

baumann.at – concepts & solutions – DI Dr. Christian Baumann

Document-Notarisation “DocNoS” – API

Version 1.6.0



Arbeitskreis Blockchain



Inhalt

1. Introduction.....	2
2. Variants of the blockchain interface	2
3. DocNoS-API.....	4
3.1. General	4
3.2. Create a notarisation.....	4
3.2.1. Request.....	5
3.2.2. Response	5
3.3. Verify a notarisation	6
3.3.1. Request.....	6
3.3.2. Response	7
4. Appendix: Test-Scripts.....	11
5. Appendix: Testsystem	14
5.1. Create a notarisation.....	14
5.2. Verifying a notarisation	17
5.3. Direct query of the data	19

1. Introduction

By means of "document notarisation", it can be proven that an electronic document existed in a certain form at a certain time and has not been changed since then. Documents are identified by their "digital fingerprints" (hash values), i.e. no data (readable in plain text) is transmitted, processed or stored.

The security and trust that the stored data cannot be manipulated is guaranteed by the blockchain technology.

This document describes the REST API of the system "DocNoS" (Document Notarisation System), which is used, among other things, in the course of the notarisation service of the **WKO**¹ (Austrian Economic Chamber) - called "Data Certification" - and the **Vienna University of Economics and Business Administration**². These two (and other) systems are combined in the "**Austrian Public Service Blockchain**", which is set up as a consortium blockchain.

Parallel to the system for the public sector, there is also a system for use by the private sector, a B2B counterpart called "**Private Sector Blockchain**". The nodes of this blockchain are operated by the members of the "**Blockchain Initiative Austria**"³.

The API described here is available across systems, i.e. can be used for both systems; any differences are described in the respective sections.

2. Variants of the blockchain interface

In order to write the data to the respective blockchain (or also to search/read it), there are basically two possibilities:

- 1) Direct access from the application to the blockchain node (via Multichain-RPC-API).
- 2) Access via the DocNoS API described in this document, which encapsulates the complexity of blockchain access.

¹ <https://www.wko.at/service/innovation-technologie-digitalisierung/blockchain.html>

² <https://www.wu.ac.at/blockchain/>

³ <https://www.bc-init.at/>

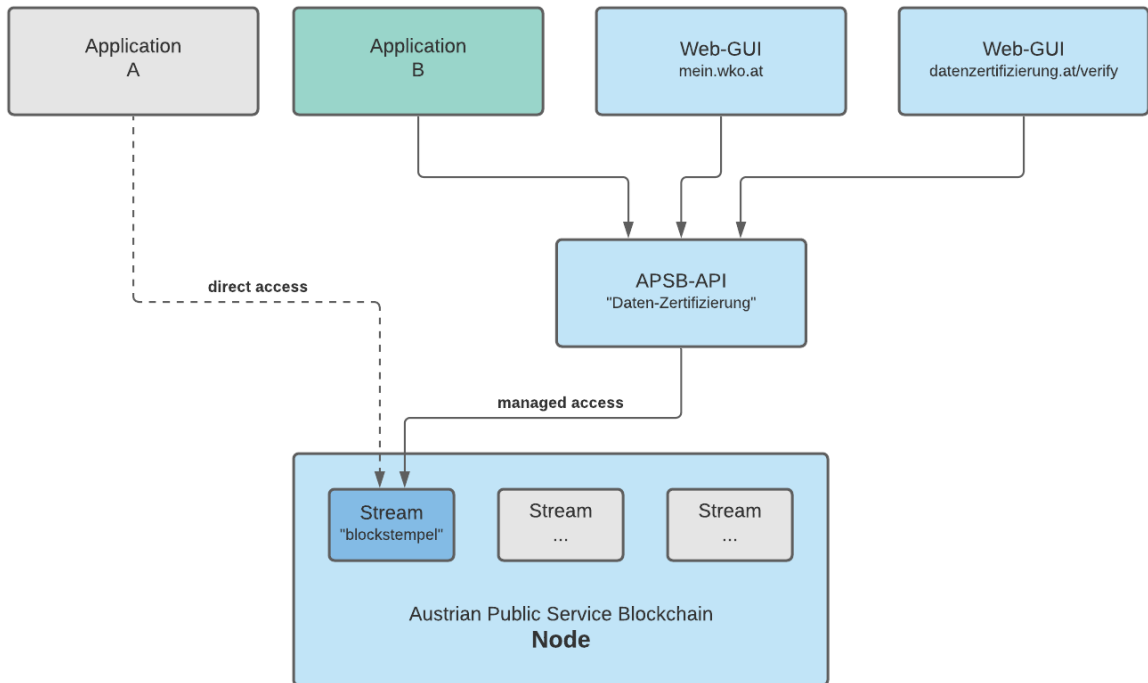


Figure 1: Access variants using the naming of the "Austrian Public Service Blockchain"

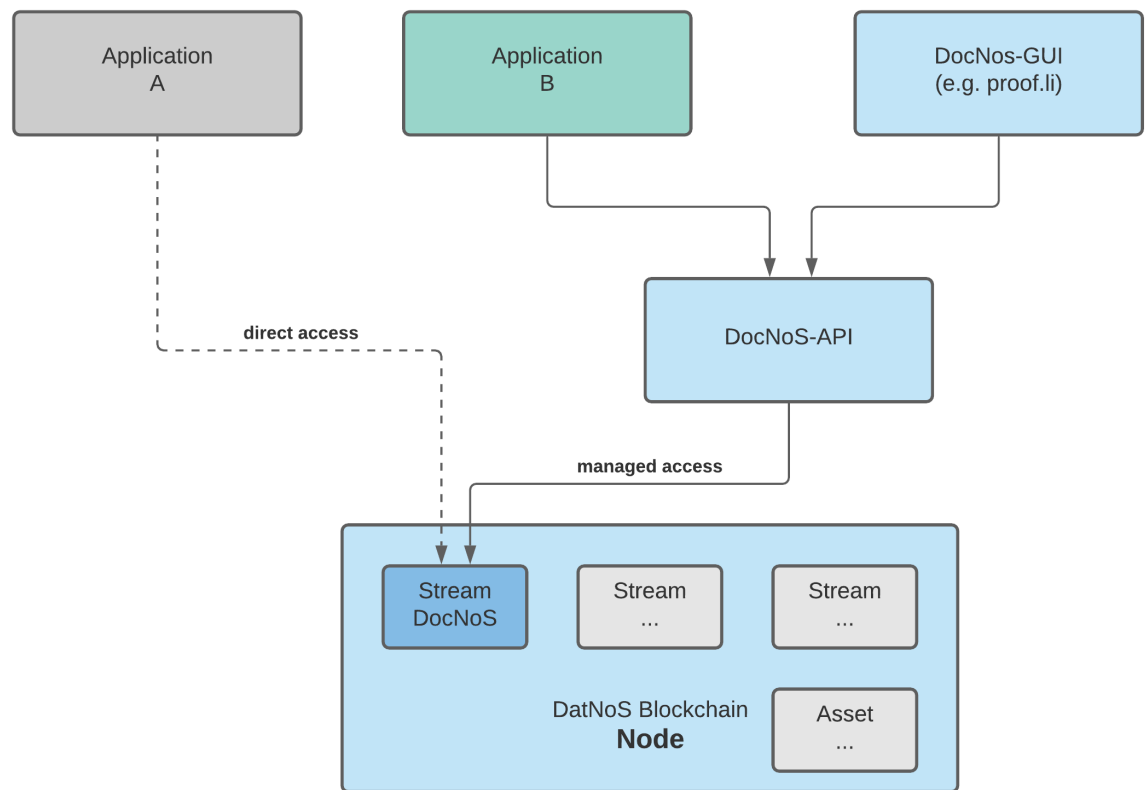


Figure 2: Access variants using the naming of the "Private Sector Blockchain"

Managed access via the DocNoS API is usually preferable, as the API creates the DocNoS data structure plus all keys as specified and can also manage access from different applications to several blockchain nodes or streams and protect them with access authorisations. Upcoming extensions can also be implemented more easily in the DocNoS API.

Direct access offers slightly better performance for high transaction numbers, but is much more complex to implement as it needs to create the complete data structure plus all keys.

An example of an "Application B" as demoscrypts and an example of a web GUI are described in the appendix to this document.

3. DocNoS-API

3.1. General

The REST API provides functions for creating and verifying notarisations. It is addressed with https POST (or alternatively with GET when verifying).

For the designation of hash procedures, the corresponding tokens are used according to the following table (lower case):

Hash-Verfahren	Token
SHA2– prefix „sha“	
Example SHA2 256 Bit	„sha256“ (must at least be present)
Example SHA2 512 Bit	„sha512“
SHA3 – prefix „sha3/“	
Example SHA3 512 Bit	„sha3/512“ ⁴

If an (optional) "document ID" is also transferred, it must be formatted as UUID (version 4).

To authenticate the client to the API, the client passes an API token (shared secret) in the http header⁵, example:

```
Content-type: application/json
Accept: application/json
Content-Length: 457
X-API-Token: pyDemo/test/44edd16fa685b3be6b874a01...
```

3.2. Create a notarisation

To create a notarisation, the client passes at least a hash value of the type SHA 256. Optionally, other types can be passed, also optionally a document ID.

⁴ Note: When coding in JSON it will become „sha3\512“

⁵ Please request the API token for the test system from C. Baumann.

The server creates the defined data structure and enters it plus keys into the configured blockchain stream. The response of the API server contains detailed data about the resulting blockchain transaction.

The appendix to this document shows sample code for generating and submitting API requests.

3.2.1. Request

The URL in the test system is: <https://blockchains.web-lab.at/docnos3-api/create/>

The user data to be transmitted are to be transferred in the http body as JSON and are structured as follows. The following applies

- There must be at least one entry (sha256) in "hashes".
- The entry "id" is optional. If it is used, the value must be a UUID (according to RFC4122).
- The entry "remarks" is also optional.

```
{
  "id": "12345678-5f7c-4eb2-9344-b35943815ed5",
  "hashes": {
    "sha256":
"8daf3a72c2bea121e7f8477141bfad7787192a2e0f82680cbf83f55a070fbdbf",
    "sha512":
"439eef199e5da58722f459a8e4088c842015fe0458ce27bf8bc5a3d2cdb2fb65b3a61ec3d750b
bd8912edc82ffb325cafe052d20ffad5dea185dab6244f6d351"
  },
  "remarks": "sent from pyDemo 0.5"
}
```

3.2.2. Response

If successful, the service returns http status 200 (OK) and the following JSON response:

```
{
  "success": "OK, data published in transaction
11fecdbfffb6f9d626c921a34d5b537ea4528e33cd074e2bf561a467c0fb52860",
  "timeStamp": "2023-07-20T09:01:05+02:00",
  "id": "12345678-5f7c-4eb2-9344-b35943815ed5",
  "txid":
"11fecdbfffb6f9d626c921a34d5b537ea4528e33cd074e2bf561a467c0fb52860",
  "service": "DocNoS receiver\/create v1.6.2",
  "infos": "client:pyDemo v:1 stream:docnos-test-1 chain:mc2b1
rpc:127.0.0.1:7222"
}
```

In case of an error, the following response is sent:

```
{
  "error": "Error 401: Unauthorized (API-Token not known\valid)",
}
```

```

    "service": "DocNoS receiver\create v1.6.2",
    "infos": "n\A"
}

```

In addition, the http status is set according to the following table:

Statuscode	Meaning
401	No (valid) API token set
403	Forbidden: <ul style="list-style-type: none"> Client either (temporarily) deactivated or IP of the client is from a not allowed IP range (subnet)
405	Method not allowed: Request is not a post-request
400	Bad request: <ul style="list-style-type: none"> No user data available or user data not (correctly) JSON encoded. No element "hashes" in the user data The hash types or hash values do not conform to the specification (or configuration). - If "id" is used but does not correspond to UUID format (RFC4122)
500	Error in the configuration of the service
503	Service unavailable: chain or stream (temporarily) deactivated

3.3. Verify a notarisation

In the course of verifying a notarisation, the desired data is searched for in the blockchain stream. To do this, the API client must pass one of four possible parameters:

- Hash value of the document in question; there may be several entries with the same hash value if the same document has been notarised several times.
- Document ID (if it was used); several entries with the same ID can exist, as the ID can be created by the API client.
- ID of the transaction; this is always unique.
- Search for block hash; several transactions can exist in one block.
- (Upcoming extension: Search for block hash with filtering by hash values and/or document IDs).

The search result can be the following

- No entry found: If a hash value was searched for, this means that the document concerned (in the version at hand) was not notarised in this blockchain system.
- One entry found
- Multiple entries found, see above for details. The oldest entry is the most relevant.

3.3.1. Request

The URL in the test system is: <https://blockchains.web-lab.at/docnos3-api/verify/>

A GET or POST request can be sent to the service for verification. One of the following parameters must be used:

- „hash“ in the form „type:value“, e.g. sha256:
8daf3a72c2bea121e7f8477141bfad7787192a2e0f82680cbf83f55a070fbdbf
- „txid“, e.g. 11fecdbff6f9d626c921a34d5b537ea4528e33cd074e2bf561a467c0fb52860
- „id“, e.g. id:12345678-5f7c-4eb2-9344-b35943815ed5
- „blockHash“, e.g. 12345678-5f7c-4eb2-9344-b35943815ed5

3.3.2. Response

If successful, the following response is reported (for the Private Sector Blockchain):

```
{
  "success": "hash found:
sha256:8daf3a72c2bea121e7f8477141bfad7787192a2e0f82680cbf83f55a070fbdbf",
  "data": [
    {
      "publisher": "13VXwdarLRtV5fyP8qdWEXebe6Ay45pgdY4Bb",
      "txid":
"11fecdbff6f9d626c921a34d5b537ea4528e33cd074e2bf561a467c0fb52860",
      "blockHash":
"0001241892b2ac07a48e8c0587f0ba9c85dd130ed956a1b5596e0a65d1361169",
      "blockTime": "2023-07-20T09:01:09+02:00",
      "confirmations": 21,
      "data": {
        "timeStamp": "2023-07-20T09:01:05+02:00",
        "client": "pyDemo",
        "version": "DocNoS-v1.1",
        "data": {
          "id": "12345678-5f7c-4eb2-9344-b35943815ed5",
          "hashes": {
            "sha256":
"8daf3a72c2bea121e7f8477141bfad7787192a2e0f82680cbf83f55a070fbdbf",
            "sha512":
"439eef199e5da58722f459a8e4088c842015fe0458ce27bf8bc5a3d2cdb2fb65b3a61ec3d750b
bd8912edc82ffb325cafe052d20ffad5dea185dab6244f6d351"
          },
          "remarks": "sent from pyDemo 0.5"
        }
      }
    }
  ],
  "service": "DocNoS receiver/verify v1.6.2",
  "infos": "client:pyDemo v:1 stream:docnos-test-1 chain:mc2b1
rpc:127.0.0.1:7222"
}
```

In the "Austrian Public Service Blockchain" variant, the response is structured as follows:

```
{
```

```
"success": "hash found",
"service": "DocNoS receiver\verify v1.40",
"data": [
  {
    "txid":
"295a9c67904b0be0edb8a8474c3e0fb0dfaef5ed7d13f6f9a593284765092914",
    "blockHash":
"00df1aeee406648f3e3ba94ae488bd376320a8e1a2c11e80438e8f49ec05e674",
    "blockTime": "2020-12-08T17:34:15+01:00",
    "confirmations": 3,
    "data": {
      "metadataInternal": {
        "app": "unknown",
        "time": "1607445243000",
        "storageType": "JSON"
      },
      "metadataExternal": {
        "additionalMetadata": null,
        "user": "pyDemo",
        "dataType": "Blockstempel-v2",
        "tags": [
          "Blockstempel-v2",
          "id:1c123b9d-5f7c-4eb2-9344-b35943815ed5",
          "hash:sha256:2bf928c9b7877fa508487a70d387bee8b66399f3f
024bf757bbd52b4ea2c3aa2",
          "hash:sha512:61154490bbcbd6080bfc1d7797d465f251c79d6dc
e2a301a3df55cce7c7ad6275d69ba6fbe7a35be64c9ff581ef04554dd42f17f7fd38375e381763
72b52f79c",
          "hash:sha3\512:9000b07534d4fe6f73dd8bfff28d5b0b7946b80
845747b8fe7b81ce2b80a284d1edba24e22441aa2f0ea235ad537570dece2b01183430cd1148a9
7d7acca62426"
        ]
      },
      "data": {
        "id": "1c123b9d-5f7c-4eb2-9344-b35943815ed5",
        "time": "2020-12-08T17:34:03+01:00",
        "hashes": {
          "sha256":
"2bf928c9b7877fa508487a70d387bee8b66399f3f024bf757bbd52b4ea2c3aa2",
          "sha512":
"61154490bbcbd6080bfc1d7797d465f251c79d6dce2a301a3df55cce7c7ad6275d69ba6fbe7a3
5be64c9ff581ef04554dd42f17f7fd38375e38176372b52f79c",
          "sha3\512":
"9000b07534d4fe6f73dd8bfff28d5b0b7946b80845747b8fe7b81ce2b80a284d1edba24e22441a
a2f0ea235ad537570dece2b01183430cd1148a97d7acca62426"
        },
        "optional": {
          "size": null
        }
      }
    }
  ]
}
```



```

    }
  }
]
}

```

If several results are found (e.g. when searching for a hash value that has been entered several times), the response is as follows - an array with the corresponding transactions is entered in the "data" element.

```

{
  "success": "hash found:
sha256:8daf3a72c2bea121e7f8477141bfad7787192a2e0f82680cbf83f55a070fbdbf",
  "data": [
    {
      "publisher": "13VXwdarLRtV5fyP8qdWEXebe6Ay45pgdY4Bb",
      "txid":
"11fecdbfffb6f9d626c921a34d5b537ea4528e33cd074e2bf561a467c0fb52860",
      "blockHash":
"0001241892b2ac07a48e8c0587f0ba9c85dd130ed956a1b5596e0a65d1361169",
      "blockTime": "2023-07-20T09:01:09+02:00",
      "confirmations": 21,
      "data": {
        ...
      }
    },
    {
      "publisher": "13VXwdarLRtV5fyP8qdWEXebe6Ay45pgdY4Bb",
      "txid":
"dfb1d39c80ee0922c676cc95f76f4a6a5a822174e109da727bec012ea38e6df1",
      "blockHash": null,
      "blockTime": null,
      "confirmations": 0,
      "data": {
        ...
      }
    }
  ],
  "service": "DocNoS receiver/verify v1.6.2",
  "infos": "client:pyDemo v:1 stream:docnos-test-1 chain:mc2b1
rpc:127.0.0.1:7222"
}

```

If the verify occurs so shortly after a create that the transaction is valid but has not yet been processed in a block, the blockTime can be zero, i.e. invalid (e.g. "blockTime": "1970-01-01T01:00:00+01:00"). In this case, the processing programme should not issue an error message but information (e.g. "Transaction not yet entered in any block").

In case of errors, the following response is transmitted:

```
{
  "error": "hash not found:
sha256:7d8028dc7273a4da78320629dd14fccc998bc9ff5a25f4152187c1404c3d9847",
  "service": "DocNoS receiver/verify v1.6.2",
  "infos": "client:pyDemo v:1 stream:docnos-test-1 chain:mc2b1
rpc:127.0.0.1:7222"
}
```

The http status codes are set according to the following table:

Statuscode	Meaning
400	Bad request: <ul style="list-style-type: none"> • None of the parameters "hash", "txid", "id" or "blockhash" were used. • The format of txid or blockhash is not correct (if used)
404	Not found: No record (transaction) with the specified value was found (in the blockchain)
401	No (valid) API token set
403	Forbidden: <ul style="list-style-type: none"> • Client either (temporarily) deactivated • or IP of the client is from a not permitted IP range (subnet)
500	Error in the configuration of the service
503	Service unavailable: chain or stream (temporarily) deactivated

4. Appendix: Test-Scripts

To provide a quick introduction to the use of the API for those interested, Python scripts have been created and published on Github, see

<https://github.com/austriapro/blockchain/tree/master/docnos3-testclient>

The script "create_notarization.py" shows how a request is built, the API token is set in the http header and the request is sent to the API server, here is an excerpt of the source code:

```
...
Simple script to test DocNoS API function "create"

configuration see config.py

@author    Chris Baumann <cba@infinite-trust-digital.com>,
<c.baumann@baumann.at>
@version   v0.5 2023/07/20

...
import sys
import datetime
import hashlib
import json
import requests # install with "pip3 install requests" if necessary

from config import *

print('-----')
print('DocNos - Test ... create')

...
A valid DocNos Create Request looks like this:
- id is an optional (v4) UUID e.g. for a document-id. If not present, it will
be generated from the API
- hashes: sha256 is required, sha512 (and sha3/512) optional
- remarks: optional, used for testing
{
    "id": "12345678-5f7c-4eb2-9344-b35943815ed5",
    "hashes": {
        "sha256":
"8daf3a72c2bea121e7f8477141bfad7787192a2e0f82680cbf83f55a070fbdbf",
        "sha512":
"439eef199e5da58722f459a8e4088c842015fe0458ce27bf8bc5a3d2cdb2fb65b3a61ec3d750b
bd8912edc82ffb325cafe052d20ffad5dea185dab6244f6d351"
    },
    "remarks": "sent from pyDemo 0.5"
}
...
```

```
# hash input data (in this case just the content, defined in config.py)
sha256 = hashlib.sha256(defaultContent.encode('utf-8')).hexdigest()
sha512 = hashlib.sha512(defaultContent.encode('utf-8')).hexdigest()

# or create uuid based on some kind of document id etc.
uuid = '12345678-5f7c-4eb2-9344-b35943815ed5'

hashes = {
    'sha256': sha256,
    'sha512': sha512
}

# example request using the optional uuid
request = {
    'id': uuid,
    'hashes': hashes,
    'remarks': 'sent from pyDemo 0.5'
}

# minimal request would be
...

request = {
    'hashes': hashes
}
...

postData = json.dumps(request)
print('JSON-Request:')
print(postData)
print('-----')

length = str(len(postData))

httpHeaders = {
    'Content-type': 'application/json',
    'Accept': 'application/json',
    'Content-Length': length,
    'X-ApiToken': apiToken}
print(httpHeaders)

response = requests.post(url_create, data=postData, headers=httpHeaders)
print(str(response))
print('RESULT: ' + response.text)
```

The script "verify_notarization.py" shows the three variants of the search, here is an excerpt of the source code:

```
...
```

Simple script to test DocNoS API function "verify"

configuration see config.py

```
@author    Chris Baumann <cba@infinite-trust-digital.com>,
<c.baumann@baumann.at>
@version   v0.5 2023/07/20

...

import sys
import hashlib
import json
import requests # install with "pip3 install requests" if necessary

from config import *

print('-----')
print('DocNos for - Test ... verify')

# SHA2 256
sha256_hash_to_verify = hashlib.sha256(
    defaultContent.encode('utf-8')).hexdigest()
# print(sha256_hash_to_verify)

httpHeaders = {
    'Accept': 'application/json',
    'X-ApiToken': apiToken
}

# search for specific hash
# prefix required
key = 'hash'
value = 'sha256:' + sha256_hash_to_verify

# OR search for an id (UUID)
# prefix required
#key = 'id'
#value = 'id:12345678-5f7c-4eb2-9344-b35943815ed5'

# OR search by transaction
#key = 'txid'
#value = '8877873041300b8ce01b0429523764e26b34422e9cfa7df532fd464a7ce89b03'

# OR search by blockhash (new in v1.6.x)
...
key = 'blockHash'
value = '00a72a6c434c46a334c0101698c6124c13b01675f2ade6b1281d00dc1827457b'
...
```

```
response = requests.get(url_verify, params={key: value}, headers=httpHeaders)

print(str(response.headers))

print(str(response))
print('Raw RESULT: ' + response.text)

parsed_res = json.loads(response.text)
beautified_res = json.dumps(parsed_res, indent=2)
print('Beautified RESULT: ')
print(beautified_res)
```

In the script "config.py", the configuration of the API client is defined, essentially the URIs for the API calls and the API token:

```
# configuration for current Test-System for contignum gmbh
url_create = 'https://blockchains.web-lab.at/docnos3-api/create/'
url_verify = 'https://blockchains.web-lab.at/docnos3-api/verify/'
apiToken = 'pyDemo/test/42edd16fa685b3be6b874a01a ... etc.'

defaultContent = 'This is just some content, which is used as input example
... 123abc '
```



5. Appendix: Testsystem

The web GUI of the current test system can be called up at <https://blockchains.web-lab.at/docnos/>⁶. This can of course also be used to verify blockchain entries created with demo scripts or, conversely, to search for entries from the web GUI with the demo script. The use of the web GUI is described briefly below.

5.1. Create a notarisation

To create a notarisation, a document must first be selected on the local system. The document is NOT loaded onto the web server, but the calculation of the digital fingerprint of the document (hash value) takes place in the user's web browser. Further additional information can be entered in the web form (annotations). The entry of additional information is voluntary and optional. If no additional information is entered, the entry is completely anonymous.

⁶ System with a GUI which can be switched to English see <https://test.proof.li> – (Production system see <https://proof.li>)



Create notarization - TEST

To create a notarization, choose a document. The file is not uploaded to the server, the hash value is calculated locally in the browser.

Select file (will NOT be uploaded to the server):

photo1675070934.jpeg

Calculated hash value (sha256):

Filename (*):

Remark (optional, *):

(*) for reference, will NOT be stored in the blockchain.

After sending the data (hash value, optional annotations), the server performs an entry in the notarisation blockchain and reports the result (timestamp, transaction ID ...) back to the user.

Result of the creation - TEST



Notarization created.

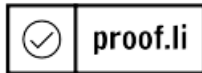
The notarization was created successfully, details are shown in the following table and can be downloaded as PDF (see link on bottom of page).

Time stamp	2023-07-20T09:42:43+02:00
Hash value	9ff27b26ea2e4ec1a5ccd2497fa77106283d13089d2ed8e04b778b8328aa07e5
Transaction-ID	5c55fef7287a58734d997e0515937e6d44b42dc897fbc4f299be5720a1e763a4
Filename (*)	photo1675070934.jpeg
Remark (*)	

(*) for reference, will NOT be stored in the blockchain.

[Back](#)

The user can now download a confirmation of the entry as a PDF document with QR code. This document contains the reference to the entry (transaction ID), other essential data (timestamp) and the optional additional information.



Document Notarization - Certificate

Created at 20.07.2023 - 09:42:43

This is to certify, that the hash value ("SHA256") of the document was securely and immutably stored in the blockchain.

The following table shows all details:

Time stamp	2023-07-20T09:42:43+02:00
Hash value	9ff27b26ea2e4ec1a5ccd2497fa77106283d13089d2ed8e04b778b8328aa07e5
Transaktions-ID	5c55fef7287a58734d997e0515937e6d44b42dc897fbc4f299be5720a1e763a4
Filename (*)	photo1675070934.jpeg
Remark (*)	

Data marked with (*) is for information and reference only and not stored in the blockchain

By using the following QR-Code or link you can invoke a verification service and pass the transaction ID.



<https://test.proof.li/test/index.php?page=verify&fileHash=9ff27b26ea2e4ec1a5ccd2497fa77106283d13089d2ed8e04b778b8328aa07e5>

5.2. Verifying a notarisation

In order to check a notarisation at a later point in time, there are two possibilities - in both cases, the hash value of the document is compared with the data stored in the blockchain.

1. In the first variant, the transaction ID shown on the confirmation is searched for in the system (e.g. with the help of the QR code) and the data on the document is displayed. The hash value of the document is calculated with any (other) tool and can now be compared.

Verify notarization - TEST

Here you can check whether/when a document was notarized, i.e. the digital fingerprint (hash value) of a file was stored in the blockchain.

To do this, select the corresponding file (the hash value is calculated automatically), or enter the hash value or the transaction ID.

Select file (will NOT be uploaded to the server) to calculate hash value:

Keine Datei ausgewählt.

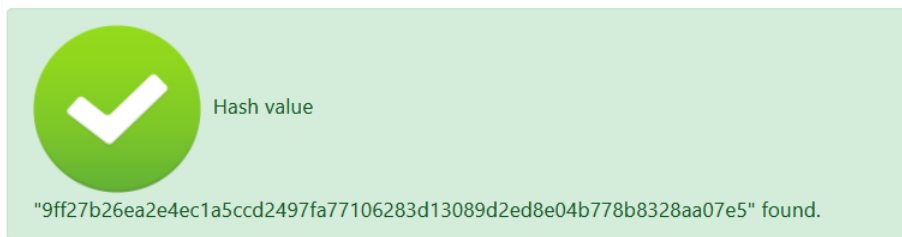
or hash value (sha256):

or Transaction-ID:

The entered data is searched in the blockchain and displayed accordingly.

2. With the second option, the hash value of the document is first calculated in the web browser, the notarisation is searched for in the blockchain using this as the key and the confirmation data created at that time is displayed. With this second option, the notarisation of a document can be proven even if the user no longer has the confirmation.

Result of the verification - TEST



Multiple entries were found, i.e. the document was notarized several times. The oldest entry (the first in the list) is therefore the most relevant.

Record 1/3

Block hash	00632eacfb296ecc6564a8b916626c354540a7dd8bc7b940433d2e1e806ee75e
Block time	2023-01-30T10:28:56+01:00
Confirmations	162296
Time stamp	2023-01-30T10:29:04+01:00
Hash value (sha256)	9ff27b26ea2e4ec1a5ccd2497fa77106283d13089d2ed8e04b778b8328aa07e5
Transaction-ID	01889862caaec0f64ba71f98a7ccc72446ce1d1688aeacd9ef544c5c10608e2a

If several entries exist for a hash value (as in this example), the oldest entry is displayed first.

A verification of a notarisation can of course take place on every participating system, not only on the one that originally created the entry.

As an example, entries created with the web GUI can be verified with the Python demo code, or notarisations created with it can be verified using the web GUI.

5.3. Direct query of the data

In the test system (Private Sector Blockchain), the data stored in the blockchain can also be accessed directly (i.e. without searching via "verification"). The url for this is <https://blockchains.web-lab.at/docnos-view>

DocNoS - Data view

Select Key

[all] - bs-client-cb1 - bs-client-jb1 - dn-client-cb2 - dn-client-jb2 - dn-client-cb3 - dn-client-jb3 - proof.li - dn-client-cb4 - bibi.li - test.meinwko - ForFor - sha512: - sha3/512: - dn-client-v3-std - test.nic.at - dn-client-v3-std-KEY - test.securikett - cardid:123 - test.ma01.wien - dn-client-cb4-std - proof.li/c2 - proof.li/c2/test - sec/forfor/test - pyDemo - Blockstempel-v2 - ABC-Test1 - proof.li/c#-client/test - proof.li/csc/test - IVM/Test - Weinand/Test - digicert/test - digicert/mei - woschitz/test - ifm.tu/test - docnos/test - MTP/Test - condignum/Test - pydemo - dnfn/test - futurelab/Test - TelegramNotarizingBot/test - matdol/Test - icomedias/Test - itreebute/Test - vecctor.de/Test

Key: [all]

10 of 54263 items

first - prev - next - last

Publishers	13VXwdarLRtV5fyP8qdWEFXebe6Ay45pgdY4Bb
Key 0	id:9d252d12-e1ff-43f4-a8d2-65c080565956
Key 1	sha256:9ff27b26ea2e4ec1a5ccd2497fa77106283d13089d2ed8e04b778b8328aa07e5
Key 2	proof.li/c2/test
JSON data	<pre>{ "timeStamp": "2023-07-20T09:42:43+02:00", "client": "proof.li\c2\test@h", "version": "DocNoS-v1.1", "data": { "id": "9d252d12-e1ff-43f4-a8d2-65c080565956", "hashes": { "sha256": "9ff27b26ea2e4ec1a5ccd2497fa77106283d13089d2ed8e04b778b8328aa07e5" } } }</pre>
Transaction	5c55fef7287a58734d997e0515937e6d44b42dc897fbc4f299be5720a1e763a4
Blocktime	2023-07-20T09:42:56+02:00
Blockhash	00ca75bd46735e70e5e8ccfa988f7a17ccb851aca116a8ad1495da7efb41eea
Confirmations	17

The page displays all the data of the transactions available in the blockchain, similar to a block explorer.

 baumann.at - concepts & solutions
 DI Dr. Christian Baumann
 e-Mail: c.baumann@baumann.at
 Tel.: +43 664 43 24 243
 Web: <http://www4.baumann.at>
