INTINOR

WE ARE DIREKT

INTINOR DIREKT PLUGIN FOR ELGATO STREAM DECK

Version 2020-06-01

Intinor Direkt link and router are advanced products for robust video live streaming via Internet. Many features are available for the administrator from the web interface but it is often nice to provide the operator with just a few buttons to press instead.

Intinor has developed a plugin module for Elgato Stream Deck, using Direkt open REST API to monitor and control any features of Direkt link and Direkt router via a PC or MAC.

The Stream Deck module can access a Direkt unit either via Intinors managament system ISS or direct using a host name of a unit.

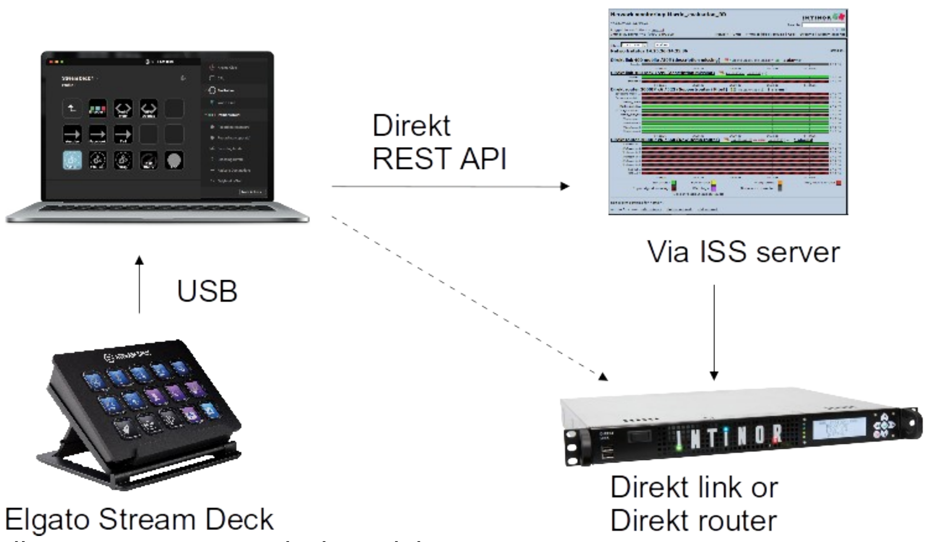
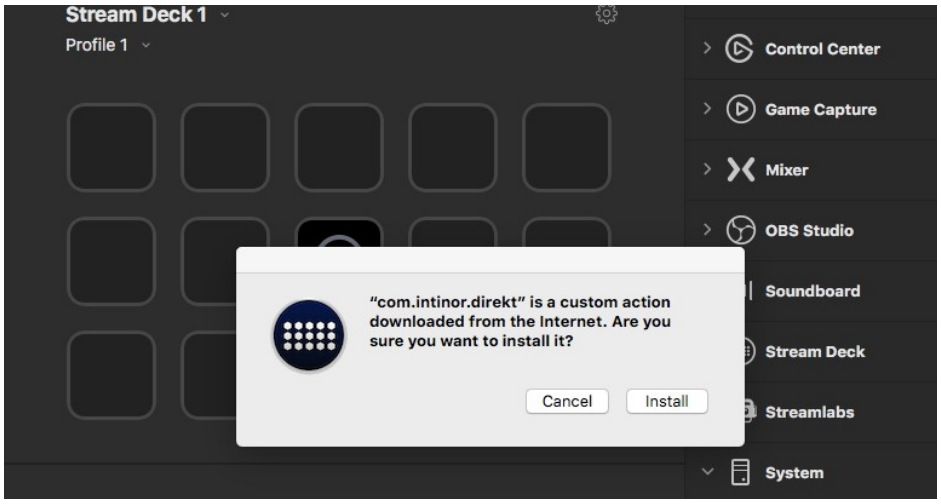


Illustration 1: Stream deck module on PC

INSTALLATION

Running the distributable `com.intinor.direkt.streamDeckPlugin` installs the Intinor Direkt Stream Deck plugin automatically.

On Windows, plugin will be installed in
`%appdata%\Elgato\StreamDeck\Plugins\com.elgato.StreamDeck\Plugins\com.intinor.direkt.sdPlugin`



"com.intinor.direkt" is a custom action downloaded from the Internet. Are you sure you want to install it?

Illustration 2: Installing the module

On Mac, plugin will be installed in
`~/Library/Application Support/com.elgato.StreamDeck/Plugins/com.intinor.direkt.sdPlugin`

When installed, the plugin should appear in the actions list in the Stream Deck application.

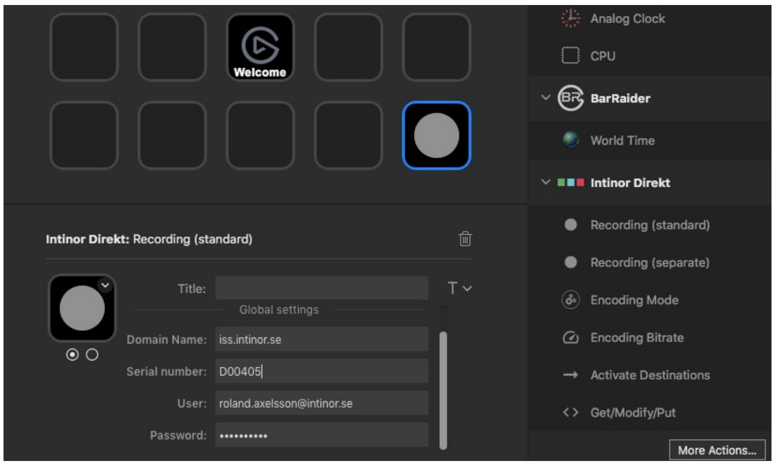
GLOBAL SETTINGS

Enter domain name (or IP) and serial number of the Direkt unit, and user name and password for API access. Global settings affect all actions of the Intinor Direkt Stream Deck plugin.

Example 1:
```
Domain name: iss.intinor.se
Serial number: D00858
User: myIssLogin
Password: ******
```

Example 2:
```
Domain name: direkt-858.intern.intinor.se
Serial number: D00858
User: admin
Password: ******
```

Accessing the Direkt API directly requires some configuration of security certificate both on the Direkt unit and on the machine running the Stream Deck software. Until the certificates are configured, it is possible to access the API via ISS. Remember that the ISS user must be mapped on the Direkt unit.



Intinor Direkt: Recording (standard)

Title:
Global settings
Domain Name: iss.intinor.se
Serial number: D00405
User: roland.axelsson@intinor.se
Password: ••••••••••

Illustration 3: Global settings

BASIC ACTIONS

Those actions are easy to use – just drag and drop to a button and configure the settings.

Recording (standard)

Starts and stops recording on the unit. Unit most not have the *separate_record_controls* option. Using Recording action in a multi-action allows the user to set a desired state, instead of toggling the recording state.

Recording (separate)

Starts and stops recording separately on an IP stream or encoder. Unit requires the *separate_record_controls* option.

Settings
Separate record control path: API path for the encoder or IP stream

Example: `/encoders/0`
Example: `/network_inputs/1`

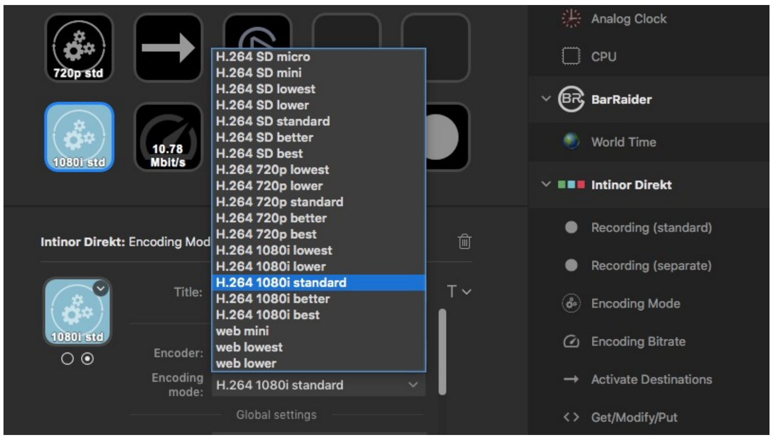
Encoding mode

Set the encoding mode of an encoder.

Settings
Encoder: API index of the encoder

Example: `0` (for the first encoder)

Encoding mode: Choose encoding mode from dropdown list.



Intinor Direkt: Encoding Mod
Title:
Encoder:
Encoding mode: H.264 1080i standard

Illustration 4: : Select encoding mode from list

The Encoding mode settings dropdown list is populated when the Stream Deck application plugin starts, or first time the property inspector of the action appears in the application. If the list does not appear correct, try entering correct global settings and then restarting the Stream Deck application.

Encoding Bitrate

Shows total bitrate of an encoder.

Settings
Encoder: API index of the encoder *Example:* `0`

Active Destinations

Toggle active state of destination matching a description, from an encoder or network stream.

Settings
Control path: API path for the encoder or network stream

Example: `/encoders/0`
Example: `/network_inputs/1`

Destination description: Description of the destinations to toggle

Using Activate Destination action in a multi-action allows the user to set a desired state, instead of toggling the state.

GET/MODIFY/PUT ACTION

Get/Modify/Put is a general purpose actions for modifying Direkt API resources with custom JavaScript scripts when the basic actions are not enough.

See example custom scripts in
`com.intinor.direkt.sdPlugin/getmodifyput_examples`

See `example_template.js` for a starting point and important comments on writing custom scripts.

Settings
Resource path: Relative path to the API resource to modify.

Example: `/recording/settings`

Custom script: File dialog to choose the custom script for the action instance

Example:
`com.intinor.direkt.sdPlugin/getmodifyput_examples/example_recording.js`

This example script toggles standard recording.

Custom args: Custom arguments can be sent to the custom script. This allows some data to be stored per button instance instead of being hardcoded in the script. Custom arguments are a comma separated list of *key: value* pairs. The example script change_video_mixer_source_profile.js uses custom arguments.

Writing custom scripts

The Get/Modify/Put action gets a resource from the from the Direkt API, modifies the resource according to a custom script, and puts the modified resource back to the API.

Custom scripts are written in the JavaScript language and should implement a `modify()` function. This function modifies the retrieved API resource on an action instance key press. The `modify()` function should take an argument, the retrieved resource as a javascript object. The function should return the modified resource object.

```
function modify(api_resource) {
    //Modify the api resource here
    return api_resource;
}
```

If the `modify()` funcion is omitted from the script, the retrieved resource is put back unmodified.

Custom scripts may implement a `verify()` function. This function sets the visual state of the Stream Deck key, based on the API resource from the put response. This function is also used to update state of the key from periodic polling of the resource. The `verify()` function should take an argument, the retrieved resource from a successfull put, as a javascript object. `verify()` should return false or true based on inspection of the argument object, to set corresponding state of the key.

```
function verify(api_resource) {
    //Inspect resource object here
   return true;
}
```

If the `verify()` function is omitted from the script, no verification of the resource from put response or polling is done.

Errors thrown by `modify()` or `verify()` functions are caught and result in a warning triangle briefly displaying on the Stream Deck key.

Outside the `modify()` or `verify()` functions custom scripts may also define other helper functions or variables. It is important that variables in a custom script are always declared with a `let`, `const` or `var` qualifier. Assigning values to undeclared variables implicitly places the variable in the global scope of the javascript execution. This can potentially interfere with the operation of the plugin in case of accidental name collisions.

One global object of interest is `$SD`. This object contains methods for communicating with the Stream Deck. Custom scripts are passed a `context` wich lets the script send messages to the Stream Deck key. For instance, calling `$SD.api.setTitle(context, "Hello");` sets the title of the key. This can be used to display text information on the Stream Deck. The show encoder bitrate example is implemented this way. (To see the set title of a key, the title must not be set manually from Stream Deck control application.) For more details, see file `js/common.js` and https://developer.elgato.com/documentation/stream-deck/sdk/events-sent/

Custom scripts can also be passed custom arguments set on the key configuration. Custom arguments may be set in the *Custom args* settings field in the property inspector for the key, in the Stream Deck application. The *Custom args* field takes a comma separate list of *key: value* pairs.

From this list is created a javascript object that is passed to the custom script. For example, *Custom args*: `x: 5, hello: "world"` passes the object

```
{
  x: 5,
  hello: "world",
}
```

to the script. The script can access the custom arguments object through the variable `custom_args`. It is worth noting, lest one forgets, that strings must be entered with quotation marks in the *Custom args* field.

Examples of custom script implementations can be found in the directory `getmodifyput_examples`

Writing custom scripts

This is to give the user a better idea of what is going on under the hood.

The custom script interface is implemented internally by wrapping the script in a function that is then executed in the global scope. When a custom script looks like this:

```
function modify(settings) {/*...*/}

function verify(settings) {/*...*/}
```

what the implementation sees looks more like this:

```
function(context, custom_args) {
    'use strict';
    function modify(settings) {/*...*/}
    function verify(settings) {/*...*/}
    return [modify, verify];
}
```

The custom script is executed on key press or when API polling starts. The `modify()` and `verify()` functions are returned to the plugin and then executed at their appropriate moments in the get/modify/put cycle.

Of course, for anyone interested in further details the source code of the plugin is available.

Debugging

A debug console and debug tools are accessible through the Chrome browser.

Remote debugging must first be enabled in Stream Deck

On macOS run the command

```
defaults write com.elgato.StreamDeck html_remote_debugging_enabled -bool
YES
```

On Windows add a DWORD `html_remote_debugging_enabled` with value `1` in the registry `@ HKEY_CURRENT_USER\Software\Elgato Systems GmbH\StreamDeck`.

After restarting the Stream Deck application, navigate to http://localhost:23654/ in Chrome, and click com.intinor.direkt.

See https://developer.elgato.com/documentation/stream-deck/sdk/create-your-own-plugin/

NOTES

Remember to commit changes in text fields by pressing enter (or marking another text field in the same action property inspector). Text field with uncommited changes appear with a red border around them.

Changing settings of buttons require that the button must be reinitialized. This can be done by switching profiles in the Stream Deck app. During configuration of a profile it is convenient to assign an empty folder to one button, for easy reinitialisation of buttons.

Polling interval for state update of buttons is two seconds between polls.

Because of the asynchronous nature of API communication, using actions in multi-actions may fail if the actions try to modify the same API resource. A workaround for this is to insert sleep between actions. This is not reliable because there are no guarantees how long a network call to the API can take.

The Intinor Direkt Stream Deck plugin is written in JavaScript. For general information on writing plugins for the Stream Deck, see:
https://developer.elgato.com/documentation/stream-deck/sdk/overview/

Stream deck is a USB-connected interactive control panel developed by Elgato
www.elgato.com