxGreen, int maxBlue);
```

PluginFillThresholdColorsMinMaxAllFramesRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillThresholdColorsMinMaxAllFramesRGBNameD(
    const char* path, double minThreshold, double minRed, double minGreen, double
minBlue,
    double maxThreshold, double maxRed, double maxGreen, double maxBlue);

// Class Plugin
double result = ChromaAnimationAPI::FillThresholdColorsMinMaxAllFramesRGBNameD(
    const char* path, double minThreshold, double minRed, double minGreen, double
minBlue,
    double maxThreshold, double maxRed, double maxGreen, double maxBlue);
```

PluginFillThresholdColorsMinMaxRGB

Fill the specified frame with the min RGB color where the animation color is less than the min threshold AND with the max RGB color where the animation is more than the max threshold. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillThresholdColorsMinMaxRGB(
    int animationId, int frameId, int minThreshold, int minRed, int minGreen,
    int minBlue, int maxThreshold, int maxRed, int maxGreen, int maxBlue);

// Class Plugin
ChromaAnimationAPI::FillThresholdColorsMinMaxRGB(
    int animationId, int frameId, int minThreshold, int minRed, int minGreen,
    int minBlue, int maxThreshold, int maxRed, int maxGreen, int maxBlue);
```

PluginFillThresholdColorsMinMaxRGBName

Fill the specified frame with the min RGB color where the animation color is less than the min threshold AND with the max RGB color where the animation is more than the max threshold. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillThresholdColorsMinMaxRGBName(
    const char* path, int frameId, int minThreshold, int minRed, int minGreen,
    int minBlue, int maxThreshold, int maxRed, int maxGreen, int maxBlue);

// Class Plugin
ChromaAnimationAPI::FillThresholdColorsMinMaxRGBName(
    const char* path, int frameId, int minThreshold, int minRed, int minGreen,
    int minBlue, int maxThreshold, int maxRed, int maxGreen, int maxBlue);
```

PluginFillThresholdColorsMinMaxRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillThresholdColorsMinMaxRGBNameD(
    const char* path, double frameId, double minThreshold, double minRed, double
minGreen,
    double minBlue, double maxThreshold, double maxRed, double maxGreen, double
maxBlue);

// Class Plugin
double result = ChromaAnimationAPI::FillThresholdColorsMinMaxRGBNameD(
    const char* path, double frameId, double minThreshold, double minRed, double
minGreen,
```

```
    double minBlue, double maxThreshold, double maxRed, double maxGreen, double
maxBlue);
```

PluginFillThresholdColorsName

Fill the specified frame with RGB color where the animation color is less than the RGB threshold. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillThresholdColorsName(
    const char* path, int frameId, int threshold, int color);

// Class Plugin
ChromaAnimationAPI::FillThresholdColorsName(
    const char* path, int frameId, int threshold, int color);
```

PluginFillThresholdColorsNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillThresholdColorsNameD(
    const char* path, double frameId, double threshold, double color);

// Class Plugin
double result = ChromaAnimationAPI::FillThresholdColorsNameD(
    const char* path, double frameId, double threshold, double color);
```

PluginFillThresholdColorsRGB

Fill the specified frame with RGB color where the animation color is less than the RGB threshold. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillThresholdColorsRGB(
    int animationId, int frameId, int threshold, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillThresholdColorsRGB(
    int animationId, int frameId, int threshold, int red, int green, int blue);
```

PluginFillThresholdColorsRGBName

Fill the specified frame with RGB color where the animation color is less than the RGB threshold. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillThresholdColorsRGBName(
    const char* path, int frameId, int threshold, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillThresholdColorsRGBName(
    const char* path, int frameId, int threshold, int red, int green, int blue);
```

PluginFillThresholdColorsRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillThresholdColorsRGBNameD(
    const char* path, double frameId, double threshold, double red, double green,
    double blue);

// Class Plugin
double result = ChromaAnimationAPI::FillThresholdColorsRGBNameD(
    const char* path, double frameId, double threshold, double red, double green,
    double blue);
```

PluginFillThresholdRGBColorsAllFramesRGB

Fill all frames with RGB color where the animation color is less than the RGB threshold. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillThresholdRGBColorsAllFramesRGB(
    int animationId, int redThreshold, int greenThreshold, int blueThreshold,
    int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillThresholdRGBColorsAllFramesRGB(
    int animationId, int redThreshold, int greenThreshold, int blueThreshold,
    int red, int green, int blue);
```

PluginFillThresholdRGBColorsAllFramesRGBName

Fill all frames with RGB color where the animation color is less than the RGB threshold. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillThresholdRGBColorsAllFramesRGBName(
    const char* path, int redThreshold, int greenThreshold, int blueThreshold,
    int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillThresholdRGBColorsAllFramesRGBName(
    const char* path, int redThreshold, int greenThreshold, int blueThreshold,
    int red, int green, int blue);
```

PluginFillThresholdRGBColorsAllFramesRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillThresholdRGBColorsAllFramesRGBNameD(
    const char* path, double redThreshold, double greenThreshold, double
blueThreshold,
    double red, double green, double blue);

// Class Plugin
double result = ChromaAnimationAPI::FillThresholdRGBColorsAllFramesRGBNameD(
    const char* path, double redThreshold, double greenThreshold, double
blueThreshold,
    double red, double green, double blue);
```

PluginFillThresholdRGBColorsRGB

Fill the specified frame with RGB color where the animation color is less than the RGB threshold. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillThresholdRGBColorsRGB(
    int animationId, int frameId, int redThreshold, int greenThreshold, int
blueThreshold,
    int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillThresholdRGBColorsRGB(
    int animationId, int frameId, int redThreshold, int greenThreshold, int
blueThreshold,
    int red, int green, int blue);
```

PluginFillThresholdRGBColorsRGBName

Fill the specified frame with RGB color where the animation color is less than the RGB threshold. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillThresholdRGBColorsRGBName(
    const char* path, int frameId, int redThreshold, int greenThreshold, int
blueThreshold,
    int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillThresholdRGBColorsRGBName(
    const char* path, int frameId, int redThreshold, int greenThreshold, int
blueThreshold,
    int red, int green, int blue);
```

PluginFillThresholdRGBColorsRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillThresholdRGBColorsRGBNameD(
    const char* path, double frameId, double redThreshold, double greenThreshold,
    double blueThreshold, double red, double green, double blue);

// Class Plugin
double result = ChromaAnimationAPI::FillThresholdRGBColorsRGBNameD(
    const char* path, double frameId, double redThreshold, double greenThreshold,
    double blueThreshold, double red, double green, double blue);
```

PluginFillZeroColor

Fill the specified frame with RGB color where the animation color is zero. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillZeroColor(
    int animationId, int frameId, int color);

// Class Plugin
ChromaAnimationAPI::FillZeroColor(
    int animationId, int frameId, int color);
```

PluginFillZeroColorAllFrames

Fill all frames with RGB color where the animation color is zero. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillZeroColorAllFrames(
    int animationId, int color);

// Class Plugin
ChromaAnimationAPI::FillZeroColorAllFrames(
    int animationId, int color);
```

PluginFillZeroColorAllFramesName

Fill all frames with RGB color where the animation color is zero. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillZeroColorAllFramesName(
    const char* path, int color);

// Class Plugin
ChromaAnimationAPI::FillZeroColorAllFramesName(
    const char* path, int color);
```

PluginFillZeroColorAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillZeroColorAllFramesNameD(
    const char* path, double color);

// Class Plugin
double result = ChromaAnimationAPI::FillZeroColorAllFramesNameD(
    const char* path, double color);
```

PluginFillZeroColorAllFramesRGB

Fill all frames with RGB color where the animation color is zero. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillZeroColorAllFramesRGB(
    int animationId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillZeroColorAllFramesRGB(
    int animationId, int red, int green, int blue);
```

PluginFillZeroColorAllFramesRGBName

Fill all frames with RGB color where the animation color is zero. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillZeroColorAllFramesRGBName(
    const char* path, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillZeroColorAllFramesRGBName(
    const char* path, int red, int green, int blue);
```

PluginFillZeroColorAllFramesRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillZeroColorAllFramesRGBNameD(
    const char* path, double red, double green, double blue);

// Class Plugin
double result = ChromaAnimationAPI::FillZeroColorAllFramesRGBNameD(
    const char* path, double red, double green, double blue);
```

PluginFillZeroColorName

Fill the specified frame with RGB color where the animation color is zero. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillZeroColorName(
    const char* path, int frameId, int color);

// Class Plugin
ChromaAnimationAPI::FillZeroColorName(
    const char* path, int frameId, int color);
```

PluginFillZeroColorNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillZeroColorNameD(
```

```
    const char* path, double frameId, double color);

// Class Plugin
double result = ChromaAnimationAPI::FillZeroColorNameD(
    const char* path, double frameId, double color);
```

PluginFillZeroColorRGB

Fill the specified frame with RGB color where the animation color is zero. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginFillZeroColorRGB(
    int animationId, int frameId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillZeroColorRGB(
    int animationId, int frameId, int red, int green, int blue);
```

PluginFillZeroColorRGBName

Fill the specified frame with RGB color where the animation color is zero. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginFillZeroColorRGBName(
    const char* path, int frameId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::FillZeroColorRGBName(
    const char* path, int frameId, int red, int green, int blue);
```

PluginFillZeroColorRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginFillZeroColorRGBNameD(
    const char* path, double frameId, double red, double green, double blue);

// Class Plugin
double result = ChromaAnimationAPI::FillZeroColorRGBNameD(
    const char* path, double frameId, double red, double green, double blue);
```

PluginGet1DColor

Get the animation color for a frame given the `1D led`. The `led` should be greater than or equal to 0 and less than the `MaxLeds`. Animation is referenced by id.

```
// DLL Interface
EXPORT_API int PluginGet1DColor(
    int animationId, int frameId, int led);

// Class Plugin
int result = ChromaAnimationAPI::Get1DColor(
    int animationId, int frameId, int led);
```

PluginGet1DColorName

Get the animation color for a frame given the `1D led`. The `led` should be greater than or equal to 0 and less than the `MaxLeds`. Animation is referenced by name.

```
// DLL Interface
EXPORT_API int PluginGet1DColorName(
    const char* path, int frameId, int led);

// Class Plugin
int result = ChromaAnimationAPI::Get1DColorName(
    const char* path, int frameId, int led);
```

PluginGet1DColorNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginGet1DColorNameD(
    const char* path, double frameId, double led);

// Class Plugin
double result = ChromaAnimationAPI::Get1DColorNameD(
    const char* path, double frameId, double led);
```

PluginGet2DColor

Get the animation color for a frame given the `2D row` and `column`. The `row` should be greater than or equal to 0 and less than the `MaxRow`. The `column` should be greater than or equal to 0 and less than the `MaxColumn`. Animation is referenced by id.

```
// DLL Interface
EXPORT_API int PluginGet2DColor(
    int animationId, int frameId, int row, int column);

// Class Plugin
int result = ChromaAnimationAPI::Get2DColor(
    int animationId, int frameId, int row, int column);
```

PluginGet2DColorName

Get the animation color for a frame given the `2D row` and `column`. The `row` should be greater than or equal to 0 and less than the `MaxRow`. The `column` should be greater than or equal to 0 and less than the `MaxColumn`. Animation is referenced by name.

```
// DLL Interface
EXPORT_API int PluginGet2DColorName(
    const char* path, int frameId, int row, int column);

// Class Plugin
int result = ChromaAnimationAPI::Get2DColorName(
    const char* path, int frameId, int row, int column);
```

PluginGet2DColorNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginGet2DColorNameD(
    const char* path, double frameId, double row, double column);

// Class Plugin
double result = ChromaAnimationAPI::Get2DColorNameD(
    const char* path, double frameId, double row, double column);
```

PluginGetAnimation

Get the animation id for the named animation.

```
// DLL Interface
EXPORT_API int PluginGetAnimation(const char* name);

// Class Plugin
int result = ChromaAnimationAPI::GetAnimation(const char* name);
```

PluginGetAnimationCount

`PluginGetAnimationCount` will return the number of loaded animations.

```
// DLL Interface
EXPORT_API int PluginGetAnimationCount();

// Class Plugin
int result = ChromaAnimationAPI::GetAnimationCount();
```

PluginGetAnimationD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginGetAnimationD(const char* name);

// Class Plugin
double result = ChromaAnimationAPI::GetAnimationD(const char* name);
```

PluginGetAnimationId

`PluginGetAnimationId` will return the `animationId` given the `index` of the loaded animation. The `index` is zero-based and less than the number returned by `PluginGetAnimationCount`. Use `PluginGetAnimationName` to get the name of the animation.

```
// DLL Interface
EXPORT_API int PluginGetAnimationId(int index);

// Class Plugin
int result = ChromaAnimationAPI::GetAnimationId(int index);
```

PluginGetAnimationName

`PluginGetAnimationName` takes an `animationId` and returns the name of the animation of the `.chroma` animation file. If a name is not available then an empty string will be returned.

```
// DLL Interface
EXPORT_API const char* PluginGetAnimationName(int animationId);
```

```
// Class Plugin
const char* result = ChromaAnimationAPI::GetAnimationName(int animationId);
```

PluginGetCurrentFrame

Get the current frame of the animation referenced by id.

```
// DLL Interface
EXPORT_API int PluginGetCurrentFrame(int animationId);

// Class Plugin
int result = ChromaAnimationAPI::GetCurrentFrame(int animationId);
```

PluginGetCurrentFrameName

Get the current frame of the animation referenced by name.

```
// DLL Interface
EXPORT_API int PluginGetCurrentFrameName(const char* path);

// Class Plugin
int result = ChromaAnimationAPI::GetCurrentFrameName(const char* path);
```

PluginGetCurrentFrameNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginGetCurrentFrameNameD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::GetCurrentFrameNameD(const char* path);
```

PluginGetDevice

Returns the `EChromaSDKDevice1DEnum` or `EChromaSDKDevice2DEnum` of a `Chroma` animation respective to the `deviceType`, as an integer upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginGetDevice(int animationId);
```

```
// Class Plugin
int result = ChromaAnimationAPI::GetDevice(int animationId);
```

PluginGetDeviceName

Returns the `EChromaSDKDevice1DEnum` or `EChromaSDKDevice2DEnum` of a `Chroma` animation respective to the `deviceType`, as an integer upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginGetDeviceName(const char* path);

// Class Plugin
int result = ChromaAnimationAPI::GetDeviceName(const char* path);
```

PluginGetDeviceNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginGetDeviceNameD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::GetDeviceNameD(const char* path);
```

PluginGetDeviceType

Returns the `EChromaSDKDeviceTypeEnum` of a `Chroma` animation as an integer upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginGetDeviceType(int animationId);

// Class Plugin
int result = ChromaAnimationAPI::GetDeviceType(int animationId);
```

PluginGetDeviceTypeName

Returns the `EChromaSDKDeviceTypeEnum` of a `Chroma` animation as an integer upon success. Returns negative one upon failure.

```

```

```
// DLL Interface
EXPORT_API int PluginGetDeviceTypeName(const char* path);

// Class Plugin
int result = ChromaAnimationAPI::GetDeviceTypeName(const char* path);
```

PluginGetDeviceTypeNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginGetDeviceTypeNameD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::GetDeviceTypeNameD(const char* path);
```

PluginGetFrame

Get the frame colors and duration (in seconds) for a `Chroma` animation referenced by id. The `color` is expected to be an array of the expected dimensions for the `deviceType/device`. The `length` parameter is the size of the `color` array. For `EChromaSDKDevice1DEnum` the array size should be `MAX_LEDS`. For `EChromaSDKDevice2DEnum` the array size should be `MAX_ROW` times `MAX_COLUMN`. Keys are populated only for EChromaSDKDevice2DEnum::DE_Keyboard and EChromaSDKDevice2DEnum::DE_KeyboardExtended. Keys will only use the EChromaSDKDevice2DEnum::DE_Keyboard `MAX_ROW` times `MAX_COLUMN` keysLength. Returns the animation id upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginGetFrame(
    int animationId, int frameId, float* duration, int* colors, int length, int*
keys,
    int keysLength);

// Class Plugin
int result = ChromaAnimationAPI::GetFrame(
    int animationId, int frameId, float* duration, int* colors, int length, int*
keys,
    int keysLength);
```

PluginGetFrameCount

Returns the frame count of a `Chroma` animation upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginGetFrameCount(int animationId);

// Class Plugin
int result = ChromaAnimationAPI::GetFrameCount(int animationId);
```

PluginGetFrameCountName

Returns the frame count of a `Chroma` animation upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginGetFrameCountName(const char* path);

// Class Plugin
int result = ChromaAnimationAPI::GetFrameCountName(const char* path);
```

PluginGetFrameCountNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginGetFrameCountNameD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::GetFrameCountNameD(const char* path);
```

PluginGetFrameDuration

Returns the duration of an animation frame in seconds upon success. Returns zero upon failure.

```
// DLL Interface
EXPORT_API float PluginGetFrameDuration(
    int animationId, int frameId);

// Class Plugin
float result = ChromaAnimationAPI::GetFrameDuration(
    int animationId, int frameId);
```

PluginGetFrameDurationName

Returns the duration of an animation frame in seconds upon success. Returns zero upon failure.

```
// DLL Interface
EXPORT_API float PluginGetFrameDurationName(
    const char* path, int frameId);

// Class Plugin
float result = ChromaAnimationAPI::GetFrameDurationName(
    const char* path, int frameId);
```

PluginGetFrameName

Get the frame colors and duration (in seconds) for a `Chroma` animation referenced by name. The `color` is expected to be an array of the expected dimensions for the `deviceType/device`. The `length` parameter is the size of the `color` array. For `EChromaSDKDevice1DEnum` the array size should be `MAX LEDS`. For `EChromaSDKDevice2DEnum` the array size should be `MAX ROW` times `MAX COLUMN`. Keys are populated only for EChromaSDKDevice2DEnum::DE_Keyboard and EChromaSDKDevice2DEnum::DE_KeyboardExtended. Keys will only use the EChromaSDKDevice2DEnum::DE_Keyboard `MAX_ROW` times `MAX_COLUMN` keysLength. Returns the animation id upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginGetFrameName(
    const char* path, int frameId, float* duration, int* colors, int length,
    int* keys, int keysLength);

// Class Plugin
int result = ChromaAnimationAPI::GetFrameName(
    const char* path, int frameId, float* duration, int* colors, int length,
    int* keys, int keysLength);
```

PluginGetKeyColor

Get the color of an animation key for the given frame referenced by id.

```
// DLL Interface
EXPORT_API int PluginGetKeyColor(
    int animationId, int frameId, int rzkey);

// Class Plugin
int result = ChromaAnimationAPI::GetKeyColor(
    int animationId, int frameId, int rzkey);
```

PluginGetKeyColorD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginGetKeyColorD(
    const char* path, double frameId, double rzkey);

// Class Plugin
double result = ChromaAnimationAPI::GetKeyColorD(
    const char* path, double frameId, double rzkey);
```

PluginGetKeyColorName

Get the color of an animation key for the given frame referenced by name.

```
// DLL Interface
EXPORT_API int PluginGetKeyColorName(
    const char* path, int frameId, int rzkey);

// Class Plugin
int result = ChromaAnimationAPI::GetKeyColorName(
    const char* path, int frameId, int rzkey);
```

PluginGetLibraryLoadedState

Returns `RZRESULT_SUCCESS` if the plugin has been initialized successfully. Returns `RZRESULT_DLL_NOT_FOUND` if core Chroma library is not found. Returns `RZRESULT_DLL_INVALID_SIGNATURE` if core Chroma library has an invalid signature.

```
// DLL Interface
EXPORT_API RZRESULT PluginGetLibraryLoadedState();

// Class Plugin
RZRESULT result = ChromaAnimationAPI::GetLibraryLoadedState();
```

PluginGetLibraryLoadedStateD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginGetLibraryLoadedStateD();

// Class Plugin
double result = ChromaAnimationAPI::GetLibraryLoadedStateD();
```

PluginGetMaxColumn

Returns the `MAX COLUMN` given the `EChromaSDKDevice2DEnum` device as an integer upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginGetMaxColumn(int device);

// Class Plugin
int result = ChromaAnimationAPI::GetMaxColumn(int device);
```

PluginGetMaxColumnD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginGetMaxColumnD(double device);

// Class Plugin
double result = ChromaAnimationAPI::GetMaxColumnD(double device);
```

PluginGetMaxLeds

Returns the MAX LEDS given the `EChromaSDKDevice1DEnum` device as an integer upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginGetMaxLeds(int device);

// Class Plugin
int result = ChromaAnimationAPI::GetMaxLeds(int device);
```

PluginGetMaxLedsD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginGetMaxLedsD(double device);

// Class Plugin
double result = ChromaAnimationAPI::GetMaxLedsD(double device);
```

PluginGetMaxRow

Returns the `MAX ROW` given the `EChromaSDKDevice2DEnum` device as an integer upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginGetMaxRow(int device);

// Class Plugin
int result = ChromaAnimationAPI::GetMaxRow(int device);
```

PluginGetMaxRowD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginGetMaxRowD(double device);

// Class Plugin
double result = ChromaAnimationAPI::GetMaxRowD(double device);
```

PluginGetPlayingAnimationCount

`PluginGetPlayingAnimationCount` will return the number of playing animations.

```
// DLL Interface
EXPORT_API int PluginGetPlayingAnimationCount();

// Class Plugin
int result = ChromaAnimationAPI::GetPlayingAnimationCount();
```

PluginGetPlayingAnimationId

`PluginGetPlayingAnimationId` will return the `animationId` given the `index` of the playing animation. The `index` is zero-based and less than the number returned by `PluginGetPlayingAnimationCount`. Use `PluginGetAnimationName` to get the name of the animation.

```
// DLL Interface
EXPORT_API int PluginGetPlayingAnimationId(int index);

// Class Plugin
int result = ChromaAnimationAPI::GetPlayingAnimationId(int index);
```

PluginGetRGB

Get the RGB color given red, green, and blue.

```
// DLL Interface
EXPORT_API int PluginGetRGB(
    int red, int green, int blue);

// Class Plugin
int result = ChromaAnimationAPI::GetRGB(
    int red, int green, int blue);
```

PluginGetRGBD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginGetRGBD(
    double red, double green, double blue);

// Class Plugin
double result = ChromaAnimationAPI::GetRGBD(
    double red, double green, double blue);
```

PluginGetTotalDuration

Returns the total duration of an animation in seconds upon success. Returns zero upon failure.

```
// DLL Interface
EXPORT_API float PluginGetTotalDuration(int animationId);

// Class Plugin
float result = ChromaAnimationAPI::GetTotalDuration(int animationId);
```

PluginGetTotalDurationName

Returns the total duration of an animation in seconds upon success. Returns zero upon failure.

```
// DLL Interface
EXPORT_API float PluginGetTotalDurationName(const char* path);

// Class Plugin
float result = ChromaAnimationAPI::GetTotalDurationName(const char* path);
```

```

```

PluginHasAnimationLoop

Check if the animation has loop enabled referenced by id.

```
// DLL Interface
EXPORT_API bool PluginHasAnimationLoop(int animationId);

// Class Plugin
bool result = ChromaAnimationAPI::HasAnimationLoop(int animationId);
```

PluginHasAnimationLoopName

Check if the animation has loop enabled referenced by name.

```
// DLL Interface
EXPORT_API bool PluginHasAnimationLoopName(const char* path);

// Class Plugin
bool result = ChromaAnimationAPI::HasAnimationLoopName(const char* path);
```

PluginHasAnimationLoopNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginHasAnimationLoopNameD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::HasAnimationLoopNameD(const char* path);
```

PluginInit

Initialize the ChromaSDK. Zero indicates success, otherwise failure. Many API methods auto initialize the ChromaSDK if not already initialized.

```
// DLL Interface
EXPORT_API RZRESULT PluginInit();

// Class Plugin
RZRESULT result = ChromaAnimationAPI::Init();
```

PluginInitD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginInitD();

// Class Plugin
double result = ChromaAnimationAPI::InitD();
```

PluginInitSDK

Initialize the ChromaSDK. AppInfo populates the details in Synapse. Zero indicates success, otherwise failure. Many API methods auto initialize the ChromaSDK if not already initialized.

```
// DLL Interface
EXPORT_API RZRESULT PluginInitSDK(
    ChromaSDK::APPINFOTYPE* AppInfo);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::InitSDK(
    ChromaSDK::APPINFOTYPE* AppInfo);
```

PluginInsertDelay

Insert an animation delay by duplicating the frame by the delay number of times. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginInsertDelay(
    int animationId, int frameId, int delay);

// Class Plugin
ChromaAnimationAPI::InsertDelay(
    int animationId, int frameId, int delay);
```

PluginInsertDelayName

Insert an animation delay by duplicating the frame by the delay number of times. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginInsertDelayName(
    const char* path, int frameId, int delay);

// Class Plugin
ChromaAnimationAPI::InsertDelayName(
    const char* path, int frameId, int delay);
```

PluginInsertDelayNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginInsertDelayNameD(
    const char* path, double frameId, double delay);

// Class Plugin
double result = ChromaAnimationAPI::InsertDelayNameD(
    const char* path, double frameId, double delay);
```

PluginInsertFrame

Duplicate the source frame index at the target frame index. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginInsertFrame(
    int animationId, int sourceFrame, int targetFrame);

// Class Plugin
ChromaAnimationAPI::InsertFrame(
    int animationId, int sourceFrame, int targetFrame);
```

PluginInsertFrameName

Duplicate the source frame index at the target frame index. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginInsertFrameName(
    const char* path, int sourceFrame, int targetFrame);

// Class Plugin
ChromaAnimationAPI::InsertFrameName(
    const char* path, int sourceFrame, int targetFrame);
```

PluginInsertFrameNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginInsertFrameNameD(
    const char* path, double sourceFrame, double targetFrame);

// Class Plugin
double result = ChromaAnimationAPI::InsertFrameNameD(
    const char* path, double sourceFrame, double targetFrame);
```

PluginInvertColors

Invert all the colors at the specified frame. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginInvertColors(
    int animationId, int frameId);

// Class Plugin
ChromaAnimationAPI::InvertColors(
    int animationId, int frameId);
```

PluginInvertColorsAllFrames

Invert all the colors for all frames. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginInvertColorsAllFrames(int animationId);

// Class Plugin
ChromaAnimationAPI::InvertColorsAllFrames(int animationId);
```

PluginInvertColorsAllFramesName

Invert all the colors for all frames. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginInvertColorsAllFramesName(const char* path);

// Class Plugin
ChromaAnimationAPI::InvertColorsAllFramesName(const char* path);
```

```

```

PluginInvertColorsAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginInvertColorsAllFramesNameD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::InvertColorsAllFramesNameD(const char* path);
```

PluginInvertColorsName

Invert all the colors at the specified frame. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginInvertColorsName(
    const char* path, int frameId);

// Class Plugin
ChromaAnimationAPI::InvertColorsName(
    const char* path, int frameId);
```

PluginInvertColorsNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginInvertColorsNameD(
    const char* path, double frameId);

// Class Plugin
double result = ChromaAnimationAPI::InvertColorsNameD(
    const char* path, double frameId);
```

PluginIsAnimationPaused

Check if the animation is paused referenced by id.

```
// DLL Interface
EXPORT_API bool PluginIsAnimationPaused(int animationId);
```

```
// Class Plugin
bool result = ChromaAnimationAPI::IsAnimationPaused(int animationId);
```

PluginIsAnimationPausedName

Check if the animation is paused referenced by name.

```
// DLL Interface
EXPORT_API bool PluginIsAnimationPausedName(const char* path);

// Class Plugin
bool result = ChromaAnimationAPI::IsAnimationPausedName(const char* path);
```

PluginIsAnimationPausedNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginIsAnimationPausedNameD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::IsAnimationPausedNameD(const char* path);
```

PluginIsDialogOpen

The editor dialog is a non-blocking modal window, this method returns true if the modal window is open, otherwise false.

```
// DLL Interface
EXPORT_API bool PluginIsDialogOpen();

// Class Plugin
bool result = ChromaAnimationAPI::IsDialogOpen();
```

PluginIsDialogOpenD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginIsDialogOpenD();
```

```
// Class Plugin
double result = ChromaAnimationAPI::IsDialogOpenD();
```

PluginIsInitialized

Returns true if the plugin has been initialized. Returns false if the plugin is uninitialized.

```
// DLL Interface
EXPORT_API bool PluginIsInitialized();

// Class Plugin
bool result = ChromaAnimationAPI::IsInitialized();
```

PluginIsInitializedD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginIsInitializedD();

// Class Plugin
double result = ChromaAnimationAPI::IsInitializedD();
```

PluginIsPlatformSupported

If the method can be invoked the method returns true.

```
// DLL Interface
EXPORT_API bool PluginIsPlatformSupported();

// Class Plugin
bool result = ChromaAnimationAPI::IsPlatformSupported();
```

PluginIsPlatformSupportedD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginIsPlatformSupportedD();

// Class Plugin
double result = ChromaAnimationAPI::IsPlatformSupportedD();
```

PluginIsPlaying

`PluginIsPlayingName` automatically handles initializing the `ChromaSDK`. The named `.chroma` animation file will be automatically opened. The method will return whether the animation is playing or not. Animation is referenced by id.

```
// DLL Interface
EXPORT_API bool PluginIsPlaying(int animationId);

// Class Plugin
bool result = ChromaAnimationAPI::IsPlaying(int animationId);
```

PluginIsPlayingD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginIsPlayingD(double animationId);

// Class Plugin
double result = ChromaAnimationAPI::IsPlayingD(double animationId);
```

PluginIsPlayingName

`PluginIsPlayingName` automatically handles initializing the `ChromaSDK`. The named `.chroma` animation file will be automatically opened. The method will return whether the animation is playing or not. Animation is referenced by name.

```
// DLL Interface
EXPORT_API bool PluginIsPlayingName(const char* path);

// Class Plugin
bool result = ChromaAnimationAPI::IsPlayingName(const char* path);
```

PluginIsPlayingNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginIsPlayingNameD(const char* path);
```

```
// Class Plugin
double result = ChromaAnimationAPI::IsPlayingNameD(const char* path);
```

PluginIsPlayingType

`PluginIsPlayingType` automatically handles initializing the `ChromaSDK`. If any animation is playing for the `deviceType` and `device` combination, the method will return true, otherwise false.

```
// DLL Interface
EXPORT_API bool PluginIsPlayingType(
    int deviceType, int device);

// Class Plugin
bool result = ChromaAnimationAPI::IsPlayingType(
    int deviceType, int device);
```

PluginIsPlayingTypeD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginIsPlayingTypeD(
    double deviceType, double device);

// Class Plugin
double result = ChromaAnimationAPI::IsPlayingTypeD(
    double deviceType, double device);
```

PluginLerp

Do a lerp math operation on a float.

```
// DLL Interface
EXPORT_API float PluginLerp(
    float start, float end, float amt);

// Class Plugin
float result = ChromaAnimationAPI::Lerp(
    float start, float end, float amt);
```

PluginLerpColor

Lerp from one color to another given t in the range 0.0 to 1.0.

```
// DLL Interface
EXPORT_API int PluginLerpColor(
    int from, int to, float t);

// Class Plugin
int result = ChromaAnimationAPI::LerpColor(
    int from, int to, float t);
```

PluginLoadAnimation

Loads `Chroma` effects so that the animation can be played immediately. Returns the animation id upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginLoadAnimation(int animationId);

// Class Plugin
int result = ChromaAnimationAPI::LoadAnimation(int animationId);
```

PluginLoadAnimationD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginLoadAnimationD(double animationId);

// Class Plugin
double result = ChromaAnimationAPI::LoadAnimationD(double animationId);
```

PluginLoadAnimationName

Load the named animation.

```
// DLL Interface
EXPORT_API void PluginLoadAnimationName(const char* path);

// Class Plugin
ChromaAnimationAPI::LoadAnimationName(const char* path);
```

PluginLoadComposite

Load a composite set of animations.

```
// DLL Interface
EXPORT_API void PluginLoadComposite(const char* name);

// Class Plugin
ChromaAnimationAPI::LoadComposite(const char* name);
```

PluginMakeBlankFrames

Make a blank animation for the length of the frame count. Frame duration defaults to the duration. The frame color defaults to color. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginMakeBlankFrames(
    int animationId, int frameCount, float duration, int color);

// Class Plugin
ChromaAnimationAPI::MakeBlankFrames(
    int animationId, int frameCount, float duration, int color);
```

PluginMakeBlankFramesName

Make a blank animation for the length of the frame count. Frame duration defaults to the duration. The frame color defaults to color. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginMakeBlankFramesName(
    const char* path, int frameCount, float duration, int color);

// Class Plugin
ChromaAnimationAPI::MakeBlankFramesName(
    const char* path, int frameCount, float duration, int color);
```

PluginMakeBlankFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginMakeBlankFramesNameD(
    const char* path, double frameCount, double duration, double color);
```

```
// Class Plugin
double result = ChromaAnimationAPI::MakeBlankFramesNameD(
    const char* path, double frameCount, double duration, double color);
```

PluginMakeBlankFramesRandom

Make a blank animation for the length of the frame count. Frame duration defaults to the duration. The frame color is random. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginMakeBlankFramesRandom(
    int animationId, int frameCount, float duration);

// Class Plugin
ChromaAnimationAPI::MakeBlankFramesRandom(
    int animationId, int frameCount, float duration);
```

PluginMakeBlankFramesRandomBlackAndWhite

Make a blank animation for the length of the frame count. Frame duration defaults to the duration. The frame color is random black and white. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginMakeBlankFramesRandomBlackAndWhite(
    int animationId, int frameCount, float duration);

// Class Plugin
ChromaAnimationAPI::MakeBlankFramesRandomBlackAndWhite(
    int animationId, int frameCount, float duration);
```

PluginMakeBlankFramesRandomBlackAndWhiteName

Make a blank animation for the length of the frame count. Frame duration defaults to the duration. The frame color is random black and white. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginMakeBlankFramesRandomBlackAndWhiteName(
    const char* path, int frameCount, float duration);

// Class Plugin
ChromaAnimationAPI::MakeBlankFramesRandomBlackAndWhiteName(
    const char* path, int frameCount, float duration);
```

PluginMakeBlankFramesRandomBlackAndWhiteNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginMakeBlankFramesRandomBlackAndWhiteNameD(
    const char* path, double frameCount, double duration);

// Class Plugin
double result = ChromaAnimationAPI::MakeBlankFramesRandomBlackAndWhiteNameD(
    const char* path, double frameCount, double duration);
```

PluginMakeBlankFramesRandomName

Make a blank animation for the length of the frame count. Frame duration defaults to the duration. The frame color is random. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginMakeBlankFramesRandomName(
    const char* path, int frameCount, float duration);

// Class Plugin
ChromaAnimationAPI::MakeBlankFramesRandomName(
    const char* path, int frameCount, float duration);
```

PluginMakeBlankFramesRandomNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginMakeBlankFramesRandomNameD(
    const char* path, double frameCount, double duration);

// Class Plugin
double result = ChromaAnimationAPI::MakeBlankFramesRandomNameD(
    const char* path, double frameCount, double duration);
```

PluginMakeBlankFramesRGB

Make a blank animation for the length of the frame count. Frame duration defaults to the duration. The frame color defaults to color. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginMakeBlankFramesRGB(
    int animationId, int frameCount, float duration, int red, int green, int
blue);

// Class Plugin
ChromaAnimationAPI::MakeBlankFramesRGB(
    int animationId, int frameCount, float duration, int red, int green, int
blue);
```

PluginMakeBlankFramesRGBName

Make a blank animation for the length of the frame count. Frame duration defaults to the duration. The frame color defaults to color. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginMakeBlankFramesRGBName(
    const char* path, int frameCount, float duration, int red, int green, int
blue);

// Class Plugin
ChromaAnimationAPI::MakeBlankFramesRGBName(
    const char* path, int frameCount, float duration, int red, int green, int
blue);
```

PluginMakeBlankFramesRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginMakeBlankFramesRGBNameD(
    const char* path, double frameCount, double duration, double red, double
green,
    double blue);

// Class Plugin
double result = ChromaAnimationAPI::MakeBlankFramesRGBNameD(
    const char* path, double frameCount, double duration, double red, double
green,
    double blue);
```

PluginMirrorHorizontally

Flips the color grid horizontally for all `Chroma` animation frames. Returns the animation id upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginMirrorHorizontally(int animationId);

// Class Plugin
int result = ChromaAnimationAPI::MirrorHorizontally(int animationId);
```

PluginMirrorVertically

Flips the color grid vertically for all `Chroma` animation frames. This method has no effect for `EChromaSDKDevice1DEnum` devices. Returns the animation id upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginMirrorVertically(int animationId);

// Class Plugin
int result = ChromaAnimationAPI::MirrorVertically(int animationId);
```

PluginMultiplyColorLerpAllFrames

Multiply the color intensity with the lerp result from color 1 to color 2 using the frame index divided by the frame count for the `t` parameter. Animation is referenced in id.

```
// DLL Interface
EXPORT_API void PluginMultiplyColorLerpAllFrames(
    int animationId, int color1, int color2);

// Class Plugin
ChromaAnimationAPI::MultiplyColorLerpAllFrames(
    int animationId, int color1, int color2);
```

PluginMultiplyColorLerpAllFramesName

Multiply the color intensity with the lerp result from color 1 to color 2 using the frame index divided by the frame count for the `t` parameter. Animation is referenced in name.

```
// DLL Interface
EXPORT_API void PluginMultiplyColorLerpAllFramesName(
    const char* path, int color1, int color2);

// Class Plugin
ChromaAnimationAPI::MultiplyColorLerpAllFramesName(
    const char* path, int color1, int color2);
```

PluginMultiplyColorLerpAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginMultiplyColorLerpAllFramesNameD(
    const char* path, double color1, double color2);

// Class Plugin
double result = ChromaAnimationAPI::MultiplyColorLerpAllFramesNameD(
    const char* path, double color1, double color2);
```

PluginMultiplyIntensity

Multiply all the colors in the frame by the intensity value. The valid the intensity range is from 0.0 to 255.0. RGB components are multiplied equally. An intensity of 0.5 would half the color value. Black colors in the frame will not be affected by this method.

```
// DLL Interface
EXPORT_API void PluginMultiplyIntensity(
    int animationId, int frameId, float intensity);

// Class Plugin
ChromaAnimationAPI::MultiplyIntensity(
    int animationId, int frameId, float intensity);
```

PluginMultiplyIntensityAllFrames

Multiply all the colors for all frames by the intensity value. The valid the intensity range is from 0.0 to 255.0. RGB components are multiplied equally. An intensity of 0.5 would half the color value. Black colors in the frame will not be affected by this method.

```
// DLL Interface
EXPORT_API void PluginMultiplyIntensityAllFrames(
    int animationId, float intensity);

// Class Plugin
ChromaAnimationAPI::MultiplyIntensityAllFrames(
    int animationId, float intensity);
```

PluginMultiplyIntensityAllFramesName

Multiply all the colors for all frames by the intensity value. The valid the intensity range is from 0.0 to 255.0. RGB components are multiplied equally. An intensity of 0.5 would half the color value. Black colors in the frame will not be affected by this method.

```
// DLL Interface
EXPORT_API void PluginMultiplyIntensityAllFramesName(
    const char* path, float intensity);

// Class Plugin
ChromaAnimationAPI::MultiplyIntensityAllFramesName(
    const char* path, float intensity);
```

PluginMultiplyIntensityAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginMultiplyIntensityAllFramesNameD(
    const char* path, double intensity);

// Class Plugin
double result = ChromaAnimationAPI::MultiplyIntensityAllFramesNameD(
    const char* path, double intensity);
```

PluginMultiplyIntensityAllFramesRGB

Multiply all frames by the RBG color intensity. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginMultiplyIntensityAllFramesRGB(
    int animationId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::MultiplyIntensityAllFramesRGB(
    int animationId, int red, int green, int blue);
```

PluginMultiplyIntensityAllFramesRGBName

Multiply all frames by the RBG color intensity. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginMultiplyIntensityAllFramesRGBName(
    const char* path, int red, int green, int blue);
```

```
// Class Plugin
ChromaAnimationAPI::MultiplyIntensityAllFramesRGBName(
    const char* path, int red, int green, int blue);
```

PluginMultiplyIntensityAllFramesRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginMultiplyIntensityAllFramesRGBNameD(
    const char* path, double red, double green, double blue);

// Class Plugin
double result = ChromaAnimationAPI::MultiplyIntensityAllFramesRGBNameD(
    const char* path, double red, double green, double blue);
```

PluginMultiplyIntensityColor

Multiply the specific frame by the RBG color intensity. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginMultiplyIntensityColor(
    int animationId, int frameId, int color);

// Class Plugin
ChromaAnimationAPI::MultiplyIntensityColor(
    int animationId, int frameId, int color);
```

PluginMultiplyIntensityColorAllFrames

Multiply all frames by the RBG color intensity. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginMultiplyIntensityColorAllFrames(
    int animationId, int color);

// Class Plugin
ChromaAnimationAPI::MultiplyIntensityColorAllFrames(
    int animationId, int color);
```

PluginMultiplyIntensityColorAllFramesName

Multiply all frames by the RBG color intensity. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginMultiplyIntensityColorAllFramesName(
    const char* path, int color);

// Class Plugin
ChromaAnimationAPI::MultiplyIntensityColorAllFramesName(
    const char* path, int color);
```

PluginMultiplyIntensityColorAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginMultiplyIntensityColorAllFramesNameD(
    const char* path, double color);

// Class Plugin
double result = ChromaAnimationAPI::MultiplyIntensityColorAllFramesNameD(
    const char* path, double color);
```

PluginMultiplyIntensityColorName

Multiply the specific frame by the RBG color intensity. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginMultiplyIntensityColorName(
    const char* path, int frameId, int color);

// Class Plugin
ChromaAnimationAPI::MultiplyIntensityColorName(
    const char* path, int frameId, int color);
```

PluginMultiplyIntensityColorNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginMultiplyIntensityColorNameD(
    const char* path, double frameId, double color);

// Class Plugin
```

```
double result = ChromaAnimationAPI::MultiplyIntensityColorNameD(
    const char* path, double frameId, double color);
```

PluginMultiplyIntensityName

Multiply all the colors in the frame by the intensity value. The valid the intensity range is from 0.0 to 255.0. RGB components are multiplied equally. An intensity of 0.5 would half the color value. Black colors in the frame will not be affected by this method.

```
// DLL Interface
EXPORT_API void PluginMultiplyIntensityName(
    const char* path, int frameId, float intensity);

// Class Plugin
ChromaAnimationAPI::MultiplyIntensityName(
    const char* path, int frameId, float intensity);
```

PluginMultiplyIntensityNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginMultiplyIntensityNameD(
    const char* path, double frameId, double intensity);

// Class Plugin
double result = ChromaAnimationAPI::MultiplyIntensityNameD(
    const char* path, double frameId, double intensity);
```

PluginMultiplyIntensityRGB

Multiply the specific frame by the RBG color intensity. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginMultiplyIntensityRGB(
    int animationId, int frameId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::MultiplyIntensityRGB(
    int animationId, int frameId, int red, int green, int blue);
```

PluginMultiplyIntensityRGBName

Multiply the specific frame by the RBG color intensity. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginMultiplyIntensityRGBName(
    const char* path, int frameId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::MultiplyIntensityRGBName(
    const char* path, int frameId, int red, int green, int blue);
```

PluginMultiplyIntensityRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginMultiplyIntensityRGBNameD(
    const char* path, double frameId, double red, double green, double blue);

// Class Plugin
double result = ChromaAnimationAPI::MultiplyIntensityRGBNameD(
    const char* path, double frameId, double red, double green, double blue);
```

PluginMultiplyNonZeroTargetColorLerp

Multiply the specific frame by the color lerp result between color 1 and 2 using the frame color value as the `t` value. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginMultiplyNonZeroTargetColorLerp(
    int animationId, int frameId, int color1, int color2);

// Class Plugin
ChromaAnimationAPI::MultiplyNonZeroTargetColorLerp(
    int animationId, int frameId, int color1, int color2);
```

PluginMultiplyNonZeroTargetColorLerpAllFrames

Multiply all frames by the color lerp result between color 1 and 2 using the frame color value as the `t` value. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginMultiplyNonZeroTargetColorLerpAllFrames(
    int animationId, int color1, int color2);
```

```
// Class Plugin
ChromaAnimationAPI::MultiplyNonZeroTargetColorLerpAllFrames(
    int animationId, int color1, int color2);
```

PluginMultiplyNonZeroTargetColorLerpAllFramesName

Multiply all frames by the color lerp result between color 1 and 2 using the frame color value as the `t` value. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginMultiplyNonZeroTargetColorLerpAllFramesName(
    const char* path, int color1, int color2);

// Class Plugin
ChromaAnimationAPI::MultiplyNonZeroTargetColorLerpAllFramesName(
    const char* path, int color1, int color2);
```

PluginMultiplyNonZeroTargetColorLerpAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginMultiplyNonZeroTargetColorLerpAllFramesNameD(
    const char* path, double color1, double color2);

// Class Plugin
double result = ChromaAnimationAPI::MultiplyNonZeroTargetColorLerpAllFramesNameD(
    const char* path, double color1, double color2);
```

PluginMultiplyNonZeroTargetColorLerpAllFramesRGB

Multiply the specific frame by the color lerp result between RGB 1 and 2 using the frame color value as the `t` value. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginMultiplyNonZeroTargetColorLerpAllFramesRGB(
    int animationId, int red1, int green1, int blue1, int red2, int green2, int
blue2);

// Class Plugin
ChromaAnimationAPI::MultiplyNonZeroTargetColorLerpAllFramesRGB(
    int animationId, int red1, int green1, int blue1, int red2, int green2, int
blue2);
```

PluginMultiplyNonZeroTargetColorLerpAllFramesRGBName

Multiply the specific frame by the color lerp result between RGB 1 and 2 using the frame color value as the `t` value. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginMultiplyNonZeroTargetColorLerpAllFramesRGBName(
    const char* path, int red1, int green1, int blue1, int red2, int green2,
    int blue2);

// Class Plugin
ChromaAnimationAPI::MultiplyNonZeroTargetColorLerpAllFramesRGBName(
    const char* path, int red1, int green1, int blue1, int red2, int green2,
    int blue2);
```

PluginMultiplyNonZeroTargetColorLerpAllFramesRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginMultiplyNonZeroTargetColorLerpAllFramesRGBNameD(
    const char* path, double red1, double green1, double blue1, double red2,
    double green2, double blue2);

// Class Plugin
double result =
ChromaAnimationAPI::MultiplyNonZeroTargetColorLerpAllFramesRGBNameD(
    const char* path, double red1, double green1, double blue1, double red2,
    double green2, double blue2);
```

PluginMultiplyTargetColorLerp

Multiply the specific frame by the color lerp result between color 1 and 2 using the frame color value as the `t` value. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginMultiplyTargetColorLerp(
    int animationId, int frameId, int color1, int color2);

// Class Plugin
ChromaAnimationAPI::MultiplyTargetColorLerp(
    int animationId, int frameId, int color1, int color2);
```

PluginMultiplyTargetColorLerpAllFrames

Multiply all frames by the color lerp result between color 1 and 2 using the frame color value as the `t` value. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginMultiplyTargetColorLerpAllFrames(
    int animationId, int color1, int color2);

// Class Plugin
ChromaAnimationAPI::MultiplyTargetColorLerpAllFrames(
    int animationId, int color1, int color2);
```

PluginMultiplyTargetColorLerpAllFramesName

Multiply all frames by the color lerp result between color 1 and 2 using the frame color value as the `t` value. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginMultiplyTargetColorLerpAllFramesName(
    const char* path, int color1, int color2);

// Class Plugin
ChromaAnimationAPI::MultiplyTargetColorLerpAllFramesName(
    const char* path, int color1, int color2);
```

PluginMultiplyTargetColorLerpAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginMultiplyTargetColorLerpAllFramesNameD(
    const char* path, double color1, double color2);

// Class Plugin
double result = ChromaAnimationAPI::MultiplyTargetColorLerpAllFramesNameD(
    const char* path, double color1, double color2);
```

PluginMultiplyTargetColorLerpAllFramesRGB

Multiply all frames by the color lerp result between RGB 1 and 2 using the frame color value as the `t` value. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginMultiplyTargetColorLerpAllFramesRGB(
    int animationId, int red1, int green1, int blue1, int red2, int green2, int
blue2);

// Class Plugin
ChromaAnimationAPI::MultiplyTargetColorLerpAllFramesRGB(
    int animationId, int red1, int green1, int blue1, int red2, int green2, int
blue2);
```

PluginMultiplyTargetColorLerpAllFramesRGBName

Multiply all frames by the color lerp result between RGB 1 and 2 using the frame color value as the `t` value. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginMultiplyTargetColorLerpAllFramesRGBName(
    const char* path, int red1, int green1, int blue1, int red2, int green2,
    int blue2);

// Class Plugin
ChromaAnimationAPI::MultiplyTargetColorLerpAllFramesRGBName(
    const char* path, int red1, int green1, int blue1, int red2, int green2,
    int blue2);
```

PluginMultiplyTargetColorLerpAllFramesRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginMultiplyTargetColorLerpAllFramesRGBNameD(
    const char* path, double red1, double green1, double blue1, double red2,
    double green2, double blue2);

// Class Plugin
double result = ChromaAnimationAPI::MultiplyTargetColorLerpAllFramesRGBNameD(
    const char* path, double red1, double green1, double blue1, double red2,
    double green2, double blue2);
```

PluginMultiplyTargetColorLerpName

Multiply the specific frame by the color lerp result between color 1 and 2 using the frame color value as the `t` value. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginMultiplyTargetColorLerpName(
    const char* path, int frameId, int color1, int color2);

// Class Plugin
ChromaAnimationAPI::MultiplyTargetColorLerpName(
    const char* path, int frameId, int color1, int color2);
```

PluginOffsetColors

Offset all colors in the frame using the RGB offset. Use the range of -255 to 255 for red, green, and blue parameters. Negative values remove color. Positive values add color.

```
// DLL Interface
EXPORT_API void PluginOffsetColors(
    int animationId, int frameId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::OffsetColors(
    int animationId, int frameId, int red, int green, int blue);
```

PluginOffsetColorsAllFrames

Offset all colors for all frames using the RGB offset. Use the range of -255 to 255 for red, green, and blue parameters. Negative values remove color. Positive values add color.

```
// DLL Interface
EXPORT_API void PluginOffsetColorsAllFrames(
    int animationId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::OffsetColorsAllFrames(
    int animationId, int red, int green, int blue);
```

PluginOffsetColorsAllFramesName

Offset all colors for all frames using the RGB offset. Use the range of -255 to 255 for red, green, and blue parameters. Negative values remove color. Positive values add color.

```
// DLL Interface
EXPORT_API void PluginOffsetColorsAllFramesName(
    const char* path, int red, int green, int blue);
```

```
// Class Plugin
ChromaAnimationAPI::OffsetColorsAllFramesName(
    const char* path, int red, int green, int blue);
```

PluginOffsetColorsAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginOffsetColorsAllFramesNameD(
    const char* path, double red, double green, double blue);

// Class Plugin
double result = ChromaAnimationAPI::OffsetColorsAllFramesNameD(
    const char* path, double red, double green, double blue);
```

PluginOffsetColorsName

Offset all colors in the frame using the RGB offset. Use the range of -255 to 255 for red, green, and blue parameters. Negative values remove color. Positive values add color.

```
// DLL Interface
EXPORT_API void PluginOffsetColorsName(
    const char* path, int frameId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::OffsetColorsName(
    const char* path, int frameId, int red, int green, int blue);
```

PluginOffsetColorsNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginOffsetColorsNameD(
    const char* path, double frameId, double red, double green, double blue);

// Class Plugin
double result = ChromaAnimationAPI::OffsetColorsNameD(
    const char* path, double frameId, double red, double green, double blue);
```

PluginOffsetNonZeroColors

This method will only update colors in the animation that are not already set to black. Offset a subset of colors in the frame using the RGB offset. Use the range of -255 to 255 for red, green, and blue parameters. Negative values remove color. Positive values add color.

```
// DLL Interface
EXPORT_API void PluginOffsetNonZeroColors(
    int animationId, int frameId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::OffsetNonZeroColors(
    int animationId, int frameId, int red, int green, int blue);
```

PluginOffsetNonZeroColorsAllFrames

This method will only update colors in the animation that are not already set to black. Offset a subset of colors for all frames using the RGB offset. Use the range of -255 to 255 for red, green, and blue parameters. Negative values remove color. Positive values add color.

```
// DLL Interface
EXPORT_API void PluginOffsetNonZeroColorsAllFrames(
    int animationId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::OffsetNonZeroColorsAllFrames(
    int animationId, int red, int green, int blue);
```

PluginOffsetNonZeroColorsAllFramesName

This method will only update colors in the animation that are not already set to black. Offset a subset of colors for all frames using the RGB offset. Use the range of -255 to 255 for red, green, and blue parameters. Negative values remove color. Positive values add color.

```
// DLL Interface
EXPORT_API void PluginOffsetNonZeroColorsAllFramesName(
    const char* path, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::OffsetNonZeroColorsAllFramesName(
    const char* path, int red, int green, int blue);
```

PluginOffsetNonZeroColorsAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginOffsetNonZeroColorsAllFramesNameD(
    const char* path, double red, double green, double blue);

// Class Plugin
double result = ChromaAnimationAPI::OffsetNonZeroColorsAllFramesNameD(
    const char* path, double red, double green, double blue);
```

PluginOffsetNonZeroColorsName

This method will only update colors in the animation that are not already set to black. Offset a subset of colors in the frame using the RGB offset. Use the range of -255 to 255 for red, green, and blue parameters. Negative values remove color. Positive values add color.

```
// DLL Interface
EXPORT_API void PluginOffsetNonZeroColorsName(
    const char* path, int frameId, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::OffsetNonZeroColorsName(
    const char* path, int frameId, int red, int green, int blue);
```

PluginOffsetNonZeroColorsNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginOffsetNonZeroColorsNameD(
    const char* path, double frameId, double red, double green, double blue);

// Class Plugin
double result = ChromaAnimationAPI::OffsetNonZeroColorsNameD(
    const char* path, double frameId, double red, double green, double blue);
```

PluginOpenAnimation

Opens a `Chroma` animation file so that it can be played. Returns an animation id >= 0 upon success. Returns negative one if there was a failure. The animation id is used in most of the API methods.

```
// DLL Interface
EXPORT_API int PluginOpenAnimation(const char* path);
```

```
// Class Plugin
int result = ChromaAnimationAPI::OpenAnimation(const char* path);
```

PluginOpenAnimationD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginOpenAnimationD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::OpenAnimationD(const char* path);
```

PluginOpenAnimationFromMemory

Opens a `Chroma` animation data from memory so that it can be played. `Data` is a pointer to BYTE array of the loaded animation in memory. `Name` will be assigned to the animation when loaded. Returns an animation id >= 0 upon success. Returns negative one if there was a failure. The animation id is used in most of the API methods.

```
// DLL Interface
EXPORT_API int PluginOpenAnimationFromMemory(
    const BYTE* data, const char* name);

// Class Plugin
int result = ChromaAnimationAPI::OpenAnimationFromMemory(
    const BYTE* data, const char* name);
```

PluginOpenEditorDialog

Opens a `Chroma` animation file with the `.chroma` extension. Returns zero upon success. Returns negative one if there was a failure.

```
// DLL Interface
EXPORT_API int PluginOpenEditorDialog(const char* path);

// Class Plugin
int result = ChromaAnimationAPI::OpenEditorDialog(const char* path);
```

PluginOpenEditorDialogAndPlay

Open the named animation in the editor dialog and play the animation at start.

```
// DLL Interface
EXPORT_API int PluginOpenEditorDialogAndPlay(const char* path);

// Class Plugin
int result = ChromaAnimationAPI::OpenEditorDialogAndPlay(const char* path);
```

PluginOpenEditorDialogAndPlayD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginOpenEditorDialogAndPlayD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::OpenEditorDialogAndPlayD(const char* path);
```

PluginOpenEditorDialogD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginOpenEditorDialogD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::OpenEditorDialogD(const char* path);
```

PluginOverrideFrameDuration

Sets the `duration` for all grames in the `Chroma` animation to the `duration` parameter. Returns the animation id upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginOverrideFrameDuration(
    int animationId, float duration);

// Class Plugin
int result = ChromaAnimationAPI::OverrideFrameDuration(
    int animationId, float duration);
```

PluginOverrideFrameDurationD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginOverrideFrameDurationD(
    double animationId, double duration);

// Class Plugin
double result = ChromaAnimationAPI::OverrideFrameDurationD(
    double animationId, double duration);
```

PluginOverrideFrameDurationName

Override the duration of all frames with the `duration` value. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginOverrideFrameDurationName(
    const char* path, float duration);

// Class Plugin
ChromaAnimationAPI::OverrideFrameDurationName(
    const char* path, float duration);
```

PluginPauseAnimation

Pause the current animation referenced by id.

```
// DLL Interface
EXPORT_API void PluginPauseAnimation(int animationId);

// Class Plugin
ChromaAnimationAPI::PauseAnimation(int animationId);
```

PluginPauseAnimationName

Pause the current animation referenced by name.

```
// DLL Interface
EXPORT_API void PluginPauseAnimationName(const char* path);

// Class Plugin
ChromaAnimationAPI::PauseAnimationName(const char* path);
```

PluginPauseAnimationNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginPauseAnimationNameD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::PauseAnimationNameD(const char* path);
```

PluginPlayAnimation

Plays the `Chroma` animation. This will load the animation, if not loaded previously. Returns the animation id upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginPlayAnimation(int animationId);

// Class Plugin
int result = ChromaAnimationAPI::PlayAnimation(int animationId);
```

PluginPlayAnimationD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginPlayAnimationD(double animationId);

// Class Plugin
double result = ChromaAnimationAPI::PlayAnimationD(double animationId);
```

PluginPlayAnimationFrame

`PluginPlayAnimationFrame` automatically handles initializing the `ChromaSDK`. The method will play the animation given the `animationId` with looping `on` or `off` starting at the `frameId`.

```
// DLL Interface
EXPORT_API void PluginPlayAnimationFrame(
    int animationId, int frameId, bool loop);

// Class Plugin
ChromaAnimationAPI::PlayAnimationFrame(
    int animationId, int frameId, bool loop);
```

PluginPlayAnimationFrameName

`PluginPlayAnimationFrameName` automatically handles initializing the `ChromaSDK`. The named `.chroma` animation file will be automatically opened. The animation will play with looping `on` or `off` starting at the `frameId`.

```
// DLL Interface
EXPORT_API void PluginPlayAnimationFrameName(
    const char* path, int frameId, bool loop);

// Class Plugin
ChromaAnimationAPI::PlayAnimationFrameName(
    const char* path, int frameId, bool loop);
```

PluginPlayAnimationFrameNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginPlayAnimationFrameNameD(
    const char* path, double frameId, double loop);

// Class Plugin
double result = ChromaAnimationAPI::PlayAnimationFrameNameD(
    const char* path, double frameId, double loop);
```

PluginPlayAnimationLoop

`PluginPlayAnimationLoop` automatically handles initializing the `ChromaSDK`. The method will play the animation given the `animationId` with looping `on` or `off`.

```
// DLL Interface
EXPORT_API void PluginPlayAnimationLoop(
    int animationId, bool loop);

// Class Plugin
ChromaAnimationAPI::PlayAnimationLoop(
    int animationId, bool loop);
```

PluginPlayAnimationName

`PluginPlayAnimationName` automatically handles initializing the `ChromaSDK`. The named `.chroma` animation file will be automatically opened. The animation will play with looping `on` or `off`.

```
// DLL Interface
EXPORT_API void PluginPlayAnimationName(
    const char* path, bool loop);

// Class Plugin
ChromaAnimationAPI::PlayAnimationName(
    const char* path, bool loop);
```

PluginPlayAnimationNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginPlayAnimationNameD(
    const char* path, double loop);

// Class Plugin
double result = ChromaAnimationAPI::PlayAnimationNameD(
    const char* path, double loop);
```

PluginPlayComposite

`PluginPlayComposite` automatically handles initializing the `ChromaSDK`. The named animation files for the `.chroma` set will be automatically opened. The set of animations will play with looping `on` or `off`.

```
// DLL Interface
EXPORT_API void PluginPlayComposite(
    const char* name, bool loop);

// Class Plugin
ChromaAnimationAPI::PlayComposite(
    const char* name, bool loop);
```

PluginPlayCompositeD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginPlayCompositeD(
    const char* name, double loop);
```

```
// Class Plugin
double result = ChromaAnimationAPI::PlayCompositeD(
    const char* name, double loop);
```

PluginPreviewFrame

Displays the `Chroma` animation frame on `Chroma` hardware given the `frameId`. Returns the animation id upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginPreviewFrame(
    int animationId, int frameId);

// Class Plugin
int result = ChromaAnimationAPI::PreviewFrame(
    int animationId, int frameId);
```

PluginPreviewFrameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginPreviewFrameD(
    double animationId, double frameId);

// Class Plugin
double result = ChromaAnimationAPI::PreviewFrameD(
    double animationId, double frameId);
```

PluginPreviewFrameName

Displays the `Chroma` animation frame on `Chroma` hardware given the `frameId`. Animaton is referenced by name.

```
// DLL Interface
EXPORT_API void PluginPreviewFrameName(
    const char* path, int frameId);

// Class Plugin
ChromaAnimationAPI::PreviewFrameName(
    const char* path, int frameId);
```

PluginReduceFrames

Reduce the frames of the animation by removing every nth element. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginReduceFrames(
    int animationId, int n);

// Class Plugin
ChromaAnimationAPI::ReduceFrames(
    int animationId, int n);
```

PluginReduceFramesName

Reduce the frames of the animation by removing every nth element. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginReduceFramesName(
    const char* path, int n);

// Class Plugin
ChromaAnimationAPI::ReduceFramesName(
    const char* path, int n);
```

PluginReduceFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginReduceFramesNameD(
    const char* path, double n);

// Class Plugin
double result = ChromaAnimationAPI::ReduceFramesNameD(
    const char* path, double n);
```

PluginResetAnimation

Resets the `Chroma` animation to 1 blank frame. Returns the animation id upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginResetAnimation(int animationId);
```

```
// Class Plugin
int result = ChromaAnimationAPI::ResetAnimation(int animationId);
```

PluginResumeAnimation

Resume the animation with loop `ON` or `OFF` referenced by id.

```
// DLL Interface
EXPORT_API void PluginResumeAnimation(
    int animationId, bool loop);

// Class Plugin
ChromaAnimationAPI::ResumeAnimation(
    int animationId, bool loop);
```

PluginResumeAnimationName

Resume the animation with loop `ON` or `OFF` referenced by name.

```
// DLL Interface
EXPORT_API void PluginResumeAnimationName(
    const char* path, bool loop);

// Class Plugin
ChromaAnimationAPI::ResumeAnimationName(
    const char* path, bool loop);
```

PluginResumeAnimationNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginResumeAnimationNameD(
    const char* path, double loop);

// Class Plugin
double result = ChromaAnimationAPI::ResumeAnimationNameD(
    const char* path, double loop);
```

PluginReverse

Reverse the animation frame order of the Chroma animation. Returns the animation id upon success. Returns negative one upon failure. Animation is referenced by id.

```
// DLL Interface
EXPORT_API int PluginReverse(int animationId);

// Class Plugin
int result = ChromaAnimationAPI::Reverse(int animationId);
```

PluginReverseAllFrames

Reverse the animation frame order of the Chroma animation. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginReverseAllFrames(int animationId);

// Class Plugin
ChromaAnimationAPI::ReverseAllFrames(int animationId);
```

PluginReverseAllFramesName

Reverse the animation frame order of the Chroma animation. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginReverseAllFramesName(const char* path);

// Class Plugin
ChromaAnimationAPI::ReverseAllFramesName(const char* path);
```

PluginReverseAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginReverseAllFramesNameD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::ReverseAllFramesNameD(const char* path);
```

PluginSaveAnimation

Save the animation referenced by id to the path specified.

```
// DLL Interface
EXPORT_API int PluginSaveAnimation(
    int animationId, const char* path);

// Class Plugin
int result = ChromaAnimationAPI::SaveAnimation(
    int animationId, const char* path);
```

PluginSaveAnimationName

Save the named animation to the target path specified.

```
// DLL Interface
EXPORT_API int PluginSaveAnimationName(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
int result = ChromaAnimationAPI::SaveAnimationName(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginSet1DColor

Set the animation color for a frame given the `1D led`. The `led` should be greater than or equal to 0 and less than the `MaxLeds`. The animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSet1DColor(
    int animationId, int frameId, int led, int color);

// Class Plugin
ChromaAnimationAPI::Set1DColor(
    int animationId, int frameId, int led, int color);
```

PluginSet1DColorName

Set the animation color for a frame given the `1D led`. The `led` should be greater than or equal to 0 and less than the `MaxLeds`. The animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSet1DColorName(
    const char* path, int frameId, int led, int color);
```

```
// Class Plugin
ChromaAnimationAPI::Set1DColorName(
    const char* path, int frameId, int led, int color);
```

PluginSet1DColorNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSet1DColorNameD(
    const char* path, double frameId, double led, double color);

// Class Plugin
double result = ChromaAnimationAPI::Set1DColorNameD(
    const char* path, double frameId, double led, double color);
```

PluginSet2DColor

Set the animation color for a frame given the `2D row` and `column`. The `row` should be greater than or equal to 0 and less than the `MaxRow`. The `column` should be greater than or equal to 0 and less than the `MaxColumn`. The animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSet2DColor(
    int animationId, int frameId, int row, int column, int color);

// Class Plugin
ChromaAnimationAPI::Set2DColor(
    int animationId, int frameId, int row, int column, int color);
```

PluginSet2DColorName

Set the animation color for a frame given the `2D row` and `column`. The `row` should be greater than or equal to 0 and less than the `MaxRow`. The `column` should be greater than or equal to 0 and less than the `MaxColumn`. The animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSet2DColorName(
    const char* path, int frameId, int row, int column, int color);

// Class Plugin
ChromaAnimationAPI::Set2DColorName(
    const char* path, int frameId, int row, int column, int color);
```

PluginSet2DColorNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSet2DColorNameD(
    const char* path, double frameId, double rowColumnIndex, double color);

// Class Plugin
double result = ChromaAnimationAPI::Set2DColorNameD(
    const char* path, double frameId, double rowColumnIndex, double color);
```

PluginSetChromaCustomColorAllFrames

When custom color is set, the custom key mode will be used. The animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetChromaCustomColorAllFrames(int animationId);

// Class Plugin
ChromaAnimationAPI::SetChromaCustomColorAllFrames(int animationId);
```

PluginSetChromaCustomColorAllFramesName

When custom color is set, the custom key mode will be used. The animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetChromaCustomColorAllFramesName(const char* path);

// Class Plugin
ChromaAnimationAPI::SetChromaCustomColorAllFramesName(const char* path);
```

PluginSetChromaCustomColorAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSetChromaCustomColorAllFramesNameD(const char* path);

// Class Plugin
```

```
double result = ChromaAnimationAPI::SetChromaCustomColorAllFramesNameD(const char*
path);
```

PluginSetChromaCustomFlag

Set the Chroma custom key color flag on all frames. `True` changes the layout from grid to key. `True` changes the layout from key to grid. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetChromaCustomFlag(
    int animationId, bool flag);

// Class Plugin
ChromaAnimationAPI::SetChromaCustomFlag(
    int animationId, bool flag);
```

PluginSetChromaCustomFlagName

Set the Chroma custom key color flag on all frames. `True` changes the layout from grid to key. `True` changes the layout from key to grid. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetChromaCustomFlagName(
    const char* path, bool flag);

// Class Plugin
ChromaAnimationAPI::SetChromaCustomFlagName(
    const char* path, bool flag);
```

PluginSetChromaCustomFlagNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSetChromaCustomFlagNameD(
    const char* path, double flag);

// Class Plugin
double result = ChromaAnimationAPI::SetChromaCustomFlagNameD(
    const char* path, double flag);
```

PluginSetCurrentFrame

Set the current frame of the animation referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetCurrentFrame(
    int animationId, int frameId);

// Class Plugin
ChromaAnimationAPI::SetCurrentFrame(
    int animationId, int frameId);
```

PluginSetCurrentFrameName

Set the current frame of the animation referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetCurrentFrameName(
    const char* path, int frameId);

// Class Plugin
ChromaAnimationAPI::SetCurrentFrameName(
    const char* path, int frameId);
```

PluginSetCurrentFrameNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSetCurrentFrameNameD(
    const char* path, double frameId);

// Class Plugin
double result = ChromaAnimationAPI::SetCurrentFrameNameD(
    const char* path, double frameId);
```

PluginSetCustomColorFlag2D

Set the custom alpha flag on the color array

```
// DLL Interface
EXPORT_API RZRESULT PluginSetCustomColorFlag2D(
    int device, int* colors);

// Class Plugin
```

```
RZRESULT result = ChromaAnimationAPI::SetCustomColorFlag2D(
    int device, int* colors);
```

PluginSetDevice

Changes the `deviceType` and `device` of a `Chroma` animation. If the device is changed, the `Chroma` animation will be reset with 1 blank frame. Returns the animation id upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginSetDevice(
    int animationId, int deviceType, int device);

// Class Plugin
int result = ChromaAnimationAPI::SetDevice(
    int animationId, int deviceType, int device);
```

PluginSetEffect

SetEffect will display the referenced effect id.

```
// DLL Interface
EXPORT_API RZRESULT PluginSetEffect(
    const ChromaSDK::FChromaSDKGuid& effectId);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::SetEffect(
    const ChromaSDK::FChromaSDKGuid& effectId);
```

PluginSetEffectCustom1D

SetEffectCustom1D will display the referenced colors immediately

```
// DLL Interface
EXPORT_API RZRESULT PluginSetEffectCustom1D(
    const int device, const int* colors);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::SetEffectCustom1D(
    const int device, const int* colors);
```

PluginSetEffectCustom2D

SetEffectCustom2D will display the referenced colors immediately.

```
// DLL Interface
EXPORT_API RZRESULT PluginSetEffectCustom2D(
    const int device, const int* colors);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::SetEffectCustom2D(
    const int device, const int* colors);
```

PluginSetEffectKeyboardCustom2D

SetEffectKeyboardCustom2D will display the referenced custom keyboard colors immediately. Colors represent a visual grid layout. Keys represent the hotkeys for any layout.

```
// DLL Interface
EXPORT_API RZRESULT PluginSetEffectKeyboardCustom2D(
    const int device, const int* colors, const int* keys);

// Class Plugin
RZRESULT result = ChromaAnimationAPI::SetEffectKeyboardCustom2D(
    const int device, const int* colors, const int* keys);
```

PluginSetIdleAnimation

When the idle animation is used, the named animation will play when no other animations are playing. Reference the animation by id.

```
// DLL Interface
EXPORT_API void PluginSetIdleAnimation(int animationId);

// Class Plugin
ChromaAnimationAPI::SetIdleAnimation(int animationId);
```

PluginSetIdleAnimationName

When the idle animation is used, the named animation will play when no other animations are playing. Reference the animation by name.

```
// DLL Interface
EXPORT_API void PluginSetIdleAnimationName(const char* path);

// Class Plugin
ChromaAnimationAPI::SetIdleAnimationName(const char* path);
```

PluginSetKeyColor

Set animation key to a static color for the given frame.

```
// DLL Interface
EXPORT_API void PluginSetKeyColor(
    int animationId, int frameId, int rzkey, int color);

// Class Plugin
ChromaAnimationAPI::SetKeyColor(
    int animationId, int frameId, int rzkey, int color);
```

PluginSetKeyColorAllFrames

Set the key to the specified key color for all frames. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetKeyColorAllFrames(
    int animationId, int rzkey, int color);

// Class Plugin
ChromaAnimationAPI::SetKeyColorAllFrames(
    int animationId, int rzkey, int color);
```

PluginSetKeyColorAllFramesName

Set the key to the specified key color for all frames. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetKeyColorAllFramesName(
    const char* path, int rzkey, int color);

// Class Plugin
ChromaAnimationAPI::SetKeyColorAllFramesName(
    const char* path, int rzkey, int color);
```

PluginSetKeyColorAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSetKeyColorAllFramesNameD(
    const char* path, double rzkey, double color);
```

```
// Class Plugin
double result = ChromaAnimationAPI::SetKeyColorAllFramesNameD(
    const char* path, double rzkey, double color);
```

PluginSetKeyColorAllFramesRGB

Set the key to the specified key color for all frames. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetKeyColorAllFramesRGB(
    int animationId, int rzkey, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::SetKeyColorAllFramesRGB(
    int animationId, int rzkey, int red, int green, int blue);
```

PluginSetKeyColorAllFramesRGBName

Set the key to the specified key color for all frames. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetKeyColorAllFramesRGBName(
    const char* path, int rzkey, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::SetKeyColorAllFramesRGBName(
    const char* path, int rzkey, int red, int green, int blue);
```

PluginSetKeyColorAllFramesRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSetKeyColorAllFramesRGBNameD(
    const char* path, double rzkey, double red, double green, double blue);

// Class Plugin
double result = ChromaAnimationAPI::SetKeyColorAllFramesRGBNameD(
    const char* path, double rzkey, double red, double green, double blue);
```

PluginSetKeyColorName

Set animation key to a static color for the given frame.

```
// DLL Interface
EXPORT_API void PluginSetKeyColorName(
    const char* path, int frameId, int rzkey, int color);

// Class Plugin
ChromaAnimationAPI::SetKeyColorName(
    const char* path, int frameId, int rzkey, int color);
```

PluginSetKeyColorNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSetKeyColorNameD(
    const char* path, double frameId, double rzkey, double color);

// Class Plugin
double result = ChromaAnimationAPI::SetKeyColorNameD(
    const char* path, double frameId, double rzkey, double color);
```

PluginSetKeyColorRGB

Set the key to the specified key color for the specified frame. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetKeyColorRGB(
    int animationId, int frameId, int rzkey, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::SetKeyColorRGB(
    int animationId, int frameId, int rzkey, int red, int green, int blue);
```

PluginSetKeyColorRGBName

Set the key to the specified key color for the specified frame. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetKeyColorRGBName(
    const char* path, int frameId, int rzkey, int red, int green, int blue);

// Class Plugin
```

```
ChromaAnimationAPI::SetKeyColorRGBName(
    const char* path, int frameId, int rzkey, int red, int green, int blue);
```

PluginSetKeyColorRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSetKeyColorRGBNameD(
    const char* path, double frameId, double rzkey, double red, double green,
    double blue);

// Class Plugin
double result = ChromaAnimationAPI::SetKeyColorRGBNameD(
    const char* path, double frameId, double rzkey, double red, double green,
    double blue);
```

PluginSetKeyNonZeroColor

Set animation key to a static color for the given frame if the existing color is not already black.

```
// DLL Interface
EXPORT_API void PluginSetKeyNonZeroColor(
    int animationId, int frameId, int rzkey, int color);

// Class Plugin
ChromaAnimationAPI::SetKeyNonZeroColor(
    int animationId, int frameId, int rzkey, int color);
```

PluginSetKeyNonZeroColorName

Set animation key to a static color for the given frame if the existing color is not already black.

```
// DLL Interface
EXPORT_API void PluginSetKeyNonZeroColorName(
    const char* path, int frameId, int rzkey, int color);

// Class Plugin
ChromaAnimationAPI::SetKeyNonZeroColorName(
    const char* path, int frameId, int rzkey, int color);
```

PluginSetKeyNonZeroColorNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSetKeyNonZeroColorNameD(
    const char* path, double frameId, double rzkey, double color);

// Class Plugin
double result = ChromaAnimationAPI::SetKeyNonZeroColorNameD(
    const char* path, double frameId, double rzkey, double color);
```

PluginSetKeyNonZeroColorRGB

Set the key to the specified key color for the specified frame where color is not black. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetKeyNonZeroColorRGB(
    int animationId, int frameId, int rzkey, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::SetKeyNonZeroColorRGB(
    int animationId, int frameId, int rzkey, int red, int green, int blue);
```

PluginSetKeyNonZeroColorRGBName

Set the key to the specified key color for the specified frame where color is not black. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetKeyNonZeroColorRGBName(
    const char* path, int frameId, int rzkey, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::SetKeyNonZeroColorRGBName(
    const char* path, int frameId, int rzkey, int red, int green, int blue);
```

PluginSetKeyNonZeroColorRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSetKeyNonZeroColorRGBNameD(
    const char* path, double frameId, double rzkey, double red, double green,
```

```
    double blue);

// Class Plugin
double result = ChromaAnimationAPI::SetKeyNonZeroColorRGBNameD(
    const char* path, double frameId, double rzkey, double red, double green,
    double blue);
```

PluginSetKeyRowColumnColorName

Set animation key by row and column to a static color for the given frame.

```
// DLL Interface
EXPORT_API void PluginSetKeyRowColumnColorName(
    const char* path, int frameId, int row, int column, int color);

// Class Plugin
ChromaAnimationAPI::SetKeyRowColumnColorName(
    const char* path, int frameId, int row, int column, int color);
```

PluginSetKeysColor

Set an array of animation keys to a static color for the given frame. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetKeysColor(
    int animationId, int frameId, const int* rzkeys, int keyCount, int color);

// Class Plugin
ChromaAnimationAPI::SetKeysColor(
    int animationId, int frameId, const int* rzkeys, int keyCount, int color);
```

PluginSetKeysColorAllFrames

Set an array of animation keys to a static color for all frames. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetKeysColorAllFrames(
    int animationId, const int* rzkeys, int keyCount, int color);

// Class Plugin
ChromaAnimationAPI::SetKeysColorAllFrames(
    int animationId, const int* rzkeys, int keyCount, int color);
```

PluginSetKeysColorAllFramesName

Set an array of animation keys to a static color for all frames. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetKeysColorAllFramesName(
    const char* path, const int* rzkeys, int keyCount, int color);

// Class Plugin
ChromaAnimationAPI::SetKeysColorAllFramesName(
    const char* path, const int* rzkeys, int keyCount, int color);
```

PluginSetKeysColorAllFramesRGB

Set an array of animation keys to a static color for all frames. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetKeysColorAllFramesRGB(
    int animationId, const int* rzkeys, int keyCount, int red, int green, int
blue);

// Class Plugin
ChromaAnimationAPI::SetKeysColorAllFramesRGB(
    int animationId, const int* rzkeys, int keyCount, int red, int green, int
blue);
```

PluginSetKeysColorAllFramesRGBName

Set an array of animation keys to a static color for all frames. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetKeysColorAllFramesRGBName(
    const char* path, const int* rzkeys, int keyCount, int red, int green, int
blue);

// Class Plugin
ChromaAnimationAPI::SetKeysColorAllFramesRGBName(
    const char* path, const int* rzkeys, int keyCount, int red, int green, int
blue);
```

PluginSetKeysColorName

Set an array of animation keys to a static color for the given frame.

```
// DLL Interface
EXPORT_API void PluginSetKeysColorName(
    const char* path, int frameId, const int* rzkeys, int keyCount, int color);

// Class Plugin
ChromaAnimationAPI::SetKeysColorName(
    const char* path, int frameId, const int* rzkeys, int keyCount, int color);
```

PluginSetKeysColorRGB

Set an array of animation keys to a static color for the given frame. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetKeysColorRGB(
    int animationId, int frameId, const int* rzkeys, int keyCount, int red, int
green,
    int blue);

// Class Plugin
ChromaAnimationAPI::SetKeysColorRGB(
    int animationId, int frameId, const int* rzkeys, int keyCount, int red, int
green,
    int blue);
```

PluginSetKeysColorRGBName

Set an array of animation keys to a static color for the given frame. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetKeysColorRGBName(
    const char* path, int frameId, const int* rzkeys, int keyCount, int red,
    int green, int blue);

// Class Plugin
ChromaAnimationAPI::SetKeysColorRGBName(
    const char* path, int frameId, const int* rzkeys, int keyCount, int red,
    int green, int blue);
```

PluginSetKeysNonZeroColor

Set an array of animation keys to a static color for the given frame if the existing color is not already black.

```
// DLL Interface
EXPORT_API void PluginSetKeysNonZeroColor(
```

```
    int animationId, int frameId, const int* rzkeys, int keyCount, int color);

// Class Plugin
ChromaAnimationAPI::SetKeysNonZeroColor(
    int animationId, int frameId, const int* rzkeys, int keyCount, int color);
```

PluginSetKeysNonZeroColorAllFrames

Set an array of animation keys to a static color for the given frame where the color is not black. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetKeysNonZeroColorAllFrames(
    int animationId, const int* rzkeys, int keyCount, int color);

// Class Plugin
ChromaAnimationAPI::SetKeysNonZeroColorAllFrames(
    int animationId, const int* rzkeys, int keyCount, int color);
```

PluginSetKeysNonZeroColorAllFramesName

Set an array of animation keys to a static color for all frames if the existing color is not already black. Reference animation by name.

```
// DLL Interface
EXPORT_API void PluginSetKeysNonZeroColorAllFramesName(
    const char* path, const int* rzkeys, int keyCount, int color);

// Class Plugin
ChromaAnimationAPI::SetKeysNonZeroColorAllFramesName(
    const char* path, const int* rzkeys, int keyCount, int color);
```

PluginSetKeysNonZeroColorName

Set an array of animation keys to a static color for the given frame if the existing color is not already black. Reference animation by name.

```
// DLL Interface
EXPORT_API void PluginSetKeysNonZeroColorName(
    const char* path, int frameId, const int* rzkeys, int keyCount, int color);

// Class Plugin
ChromaAnimationAPI::SetKeysNonZeroColorName(
    const char* path, int frameId, const int* rzkeys, int keyCount, int color);
```

PluginSetKeysNonZeroColorRGB

Set an array of animation keys to a static color for the given frame where the color is not black. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetKeysNonZeroColorRGB(
    int animationId, int frameId, const int* rzkeys, int keyCount, int red, int
green,
    int blue);

// Class Plugin
ChromaAnimationAPI::SetKeysNonZeroColorRGB(
    int animationId, int frameId, const int* rzkeys, int keyCount, int red, int
green,
    int blue);
```

PluginSetKeysNonZeroColorRGBName

Set an array of animation keys to a static color for the given frame where the color is not black. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetKeysNonZeroColorRGBName(
    const char* path, int frameId, const int* rzkeys, int keyCount, int red,
    int green, int blue);

// Class Plugin
ChromaAnimationAPI::SetKeysNonZeroColorRGBName(
    const char* path, int frameId, const int* rzkeys, int keyCount, int red,
    int green, int blue);
```

PluginSetKeysZeroColor

Set an array of animation keys to a static color for the given frame where the color is black. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetKeysZeroColor(
    int animationId, int frameId, const int* rzkeys, int keyCount, int color);

// Class Plugin
ChromaAnimationAPI::SetKeysZeroColor(
    int animationId, int frameId, const int* rzkeys, int keyCount, int color);
```

PluginSetKeysZeroColorAllFrames

Set an array of animation keys to a static color for all frames where the color is black. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetKeysZeroColorAllFrames(
    int animationId, const int* rzkeys, int keyCount, int color);

// Class Plugin
ChromaAnimationAPI::SetKeysZeroColorAllFrames(
    int animationId, const int* rzkeys, int keyCount, int color);
```

PluginSetKeysZeroColorAllFramesName

Set an array of animation keys to a static color for all frames where the color is black. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetKeysZeroColorAllFramesName(
    const char* path, const int* rzkeys, int keyCount, int color);

// Class Plugin
ChromaAnimationAPI::SetKeysZeroColorAllFramesName(
    const char* path, const int* rzkeys, int keyCount, int color);
```

PluginSetKeysZeroColorAllFramesRGB

Set an array of animation keys to a static color for all frames where the color is black. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetKeysZeroColorAllFramesRGB(
    int animationId, const int* rzkeys, int keyCount, int red, int green, int
blue);

// Class Plugin
ChromaAnimationAPI::SetKeysZeroColorAllFramesRGB(
    int animationId, const int* rzkeys, int keyCount, int red, int green, int
blue);
```

PluginSetKeysZeroColorAllFramesRGBName

Set an array of animation keys to a static color for all frames where the color is black. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetKeysZeroColorAllFramesRGBName(
    const char* path, const int* rzkeys, int keyCount, int red, int green, int
blue);

// Class Plugin
ChromaAnimationAPI::SetKeysZeroColorAllFramesRGBName(
    const char* path, const int* rzkeys, int keyCount, int red, int green, int
blue);
```

PluginSetKeysZeroColorName

Set an array of animation keys to a static color for the given frame where the color is black. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetKeysZeroColorName(
    const char* path, int frameId, const int* rzkeys, int keyCount, int color);

// Class Plugin
ChromaAnimationAPI::SetKeysZeroColorName(
    const char* path, int frameId, const int* rzkeys, int keyCount, int color);
```

PluginSetKeysZeroColorRGB

Set an array of animation keys to a static color for the given frame where the color is black. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetKeysZeroColorRGB(
    int animationId, int frameId, const int* rzkeys, int keyCount, int red, int
green,
    int blue);

// Class Plugin
ChromaAnimationAPI::SetKeysZeroColorRGB(
    int animationId, int frameId, const int* rzkeys, int keyCount, int red, int
green,
    int blue);
```

PluginSetKeysZeroColorRGBName

Set an array of animation keys to a static color for the given frame where the color is black. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetKeysZeroColorRGBName(
    const char* path, int frameId, const int* rzkeys, int keyCount, int red,
    int green, int blue);

// Class Plugin
ChromaAnimationAPI::SetKeysZeroColorRGBName(
    const char* path, int frameId, const int* rzkeys, int keyCount, int red,
    int green, int blue);
```

PluginSetKeyZeroColor

Set animation key to a static color for the given frame where the color is black. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetKeyZeroColor(
    int animationId, int frameId, int rzkey, int color);

// Class Plugin
ChromaAnimationAPI::SetKeyZeroColor(
    int animationId, int frameId, int rzkey, int color);
```

PluginSetKeyZeroColorName

Set animation key to a static color for the given frame where the color is black. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetKeyZeroColorName(
    const char* path, int frameId, int rzkey, int color);

// Class Plugin
ChromaAnimationAPI::SetKeyZeroColorName(
    const char* path, int frameId, int rzkey, int color);
```

PluginSetKeyZeroColorNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSetKeyZeroColorNameD(
    const char* path, double frameId, double rzkey, double color);

// Class Plugin
double result = ChromaAnimationAPI::SetKeyZeroColorNameD(
    const char* path, double frameId, double rzkey, double color);
```

PluginSetKeyZeroColorRGB

Set animation key to a static color for the given frame where the color is black. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSetKeyZeroColorRGB(
    int animationId, int frameId, int rzkey, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::SetKeyZeroColorRGB(
    int animationId, int frameId, int rzkey, int red, int green, int blue);
```

PluginSetKeyZeroColorRGBName

Set animation key to a static color for the given frame where the color is black. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSetKeyZeroColorRGBName(
    const char* path, int frameId, int rzkey, int red, int green, int blue);

// Class Plugin
ChromaAnimationAPI::SetKeyZeroColorRGBName(
    const char* path, int frameId, int rzkey, int red, int green, int blue);
```

PluginSetKeyZeroColorRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSetKeyZeroColorRGBNameD(
    const char* path, double frameId, double rzkey, double red, double green,
    double blue);

// Class Plugin
double result = ChromaAnimationAPI::SetKeyZeroColorRGBNameD(
```

```
    const char* path, double frameId, double rzkey, double red, double green,
    double blue);
```

PluginSetLogDelegate

Invokes the setup for a debug logging callback so that `stdout` is redirected to the callback. This is used by `Unity` so that debug messages can appear in the console window.

```
// DLL Interface
EXPORT_API void PluginSetLogDelegate(DebugLogPtr fp);

// Class Plugin
ChromaAnimationAPI::SetLogDelegate(DebugLogPtr fp);
```

PluginSetStaticColor

Sets the target device to the static color.

```
// DLL Interface
EXPORT_API void PluginSetStaticColor(
    int deviceType, int device, int color);

// Class Plugin
ChromaAnimationAPI::SetStaticColor(
    int deviceType, int device, int color);
```

PluginSetStaticColorAll

Sets all devices to the static color.

```
// DLL Interface
EXPORT_API void PluginSetStaticColorAll(int color);

// Class Plugin
ChromaAnimationAPI::SetStaticColorAll(int color);
```

PluginStaticColor

Sets the target device to the static color.

```
// DLL Interface
EXPORT_API void PluginStaticColor(
```

```
    int deviceType, int device, int color);

// Class Plugin
ChromaAnimationAPI::StaticColor(
    int deviceType, int device, int color);
```

PluginStaticColorAll

Sets all devices to the static color.

```
// DLL Interface
EXPORT_API void PluginStaticColorAll(int color);

// Class Plugin
ChromaAnimationAPI::StaticColorAll(int color);
```

PluginStaticColorD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginStaticColorD(
    double deviceType, double device, double color);

// Class Plugin
double result = ChromaAnimationAPI::StaticColorD(
    double deviceType, double device, double color);
```

PluginStopAll

`PluginStopAll` will automatically stop all animations that are playing.

```
// DLL Interface
EXPORT_API void PluginStopAll();

// Class Plugin
ChromaAnimationAPI::StopAll();
```

PluginStopAnimation

Stops animation playback if in progress. Returns the animation id upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginStopAnimation(int animationId);

// Class Plugin
int result = ChromaAnimationAPI::StopAnimation(int animationId);
```

PluginStopAnimationD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginStopAnimationD(double animationId);

// Class Plugin
double result = ChromaAnimationAPI::StopAnimationD(double animationId);
```

PluginStopAnimationName

`PluginStopAnimationName` automatically handles initializing the `ChromaSDK`. The named `.chroma` animation file will be automatically opened. The animation will stop if playing.

```
// DLL Interface
EXPORT_API void PluginStopAnimationName(const char* path);

// Class Plugin
ChromaAnimationAPI::StopAnimationName(const char* path);
```

PluginStopAnimationNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginStopAnimationNameD(const char* path);

// Class Plugin
double result = ChromaAnimationAPI::StopAnimationNameD(const char* path);
```

PluginStopAnimationType

`PluginStopAnimationType` automatically handles initializing the `ChromaSDK`. If any animation is playing for the `deviceType` and `device` combination, it will be stopped.

```
// DLL Interface
EXPORT_API void PluginStopAnimationType(
    int deviceType, int device);

// Class Plugin
ChromaAnimationAPI::StopAnimationType(
    int deviceType, int device);
```

PluginStopAnimationTypeD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginStopAnimationTypeD(
    double deviceType, double device);

// Class Plugin
double result = ChromaAnimationAPI::StopAnimationTypeD(
    double deviceType, double device);
```

PluginStopComposite

`PluginStopComposite` automatically handles initializing the `ChromaSDK`. The named animation files for the `.chroma` set will be automatically opened. The set of animations will be stopped if playing.

```
// DLL Interface
EXPORT_API void PluginStopComposite(const char* name);

// Class Plugin
ChromaAnimationAPI::StopComposite(const char* name);
```

PluginStopCompositeD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginStopCompositeD(const char* name);

// Class Plugin
double result = ChromaAnimationAPI::StopCompositeD(const char* name);
```

PluginSubtractColor

Return color1 - color2

```
// DLL Interface
EXPORT_API int PluginSubtractColor(
    const int color1, const int color2);

// Class Plugin
int result = ChromaAnimationAPI::SubtractColor(
    const int color1, const int color2);
```

PluginSubtractNonZeroAllKeys

Subtract the source color from the target color for the frame where the target color is not black. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginSubtractNonZeroAllKeys(
    int sourceAnimationId, int targetAnimationId, int frameId);

// Class Plugin
ChromaAnimationAPI::SubtractNonZeroAllKeys(
    int sourceAnimationId, int targetAnimationId, int frameId);
```

PluginSubtractNonZeroAllKeysAllFrames

Subtract the source color from the target color for all frames where the target color is not black. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginSubtractNonZeroAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);

// Class Plugin
ChromaAnimationAPI::SubtractNonZeroAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);
```

PluginSubtractNonZeroAllKeysAllFramesName

Subtract the source color from the target color for all frames where the target color is not black. Source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginSubtractNonZeroAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
ChromaAnimationAPI::SubtractNonZeroAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginSubtractNonZeroAllKeysAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSubtractNonZeroAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
double result = ChromaAnimationAPI::SubtractNonZeroAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginSubtractNonZeroAllKeysAllFramesOffset

Subtract the source color from the target color for all frames where the target color is not black starting at offset for the length of the source. Source and target are referenced by id.

```
// DLL Interface
EXPORT_API void PluginSubtractNonZeroAllKeysAllFramesOffset(
    int sourceAnimationId, int targetAnimationId, int offset);

// Class Plugin
ChromaAnimationAPI::SubtractNonZeroAllKeysAllFramesOffset(
    int sourceAnimationId, int targetAnimationId, int offset);
```

PluginSubtractNonZeroAllKeysAllFramesOffsetName

Subtract the source color from the target color for all frames where the target color is not black starting at offset for the length of the source. Source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginSubtractNonZeroAllKeysAllFramesOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int offset);

// Class Plugin
```

```
ChromaAnimationAPI::SubtractNonZeroAllKeysAllFramesOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int offset);
```

PluginSubtractNonZeroAllKeysAllFramesOffsetNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSubtractNonZeroAllKeysAllFramesOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double offset);

// Class Plugin
double result = ChromaAnimationAPI::SubtractNonZeroAllKeysAllFramesOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double offset);
```

PluginSubtractNonZeroAllKeysName

Subtract the source color from the target color for the frame where the target color is not black. Source and target are referenced by name.

```
// DLL Interface
EXPORT_API void PluginSubtractNonZeroAllKeysName(
    const char* sourceAnimation, const char* targetAnimation, int frameId);

// Class Plugin
ChromaAnimationAPI::SubtractNonZeroAllKeysName(
    const char* sourceAnimation, const char* targetAnimation, int frameId);
```

PluginSubtractNonZeroAllKeysOffset

Subtract the source color from the target where color is not black for the source frame and target offset frame, reference source and target by id.

```
// DLL Interface
EXPORT_API void PluginSubtractNonZeroAllKeysOffset(
    int sourceAnimationId, int targetAnimationId, int frameId, int offset);

// Class Plugin
ChromaAnimationAPI::SubtractNonZeroAllKeysOffset(
    int sourceAnimationId, int targetAnimationId, int frameId, int offset);
```

PluginSubtractNonZeroAllKeysOffsetName

Subtract the source color from the target where color is not black for the source frame and target offset frame, reference source and target by name.

```
// DLL Interface
EXPORT_API void PluginSubtractNonZeroAllKeysOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
offset);

// Class Plugin
ChromaAnimationAPI::SubtractNonZeroAllKeysOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
offset);
```

PluginSubtractNonZeroAllKeysOffsetNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSubtractNonZeroAllKeysOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double offset);

// Class Plugin
double result = ChromaAnimationAPI::SubtractNonZeroAllKeysOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double offset);
```

PluginSubtractNonZeroTargetAllKeysAllFrames

Subtract the source color from the target color where the target color is not black for all frames. Reference source and target by id.

```
// DLL Interface
EXPORT_API void PluginSubtractNonZeroTargetAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);

// Class Plugin
ChromaAnimationAPI::SubtractNonZeroTargetAllKeysAllFrames(
    int sourceAnimationId, int targetAnimationId);
```

PluginSubtractNonZeroTargetAllKeysAllFramesName

Subtract the source color from the target color where the target color is not black for all frames. Reference source and target by name.

```
// DLL Interface
EXPORT_API void PluginSubtractNonZeroTargetAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
ChromaAnimationAPI::SubtractNonZeroTargetAllKeysAllFramesName(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginSubtractNonZeroTargetAllKeysAllFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSubtractNonZeroTargetAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);

// Class Plugin
double result = ChromaAnimationAPI::SubtractNonZeroTargetAllKeysAllFramesNameD(
    const char* sourceAnimation, const char* targetAnimation);
```

PluginSubtractNonZeroTargetAllKeysAllFramesOffset

Subtract the source color from the target color where the target color is not black for all frames starting at the target offset for the length of the source. Reference source and target by id.

```
// DLL Interface
EXPORT_API void PluginSubtractNonZeroTargetAllKeysAllFramesOffset(
    int sourceAnimationId, int targetAnimationId, int offset);

// Class Plugin
ChromaAnimationAPI::SubtractNonZeroTargetAllKeysAllFramesOffset(
    int sourceAnimationId, int targetAnimationId, int offset);
```

PluginSubtractNonZeroTargetAllKeysAllFramesOffsetName

Subtract the source color from the target color where the target color is not black for all frames starting at the target offset for the length of the source. Reference source and target by name.

```
// DLL Interface
EXPORT_API void PluginSubtractNonZeroTargetAllKeysAllFramesOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int offset);

// Class Plugin
```

```
ChromaAnimationAPI::SubtractNonZeroTargetAllKeysAllFramesOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int offset);
```

PluginSubtractNonZeroTargetAllKeysAllFramesOffsetNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSubtractNonZeroTargetAllKeysAllFramesOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double offset);

// Class Plugin
double result =
ChromaAnimationAPI::SubtractNonZeroTargetAllKeysAllFramesOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double offset);
```

PluginSubtractNonZeroTargetAllKeysOffset

Subtract the source color from the target color where the target color is not black from the source frame to the target offset frame. Reference source and target by id.

```
// DLL Interface
EXPORT_API void PluginSubtractNonZeroTargetAllKeysOffset(
    int sourceAnimationId, int targetAnimationId, int frameId, int offset);

// Class Plugin
ChromaAnimationAPI::SubtractNonZeroTargetAllKeysOffset(
    int sourceAnimationId, int targetAnimationId, int frameId, int offset);
```

PluginSubtractNonZeroTargetAllKeysOffsetName

Subtract the source color from the target color where the target color is not black from the source frame to the target offset frame. Reference source and target by name.

```
// DLL Interface
EXPORT_API void PluginSubtractNonZeroTargetAllKeysOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
offset);

// Class Plugin
ChromaAnimationAPI::SubtractNonZeroTargetAllKeysOffsetName(
    const char* sourceAnimation, const char* targetAnimation, int frameId, int
offset);
```

PluginSubtractNonZeroTargetAllKeysOffsetNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSubtractNonZeroTargetAllKeysOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double offset);

// Class Plugin
double result = ChromaAnimationAPI::SubtractNonZeroTargetAllKeysOffsetNameD(
    const char* sourceAnimation, const char* targetAnimation, double frameId,
    double offset);
```

PluginSubtractThresholdColorsMinMaxAllFramesRGB

Subtract all frames with the min RGB color where the animation color is less than the min threshold AND with the max RGB color where the animation is more than the max threshold. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSubtractThresholdColorsMinMaxAllFramesRGB(
    const int animationId, const int minThreshold, const int minRed, const int
minGreen,
    const int minBlue, const int maxThreshold, const int maxRed, const int
maxGreen,
    const int maxBlue);

// Class Plugin
ChromaAnimationAPI::SubtractThresholdColorsMinMaxAllFramesRGB(
    const int animationId, const int minThreshold, const int minRed, const int
minGreen,
    const int minBlue, const int maxThreshold, const int maxRed, const int
maxGreen,
    const int maxBlue);
```

PluginSubtractThresholdColorsMinMaxAllFramesRGBName

Subtract all frames with the min RGB color where the animation color is less than the min threshold AND with the max RGB color where the animation is more than the max threshold. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSubtractThresholdColorsMinMaxAllFramesRGBName(
    const char* path, const int minThreshold, const int minRed, const int
minGreen,
    const int minBlue, const int maxThreshold, const int maxRed, const int
```

```
maxGreen,
    const int maxBlue);

// Class Plugin
ChromaAnimationAPI::SubtractThresholdColorsMinMaxAllFramesRGBName(
    const char* path, const int minThreshold, const int minRed, const int
minGreen,
    const int minBlue, const int maxThreshold, const int maxRed, const int
maxGreen,
    const int maxBlue);
```

PluginSubtractThresholdColorsMinMaxAllFramesRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSubtractThresholdColorsMinMaxAllFramesRGBNameD(
    const char* path, double minThreshold, double minRed, double minGreen, double
minBlue,
    double maxThreshold, double maxRed, double maxGreen, double maxBlue);

// Class Plugin
double result =
ChromaAnimationAPI::SubtractThresholdColorsMinMaxAllFramesRGBNameD(
    const char* path, double minThreshold, double minRed, double minGreen, double
minBlue,
    double maxThreshold, double maxRed, double maxGreen, double maxBlue);
```

PluginSubtractThresholdColorsMinMaxRGB

Subtract the specified frame with the min RGB color where the animation color is less than the min threshold AND with the max RGB color where the animation is more than the max threshold. Animation is referenced by id.

```
// DLL Interface
EXPORT_API void PluginSubtractThresholdColorsMinMaxRGB(
    const int animationId, const int frameId, const int minThreshold, const int
minRed,
    const int minGreen, const int minBlue, const int maxThreshold, const int
maxRed,
    const int maxGreen, const int maxBlue);

// Class Plugin
ChromaAnimationAPI::SubtractThresholdColorsMinMaxRGB(
    const int animationId, const int frameId, const int minThreshold, const int
minRed,
    const int minGreen, const int minBlue, const int maxThreshold, const int
```

```
maxRed,
    const int maxGreen, const int maxBlue);
```

PluginSubtractThresholdColorsMinMaxRGBName

Subtract the specified frame with the min RGB color where the animation color is less than the min threshold AND with the max RGB color where the animation is more than the max threshold. Animation is referenced by name.

```
// DLL Interface
EXPORT_API void PluginSubtractThresholdColorsMinMaxRGBName(
    const char* path, const int frameId, const int minThreshold, const int minRed,
    const int minGreen, const int minBlue, const int maxThreshold, const int
maxRed,
    const int maxGreen, const int maxBlue);

// Class Plugin
ChromaAnimationAPI::SubtractThresholdColorsMinMaxRGBName(
    const char* path, const int frameId, const int minThreshold, const int minRed,
    const int minGreen, const int minBlue, const int maxThreshold, const int
maxRed,
    const int maxGreen, const int maxBlue);
```

PluginSubtractThresholdColorsMinMaxRGBNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginSubtractThresholdColorsMinMaxRGBNameD(
    const char* path, const int frameId, const int minThreshold, const int minRed,
    const int minGreen, const int minBlue, const int maxThreshold, const int
maxRed,
    const int maxGreen, const int maxBlue);

// Class Plugin
double result = ChromaAnimationAPI::SubtractThresholdColorsMinMaxRGBNameD(
    const char* path, const int frameId, const int minThreshold, const int minRed,
    const int minGreen, const int minBlue, const int maxThreshold, const int
maxRed,
    const int maxGreen, const int maxBlue);
```

PluginTrimEndFrames

Trim the end of the animation. The length of the animation will be the lastFrameId plus one. Reference the animation by id.

```
// DLL Interface
EXPORT_API void PluginTrimEndFrames(
    int animationId, int lastFrameId);

// Class Plugin
ChromaAnimationAPI::TrimEndFrames(
    int animationId, int lastFrameId);
```

PluginTrimEndFramesName

Trim the end of the animation. The length of the animation will be the lastFrameId plus one. Reference the animation by name.

```
// DLL Interface
EXPORT_API void PluginTrimEndFramesName(
    const char* path, int lastFrameId);

// Class Plugin
ChromaAnimationAPI::TrimEndFramesName(
    const char* path, int lastFrameId);
```

PluginTrimEndFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginTrimEndFramesNameD(
    const char* path, double lastFrameId);

// Class Plugin
double result = ChromaAnimationAPI::TrimEndFramesNameD(
    const char* path, double lastFrameId);
```

PluginTrimFrame

Remove the frame from the animation. Reference animation by id.

```
// DLL Interface
EXPORT_API void PluginTrimFrame(
    int animationId, int frameId);

// Class Plugin
```

```
ChromaAnimationAPI::TrimFrame(
    int animationId, int frameId);
```

PluginTrimFrameName

Remove the frame from the animation. Reference animation by name.

```
// DLL Interface
EXPORT_API void PluginTrimFrameName(
    const char* path, int frameId);

// Class Plugin
ChromaAnimationAPI::TrimFrameName(
    const char* path, int frameId);
```

PluginTrimFrameNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginTrimFrameNameD(
    const char* path, double frameId);

// Class Plugin
double result = ChromaAnimationAPI::TrimFrameNameD(
    const char* path, double frameId);
```

PluginTrimStartFrames

Trim the start of the animation starting at frame 0 for the number of frames. Reference the animation by id.

```
// DLL Interface
EXPORT_API void PluginTrimStartFrames(
    int animationId, int numberOfFrames);

// Class Plugin
ChromaAnimationAPI::TrimStartFrames(
    int animationId, int numberOfFrames);
```

PluginTrimStartFramesName

Trim the start of the animation starting at frame 0 for the number of frames. Reference the animation by name.

```
// DLL Interface
EXPORT_API void PluginTrimStartFramesName(
    const char* path, int numberOfFrames);

// Class Plugin
ChromaAnimationAPI::TrimStartFramesName(
    const char* path, int numberOfFrames);
```

PluginTrimStartFramesNameD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginTrimStartFramesNameD(
    const char* path, double numberOfFrames);

// Class Plugin
double result = ChromaAnimationAPI::TrimStartFramesNameD(
    const char* path, double numberOfFrames);
```

PluginUninit

Uninitializes the `ChromaSDK`. Returns 0 upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API RZRESULT PluginUninit();

// Class Plugin
RZRESULT result = ChromaAnimationAPI::Uninit();
```

PluginUninitD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginUninitD();

// Class Plugin
double result = ChromaAnimationAPI::UninitD();
```

PluginUnloadAnimation

Unloads Chroma effects to free up resources. Returns the animation id upon success. Returns negative one upon failure. Reference the animation by id.

```
// DLL Interface
EXPORT_API int PluginUnloadAnimation(int animationId);

// Class Plugin
int result = ChromaAnimationAPI::UnloadAnimation(int animationId);
```

PluginUnloadAnimationD

D suffix for limited data types.

```
// DLL Interface
EXPORT_API double PluginUnloadAnimationD(double animationId);

// Class Plugin
double result = ChromaAnimationAPI::UnloadAnimationD(double animationId);
```

PluginUnloadAnimationName

Unload the animation effects. Reference the animation by name.

```
// DLL Interface
EXPORT_API void PluginUnloadAnimationName(const char* path);

// Class Plugin
ChromaAnimationAPI::UnloadAnimationName(const char* path);
```

PluginUnloadComposite

Unload the the composite set of animation effects. Reference the animation by name.

```
// DLL Interface
EXPORT_API void PluginUnloadComposite(const char* name);

// Class Plugin
ChromaAnimationAPI::UnloadComposite(const char* name);
```

PluginUnloadLibrarySDK

Unload the Razer Chroma SDK Library before exiting the application.

```
// DLL Interface
EXPORT_API void PluginUnloadLibrarySDK();

// Class Plugin
ChromaAnimationAPI::UnloadLibrarySDK();
```

PluginUnloadLibraryStreamingPlugin

Unload the Razer Chroma Streaming Plugin Library before exiting the application.

```
// DLL Interface
EXPORT_API void PluginUnloadLibraryStreamingPlugin();

// Class Plugin
ChromaAnimationAPI::UnloadLibraryStreamingPlugin();
```

PluginUpdateFrame

Updates the `frameId` of the `Chroma` animation referenced by id and sets the `duration` (in seconds). The `color` is expected to be an array of the dimensions for the `deviceType/device`. The `length` parameter is the size of the `color` array. For `EChromaSDKDevice1DEnum` the array size should be `MAX LEDS`. For `EChromaSDKDevice2DEnum` the array size should be `MAX ROW` times `MAX COLUMN`. Keys are populated only for EChromaSDKDevice2DEnum::DE_Keyboard and EChromaSDKDevice2DEnum::DE_KeyboardExtended. Keys will only use the EChromaSDKDevice2DEnum::DE_Keyboard `MAX_ROW` times `MAX_COLUMN` keysLength.

```
// DLL Interface
EXPORT_API int PluginUpdateFrame(
    int animationId, int frameId, float duration, int* colors, int length, int*
keys,
    int keysLength);

// Class Plugin
int result = ChromaAnimationAPI::UpdateFrame(
    int animationId, int frameId, float duration, int* colors, int length, int*
keys,
    int keysLength);
```

PluginUpdateFrameName

Update the `frameId` of the `Chroma` animation referenced by name and sets the `duration` (in seconds). The `color` is expected to be an array of the dimensions for the `deviceType/device`. The `length` parameter is the

size of the `color` array. For `EChromaSDKDevice1DEnum` the array size should be `MAX_LEDS`. For `EChromaSDKDevice2DEnum` the array size should be `MAX_ROW` times `MAX_COLUMN`. Keys are populated only for EChromaSDKDevice2DEnum::DE_Keyboard and EChromaSDKDevice2DEnum::DE_KeyboardExtended. Keys will only use the EChromaSDKDevice2DEnum::DE_Keyboard `MAX_ROW` times `MAX_COLUMN` keysLength. Returns the animation id upon success. Returns negative one upon failure.

```
// DLL Interface
EXPORT_API int PluginUpdateFrameName(
    const char* path, int frameId, float duration, int* colors, int length, int*
keys,
    int keysLength);

// Class Plugin
int result = ChromaAnimationAPI::UpdateFrameName(
    const char* path, int frameId, float duration, int* colors, int length, int*
keys,
    int keysLength);
```

PluginUseForwardChromaEvents

On by default, `UseForwardChromaEvents` sends the animation name to `CoreSetEventName` automatically when `PlayAnimationName` is called.

```
// DLL Interface
EXPORT_API void PluginUseForwardChromaEvents(bool flag);

// Class Plugin
ChromaAnimationAPI::UseForwardChromaEvents(bool flag);
```

PluginUseIdleAnimation

When the idle animation flag is true, when no other animations are playing, the idle animation will be used. The idle animation will not be affected by the API calls to PluginIsPlaying, PluginStopAnimationType, PluginGetPlayingAnimationId, and PluginGetPlayingAnimationCount. Then the idle animation flag is false, the idle animation is disabled. `Device` uses `EChromaSDKDeviceEnum` enums.

```
// DLL Interface
EXPORT_API void PluginUseIdleAnimation(
    int device, bool flag);

// Class Plugin
ChromaAnimationAPI::UseIdleAnimation(
    int device, bool flag);
```

PluginUseIdleAnimations

Set idle animation flag for all devices.

```
// DLL Interface
EXPORT_API void PluginUseIdleAnimations(bool flag);

// Class Plugin
ChromaAnimationAPI::UseIdleAnimations(bool flag);
```

PluginUsePreloading

Set preloading animation flag, which is set to true by default. Reference animation by id.

```
// DLL Interface
EXPORT_API void PluginUsePreloading(
    int animationId, bool flag);

// Class Plugin
ChromaAnimationAPI::UsePreloading(
    int animationId, bool flag);
```

PluginUsePreloadingName

Set preloading animation flag, which is set to true by default. Reference animation by name.

```
// DLL Interface
EXPORT_API void PluginUsePreloadingName(
    const char* path, bool flag);

// Class Plugin
ChromaAnimationAPI::UsePreloadingName(
    const char* path, bool flag);
```

(End of automation)