WebNFC Bridge

WebNFC[1] Use-case for Secure Interaction Between
an Untrusted Web-page and a Connecting Mobile Device

The following *conceptual specification* builds on the same core as the Web2Native Bridge[2]. In fact, the *intention* is that invoked native applications would be *identical* for both schemes.

On the Web-side there would be minor differences, since NFC and local application invocation have distinct JavaScript interfaces, whereas the actual *application code* should be identical, including error handling.

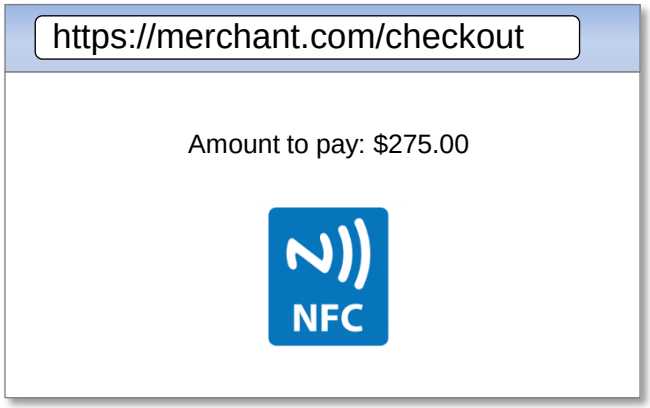
To create a good user experience, the actual transactions are supposed to be carried out over BLE (Bluetooth Low Energy). That is, NFC is only used for BLE paring and naming the application to invoke.

Note that this specification does not include a security element since such functionality can be supplied in many different ways when needed.

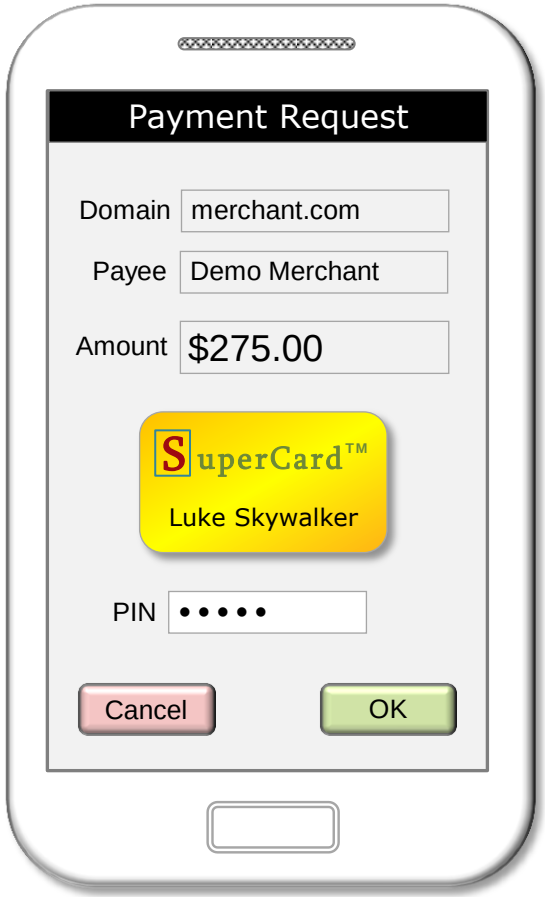
1. https://www.w3.org/community/web-nfc/
2. https://cyberphone.github.io/openkeystore/resources/docs/web2native-bridge.pdf

WebNFC Bridge – Typical Use Case

① User interacts with a Web application on a PC, ATM, POS terminal, Vending machine, etc.

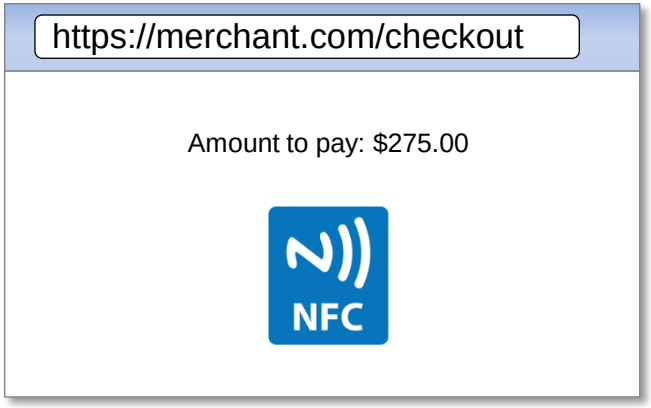


② User performs the NFC connection



③ User finishes the request in the securely connected mobile device

WebNFC Bridge – Web Side Agent



The WebNFC Bridge depends on a JavaScript extension in Web-browsers that enables ordinary (untrusted) web-pages to write NFC NDEF records to an NFC adapter available in the PC, POS, etc.

The information in the NDEF records minimally include:

- The name of the application to invoke in the connecting mobile device
- BLE pairing data including a random number used as password
- Domain data of the requesting Web-page (URL + Server certificate path)

The JavaScript extension is implemented in a Web-side agent which also deals with the BLE setup and communication.

After successful BLE setup, the Web-side agent forwards incoming BLE data to the originating Web-page as well as sending data going out from the Web-page back to the device.

In a calling Web-page the channel is expressed through **`addEventListener`** and **`postMessage`** methods used in many other places in a Web-browser.

The created channel is opaque (with respect to the Web-side agent), because the interface is deliberately application-neutral. The exact meaning of the data is thus a convention between the invoked application in the connecting device and the requesting Web-page.

WebNFC Bridge – Device Side Agent

The Device-side agent listens to incoming NFC NDEF records. This agent is a security-critical system since it eventually invokes a native (local) application without necessary asking the user first. This works because applications are supposed to be vetted for invocation from an untrusted Web.

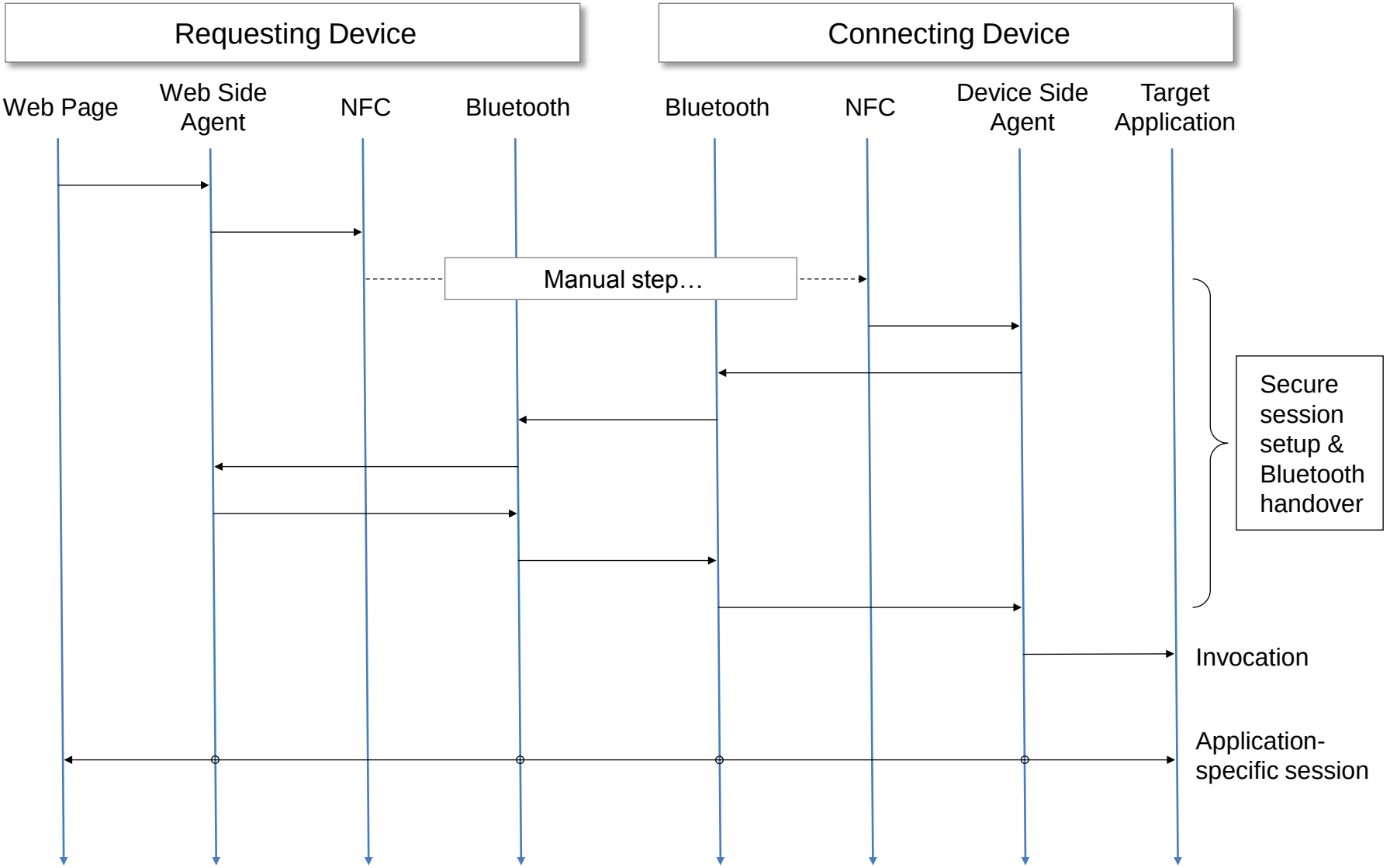
Before the application named in the NDEF record is invoked, the Device-side agent setups the channel using BLE.

The invocation of the target application minimally includes the following data:

- NFC device information used for logging purposes
- Handle to the communication channel interface which in most operating system would be based on **`stdin`** and **`stdout`**
- Domain data of the requesting Web-page (URL + Server certificate path)

Even after invocation the Device-side agent plays a vital role since it converts the data to and from the target application channel into BLE messaging as well as dealing with errors and shutdowns.

WebNFC Bridge – Sequence Diagram



WebNFC Bridge – Applications

- Secure Web Payments
- User Authentication
- Supplying User-data
- Games
- Virtual Passports & Visas
- Boarding Cards
- Interaction with TVs including remote control
- You name it…