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

Dokumenten-Notarisierung “DocNoS” – API

Version 1.6.0

AUSTRIA PRO
Arbeitskreis Blockchain

BCI
Blockchain Initiative Austria

Inhalt

1. Einleitung

Mittels „Dokumenten-Notarisierung“ kann bewiesen werden, dass ein elektronisches Dokument zu einem bestimmten Zeitpunkt in einer bestimmten Form existiert hat und seither nicht verändert wurde. Dokumente werden dabei durch ihre „digitalen Fingerabdrücke“ (Hashwerte) identifiziert, d.h. es werden keinerlei (im Klartext lesbare) Daten übertragen, verarbeitet oder gespeichert.

Die Sicherheit und das Vertrauen, dass die hinterlegten Daten nicht manipuliert werden können, wird dabei durch die Blockchain-Technologie gewährleistet.

Dieses Dokument beschreibt das REST-API des Systems „DocNoS“ (Document Notarisation System), welches u.a. im Zuge des Notarisierungs-Service der **WKO**[1] (mit der Bezeichnung „**Daten-Zertifizierung**“) und der **Wirtschaftuniversität Wien**[2] eingesetzt wird. Diese beiden (und weitere) Systeme sind in der als Konsortialblockchain aufgebauten „**Austrian Public Service Blockchain**“ zusammengeschlossen.

Parallel zu dem System für den öffentlichen Bereich läuft auch ein System für die Nutzung durch die Privatwirtschaft, ein B2B-Pendant mit der Bezeichnung „**Private Sector Blockchain**“. Die Knoten dieser Blockchain werden von den Mitgliedern der „**Blockchain Initiative Austria**[3]“ betrieben.

Das hier beschriebene API ist systemübergreifend verfügbar, d.h. kann für beide Systeme eingesetzt werden, etwaige Unterschiede sind in den jeweiligen Abschnitten beschrieben.

2. Varianten der Blockchain-Schnittstelle

Um die Daten in die jeweilige Blockchain zu schreiben (bzw. auch zu suchen/lesen), gibt es grundsätzlich zwei Möglichkeiten:

1) Direkter Zugriff von der Anwendung auf den Blockchain-Node (per Multichain-RPC-API).
2) Zugriff über das in diesem Dokument beschriebene DocNoS-API, welches die Komplexität des Blockchain-Zugriffes kapselt.

[1] https://www.wko.at/service/innovation-technologie-digitalisierung/blockchain.html
[2] https://www.wu.ac.at/blockchain/
[3] https://www.bc-init.at/

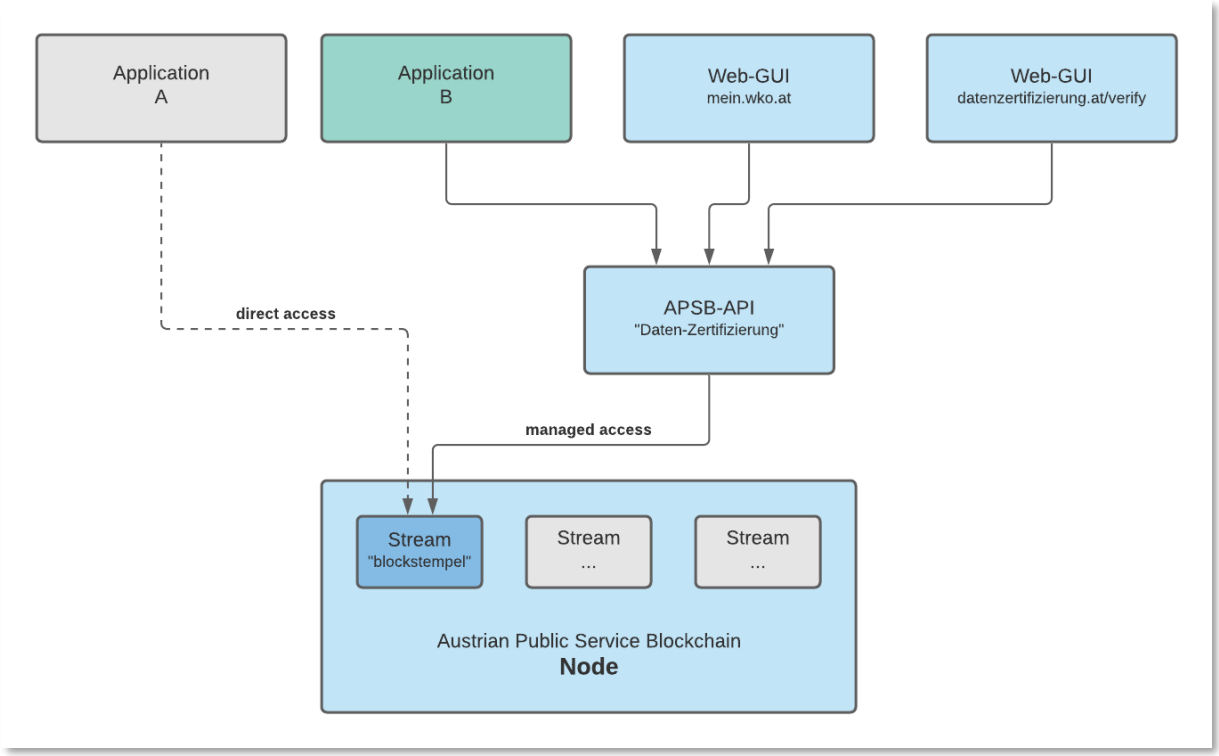


Abbildung 1: Zugriffsvarianten mit den Bezeichnungen der „Austrian Public Service Blockchain“

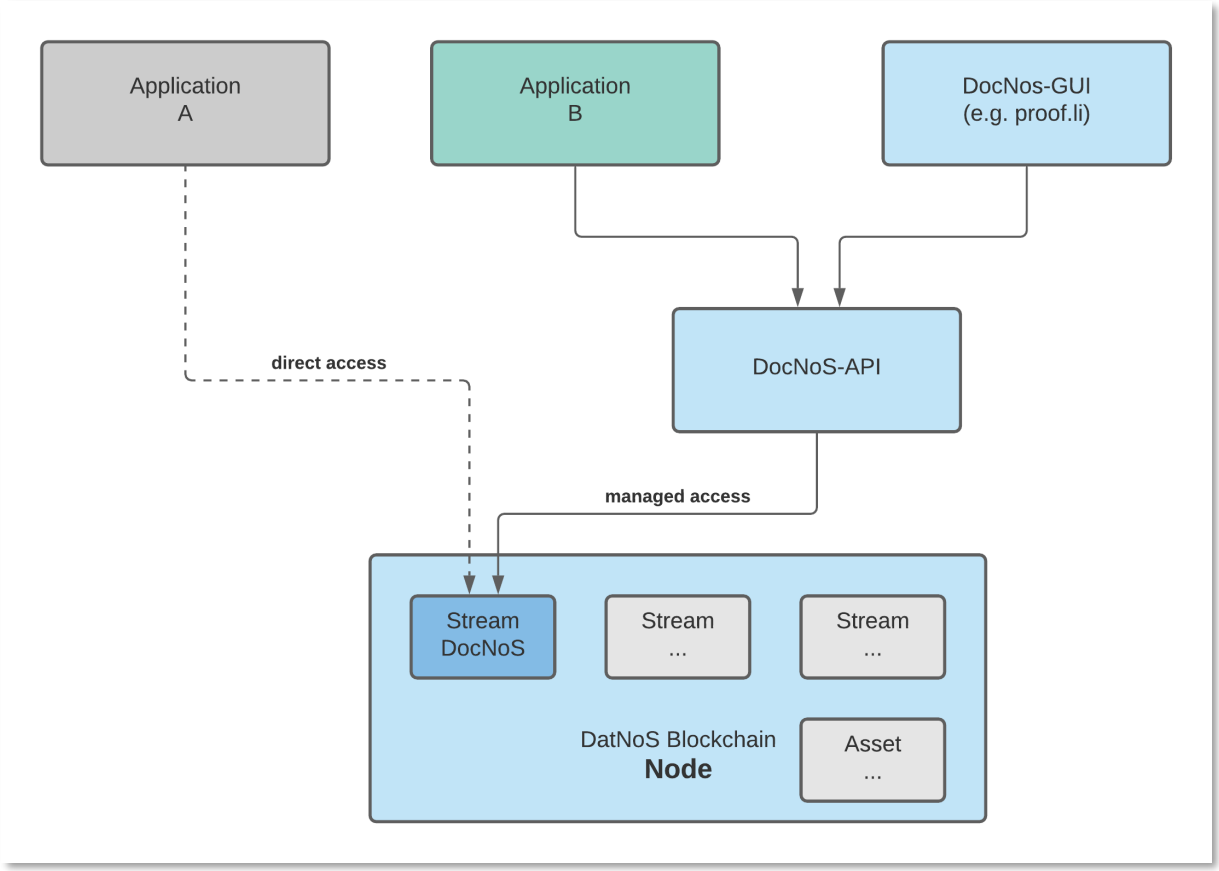


Abbildung 2: Zugriffsvarianten mit den Bezeichnungen der "Private Sector Blockchain"

Der gemanagte Zugriff über das DocNoS-API ist i.d.R. zu bevorzugen, da das API die DocNoS-Datenstruktur[4] plus alles Keys spezifikationsgemäß erstellt und darüber hinaus den Zugriff von unterschiedlichen Applikationen auf mehrere Blockchain-Nodes bzw. Streams verwalteten und mit Zugriffsberechtigungen schützen kann. Auch kommende Erweiterungen können im DocNoS-API einfacher implementiert werden.

Der direkte Zugriff bietet bei hohen Transaktionszahlen zwar etwas bessere Performance, ist jedoch deutlich komplexer zu implementieren, da er die komplette Datenstruktur plus alle Keys erstellen muss.

Ein Beispiel für eine „Application B“ als Demoscripts und ein Beispiel für ein Web-GUI sind im Anhang zu diesem Dokument beschrieben.

3. DocNoS-API

3.1. Allgemeines

Das REST-API stellt Funktionen für das Erstellen und das Verifizieren von Notarisierungen[5] bereit. Es wird mit https POST (bzw. beim Verifizieren alternativ mit GET) angesprochen.

Zur Bezeichnung von Hash-Verfahren werden die entsprechenden Token laut folgender Tabelle verwendet (Kleinschreibung):

Hash-Verfahren	**Token**
SHA2– prefix „sha“	
Beispiel SHA2 256 Bit	„sha256“ (muss mindestens vorhanden sein)
Beispiel SHA2 512 Bit	„sha512“
SHA3 – prefix „sha3/“	
Beispiel SHA3 512 Bit	„sha3/512“ [6]

Wenn eine (optionale) „Dokumenten-ID“ mit übergeben wird, so ist diese als UUID (Version 4) zu formatieren.

Zur Authentifizierung des Clients am API übergibt der Client ein API-Token (shared secret) im http Header[7], Beispiel:

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

[4] Siehe „DocNoS Datenstruktur v1.2“
[5] In der WKO Benennung „Daten-Zertifizierungen“
[6] Bitte beachten: Beim Codieren nach JSON wird dies zu „sha3\/512“
[7] Das API-Token für das Testsystem bitte bei C. Baumann anfordern.

3.2. Notarisierung erstellen

Zum Erstellen einer Notarisierung übergibt der Client zumindest einen Hashwert des Typs SHA 256. Optional können weitere Typen übergeben werden, ebenfalls optional eine Dokumenten-ID.

Der Server erstellt die definierte Datenstruktur und trägt diese plus Keys in den konfigurierten Blockchain-Stream ein. Der Response des API-Servers beinhaltet detaillierte Daten über die dabei entstandene Blockchain-Transaktion.

Im Anhang zu diesem Dokument ist Beispielcode für das Generieren und Absetzen von API-Requests dargestellt.

3.2.1. Request

Der URL im Testsystem lautet: https://blockchains.web-lab.at/docnos3-api/create/

Die zu übermittelnden Nutzdaten sind im http Body als JSON zu übergeben und folgendermaßen aufgebaut. Dabei gilt

- Es muss mindestens ein Eintrag (sha256) unter „hashes“ vorhanden sein.
- Der Eintrag „id“ ist optional. Fall er verwendet wird, muss der Wert ein UUID sein (lt. RFC4122)
- Der Eintrag „remarks“ ist ebenfalls optional.

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

3.2.2. Response

Im Erfolgsfall retourniert das Service http Status 200 (OK) und folgenden JSON-Response:

```
{
    "success": "OK, data published in transaction
11fecdbffb6f9d626c921a34d5b537ea4528e33cd074e2bf561a467c0fb52860",
    "timeStamp": "2023-07-20T09:01:05+02:00",
    "id": "12345678-5f7c-4eb2-9344-b35943815ed5",
    "txid":
"11fecdbffb6f9d626c921a34d5b537ea4528e33cd074e2bf561a467c0fb52860",
    "service": "DocNoS receiver\/create v1.6.2",
    "infos": "client:pyDemo v:1 stream:docnos-test-1 chain:mc2b1
rpc:127.0.0.1:7222"
}
```

Im Fehlerfall wird folgender Response gesendet:

```
{
    "error": "Error 401: Unauthorized (API-Token not known\/valid)",
    "service": "DocNoS receiver\/create v1.6.2",
    "infos": "n\/a"
}
```

Zusätzlich wird der http Status laut folgender Tabelle gesetzt:

Statuscode	Bedeutung
401	Kein (gültiges) API-Token gesetzt
403	Forbidden: • Client entweder (temporär) deaktiviert • oder IP des Clients ist aus einem nicht erlaubten IP-Range (Subnetz)
405	Method not allowed: Request ist kein Post-Request
400	Bad request: • Keine Nutzdaten vorhanden oder Nutzdaten nicht (korrekt) JSON codiert • Kein Element „hashes“ in den Nutzdaten • Die Hashtypen oder Hashwerte entsprechen nicht der Spezifikation (bzw. Konfiguration) • Wenn „id“ verwendet wird, aber nicht UUID Format (RFC4122) entspricht
500	Fehler in der Konfiguration des Services
503	Service unavailable: chain oder stream (temporär) deaktiviert

3.3. Notarisierung verifizieren

Im Zuge der Verifikation einer Notarisierung werden die gewünschten Daten im Blockchain-Stream gesucht. Dazu muss der API-Client einen von vier möglichen Parametern übergeben:

- Hashwert des betreffenden Dokumentes; es können mehrere Einträge mit demselben Hashwert existieren, wenn dasselbe Dokument mehrfach notarisiert wurde.
- Dokumenten-ID (falls sie verwendet wurde); es können mehrere Einträge mit derselben ID existieren, da die ID vom API-Client erstellt werden kann.
- ID der Transaktion, diese ist immer einmalig bzw. eindeutig.
- Suche nach Block-Hash; in einem Block können mehrere Transaktionen vorhanden sein.
- (Kommende Erweiterung: Suche nach Block-Hash mit Filterung nach Hashwerten und/oder Dokumenten-IDs)

Das Suchergebnis kann folgendes sein

- Kein Eintrag gefunden: Wenn nach einem Hashwert gesucht wurde bedeutet das, dass das betroffene Dokument (in der vorliegenden Version) nicht in diesem Blockchainsystem notarisiert wurde.

- Ein Eintrag gefunden
- Mehrere Einträge gefunden, Details siehe oben. Der älteste Eintrag ist der relevanteste.

3.3.1. Request

Der URL im Testsystem lautet: https://blockchains.web-lab.at/docnos3-api/verify/

Zum Verifizieren kann ein GET oder POST Request an das Service gesendet werden. Dabei muss einer der folgenden Parameter verwendet werden:

- „hash“ in der Form „type:value“, z.B. `sha256:8daf3a72c2bea121e7f8477141bfad7787192a2e0f82680cbf83f55a070fbdbf`
- „txid“, z.B. `11fecdbffb6f9d626c921a34d5b537ea4528e33cd074e2bf561a467c0fb52860`
- „id“, z.B. `id:12345678-5f7c-4eb2-9344-b35943815ed5`
- „blockHash“, z.B. `12345678-5f7c-4eb2-9344-b35943815ed5`

3.3.2. Response

Im Erfolgsfall wird (bei der Private Sector Blockchain) der folgende Response gemeldet:


```
{
    "success": "hash found:
sha256:8daf3a72c2bea121e7f8477141bfad7787192a2e0f82680cbf83f55a070fbdbf",
    "data": [
        {
            "publisher": "13VXwdarLRtV5fyP8qdWEFXebe6Ay45pgdY4Bb",
            "txid":
"11fecdbffb6f9d626c921a34d5b537ea4528e33cd074e2bf561a467c0fb52860",
            "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 der Variante „Austrian Public Service Blockchain“ ist der Response folgendermaßen aufgebaut:


```
{
    "success": "hash found",
    "service": "DocNoS receiver\/verify v1.40",
    "data": [
        {
            "txid":
"295a9c67904b0be0edb8a8474c3e0fb0dfaef5ed7d13f6f9a593284765092914",
            "blockHash":
"00df1aaee406648f3e3ba94ae488bd376320a8e1a2c11e80438e8f49ec05e674",
            "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\/512:9000b07534d4fe6f73dd8bff28d5b0b7946b80
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":
"9000b07534d4fe6f73dd8bff28d5b0b7946b80845747b8fe7b81ce2b80a284d1edba24e22441a
a2f0ea235ad537570dece2b01183430cd1148a97d7acca62426"
                    },
                    "optional": {
                        "size": null
                    }
                }
            }
        }
    ]
}
```


Falls mehrere Ergebnisse gefunden werden (z.B. bei der Suche nach einem Hashwert, der mehrfach eingetragen wurde), ist der Response wie folgt – im Element „data“ ist ein Array mit den entsprechenden Transaktionen eingetragen.


```
{
  "success": "hash found:
sha256:8daf3a72c2bea121e7f8477141bfad7787192a2e0f82680cbf83f55a070fbdbf",
  "data": [
    {
      "publisher": "13VXwdarLRtV5fyP8qdWEFXebe6Ay45pgdY4Bb",
      "txid":
"11fecdbffb6f9d626c921a34d5b537ea4528e33cd074e2bf561a467c0fb52860",
      "blockHash":
"0001241892b2ac07a48e8c0587f0ba9c85dd130ed956a1b5596e0a65d1361169",
      "blockTime": "2023-07-20T09:01:09+02:00",
      "confirmations": 21,
      "data": {
        ...
      }
    },
    {
      "publisher": "13VXwdarLRtV5fyP8qdWEFXebe6Ay45pgdY4Bb",
      "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"
}
```


Falls das Verify so kurz nach einem Create erfolgt, dass die Transaktion zwar gültig, aber noch in keinem Block verarbeitet wurde, kann die blockTime null d.h. ungültig sein (z.B. "blockTime":"1970-01-01T01:00:00+01:00"). In diesem Fall soll vom verarbeitenden Programm keine Fehlermeldung, sondern eine Information (z.B. „Transaktion noch in keinem Block eingetragen“) ausgegeben werden.

Bei Fehlern wird folgender Response übermittelt:


```
{
    "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"
}
```


Die http Statuscodes werden lt. folgender Tabelle gesetzt:

Statuscode	Bedeutung
400	Bad request: • Es wurde keiner der Parameter „hash“, „txid“, „id“ oder „blockhash“ • Das Format von txid oder blockhash ist nicht korrekt (falls verwendet)
404	Not found: Es wurde kein Datensatz (Transaktion) mit dem angegebenen Wert (in der Blockchain) gefunden
401	Kein (gültiges) API-Token gesetzt
403	Forbidden: • Client entweder (temporär) deaktiviert • oder IP des Clients ist aus einem nicht erlaubten IP-Range (Subnetz)
500	Fehler in der Konfiguration des Services
503	Service unavailable: chain oder stream (temporär) deaktiviert

4. Anhang: Test-Scripts

Um für Interessierte einen schnellen Einstieg in die Nutzung des APIs zu ermöglichen, wurden Python-Scripts erstellt und auf Github veröffentlicht, siehe
https://github.com/austriapro/blockchain/tree/master/docnos3-testclient

Mit dem Script „`create_notarization.py`“ wird gezeigt, wie ein Request aufgebaut wird, das API-Token im http Header gesetzt wird und der Request an den API-Server gesendet wird, hier ein Ausschnitt des Sourcecodes:

```
'''
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)
```

Das Script „`verify_notarization.py`“ zeigt die drei Varianten der Suche, hier ein Ausschnitt des Sourcecodes:

```
'''
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)
```

Im Script „`config.py`“ wird die Konfiguration des API-Clients definiert, im Wesentlichen die URls für die API-Calls und das 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 '
```

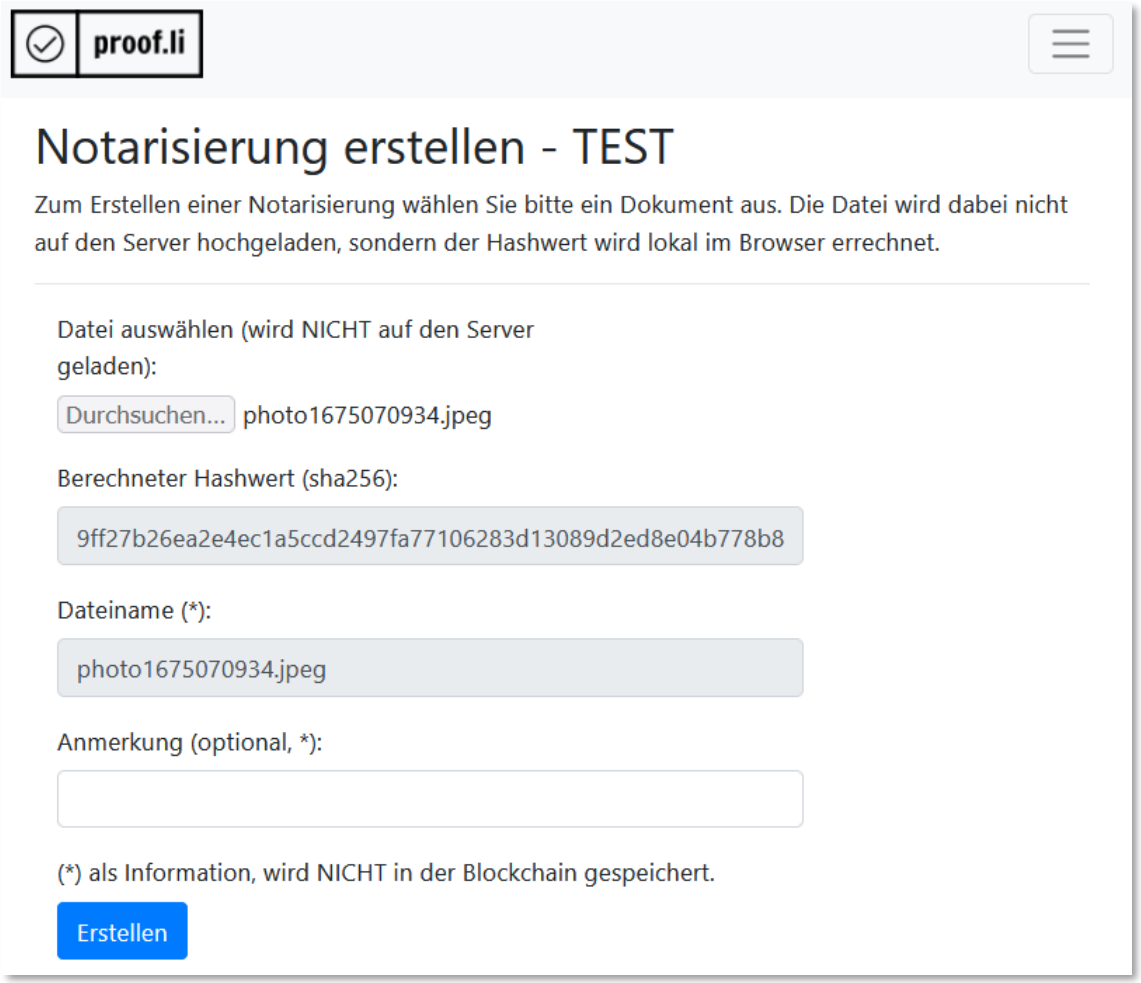
5. Anhang: Testsystem

Unter https://blockchains.web-lab.at/docnos/ kann das Web-GUI des aktuellen Testsystems aufgerufen werden[8]. Mit diesem können natürlich auch mittels Demo-Scripts erstellte Blockchaineinträge verifiziert werden bzw. umgekehrt Einträge vom Web-GUI mit dem Demo-Script gesucht. In Folge wird die Nutzung des Web-GUIs kurz beschrieben.

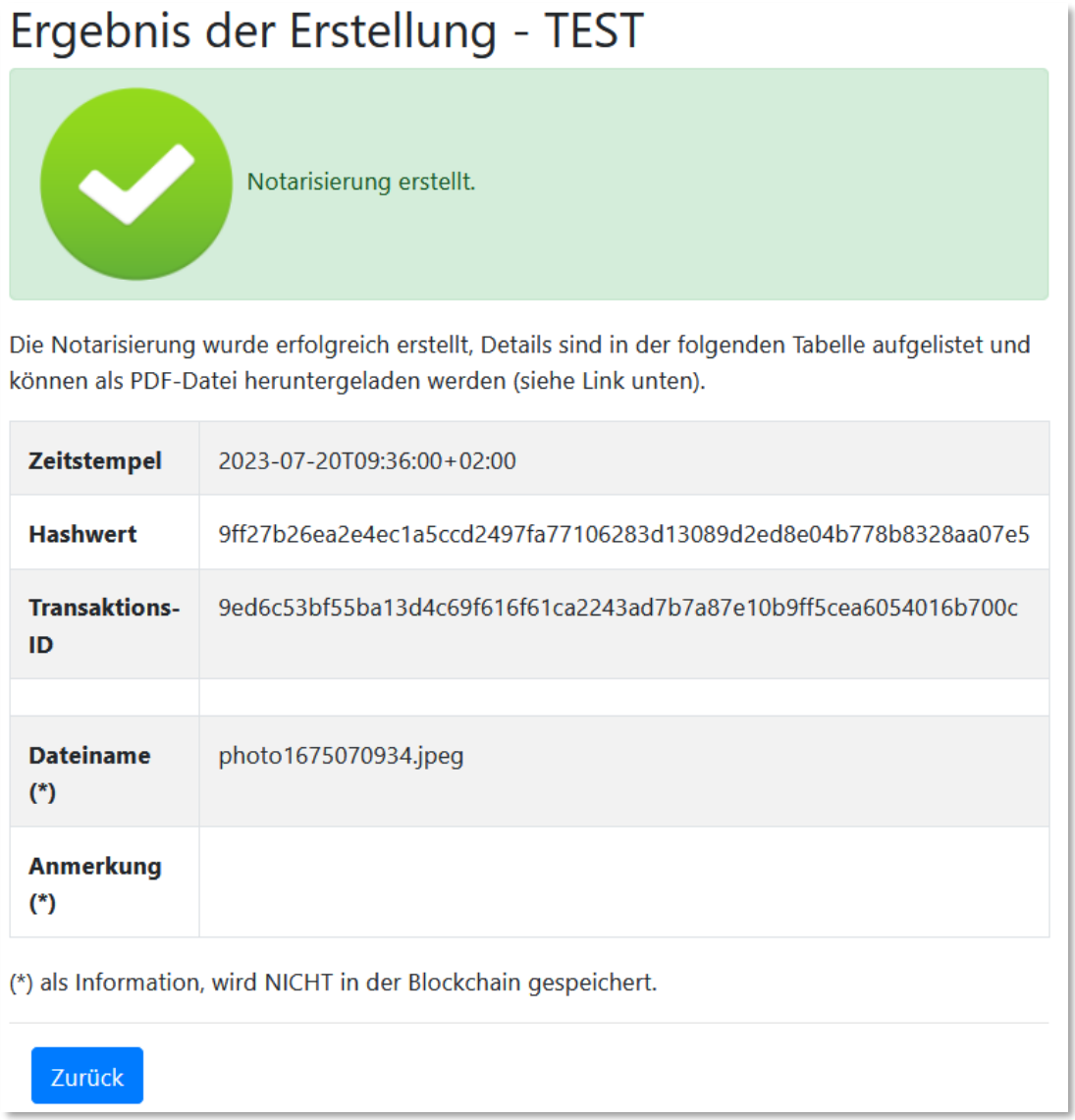
5.1. Erstellen einer Notarisierung

Um eine Notarisierung zu erstellen, ist zuerst ein Dokument am lokalen System auszuwählen. Das Dokument wird dabei NICHT auf den Webserver geladen, sondern die Berechnung des digitalen Fingerabdrucks des Dokuments (Hashwert) findet im Webbrowser des Benutzers statt. Im Webformular können weitere Zusatzinformationen erfasst werden (Anmerkungen). Die Eingabe von Zusatzinformationen ist freiwillig und optional. Wenn keine Zusatzdaten angegeben werden, ist der Eintrag völlig anonym.

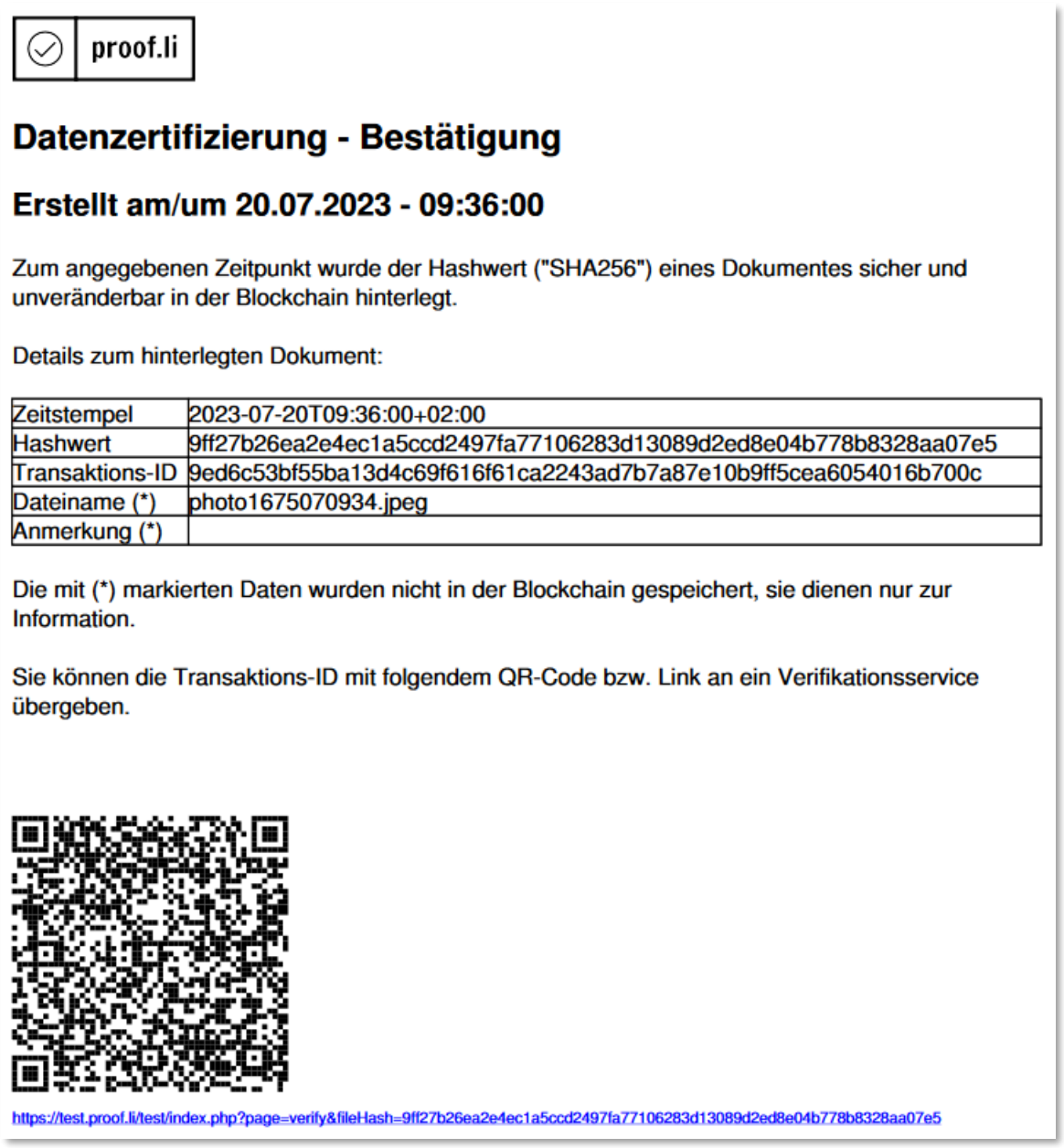
[8] System mit auf Englisch umschaltbarem GUI siehe https://test.proof.li – (Echtsystem unter https://proof.li)



Nach dem Absenden der Daten (Hashwert, optionale Anmerkungen) wird vom Server ein Eintrag in der Notarisierungs-Blockchain durchgeführt und das Ergebnis (Zeitstempel, Transaktions-ID …) an den Benutzer zurückgemeldet.



Der Benutzer kann nun eine Bestätigung des Eintrages als PDF Dokument mit QR-Code herunterladen. Dieses Dokument enthält die Referenz auf den Eintrag (Transaktions-ID), andere wesentliche Daten (Zeitstempel) sowie die optionalen Zusatzinformationen.



proof.li

Datenzertifizierung - Bestätigung

Erstellt am/um 20.07.2023 - 09:36:00

Zum angegebenen Zeitpunkt wurde der Hashwert ("SHA256") eines Dokumentes sicher und unveränderbar in der Blockchain hinterlegt.

Details zum hinterlegten Dokument:

Zeitstempel	2023-07-20T09:36:00+02:00
Hashwert	9ff27b26ea2e4ec1a5ccd2497fa77106283d13089d2ed8e04b778b8328aa07e5
Transaktions-ID	9ed6c53bf55ba13d4c69f616f61ca2243ad7b7a87e10b9ff5cea6054016b700c
Dateiname (*)	photo1675070934.jpeg
Anmerkung (*)	

Die mit (*) markierten Daten wurden nicht in der Blockchain gespeichert, sie dienen nur zur Information.

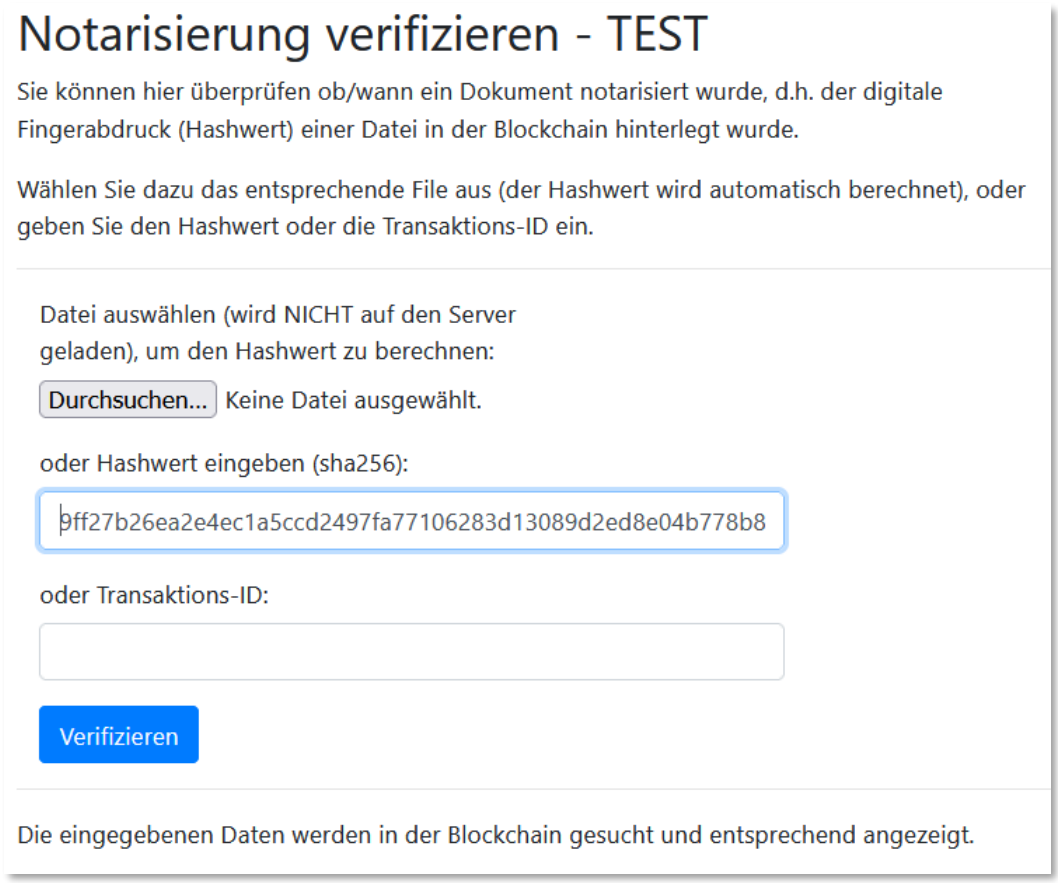
Sie können die Transaktions-ID mit folgendem QR-Code bzw. Link an ein Verifikationsservice übergeben.

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

5.2. Prüfen einer Notarisierung

Um eine Notarisierung zu einem späteren Zeitpunkt zu prüfen, gibt es zwei Möglichkeiten – bei beiden wird der Hashwert des Dokumentes mit den in der Blockchain gespeicherten Daten verglichen.

1. Bei der ersten Variante wird die auf der Bestätigung dargestellte Transaktions-ID (z.B. mit Hilfe des QR-Codes) im System gesucht und die Daten zum Dokument angezeigt. Der Hashwert des Dokumentes wird mit einem beliebigen (anderen) Tool errechnet und kann nun verglichen werden.



2. Bei der zweiten Möglichkeit wird zuerst wieder der Hashwert des Dokumentes im Webbrowser errechnet, mit diesem als Schlüssel die Notarisierung in der Blockchain gesucht und die damals erstellten Bestätigungsdaten angezeigt. Mit dieser zweiten Variante kann die Notarisierung eines Dokumentes also auch dann bewiesen werden, wenn dem Benutzer die Bestätigung nicht mehr vorliegt.

Ergebnis der Verifikation - TEST

Hashwert

"9ff27b26ea2e4ec1a5ccd2497fa77106283d13089d2ed8e04b778b8328aa07e5" gefunden.

Es wurden mehrere Einträge gefunden, d.h. das Dokument wurde mehrfach notarisiert. Der älteste Eintrag (der erste in der Liste) ist daher der relevanteste.

Eintrag 1/2

Blockhash	00632eacfb296ecc6564a8b916626c354540a7dd8bc7b940433d2e1e806ee75e
Blockzeit	2023-01-30T10:28:56+01:00
Bestätigungen	162278
Zeitstempel	2023-01-30T10:29:04+01:00
Hashwert (sha256)	9ff27b26ea2e4ec1a5ccd2497fa77106283d13089d2ed8e04b778b8328aa07e5
Transaktions-ID	01889862caaec0f64ba71f98a7ccc72446ce1d1688aeacd9ef544c5c10608e2a

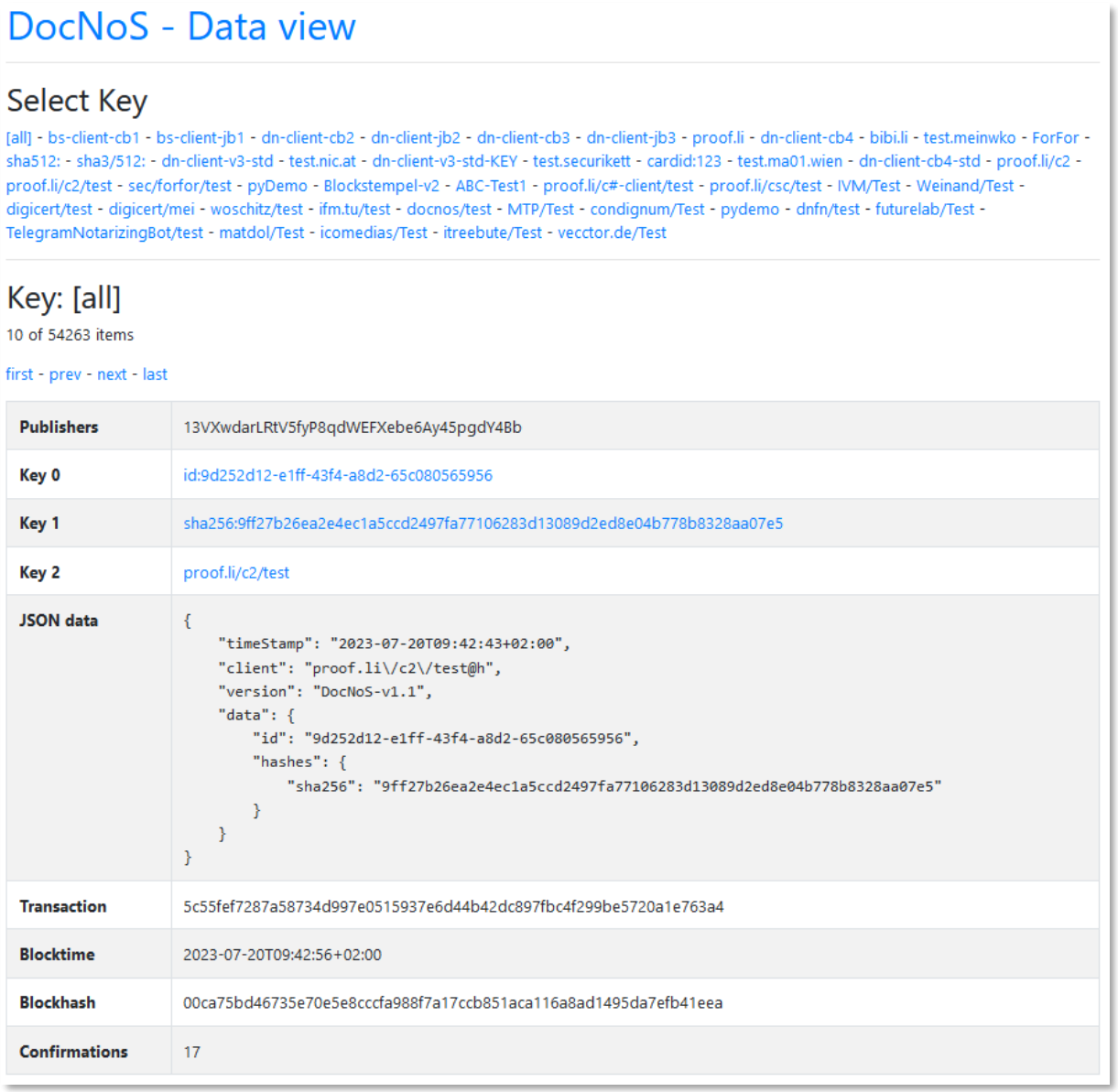
Falls mehrere Einträge zu einem Hashwert existieren (wie in diesem Beispiel) wird der älteste Eintrag zuerst angezeigt.

Eine Prüfung einer Notarisierung kann natürlich auf jedem teilnehmenden System erfolgen, nicht nur auf dem, welches den Eintrag ursprünglich erstellt hat.

Exemplarisch können Einträge, die mit dem Web-GUI erstellt wurden, mit dem Python-Democode verifiziert werden bzw. mit diesem erstellte Notarisierungen mittels Web-GUI verifiziert werden.

5.3. Direkte Abfrage der Daten

Im Testsystem (Private Sector Blockchain) können die in der Blockchain verspeicherten Daten auch direkt (d.h. ohne Suche über „Verifikation“) abgerufen werden. Der Url dazu lautet 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	{
	"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"
	}
	}
	}

Transaction	5c55fef7287a58734d997e0515937e6d44b42dc897fbc4f299be5720a1e763a4
Blocktime	2023-07-20T09:42:56+02:00
Blockhash	00ca75bd46735e70e5e8cccfa988f7a17ccb851aca116a8ad1495da7efb41eea
Confirmations	17

Auf der Seite werden alle in der Blockchain zu den Transaktionen vorhandenen Daten dargestellt, ähnlich wie in einem 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
