

# Ludovic Rousseau's blog

My activities related to smart card and Free Software (as in free speech).

Tuesday, March 14, 2017

## PC/SC sample in Smart Card Connector on Chromebook

To continue the list of PC/SC wrappers initiated in 2010 with "[PC/SC sample in different languages](#)" I now present a sample in Smart Card Connector on Chromebook (or Chrome browser).

### Smart Card Connector

Smart Card Connector is a Chrome extension developed by Google and available at <https://chrome.google.com/webstore/detail/smart-card-connector/khpfeaanjngmcnplbdlpegiifgpgdco>.

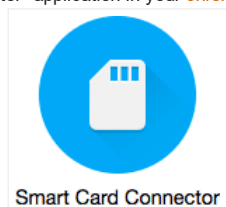
It allows to use the PC/SC API from a JavaScript application in a [Chromebook](#). The project is a port of pcsc-lite, libccid and libusb to ChromeOS and is available at [chromeos\\_smart\\_card\\_connector](#) under the Apache v2 license.

I do not have a Chromebook myself but it is possible to use the Chrome browser instead (with some limitations).

### Installation

Go to <https://chrome.google.com/webstore/detail/smart-card-connector/khpfeaanjngmcnplbdlpegiifgpgdco> and select "Add to Chrome".

You should see the "Smart Card Connector" application in your <chrome://apps/>.



If you click on the application icon you should see something like:

#### Google+ Badge

Ludovic Rousseau blog



Follow

#### Blog Archive

- ▼ 2017 (33)
  - November (1)
  - October (3)
  - September (4)
  - August (1)
  - June (2)
  - May (5)
  - April (1)
  - ▼ March (6)
    - "PC/SC" sample in Objective-C (synchronous)
    - Gemalto IDBridge K30, K50, CT30 and Zero Length Pa...
    - PC/SC sample in Rust
    - ATR statistics: TC2 - Specific to T=0
    - PC/SC sample in Smart Card Connector on Chromebook...
    - macOS Sierra bug: SCardTransmit() silently truncat...
- February (4)
- January (6)
- 2016 (49)
- 2015 (51)
- 2014 (61)
- 2013 (38)
- 2012 (27)
- 2011 (46)
- 2010 (55)

#### Search This Blog

Search

#### Subscribe To

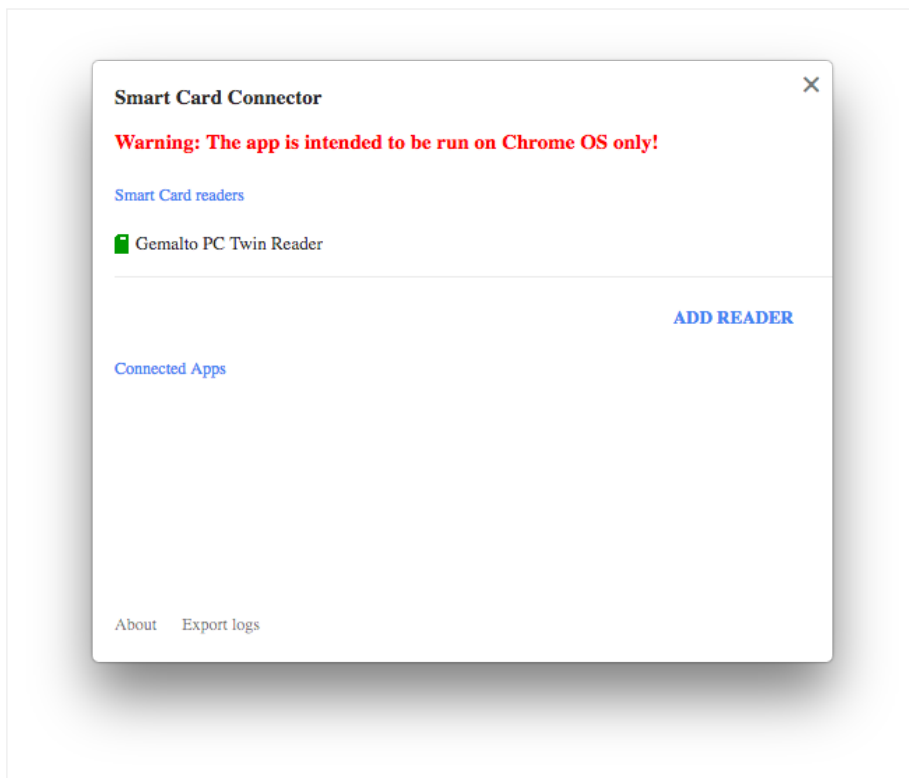
Posts



Comments



#### Google+ Followers



Since the "Smart Card Connector" application completely replaces pcsc-lite and the CCID driver the normal pcsc-lite provided by the system must be stopped. See "[Troubleshooting Apps under desktop OSes](#)" to know how to stop pcscd.

## Sample application

I created a simple test application for ChromeOS. The application is available from [google-smart-card-client-library-hello-world](#).

You need to fetch a client library [google-smart-card-client-library.js](#) from Google and also [jQuery](#). Just follow the [Building](#) instructions.

My sample application is greatly inspired from the [Hello World](#) Chrome application and [Example JavaScript Smart Card Client app](#). My contribution is (mainly) the last 4 JavaScript functions.

## Sample source code

I only provide the source code for the file [pcsc\\_appli.js](#).

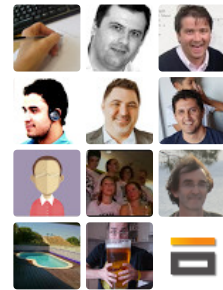
```
/** @license
 * Copyright 2017 Google Inc. ALL Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * Limitations under the License.
 */

/**
 * @fileoverview Entry point of the Smart Card Client App background script (see
 * https://developer.chrome.com/apps/event\_pages).
 */

/**
 * Client title for the connection to the server App.
```

Ludovic Rousseau b...

Follow



336 have us in circles

[View all](#)

```

*
* Currently this is only used for the debug Logs produced by the server App.
* @const
*/
var CLIENT_TITLE = 'example_js_client_app';

/**
 * Context for talking to the Smart Card Connector app for making PC/SC API
 * requests.
 * @type {GoogleSmartCard.PcscLiteClient.Context}
 */
var apiContext = null;

/**
 * Object that allows to make PC/SC API requests to the Smart Card Connector
 * app.
 * @type {GoogleSmartCard.PcscLiteClient.API}
 */
var api = null;

var API = GoogleSmartCard.PcscLiteClient.API;

/**
 * PC/SC-Lite SCard context.
 * @type {int}
 */
var sCardContext = null;

function initialize() {
  myLog('Establishing connection to the Connector app...');
  console.log('Establishing connection to the Connector app...');
  apiContext = new GoogleSmartCard.PcscLiteClient.Context(CLIENT_TITLE);
  apiContext.addOnInitializedCallback(onInitializationSucceeded);
  apiContext.addOnDisposeCallback(contextDisposedListener);
  apiContext.initialize();
}

function onInitializationSucceeded(constructedApi) {
  myLog('Successfully connected to the Connector app');
  console.log('Successfully connected to the Connector app');
  api = constructedApi;
  establishContext();
}

function establishContext() {
  myLog('Establishing PC/SC-Lite context...');
  console.log('Establishing PC/SC-Lite context...');
  api.SCardEstablishContext(
    GoogleSmartCard.PcscLiteClient.API.SCARD_SCOPE_SYSTEM, null, null).then(
    function(result) {
      result.get(onContextEstablished, onPcscLiteError);
    }, onRequestFailed);
}

/** @param {int} establishedSCardContext PC/SC-Lite SCard context. */
function onContextEstablished(establishedSCardContext) {
  myLog('Established PC/SC-Lite context ' + establishedSCardContext);
  console.log('Established PC/SC-Lite context ' + establishedSCardContext);
  sCardContext = establishedSCardContext;
  listReaders();
}

function listReaders() {
  myLog('Obtaining list of PC/SC-lite readers...');
  console.log('Obtaining list of PC/SC-lite readers...');
  api.SCardListReaders(sCardContext, null).then(function(result) {
    result.get(onReadersListed, onPcscLiteError);
  }, onRequestFailed);
}

```

```

/** @param {!Array.<string>} readers List of reader names. */
function onReadersListed(readers) {
  myLog('List of PC/SC-Lite readers: ' + readers);
  console.log('List of PC/SC-Lite readers: ' + readers);

  // Use the 1st reader
  reader = readers[0];
  myLog('Using reader: ' + reader);

  myCode(reader);
}

function contextDisposedListener() {
  myLog('Connection to the server app was shut down');
  console.warn('Connection to the server app was shut down');
  sCardContext = null;
  api = null;
  apiContext = null;
}

/** @param {int} errorCode PC/SC-Lite error code. */
function onPcscLiteError(errorCode) {
  myLog('PC/SC-Lite request failed with error code ' + errorCode);
  console.warn('PC/SC-Lite request failed with error code ' + errorCode);
}

/** @param {*} error The exception that happened during the request. */
function onRequestFailed(error) {
  myLog('Failed to perform request to the Smart Card Connector app: ' +
    error);
  console.warn('Failed to perform request to the Smart Card Connector app: ' +
    error);
}

function myLog(text)
{
  $("#logs").append($("#<li>").text(text));
}

function myCode(readerName)
{
  myLog('Connect to the reader: ' + readerName);
  api.SCARDConnect(sCardContext,
    readerName,
    API.SCARD_SHARE_SHARED,
    API.SCARD_PROTOCOL_ANY).then(function(result) {
      result.get(onConnected, onPcscLiteError);
    }, onRequestFailed);
}

function onConnected(cardHandle, protocol)
{
  APDU_SELECT = [0x00, 0xA4, 0x04, 0x00, 0x0A, 0xA0, 0x00, 0x00, 0x00,
    0x62, 0x03, 0x01, 0x0C, 0x06, 0x01];

  myLog('Sending ' + dump(APDU_SELECT));
  api.SCARDTransmit(
    cardHandle,
    protocol == API.SCARD_PROTOCOL_T0 ?
      API.SCARD_PCI_T0 : API.SCARD_PCI_T1,
    APDU_SELECT).then(function(result) {
      result.get(function(ioRecvPci, response) {
        myLog('response: ' + dump(response));

        // get the 2 last bytes
        sw = response.slice(-2);
        if (sw[0] == 0x90 && sw[1] == 0x00)
        {

```

```

APDU_COMMAND = [0x00, 0x00, 0x00, 0x00];
myLog('Sending ' + dump(APDU_COMMAND));
api.SCardTransmit(
  cardHandle,
  protocol == API.SCARD_PROTOCOL_T0 ?
  API.SCARD_PCI_T0 : API.SCARD_PCI_T1,
  APDU_COMMAND).then(function(result) {
    result.get(function(ioRecvPci, response) {
      myLog('response: ' + dump(response));

      // get the 2 last bytes
      sw = response.slice(-2);
      if (sw[0] == 0x90 && sw[1] == 0x00)
      {
        // convert to an ASCII string
        result = "";
        for (var i = 0; i < response.length; i++)
        {
          result += String.fromCharCode(response[i]);
        }
        myLog('response: ' + result);
      }
    }, onPcscLiteError);
  }, onRequestFailed);
}, onPcscLiteError);
}, onRequestFailed);

function dump(bytes)
{
  return (bytes.map(function (x) {
    return ('00' + x.toString(16).toUpperCase()).substr(-2)
  })).join(" ");
}

window.onload = initialize;

```

### Remarks

This API is very verbose and low level. You can compare it to the node-pcsc-lite project API, also in JavaScript, I used in a previous article ["PCSC sample in JavaScript \(Node.js\)"](#).

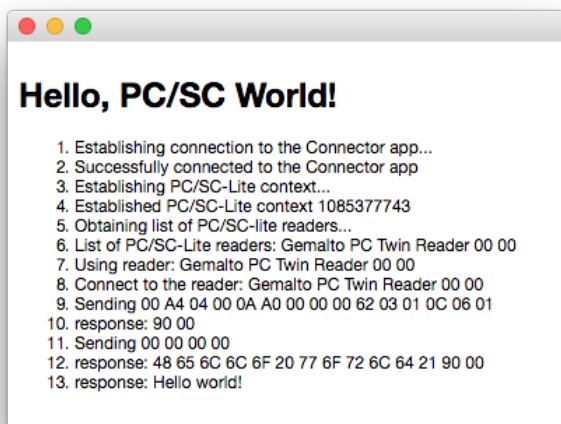
The API uses a lot of call back functions. But that is not surprising for a JavaScript code.

### Installation

To install the application in Chrome go to <chrome://extensions/> and click on the "Load non packaged extension" button (label translated from French so the real English text may be different) and select the root directory of the sample application.

You should then see a new "Hello World PC/SC" extension in the list. Click on the "Run" link to start the extension. A new window should be created as displayed in the Output section below.

### Output



## Conclusion

This API is useful mainly/only on Chromebook computers. I guess it is the only smart card interface for this kind of computer.

If you know a PC/SC wrapper that is not yet in [my list](#) then please contact me.



Labels: [code](#)

[Newer Post](#)

[Home](#)

[Older Post](#)

Bitcoin



License: [by-nc-sa](#)



This blog by [Ludovic Rousseau](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](#).

Simple theme. Powered by [Blogger](#).