



Circle Web Smart Card SDK Developer Guide

Version 1.00 | March 2025



Revision History

Version	Date	Details
1.00	31/3/2025	Initial release for SDK v1.0.0.0



Contents

- 1.0. Introduction 4
 - 1.1. Purpose4
 - 1.2. Glossary4
- 2.0. System Overview 5
 - 2.1. How it works5
 - 2.2. Data Flow5
- 3.0. SDK Overview and Components 6
 - 3.1. Prerequisites6
 - 3.2. SDK Overview6
 - 3.3. SDK Components 7
 - 3.3.1. Circle Web Smart Card Service7
 - 3.3.2. Web Browser JavaScript Library9
 - 3.3.2.1. Functions9
 - 3.3.2.2. How to Use the JavaScript Library9
 - 3.3.2.3. Key Features9
 - 3.3.3. Circle Web Smart Card Demo Website 10
 - 3.3.3.1. File Structure10
 - 3.3.3.2. How to Run the Demo Website10
 - 3.3.3.3. Layout and Key Features 11
 - 3.3.4. Configuring the Circle Web Smart Card Service 12
 - 3.3.4.1. Configuration Parameters 12
 - 3.3.4.2. How to Change the Configuration Parameters? 13
- 4.0. Example Usage14



1.0. Introduction

1.1. Purpose

The Circle Web Smart Card SDK is a comprehensive solution for system integrators and software developers to seamlessly integrate Circle smart card readers with web applications on Windows platform. It enables real-time communication between smart cards and web browsers using WebSockets, allowing for monitoring of reader status, connection/disconnection of readers, sending of APDU commands, and configuration of reader settings (Escape commands) from a web browser.

The Circle Web Smart Card SDK consists of three key components:

- Circle Web Smart Card Service
- Web Browser JavaScript Library
- Circle Web Smart Card Demo Website

Should you have questions, please contact support@abcircle.com.

1.2. Glossary

Term	Description
APDU	Application Protocol Data Unit
API	Application Programming Interface
CPU	Central Processing Unit
PC/SC	Personal Computer/Smart Card
SDK	Software Development Kit



2.0. System Overview

2.1. How it works

- The **backend** (Circle Web Smart Card Service) is a Windows service that communicates with smart card readers via PC/SC APIs.
- The **frontend** (Web Browser JavaScript Library) interacts with the backend through WebSockets to send and receive data in real time.

2.2. Data Flow

1. The Circle Web Smart Card Service detects and manages smart card readers connected to the computer.
2. The backend opens a WebSocket server (`ws://127.0.0.1:55002`) that allows web browsers to communicate with smart card readers.
3. The frontend (`SmartCardReader.js`) connects to the WebSocket server, allowing JavaScript to send commands (`connect`, `disconnect`, `send APDU/escape commands`).
4. The backend polls the smart card reader every 100ms and sends status updates back to the frontend if card is presented or removed.
5. The web application, e.g. Circle Web Smart Card Demo Website, updates the UI dynamically based on real-time reader status.



3.0. SDK Overview and Components

3.1. Prerequisites

- **Operating System:** Windows 8.1 and above
- **Web browser:** Any web browser with JavaScript support, e.g. Google Chrome, Microsoft Edge and Mozilla Firefox.
- **Smart card reader:** Any Circle smart card reader compatible with Windows (For details, refer to www.abcircle.com)
- **Smart card reader driver:** Installation of corresponding reader driver is required (All Circle smart card reader drivers can be downloaded from www.abcircle.com)

3.2. SDK Overview

The Circle Web Smart Card SDK is provided as a single package (CircleWebSC_SDK_vx.x.x.x). It includes three main components, distributed across three folders.

1. Circle Web Smart Card Service

- Location: CircleWebSCService_Installer_vx.x.x.x folder
- Description: A Windows background service that manages communication between web browsers and smart card readers. It uses PC/SC API to interact with the hardware and exposes a WebSocket server (ws://127.0.0.1:55002) for real-time interaction with web applications. This service must be installed and running on the user's computer for the web applications to function properly.

2. Web Browser JavaScript Library (SmartCardReader.js)

- Location: CircleWebSC_JS_Library_vx.x.x.x folder
- Description: A JavaScript library designed to facilitate the integration of smart card reader interactions into web applications. It communicates with the backend via WebSockets to transmit commands and receive real-time card status updates. This library enables developers to efficiently incorporate smart card communication into their web applications.

3. Circle Web Smart Card Demo Website

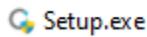
- Location: CircleWebSC_Demo_Website_vx.x.x.x folder
- Description: A sample web application that demonstrates how to use SmartCardReader.js to interact with smart card readers. It showcases the SDK's capabilities, including card connection/disconnection, sending APDU and escape commands, receiving responses, and handling real-time status updates. This demo serves as a reference for developers integrating smart card functionality into their web applications.

3.3. SDK Components

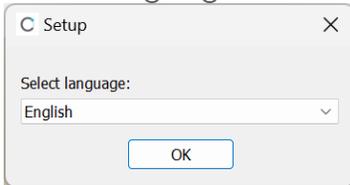
3.3.1. Circle Web Smart Card Service

This section illustrates the steps to install the Circle Web Smart Card Service.

1. Open the folder "CircleWebSCService_Installer_vx.x.x.x".
2. Run the installer "Setup.exe" and follow the on-screen instructions.



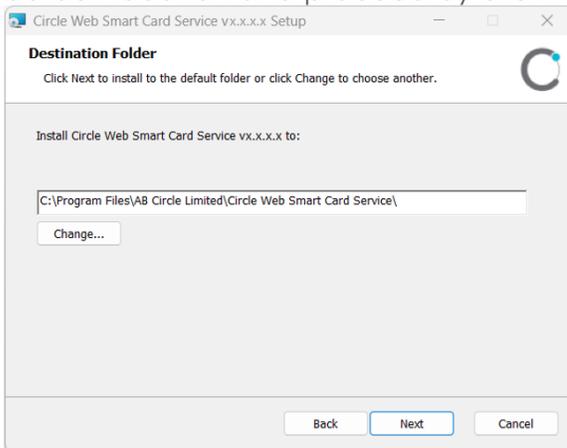
3. Select language and click "OK".



4. Click "Next" on the welcome screen.

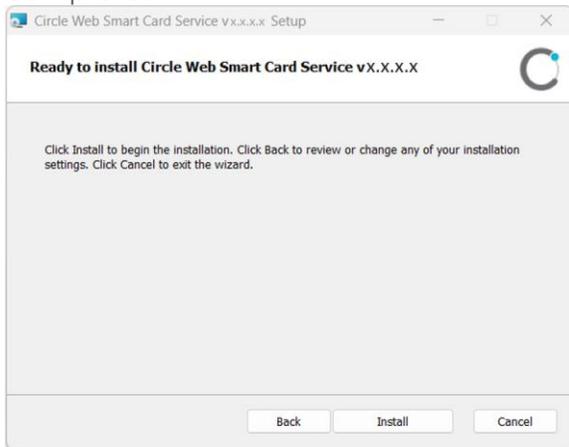


5. Click "Change" to choose the installation directory if needed, or accept the default location and proceed by clicking "Next".





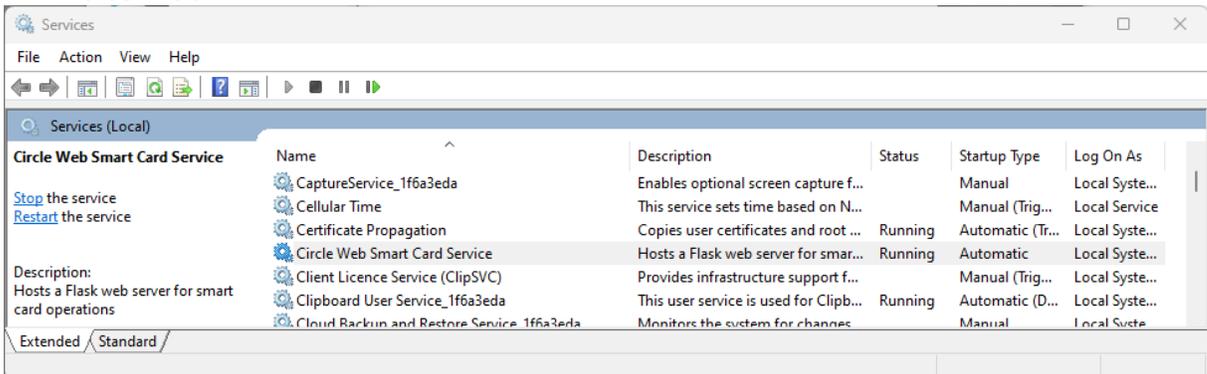
- Click "Install" to begin the installation process and wait for the progress bar to complete.



- Click "Finish" to close the installer. Once installed, the service starts automatically and appears in Windows Services.



- Verify the service is running:
 - Search for "Services" in the Windows Start menu to open the Service window.
 - Look for "Circle Web Smart Card Service" and ensure its status is "Running." If it is not running, restart the computer or right-click to select "Start" to manually start the service.





3.3.2. Web Browser JavaScript Library

This section provides an overview and usage guidelines of the Web Browser JavaScript Library (SmartCardReader.js) to help developers implementing smart card support into their web applications.

3.3.2.1. Functions

- Support multiple readers
- Auto-detect smart cards, i.e. card presentation or removal.
- Connect/disconnect readers
- Send APDU/escape commands

3.3.2.2. How to Use the JavaScript Library

1. Import the library
`import SmartCardReader from './SmartCardReader.js';`
2. Create an Instance
`const smartCardReader = new SmartCardReader('ws://127.0.0.1:55002');`
3. Retrieve a list of available smart card readers
`const readers = await reader.getReaders();`
4. Connect to a Reader
`await reader.connect(reader1, 'shared', 't1');`
5. Send an APDU Command
`const response = await reader.sendAPDUcommand('00A40400', 'reader1');`
6. Send a Control (Escape) command to a connected reader
`const response = await reader.sendControlcommand('FF00010000', 'reader1');`
7. Disconnect from a Reader
`await reader.disconnect(reader1);`

3.3.2.3. Key Features

- **Handling Asynchronous Operations:**
The functions in SmartCardReader.js are asynchronous, meaning they return a Promise. To ensure proper execution, developers should use `await` when calling these functions inside an `async` function.
- **Understanding `messageId` for WebSocket Communication:**
SmartCardReader.js uses `messageId` to track individual WebSocket requests and match them with the correct response. This is important because multiple smart card readers might be communicating at the same time.



Example message Flow:

```
// Sent request
{
  "endpoint": "/reader_APDUcommand",
  "messageId": 17121234123,
  "payload": { "command": "00A40400", "reader_name": "Reader1" }
}

// Received response
{
  "type": "response",
  "messageId": 17121234123,
  "response": "9000" // Success response from the smart card
}
```

3.3.3. Circle Web Smart Card Demo Website

This section outlines the key features of the Circle Web Smart Card Demo Website, providing instructions on how to run it and change its configurations.

3.3.3.1. File Structure

- **index.html:** Main entry point for the web interface.
- **styles.css:** Styles for the user interface.
- **CircleWebDemo.js:** Demonstration of SDK integration and smart card interactions.
- **SmartCardReader.js:** Core JavaScript library for handling smart card communication.

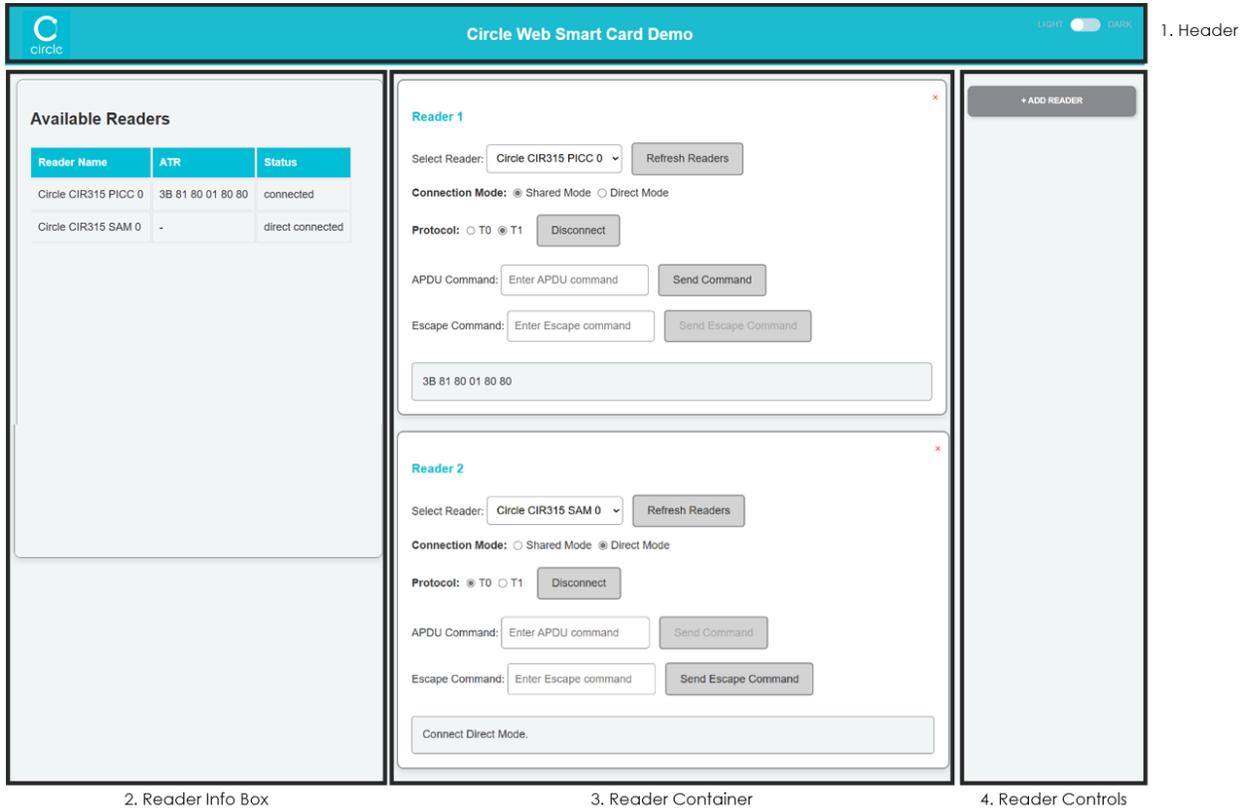
3.3.3.2. How to Run the Demo Website

1. Ensure the Circle Web Smart Card Service is running. (See section 3.3.1 for more details.)
2. Do not open index.html directly due to browser security restrictions (CORS policy error).
3. Instead, start a local web server by using Python:
python -m http.server 8000
4. Open a web browser and navigate to http://localhost:8000/ (or the port used in your local web server).



3.3.3.3. Layout and Key Features

The demo website consists of four sections, each with features described below:



- Header:**
Display the Circle logo and demo title, with a light/dark mode toggle.
- Reader Info Box:**
Show real-time status of all detected readers automatically, including
 - Reader Name(s)
 - ATR(s) if a card is presented, and
 - Status of each detected reader interface.
- Reader Container:**
Interact with individual readers for connection and sending commands. The following functions can be performed.
 - **Reader Selection:**
 - Select Reader: Choose a reader from the dropdown.
 - Refresh Readers: Update the reader(s) in the dropdown list, if needed.
 - **Reader Connection:**
 - Connection Mode: Select "Shared Mode" (multiple apps can access) or "Direct Mode" (exclusive); "T0" or "T1" protocol
 - Connect/Disconnect: Click "Connect" to link to the selected reader; "Disconnect" to release it.



- **Reader Communication:**
 - Send APDU Command: Enter a hex command, e.g. 00 A4 04 00 and click "Send Command".
 - Send Escape Command: Enter a hex control command, e.g. FF 00 01 00 00, and click "Send Escape Command".
 - Corresponding response to the sent command is then displayed in the response area
- **Removing the reader box:** Click "x" in the top right corner (pop-up for confirmation).

4. Reader Controls:

Manage the addition of new reader boxes.

- **Adding a new reader box:** Click "+ Add Reader" to create a new box for establishing a reader connection and communication. Multiple boxes can be created and used at the same time.

3.3.4. Configuring the Circle Web Smart Card Service

The Circle Web Smart Card Service allows customization of certain settings through the 'config.ini' file for the web application. This file enables the modification of key parameters, including the WebSocket service port, polling interval, and logging preferences.

After installation of Circle Web Smart Card Service, the "config.ini" file is located in the same directory as the executable (.exe). The default location is:

C:\Program Files\AB Circle Limited\Circle Web Smart Card Service\config.ini

3.3.4.1. Configuration Parameters

The following four parameters can be configured in the "config.ini".

```

config.ini
File Edit View
[Service]
1 port = 55002
2 polling_interval = 0.1

[Logging]
3 rotate = no
4 level = INFO

```

1. Port
 - This specifies the port number on which the WebSocket server listens for connections from the frontend. The default is 55002.
 - The port can be changed if 55002 is already in use by another application on the system
 - Port must be between 1024 and 65535 to avoid reserved system ports.



2. Polling interval
 - This defines how often (in seconds) the service checks for changes in smart card status, i.e. card presentation or removal. The default is 0.1 seconds.
 - A smaller interval, e.g. 0.05, provides faster updates but increases CPU usage. A larger interval, e.g. 0.5, reduces CPU load but delays status updates.
3. Log rotates
 - This setting controls whether log rotation is enabled. The default is "no", i.e. not currently implemented.
 - Setting to "yes" would rotate logs to prevent them from growing excessively large.
4. Log level
 - This sets the verbosity of the logs. Four options are available.
 - (i) DEBUG: Log all details, useful for troubleshooting.
 - (ii) INFO: Log important events, suitable for normal operation. (Default)
 - (iii) WARNING: Log warnings and errors, capturing potential issues without excessive details.
 - (iv) ERROR: Log only errors, minimizing log output.
 - The log files can be found at `C:\Windows\System32\logs` and are used to monitor the operation and diagnose any issues with the web application.

3.3.4.2. How to Change the Configuration Parameters?

To modify "config.ini", follow the steps below:

1. Navigate to the directory where the Circle Web Smart Card Service is installed.

The default location is:

`C:\Program Files\AB Circle Limited\Circle Web Smart Card Service\config.ini.`

2. Open "config.ini" using a text editor. If editing the file, administrator privileges are required:
Search for "Notepad" in the Windows Start menu, right-click and select "Run as administrator" to open it, then open "config.ini" within text editor.
3. Modify the parameters and adjust the settings as needed.
4. Save the file.
5. Restart the service for the changes to apply:
 - Search for "Services" in the Windows Start menu to open the Services window.
 - Look for "Circle Web Smart Card Service", right-click and select "Restart".



4.0. Example Usage

Below is a sample script (CircleWebDemo.js) using the library:

```
// Import the SmartCardReader library for smart card interactions
import SmartCardReader from './SmartCardReader.js';
// Initialize the SmartCardReader instance with the WebSocket server URL
const smartCardReader = new SmartCardReader('ws://127.0.0.1:55002');

async function init() {
  // List available readers
  const readers = await reader.getReaders();
  console.log('Available Readers:', readers);

  // Connect to the first reader
  if (readers.length > 0) {
    const response = await reader.connect(readers[0], 'shared', 't1');
    console.log('Connect Response:', response);

    // Send an APDU command
    const apduResponse = await reader.sendAPDUcommand('00A40400',
readers[0]);
    console.log('APDU Response:', apduResponse);

    // Disconnect
    await reader.disconnect(readers[0]);
    console.log('Disconnected');
  }
}

init().catch(console.error);
```