2 Interface Specifications

This interface specification defines the communication requirements between the backend and frontend components of the certificate authenticity verification system utilizing the Cardano blockchain.

2.1 Input File Specifications

The following describes the specifications of input files used in the system.

2.1.1 Certificate File (Credential)

The certificate file is a JSON-formatted document issued by a certificate issuance tool. It can be uploaded through the browser interface using a file selection dialog.

- Format:

  JSON Format

- Example Structure:

```
{
  "height": 123456,
  "payload": "<HEX-Encoded Transaction>",
  "merkle": [{ "hash": "abc123...", "position": "left" }, ...]
}
```

- Usage:
  - Target transaction for verification
  - Merkle path components
  - Block number reference for locating the correct header

2.1.2 IPDC Block Header (BlockHeader)

Block information transmitted as IPDC block headers via broadcast is received and managed by the IPDC reception system.
From the managed block data, the relevant block is retrieved by specifying its block height (BlockHeight). In this tool, the file can be uploaded via the browser using a file selection interface.

- Format:

  JSON Format

- Example Structure:


```
{
  "time":1730131262,
  "height": 123456,
  "hash":"b3774ece7f20d75e...",
  "slot":74448062,
  "epoch":176,
  "epoch_slot":57662,
  "slot_leader":"pool13m26ky08vz2...",
  "size":4,
  "tx_count":0,
  "block_vrf":"vrf_vk1zqpmsxhsn3...",
  "op_cert":"873bdd3584228477e...",
  "op_cert_counter":"6",
  "previous_block":"eebd878bb3811e0f...",
  "next_block":null,
  "confirmations":0,
  "ipdc_previous_block":"a7ffc6f8bf1ed766...",
  "block_body_hash":"a7ffc6f8bf1ed766...",
  "ipdc_hash":"ceb38d17d35384a9..."
}
```


- Usage:
  - Block number identification
  - EPOCH information
  - Hash values of various data elements

2.1.3 Issuer Information (Issuer)

This is the public key information used at the time of certificate issuance.
In this tool, the issuer file can be registered via the browser using a file selection interface.

- Format:

  JSON Format

- Example Structure:

```
{
  "keys": [
    {
      "account": {
      "address":"addr_test1qq54...",
      "publicKey":"3745f6098287..."
      }
    }
  ]
}
```

- Usage:
  - Public key (Key) information

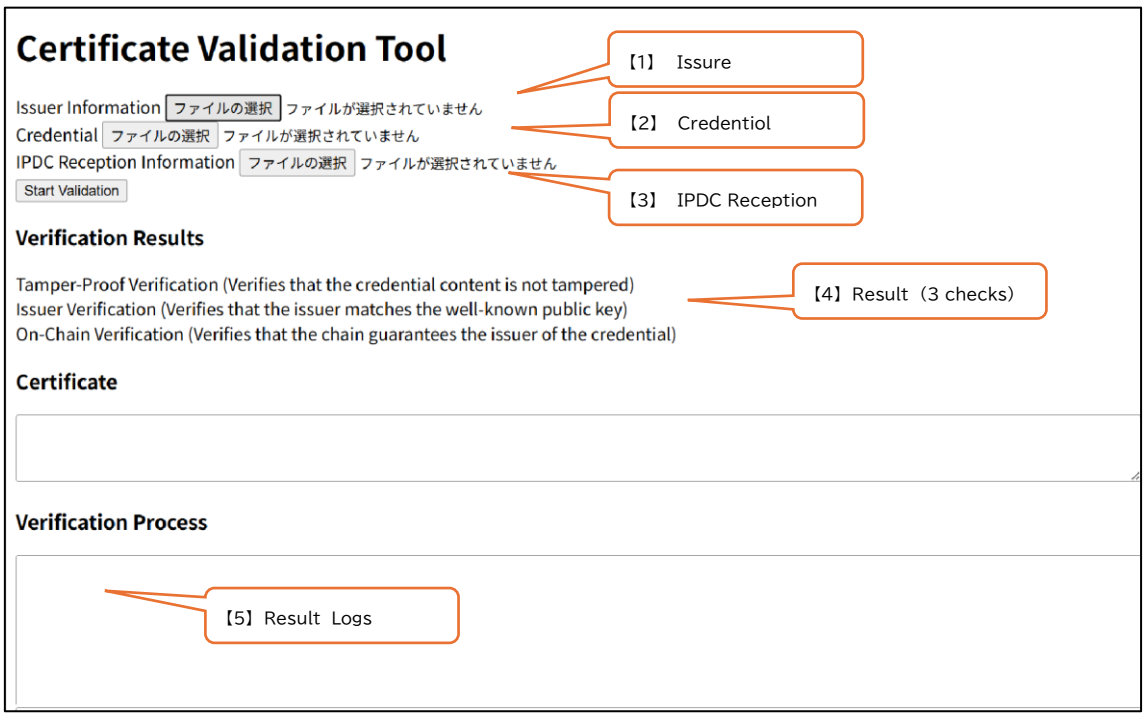
2.2 Internal Function Specification

The following table outlines the Internal Function Specification.

Function Name	Arguments	Description
clickStartVerify()	None	Loads three input files and then calls verify()
verify()	credential, blockheader, issuer	Extracts metadata, verifies the signature, and performs Merkle verification
verifyMerklePath()	leaf, path, expectedRoot	Calculates the hash path using SHA3-256 and compares it with the expected root
readFile()	input[type=file]	Uses FileReader to read the file as Base64

2.3 UI Element Specifications

The following table lists the corresponding UI elements and their roles.



UI Label	Element ID	Associated Variable / Process
【1】Issuer	issuer	Load issuer JSON
【2】Credential	credential	Load credential JSON and Merkle information
【3】IPDC Reception Info	blockheader	IPDC block header data
【4】Result(Tamper, Issuer, On-Chain)	res_signature	Display verification results
【5】Verification Output Log	output	Display detailed logs in HTML <textarea>

2.4 Additional Information

The following additional information applies to the operational environment of this system:

- The environment must be set up with the following libraries: cardano_serialization_lib.js, sha3.min.js, and buffer@6.0.3.js.
- The use of **Merkle Path** verification and **signature validation** requires the loading of cardano-serialization-lib.
- **Blockchain consistency checks** are executed on the **Node.js backend** as a separate process.