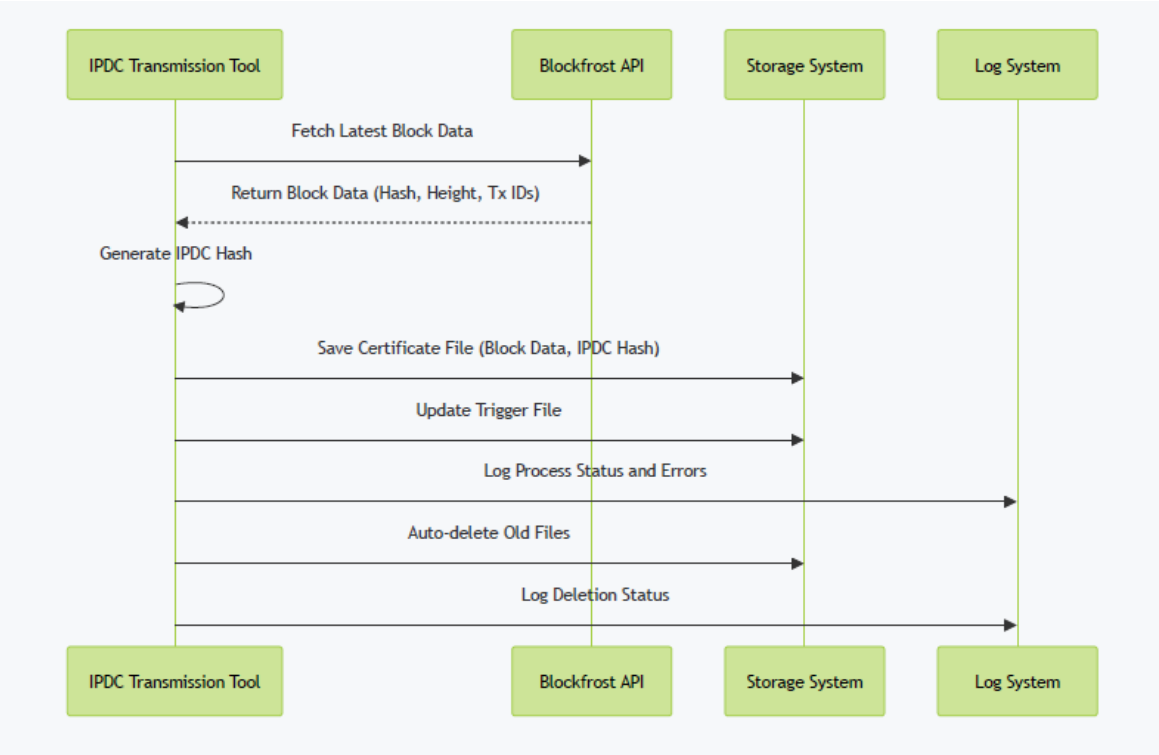
2.4 Data Flow and Algorithm Design

2.4.1 Data Flow and Algorithm

The IPDC Certificate Transmission Support Tool performs a series of processes, including retrieving the latest block information, generating an IPDC hash, creating and saving certificate files, and automatically deleting old files. Below, we describe the algorithms used in each step and the overall data flow.

The data flow is illustrated as follows:



2.4.1.1 Retrieving Block Information

- **Objective**

  Retrieve the latest block information from the Cardano blockchain to use as fundamental data for processing within the tool.

- **Algorithm**

  The latest block data is assigned an IPDC-format hash to ensure data integrity and consistency.

- **Error Handling**

  If the API call fails, the error is logged, and the process is retried in the next retrieval cycle.

2.4.1.2 Generating IPDC Hash

- **Objective**

  Use the **CardanoWasm** library to encode the credential certificate data as **Auxiliary Data** and generate a transaction.

- **Algorithm**

  ＜Retrieving Previous IPDC Hash＞

  The previously generated IPDC hash is referenced and combined with the current block data to generate a new hash.

  ＜Hash Generation Method＞

  The sha3_256 algorithm is used to hash the following data:

  - The previous IPDC hash
  - The latest block hash
  - The transaction ID list

＜**Generation Result**＞

The newly computed IPDC hash is stored and referenced in the next cycle.

- **Error Handling**

  If hash generation fails, the error is logged, and the process is retried in the next cycle.

2.4.1.3 **Generating and Saving Certificate Files**

- **Objective**

  Save the computed IPDC hash and block information as a **JSON** file for data verification and integration with external systems.

- **Processing Details**

  The IPDC hash and block information are structured in JSON format, and a certificate file is generated.　The file includes:

  Block height、Block hash、IPDC hash、Transaction ID list

  ＜**Trigger File Update**＞

  The new certificate file name is appended to **"trigger.txt"**, allowing other systems to detect newly generated files.

- **Error Handling**

  If file generation fails, the error is logged, and the process is retried in the next cycle.

2.4.1.4 Automatic Deletion of Old Files

- **Objective**

  Remove certificate files that exceed a predefined retention period to manage system storage efficiently.

- **Processing Details**

  The tool scans the certificate file storage directory and automatically deletes files that have exceeded the specified number of days (e.g., **7 days**).

  This process runs **once per day**, periodically clearing unnecessary data.

- **Error Handling**

  If the deletion process fails, the error is logged, and the process is retried in the next execution cycle.

2.4.2 Algorithm details

2.4.2.1 ブ Block information acquisition algorithm

- **Procedure**
  1. Send a request to the Blockfrost API endpoint at 15-second intervals.
  2. Retrieve the latest block hash, block height, previous block information, and transaction ID list.
  3. Pass the retrieved block information to the next IPDC hash generation process.
- **Error Handling**

  If an error is returned from the Blockfrost API, the system will not retry and will attempt to retrieve the data in the next cycle.

2.4.2.2 IPDC Hash Generation Algorithm

- **Procedure**
  1. Retrieve the IPDC hash generated in the previous cycle.
  2. Combine the latest block hash and the transaction ID list, then generate a hash using the **SHA3-256** algorithm.
  3. Save the generated IPDC hash as input data for the next cycle.
- **Error Handling**

  If hash generation fails, log the error and retry in the next cycle.

2.4.2.3 Certificate File Generation Algorithm

- **Procedure**

  1. Format the IPDC hash and block information into JSON format.
  2. Generate and save a certificate file (e.g., block_<height>.json) based on the JSON data.
  3. Append the new file name to **"trigger.txt"**.

- **Error Handling**

  If file generation fails, log the error and retry in the next cycle.

2.4.2.4 **Old File Deletion Algorithm**

- **Procedure**

  1. Scan the directory where certificate files are stored.
  2. Check the creation date of each file and delete files that exceed the specified retention period.
  3. Execute once per day to automatically remove old data.

- **Error Handling**

  If a file fails to be deleted, log the error and retry during the next execution.

2.4.2 Data Flow Summary

2.4.2.1 Data Flow

1. **Retrieve block information** from the API
2. **Generate the IPDC hash**
3. **Generate and save the certificate file**
4. **Log the process**
5. **Delete old files**

This data flow and algorithm enable the tool to periodically manage the latest block information in **IPDC format**, automate file storage, and handle old data management. Additionally, error handling is integrated into each processing step, ensuring stable operation.

2.5 Data Design

The IPDC Certificate Transmission Support Tool handles various types of data, including block information, IPDC hashes, trigger files, and logs. The following sections describe the role, structure, and storage format of each data type.

2.5.1 Data Items

2.5.1.1 Block Information

- **Description**

  The latest block information retrieved from the Cardano blockchain using the Blockfrost API.

- **Content**

  <Block Hash>

  A hash value used as a unique identifier for the block.

  <Block Height>

  A numeric value indicating the block's position in the blockchain.

  <Transaction ID List>

  A list of all transaction IDs included in the block.

- **Example Data**

```
{
   "hash": "abc123...",
   "height": 123456,
   "previous_block": "def456...",
   "transactions": ["tx1", "tx2", "tx3"]
}
```

2.5.1.2 IPDC Hash

- **Description**

  The IPDC hash assigned to each block is generated by combining the previous block's IPDC hash with the current block data. It is used to ensure data integrity and continuity.

- **Generation Method**

  The IPDC hash is calculated using the following data:

  - Previous IPDC Hash
  - Current Block Hash
  - Transaction ID List

- **Data Format**

  String (Hexadecimal hash value)

- **Example Data**

```
"ipdc_hash": "789abc..."
```

2.5.1.3 Certificate File

- **Description**

  A JSON-formatted file containing block information and the IPDC hash, generated for each block. This file is used for data integration and verification with external systems.

- **Storage Location**

  Saved in a directory specified in the tool settings. The file name includes the block height (e.g., block_<height>.json).

- **Content**

  <block height> The block height at the time of generation.

  <block hash> The hash value of the current block.

  <IPDC hash> The latest IPDC hash.

  <transaction ID List> The list of transaction IDs in the block.

- **Data Format**

  JSON format

- **Example Data**

```
{
   "height": 123456,
   "block_hash": "abc123...",
   "ipdc_hash": "789abc...",
   "transactions": ["tx1", "tx2", "tx3"]
}
```

2.5.1.4 Trigger File (trigger.txt)

- **Description**

  A text file used to notify external systems of newly generated certificate files. Each new certificate file path is appended to this file.

- **Content**

  The file path of the latest generated certificate file is added to the text file.

- **Data Format**

  Text file (.txt)

- **Example Data (trigger.txt)**

```
/path/to/certificate/block_123456.json
/path/to/certificate/block_123457.json
```

2.5.1.5 Log File

- **Description**

  The log file records the execution status, processing results, and error information of the tool. It is used for operation monitoring and troubleshooting.

- **Content**
  - Execution results of normal processing
    (e.g., successful block information retrieval, completion of IPDC hash generation)
  - Error messages
    (e.g., API errors, file save failures)
  - Processing start and end timestamps

- **Data Format**

  Text file (.log format)

- **Example Data**

```
[INFO] 2024-11-01 15:00:00 - Block data fetched successfully for block height 123456
[ERROR] 2024-11-01 15:15:00 - Failed to fetch block data - API Error: 503 Service Unavailable
```

2.5.1.6 Deletion of Old Files

- **Description**

  This process automatically deletes certificate files that have exceeded a predefined retention period to efficiently manage system storage. Files stored in the designated directory that are older than the specified number of days are targeted for deletion.

- **Data Format**

  List of file paths of deleted files

- **Example Data**

  ```
  /path/to/certificate/block_123456.json
  ```

2.5.2 Summary of Data Design

- **Block Information**

  Retrieves the latest block data and serves as the foundation for generating the IPDC hash.

- **IPDC Hash**

  A unique hash assigned to each block, ensuring data integrity and continuity.

- **Certificate File**

  Contains IPDC hash and block information, used for data integration with external systems.

- **Trigger File**

  Notifies external systems about **newly generated certificate files.**

- **Log File**

  Records processing status and error details, aiding operation monitoring and troubleshooting.

- **Deletion of Old Files**

  Periodically removes unnecessary files to maintain efficient storage management.

This **data design** ensures **organized data management within the tool**, defining the **storage format of each data type**, while maintaining **reliability and consistency in processing.**