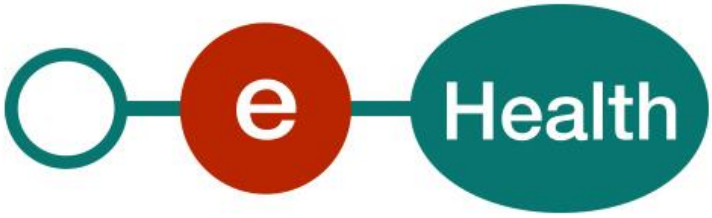

TimeStamping Client V2
User Guide
Version 1.3

This document is provided to you free of charge by the

eHealth platform

Willebroekkaai 38 – 1000 Brussel

38, Quai de Willebroeck – 1000 Bruxelles

Table of contents

Table of Contents

To the attention of: “IT expert” willing to integrate this service.

1. Document management

1.1 Document history

Version	Date	Author	Description of changes / remarks
1.0	25/06/2019	eHealth platform	Initial version
1.1	29/10/2019	eHealth platform	Added dbcr that include missing database indexes Added process for an installation from scratch of the database
1.2	12/10/2020	eHealth platform	Updates for new installations
1.3	26/04/2021	eHealth platform	Updated references and link

1.2 eHealth document references

All the document references can be found on the eHealth portal. These versions or any following versions can be used for the eHealth service.

ID	Title	Version	Date	Author
1	Cookbook Timestamp Authority V2	1.4	23/04/2021	eHealth platform
2	Cookbook Timestamp Consult V2	1.4	23/04/2021	eHealth platform
3	XSD Annex – TimeStamping Client V2.zip	1.0	05/06/2019	eHealth platform

2. Introduction

Within the Belgian hospitals, more and more ICT systems are being introduced to support the daily operations. Therefore, the hospitals are migrating towards a paperless environment, where the information is stored only electronically.

This makes it more difficult for the supervising authorities to fulfil their duties, as today the paper documents are the most important source of proof of activities.

However, migrating to an environment where all information is stored in an ICT environment also introduces new opportunities. One of these opportunities is the possibility to provide an irrefutable proof a certain piece of information existed on a given date by means of a trusted third party timestamp.

The aim of this project is

- to elaborate a specification and a reference implementation of such a timestamping system for the Belgian hospitals within the eHealth platform framework;
- to use this system in full-scale production for medication prescriptions.

A hospital wishing to introduce this timestamping system in its ICT system can implement the reference implementation. Nevertheless, it is also possible to adapt the reference implementation so that it suites better the existing ICT environment, or even make a new implementation from scratch. As long as the new implementation adheres to the specification, interoperability with the eHealth Platform timestamp server guaranteed.

3. Design overview

When designing the eHealth platform timestamp service and the reference implementation we choose not to opt for a close integration with the hospital information system (HIS). We choose to build a **journal** alongside the HIS. This journal contains dated (timestamped) **journal entries** about the operations of the hospital. Once an entry has been inserted into the journal, it cannot be modified or deleted without leaving traces. It is also impossible to insert a journal entry with an earlier date.

A journal entry contains a statement about some activity in the hospital. Technically a journal entry consists out of a format code and a sequence of bytes. The reference implementation of the eHealth platform trusted timestamp service is independent of the format of the journal entries.

The HIS should drop new journal entries into a buffer database.

To avoid an excessive growth of the load on the eHealth platform trusted timestamp servers, the trusted timestamp service will not request a timestamp for each individual journal entry. Instead, a number of journal entries will be grouped in to a timestamp bag (TSBag), and a timestamp will be requested on a timestamp bag. This does not impair the protection against changes of the journal indicated above.

Every five minutes, the trusted timestamp client program will pick the new journal entries in the buffer database, construct a TSBag of them and send the TSBag to the eHealth Platform trusted timestamp server to obtain a timestamp. Once the timestamp has been obtained, the trusted timestamp client program will verify the timestamp token returned in the response. Then it will move the newly timestamped journal entries, along with the timestamp bag, the timestamp and the search field information into the hospital archive.

Each trusted timestamp client and corresponding hospital archive must have a unique trusted timestamp client identification so that the eHealth Platform trusted timestamp server can distinguish its different clients.

To assist in later discussions on the correctness of the data and to provide extra information for forensic analysis the eHealth Platform trusted timestamp server will store all received TSBags and the delivered timestamps in the eHealth Platform archive.

The hospital archive verifier will verify the internal consistency of the hospital archive: correctness of the relationship between journal entries and timestamp bags, relationship between timestamp bags and timestamps, correctness of the timestamps and also the consistency between the eHealth platform archive and the hospital archive. This program can be run once a day (e.g. every night) to check the correct registration of the journal entries registered the previous day.

Fig 1 below illustrates the overall structure of the eHealth platform timestamp setup.

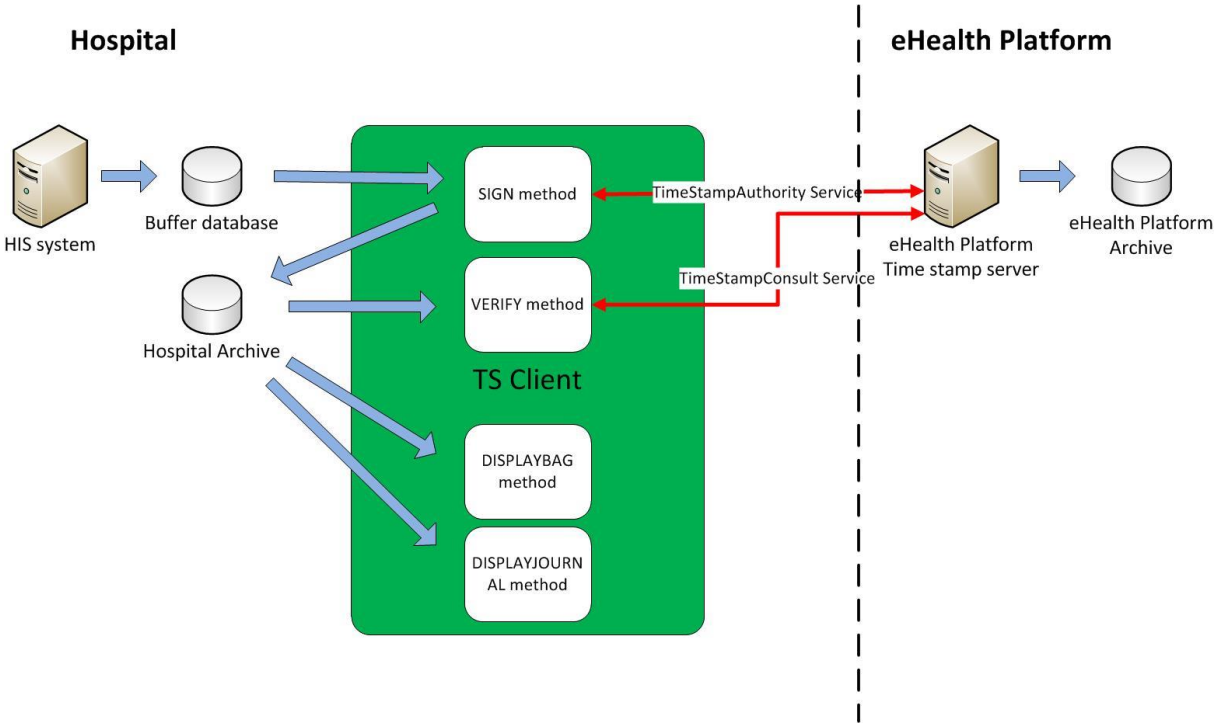

Fig 1

Any party wishing to implement its own implementation of the trusted timestamp software is free to do so. Obviously, the interfaces towards the other parties should be implemented according to the specifications described further in this document. These interfaces are marked in red in

Fig 1:

- The TimeStamp Authority service interface
- The TimeStamp Consult service interface

For all services, the eHealth platform has set up an acceptance and a production environment.

The acceptance environment is operational but the timestamps are generated by software and in a less secure environment (no hardware security environment). This environment can be used to test new implementations.

It is understood that in order to have a real valid timestamp, the eHealth platform production environment should be used.

4. Design details

4.1 Exporting journal entries or not?

Using timestamps in a given environment is relatively easy: one should extract the information to be timestamped into a journal entry, request a timestamp on that journal entry and archive the timestamp.

To verify a timestamp, one should reconstruct the journal entry and verify the timestamp.

When introducing timestamping in the healthcare environment we must take into account that information has a lifetime of several years, even several tens of years.

That means that at the moment one wants to verify a timestamp, the hospital information system is likely to have evolved, so that reconstructing the journal entry might not be straightforward, especially as –to prove the correctness of the timestamp- the reconstructed journal entry should be identical to the original one up to the bit level. We consider this as too error prone and costly to implement. It would require that with each version change of a clinical system some sort of bit-level compatibility should be assured, checked and maintained over the years. Changing to a very different clinical system would require complete embalmment of the old system in a resurrect able state in order to be able to regenerate the original statements.

Therefore, we will archive not only the timestamps, but also the journal entries that have been timestamped. The set of these journal entries will constitute the journal (logboek, journal de bord) of the hospital.

The information to be time stamped will be extracted from the hospital information system and archived independently from it.

These journal entries will be self-contained, so that they can be interpreted without consulting external information sources.

All documents will contain the identification of the doctor involved, the patient involved, the type of journal entry and the time the document was generated.

4.2 Handling different journal entry formats

The first practical application of the eHealth platform trusted timestamping system will be the electronic medication prescription. As discussed in chapter 4 we propose to use a KMHER format to format the journal entries for medication prescriptions. Using the KMEHR standard will already allow a great flexibility to represent various types of information, e.g. lab prescriptions …

Nevertheless, we believe the eHealth platform trusted timestamping system should allow even greater flexibility and be open for totally other journal entry formats.

For the core of the eHealth platform trusted timestamp service the format of the journal entries is irrelevant. Indeed the eHealth platform trusted timestamp servers handle only hash codes of journal entries.

However, it is important for the supervising authorities that the journal extracts can be handled flexibly.

4.3 Timestamping individual journal entries or timestamp bags?

The eHealth platform trusted timestamp service could be developed so that a timestamp is requested for each individual journal entry.

An alternative is to introduce a summary message we could call “timestamp bag” (TSBag). In our proposal, this timestamp bag contains containing references of individual journal entries and their hash codes. Timestamps could then be requested on the timestamp bags and not on the individual journal entries.

The concept of these TSBags is illustrated in Fig 2

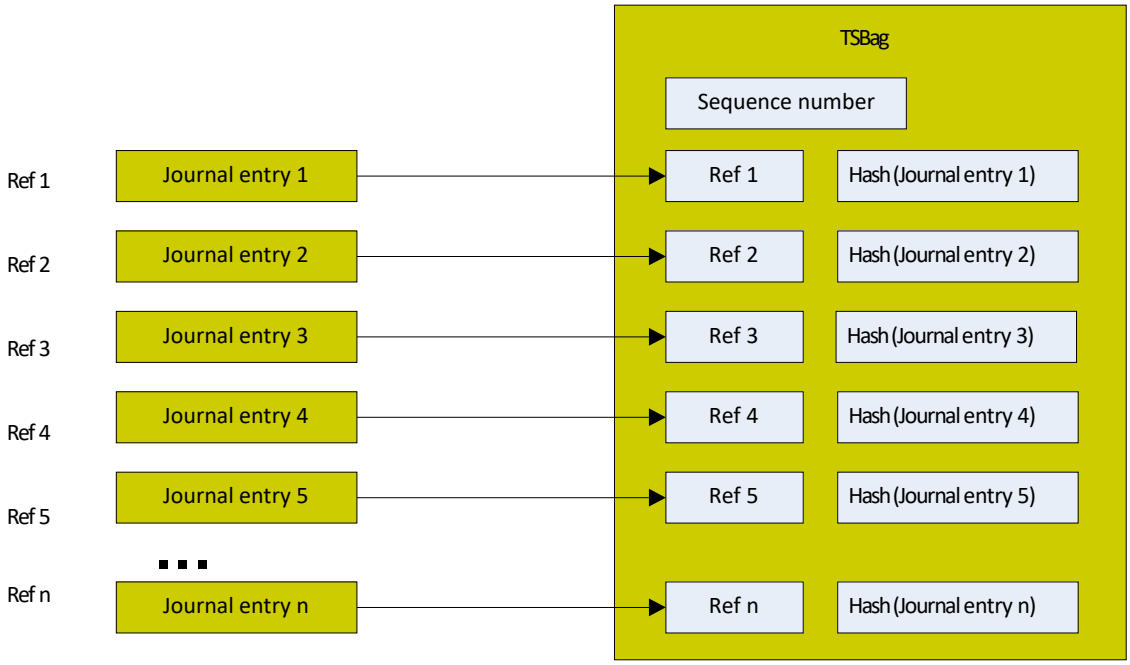

Fig 2

We can then develop a program that runs e.g. every 5 minutes, constructs a TSBag referencing all newly arrived journal entries and request a timestamp on this TSBag.

The introduction of the TSBag does not impair the proof of existence of the base journal entries; it makes it only slightly more complex, as you need 2 steps to establish the proof. To prove the existence of a particular journal entry at the time of the timestamp, one should:

- Prove the existence of the TSBag linked to the document using the standard timestamp techniques
- Verify that the hash code of the base journal entry is listed in the TSBag.

Using the TSBags has a number of advantages:

- The load on the timestamp servers is predictable: one timestamp every 5 minutes for each connected system
- The load is virtually independent of start-up of new applications: introducing new applications implies that the timestamp bags become larger. Larger timestamp bags will cause far less extra load on the timestamp servers than extra timestamp bags would cause.

We will use timestamp bags in the communication between time stamp client and the timestamp server.

We will send the complete TSBag to the timestamp server, which implies that there will be a copy of the TSBag in the log of the timestamp server.

The TSBags will also hold an identification number (TimestampID) to facilitate the search of the logs of the timestamp server for a particular TSBag.

The identification number will be unique, without a significance attached to it. Its only use is to facilitate communication when exchanging logs.

Below you find the xml schema for a TSBag:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd=" http://www.w3.org /2001/XMLSchema " xmlns:jxb="http://java.sun.com/xml/ns/jaxb
" jxb:version="1.0">
        <xsd:element name ="Bag" type="bagType"/>
        <xsd:complexType name="bagType">
                <xsd:sequence>
                        <xsd:element name="BagEntry" type="bagEntryType" minOccurs=" 1"
maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:attribute name="TrustedTimeStampClientIdentification" type="xsd:string"
use="required">
                        <xsd:annotation>
                                <xsd:documentation>ID identifying the archive and its owner
(technically this is the TTS Client ID)
                                </xsd:documentation>
                        </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="digestAlgorithm" type="digestAlgorithmType" use="required">
                        <xsd:annotation>
                                <xsd:documentation>Identifies the algorithm that is used to calculate
the hash values of the journal
                                        entries in this bag.For now it should always be SHA-256
                                </xsd:documentation>
                        </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="id" type="xsd:long" use="required">
                        <xsd:annotation>
                                <xsd:documentation>The ID of t he bag. The id is generated by the
archive owner an needs to be unique in the context of that archive only. The archive is identified by
its JournalID.
                                </xsd:documentation>
                        </xsd:annotation>
                </xsd:attribute>
        </xsd:complexType>
        <xsd:complexType name="bagEntryType">
                <xsd:sequence>
                        <xsd:element name="Hash" type="xsd:string">
                                <xsd:annotation>
                                        <xsd:documentation>The calculated hash of the journal entry in
base 64 format</xsd:documentation>
                                </xsd:annotation>
                        </xsd:element>
                        <xsd:element name="ID" type="xsd:long">
                                <xsd:annotation>
                                        <xsd:documentation>Unique number identifying the journal
entry</xsd:documentation>
                                </xsd:annotation>
                        </xsd:element>
                </xsd:sequence>
        </xsd:complexType>
        <xsd:simpleType name="digestAlgorithmType">
                <xsd:restriction base="xsd:NMTOKEN">
                        <xsd:enumeration value="SHA-256"/>
                        </xsd:restriction>
        </xsd:simpleType>
</xsd:schema>
```

4.4 Protocol between timestamp client and timestamp server

We have done a study of the literature on existing protocols to handle timestamp requests and replies.

Two protocols were examined more in detail: RFC3161 (see http://www.ietf.org/rfc/rfc3161.txt) and Oasis-DSS protocol with the timestamp profile (see ***http://www.oasis-open.org/committees/dss/***).

Based on following elements Oasis-DSS should be preferred:

- RFC3161 states explicitly that no authentication mechanism is specified. Oasis-DSS includes specifications for authentication mechanisms. For the project, we need a mechanism to authenticate the clients in order to avoid abuse.
- Oasis-DSS allows the client to send a document to be timestamped, whereas RFC3161 only accepts a hash code. This implies that Oasis-DSS allows for a more detailed log of the requests, which we deem useful to settle discussions later on.

The timestamp server will implement the Oasis-DSS protocol with the timestamp profile to accept timestamping requests from the hospitals.
Only the features really needed for the timestamp service will be implemented.
The hospitals will access the eHealth platform timestamp server through the internet. This implies, that an authentication scheme should be put in place to limit the access to the intended users.
In a first layer of access control, eHealth platform will only accept connections from registered IP addresses. This will allow keeping off eventual denial of service attacks with the minimal impact possible.

4.5 Handling multiple clinical systems in one hospital

When a hospital has more than one clinical system, each generating messages to be timestamped, 2 approaches are possible:

- one could set up a single timestamping client environment, let all clinical systems deliver their messages to be timestamped to the single timestamping environment, which collects messages from all clinical systems into one TSBag and sends it off to the timestamp server. See Fig 3 below.

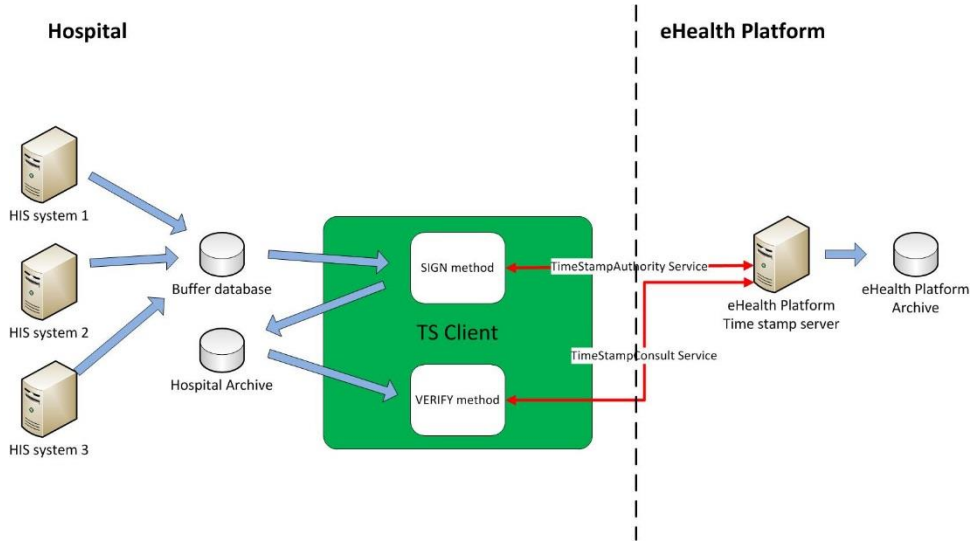

Fig 3

- one could also set up a timestamping client for each clinical system, so that each clinical system is responsible for getting its messages timestamped. See Fig 4 below.

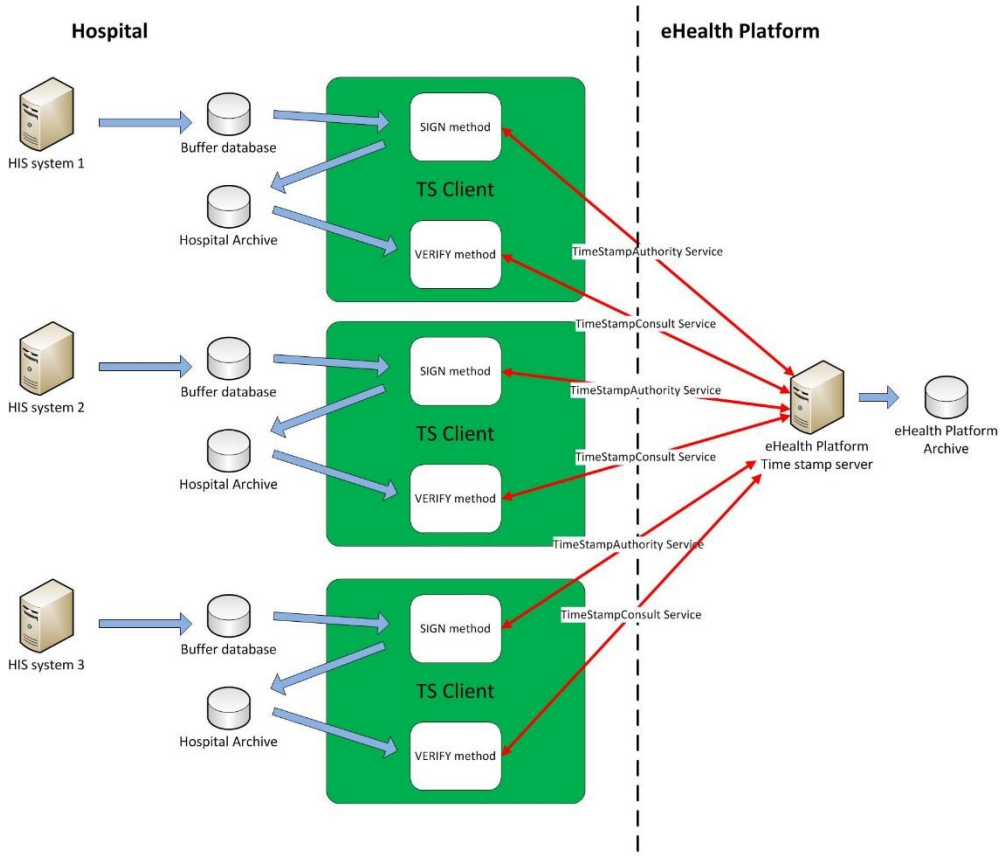

Fig 4

The choices made on this issue by an individual hospital do not affect the validity of the timestamps. The only impact is on the load on the eHealth platform timestamp servers.
A hospital is free to choose the option that is most suited for its environment, as long as the hospital does not exceed the load taken into account for each hospital.
Hospitals should be aware that it is not possible to merge 2 hospital archives, as this would violate the consistency checks with the eHealth platform trusted timestamp service archive.
When a separate eHealth platform trusted timestamp client environment has been set up, it should be maintained as a separate entity for as long as the archive has to be kept. So decisions on this issue should not be made lightly.
Each individual timestamp client and hospital archive should have its own trusted time stamp client identification. Failing to adhere to this rule will make all consistency checks between the hospital archive and the eHealth platform archive fail!

4.6 eHealth Platform trusted timestamp archive

We should take into account that the data processed by the timestamping system is to be archived for a considerable period and will be used infrequently. This implies that accidental or malicious alterations to the archived data might be discovered considerable time after they occurred. At that moment, a serious discussion might arise on what went wrong. There might even be a discussion whether the hospital's archiving and security system has failed or the eHealth platform timestamping system had delivered an incorrect timestamp. Based on the experience with similar systems we consider it useful to keep a log of all requests and replies at the server side. As the log is stored at the server side, it is an independent source of information that can help to settle discussions between a hospital and a supervising authority.

The timestamp server stores all received requests and all replies in an archive. This archive will be stored for the period the hospital has to archive the messages that have been time stamped. (30 years, whereof 5 years will be stored online)

4.7 Hospital buffer overview

4.7.1 Table JOURNAL (DEPRECATED)

This table exists only in installations that were installed with the first version of the timestamp client. Existing installations may continue to use it to insert its prescriptions in order to process them by the TSClient. The TSClient will copy the data of this table to the table JOURNAL_STAGING.

Element	Description
Id	Identification number of the journal entry
BYTES	Blob containing the journal entry
Type	Type/Format of the journal entry. The TSClient uses the format KMEHR except for the incident, in this case the value is « INCIDENT »

4.7.2 Table JOURNAL_STAGING

This table is the one in which the hospital will insert its prescriptions in order to create the TSBag that will be timestamped. It also contains the incident reports generated by the TS Client before they are packed in a TSBag and timestamped.

This table has a buffer role, meaning if the bag is correctly created, timestamped and stored in the ARCHIVE DB, the content of this table is emptied. On the contrary, if there is an issue in the sign process (e.g. internet failure), the information are kept in table JOURNAL_ARCHIVE in order to generate the same TSBag and timestamp it in the next run of the TS Client.

Element	Description
Id	Identification number of the journal entry
Content	Blob containing the journal entry
Type	Type/Format of the journal entry. The TSClient uses the format KMEHR except for the incident, in this case the value is « INCIDENT »

BAGID	This field is the identification number of the bag in which the journal entry is added. The value is completed when the bag containing the journal entry is created
DATEBAGGED	Date when the bag containing the journal entry was created. The value of this field is also completed when the bag containing the journal entry is created

4.8 Hospital archive overview

4.8.1 Table BAG_ARCHIVE

This table contains the timestamped bags of the hospital

Element	Description
Id	Identification number of the TSBag
Content	Content of the TSBag which was sent to eHealth Platform
DateCreated	Creation date of the TSBag
DateSent	Date when the TSBag was sent to eHealth Platform
Response	Response returned by the TimeStamp Authority service (this response contains the timestamp token of the TSBag)
TimestampId	The sequence number of the timestamp as generated by the timestamp server
TimestampDate	The date and time of the timestamp, generated by the timestamp server

4.8.2 Table JOURNAL_ARCHIVE

This table contains the journal entries linked to the timestamped bags of table BAG_ARCHIVE

Element	Description
BagId	Id of the TSBag in which the journal entry was added
Id	Identification number of the journal entry
Content	Blob containing the journal entry
Type	Type/Format of the journal entry. The TSClient uses the format KMEHR except for the incident, in this case the value is « INCIDENT »

The hospitals and eHealth Platform will need to set up an archive that guarantees that the hospital journal, the TSBags and the ti mestamps are stored safely and completely unchanged for as long as the hospital journal is to be kept.
Both hospital and the eHealth Platform archive will use the identification of the requesting timestamp client, the date and time of the timestamp and the sequence number of the timestamp as keys to access the archives. This will make it easy to match both archives upon inspection.

4.9 Archiving period

The current legislation specifies that the medication prescriptions should be archived for 10 years. The medical record of a patient should be archived for 30 years. If we assume all information in the journal is relevant for the medical record of the patient, we should archive the journal entries, the TSBags and the timestamps for 30 years.

4.10 What in case of Internet failures?

We should take into account that the network connection between the timestamp client in a hospital and the eHealth Platform timestamp server can break down. The connection could even break down in the middle of a timestamping operation: we could end up in a situation where the timestamp client has sent a request, the timestamp server has processed it and sent the reply, but the timestamp client never received the reply.

The regulatory environment of the timestamping system should allow the hospital to continue using its internal IT system in case of failure of the timestamp system, no matter what caused the failure: internal problem within the hospital, problems with the internet or problems at the eHealth platform site.

It is understood that hospitals and eHealth platform should set up the timestamp system so that a breakdown is an exceptional event. If the timestamp system would breakdown regularly, it would serve no purpose at all.

When we end up in a situation as described above, where a TSBag has been processed by the timestamp server but the reply got lost, the timestamp client obviously should request again a timestamp for the same journal entries.

There are a number of alternatives to handle this situation:

- One could repeat the request for exactly the same TSBag, with exactly the same content
- One could create a new TSBag, containing the journal entry hashes contained in the failed TSBag and extra journal entry hashes.

The latter alternative could raise suspicions that the hospital has been using a scenario as discussed in paragraph “Risk analysis”, trying to cover up changes in the journal.

To avoid this suspicion, it is better to use the first alternative.

When the timestamp client did not received a correct reply for a timestamp request, the timestamp client will keep on repeating the request using **exactly** the same TSBag, until a correct reply is received.

We should also take into account that a supervising authority might ask questions on the reason why there is a time gap in the sequence of TSBags.

To facilitate answering questions on time gaps in the sequence of TSBags, the hospitals should insert an "incident report" in the journal and have it timestamped along with the other journal entries.

4.11 Facilities to consult the eHealth Platform archive

The eHealth Platform timestamp server offers 2 interfaces to extract archive information through the TimeStamp Consult web service:

4.11.1 Completeness check

An interface to verify the completeness of a set of timestamps: this interface takes as arguments:

- IDHospital (= trusted timestamp client identification)
- a time period
- a list of TimeStampIdentifications: these are the sequence numbers of the timestamps delivered

As result, the eHealth platform will reply with a flag indicating whether or not the list was complete and if it was not complete, the list of missing TimeStampIdentifications.

The "verify" method integrated into the TimeStamp Client performs a call to this interface in order to also verify the completeness of the hospital archive.

This service will take into account the handling of internet failures as explained in paragraph 0. So in case an extra timestamp is found in the eHealth platform archive, but the timestamp bag of the missing one is the same as the next one, this will not be reported as a missing timestamp, assuming the eHealth platform trusted timestamp client had never received the reply and had repeated the request.

For now, the period that can be specified as argument can be at most 24 hours.

See more information about this interface in cookbook of web service TimeStamp Consult V2:

Dutch version: https://www.ehealth.fgov.be/ehealthplatform/nl/service-elektronische-datering-timestamping

French version: https://www.ehealth.fgov.be/ehealthplatform/fr/service-datation-electronique-timestamping

4.11.2 Get a TSBag back from the eHealth platform archive

The second service allows extracting a TSBag and the corresponding timestamp from the eHealth platform archive. The identification of requesting timestamp client, the date and time it was processed and the sequence number will be used as key to specify which TSBag and timestamp should be returned.

This interface is not integrated in the method proposed by the TS Client but you can find more information about it in the cookbook of web service TimeStamp Consult V2

4.11.3 Authentication

These requests to consult the eHealth platform trusted timestamp service archive must be signed in the same way the timestamping sign requests have to be signed. The eHealth platform trusted timestamp service implements access control based on the signature used. To access the archive of the data registered for one of the timestamp clients of a hospital you must sign the consultation request with a signature registered for that timestamp client.

4.12 Choosing the key length for the digital signature and the hashing algorithm

The RSA algorithm is the key algorithm for digital signatures. As with almost any algorithm, RSA can be under a brute force attack. This is a type of attack where one simply tries every possible key until the correct one is found.

The number of possible keys is obviously function of the key length: a greater key length implies a greater number of possible keys. Therefore, using an appropriate key length is an important element in the security of a digital signature. The keys should be long enough to make a brute force attack computationally infeasible. This means that in practice it is impossible to mount a brute force attack, as it would take far too much computing resources. On the other hand, a key that is too long wastes computing power.

On the website ***http://www.keylength.com/*** one can find a number of suggestions on this theme. This web site contains information based on the recommendations of the US National Institute of Standards and Technology. We find there a recommendation to use a RSA key length of 2048 bits to keep the information secure until the year 2030.

A similar reasoning can be held on the security of hashing algorithms.

A cryptographic hash function is considered insecure if either of the following is computationally feasible:

- to find a (previously unseen) message that matches a given digest
- to find a "collision", where two different messages have the same message digest.

On the same website mentioned above we find also an overview on the security of hash algorithms.

To keep information secure until 2030, the timestamp client of the hospitals and the timestamp server of eHealth Platform should use at least SHA 224.

As implementations of SHA 256 are more readily available, the reference implementation will use SHA 256.

The state of the art in this area should be monitored permanently. When the security of the chosen key length and/or the chosen hashing algorithm becomes questionable, the eHealth platform should take measures to migrate the service to a higher key length or a better hashing algorithm.

In this case, measures should also be taken to timestamp the existing archive again using the new settings. As this will not be necessary in the near future, we will not implement this and leave this for future extensions.

4.13 Regulatory aspects within the hospital

According to the existing legislation, some persons within the hospital are responsible for keeping the archive of a number of documents. E.g.: the head pharmacist in each hospital is legally responsible for archiving the medication prescriptions.

When a hospital moves to a paperless environment, the archive will be stored on some IT system, probably a centralized IT system that is not under direct supervision of the head pharmacist.

To officially transfer the responsibility for keeping the archive, each hospital will need to make an agreement between the head pharmacist, the head of the IT department and the hospital management stating the transfer of the responsibility.

5. Journal entry format for medication prescriptions

One of the objectives of this project is to propose a format for a journal entry representing a medication prescription.

A lot of work has already been done in developing standards for messages to exchange information between different healthcare workers in het KMEHR projects. When it comes to the level of coding of the information, the KMEHR formats allow for different levels: level 1, which has almost no codes up to level 4 where all information is well structured and coded. With “coded” in this context, we mean that all information refers to coding tables, e.g. the medication to be administered is not represented by its name, but by an index in a list of possible medications.

In the day-to-day environment it is very well possible to code the references to the patient and doctor involved, using e.g. the registration number at the Belgian social security crossroads database. However, when it comes to a structured and coded representation of the medication this is far less obvious.

Other projects are currently working on establishing structured representations and coding lists for medication.

We feel that, given the current state of the art, the best solution is to use a KMEHR level 2 format for a journal entry that represents a medication prescription, where patient and doctor are represented in a structured way, but the medication is represented as free text.

How this might look like is illustrated below:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<kmehrmessage xmlns="http://www.health.fgov.be/telematics/kmehr/schema">
        <header>
                <standard>
                        <cd SV="1.0" S="CD-STANDARD">20030909</cd>
                </standard>
                <id SV="1.0" S="ID-KMEHR">710-32209-000.1220439679382.level2</id>
                <date>2008-09-03</date>
                <time>13:01:19</time>
                <sender>
                        <hcparty>
                                <id SV="1.0" S="ID-HCPARTY">710-32209-000</id>
                                <cd SV="1.0" S="CD-HCPARTY">orghospital</cd>
                                <name>UZ Leuven</name>
                        </hcparty>
                </sender>
                <recipient>
                        <hc party>
                                <id SV="1.0" S="ID-HCPARTY">710-32209-000</id>
                                <cd SV="1.0" S="CD-HCPARTY">orghospital</cd>
                                <name>UZ Leuven</name>
                        </hcparty>
                </recipient>
        </header>
        <folder>
                <id SV="1.0" S="ID-KMEHR">1</id>
                <patient>
                        <id SV="1.0" S="ID-PATIENT">39082920591</id>
                        <id SV="1.0" SL="EADNR" S="LOCAL">72913841</id>
                        <firstname>voornaam</firstname>
                        <familyname>familienaam</familyname>
                        <birthdate>
                                <date>1939-08-69</date>
                        </birthdate>
                        <sex>
                                <cd SV="1.0" S="CD-SEX">male</cd>
                        </sex>
                </patient>
                <transaction>
                        <id SV="1.0" S="ID-KMEHR">1</id>
                        <cd SV="1.0" SL="VALIDATIE_OMIT" S="LOCAL">i</cd>
                        <cd SV="1.0" SL="TUPLE_TYPE" S="LOCAL">medischVoorschrift</cd>
                        <cd SV="1.0" SL="TUPLE_DKEY" S="LOCAL">2900d9044177</cd>
                        <cd SV="1.0" SL="VALIDATIE_UID" S="LOCAL">hdprae0</cd>
                        <cd SV="1.0" S="CD-TRANSACTION">treatment</cd>
```

```
                        <date>2008-08-21</date>
                        <time>11:54:13</time>
                        <author>
                                <hcparty>
                                        <id SV="1.0" S="ID-HCPARTY">1-73629-01-001</id>
                                        <cd SV="1.0" S="CD-HCPARTY">persphysician</cd>
                                        <firstname>Herbert</firstname>
                                        <familyname>De Praetere</familyname>
                                </hcparty>
                        </author>
                        <iscomplete>true</iscomplete>
                        <isvalidated>true</isvalidated>
                        <text L="nl">Elektronisc h Medic atie Voorschrift</text>
                        <text L="nl">Gevalideerd door: dokter x (riziv nummer>)</text>
                        <text L="nl">Enkelvoudige medic atie: ALDACTONE</text>
                        <text L="nl">Dosis: 1.0 tabl</text>
                        <text L="nl">Toedieningsweg: PO</text>
                        <text L="nl">Moment van to ediening: 26-08-2008 08:00</text>
                </transaction>
        </folder>
</kmehrmessage>
```

The KMHER format has been developed to exchange information between different healthcare parties. That is why there are mandatory “sender” and “recipient” entries. As we are using the KMEHR format here to store information, these entries serve little purpose. Nevertheless, as they are mandatory, they are present, both referring to the hospital concerned.

As already pointed out, the eHealth platform trusted timestamp service is open to handle any format of journal entry, so, when standards evolve, new formats can easily be adopted.

6. Risk analysis

6.1 Can a hospital prove that a certain document existed at the moment it was timestamped?

Yes, the hospital can do this based on the information stored in its own timestamping archive. Let us assume that, the hospital wants to prove that a journal entry ((journalEntry8) existed at the moment the corresponding TSBag (TSBag2) was timestamped. This takes the following steps:

- Verify the digital signature of the timestamp of TSBag2. This proves that the timestamp is valid and was generated by the eHealth platform trusted timestamp service.
- Compute the hash code of the TSBag and verify that it is equal to the hash code mentioned in the timestamp. This proves that the timestamp is a timestamp of TSBag2 and TSBag2 existed at the time mentioned it the timestamp.
- Compute the hash code of journalEntry8 and verify that this hash code is equal to the hash code of this document mentioned in the TSBag. This proves that journalEntry8 was indeed the document referred to in TSBag2, timestamped at the time mentioned in the timestamp.

So, to prove that a given journal entry existed at the specified moment, one only needs the journal entry concerned, the corresponding timestamp bag and the corresponding timestamp.

6.2 What if the timestamp bag has been altered after timestamping it?

Let us start from the same question as in paragraph 0: the hospital wants to prove that journal entry 8 existed at the moment the timestamp of TSBag2 existed, but someone has modified journal entry 7 (also linked to TSBag2). Let us also assume the intruder has gone one step further and also has modified TSBag2, so that it now contains the new hash code of journal entry 7. Modifying the timestamp is impossible without leaving traces as the timestamp is digitally signed by the eHealth Platform trusted timestamp service.

When following the procedure depicted in the paragraph above, step 2 will fail as the TSBag has been modified, thus invalidating the proof of existence of journal entries 7 and 8.

To figure out what has happened, the hospital or the supervising authority can ask eHealth Platform to receive a copy of TSBag2 and its timestamp. Based on this information the hospital can figure out what information has been altered

It is obvious that the original information cannot be recovered.

The loss of the original data can only be avoided by an appropriate archiving system with adequate protection within the hospital. Hospitals have set up procedures to archive the current paper documents in a proper way. Similar procedures should be set up to ensure proper archiving of the electronic documents.
The timestamping system can only help to prove to third parties that the hospitals archive was not corrupted.

6.3 Can a hospital correct the hospital journal without leaving traces?

This aspect was not included in the original project plan where the only aim was to prove that a journal entry existed at a given moment in time.

However, we believe that, in case of a juridical investigation, it might be useful to be able to prove completeness and originality of the hospital journal. Moreover we believe this question can be handled to a very reasonable extend without extra overhead.

The answer on this question depends on the timing.

- If the journal entry is inserted in the hospital journal, but it has not yet been processed into a TSBag, then obviously the hospital can modify the journal entry without leaving any trace. As we propose to generate a TSBag every 5 minutes, the hospital has on average a time window of 2.5 minutes to correct the journal. Given the small time window, it is very unlikely that this can be used to cover up something.
- If the document has already been processed into a TSBag, the hospital might try to cover up the traces by reusing the TSBag sequence number. Let us assume that we want to change journalEntry8 some 2 hours after it was created. So TSBag2 up to TSBag24 exist. One could then delete TSBag2 up to TSBag24, generate a new TSBag2 referring to all documents that were referred to by TSBag2 up to TSBag24. Upon inspection, it will be obvious that something went wrong, as there are no TSBags for a period of 2 hours. The hospital could pretend that this is due to a temporary disturbance of the internet connection that caused a temporary unavailability of the timestamp system. As the sequence numbers of the TSBags are correct, the hospital will pass a superficial inspection. However, when the inspector compares the TSBags as presented by the hospital with the archive of eHealth Platform, then it becomes obvious that a number of TSBags have been omitted.

We conclude that in this situation, it is impossible to change the log without leaving traces

6.4 Can a hospital prove that no information has been omitted from the hospital journal?

In fact, this is a variation of the question in paragraph 6.1, the answer is also similar: when the information has not been timestamped yet, information can be modified, but also deleted.

One can however prove that no information timestamped in a given period has been omitted:

- In a first step, one can check the internal consistency of the hospital journal: digital signatures of the timestamps, correspondence of the timestamps to the timestamp bags and correspondence of the journal entries to the timestamp bags. This proves that all journal entries in the hospital journal are legitimate, but does not prove that no information has been omitted by deleting a timestamp bag and all related journal entries.
- The proof that no information has been omitted from the hospital journal can be obtained by comparing the hospital archive with the eHealth Platform archive. If this comparison shows that no timestamps are missing from the hospital archive, then one can be assured the presented hospital journal is complete.

We must take into account that this might cause a privacy problem: suppose a hospital is requested to prove that all hospital journal documents on a given patient are presented. To prove the completeness, one must have access to all information stored in the hospital journal, thus to the information on other patients. The best solution to circumvent this issue is probably to involve a trusted third party.

6.5 How can a hospital be sure that all relevant information is stored in the journal?

This problem is very similar to a problem hospitals have to cope with today: how does the hospital make sure that all relevant paper documents are archived?

Today, every hospital has set up a procedure to collect and archive all relevant paper documents. In the new environment with only electronic documents, the hospitals will need to set up procedures (computer programs) to collect all relevant documents and insert them into the hospital journal.

Obviously, the timestamping system can only offer a facility to prove that a journal entry existed at a given moment in time. Nothing can be done about information that was never inserted into the journal.

Hospitals are encouraged to check the completeness of the hospital journal on a regular basis, e.g. by running a program every day that counts the number of prescriptions in the operational IT system of the hospital and in the journal. These procedures will be specific for each hospital information system.

6.6 Verifying the correct working of the hospital journal

By design, the hospital journal is an environment that is not used by the IT system supporting the day-to-day operations of the hospital. Therefore, errors in this system and even a breakdown of this environment might go by unnoticed. This should be avoided, given the importance of the hospital journal.

The hospitals should set up a procedure to verify the correct working of the trusted timestamp client. Hospitals are encouraged to set up procedures to verify the completeness and correctness of the contents of the hospital journal on a day-to-day basis. One method might be to include sequence numbers in the journal entries and let the timestamp client verify the sequence, so doubles or omissions do not go by unnoticed. One could also consider running consistency checks e. g once a day to compare today's hospital journal with the operational hospital information system.

7. The reference implementation

7.1 Installation

7.1.1 Before the migration

(If it is your first installation of the eHealth TS Client, you can skip this paragraph)

System setup before the migration:

1. Stop your third party software from inserting entries/prescriptions in the TSBuffer. journal table.
2. Stop the timeStamp client v1 Windows Service and other scheduled tasks (checkArchiveDb, ...).
3. All tables in TSBuffer should be empty.
4. Make a backup of the whole database TS_ARCHIVE, the configuration files and the installed client executable code.

7.1.2 Installation and configuration

7.1.2.1 eHealth configuration

In order to be able to access eHealth timestamp webservices, you need to have:

1. A certificate defined/installed in the specific eHealth environment; you can reuse the ones (one for acceptance and one for production) from the old client.
2. A tsa client name (configured for using Timestamping Authority and Consult Service V2) that identifies uniquely your client that will be linked to the certificate. Again, it can be the same as the v1's.

You can request the above items from eHealth via the email address ***integration-support@ehealth.fgov.be***.

You can have a specific configuration for acceptance testing (specific acceptance certificate but same tsa client name than production) for acceptance if you want to test your client on our acceptance environment (strongly recommended).

7.1.2.2 Server environment

- **JAVA** should be installed. The Java OpenJdk 8 (currently 1.8.0_171-1-ojdkbuild) is the version supported by the eHealth platform, because it is in Oracle Long Term Support. If you really want a lower version, please ensure that TSL 1.2 is supported.
- Official **WINDOWS** version supported is Microsoft Windows 7 Enterprise, Version 6.1.7601, Service Pack 1 Build 7601
- Official **LINUX** version supported is Red Hat Enterprise Linux Server release 6.10 (Santiago)
- **SQL server 2008 R2 SP2** is the supported version for the database.

By ***official version supported***, we mean that we test the client on these systems before each new release. Other versions can be used but there is no guarantee that the client will work properly.

7.1.2.3 Client installation

1. Create a new root folder for your client, let's call it **TSA_HOME**
2. Download the latest release from eHealth public repository and copy them to your **TSA_HOME** directory:
   a. The timeStamp client v2 executable:
   http://repo.ehealth.fgov.be/artifactory/webapp/#/artifacts/browse/tree/General/maven2/be/fgov/ehealth/timestamping/timestamping-client-boot

   b. The timeStamp client v2 configuration validator:

http://repo.ehealth.fgov.be/artifactory/webapp/#/artifacts/browse/tree/General/maven2/be/fgov/ehealth/timestamping/timestamping-client-validator

c. A configuration template:
http://repo.ehealth.fgov.be/artifactory/webapp/#/artifacts/browse/tree/General/maven2/be/fgov/ehealth/timestamping/timestamping-client-configurationtemplate

3. Extract the **configurationtemplate** jar content under the **TSA_HOME** directory
4. Copy your client certificate (the certificate that identifies your hospital, obtained in the **eHealth configuration** upper section) to authenticate your requests for eHealth timestamp webservices security into the config/certificates folder (e.g. in the config-acc/keystores directory for the acceptation environment).
5. Remove the file named **hospitalSigningCertificate.p12** then rename your client certificate to **hospitalSigningCertificate.p12**.

You should now have something like this:

```
TSA_HOME
    timestamping-client-boot-1.0.0.jar
    timestamping-client-validator-1.0.0.jar
    db
        | client_v2_ddl.sql
    META-INF
        | ...
    config-acc
        | keystores
            | hospitalSigningCertificate.p12
            | acc
                | caCertificateKeystore.jks
                | truststore.jks
                | tsacertificate.jks
                | tslostore.jks
            | prd
                | caCertificateKeystore.jks
                | truststore.jks
                | tsacertificate.jks
                | tslostore.jks
        | application.properties
        | be.ehealth.technicalconnector.properties
        | logback.xml
    config-prd
    ... (same file structure)
```

7.1.2.4 Database

Installation from scratch

1. Run the script *TSA_HOME/db/**ddl_sqlserver_v2_scratch.sql*** to create the new tables and triggers for the client
2. Execute the dbcr TSA_HOME/db/**ddl_sqlserver_v2_dbcr_1.sql** that add missing indexes

Migration from TS Client V1

1. Run the script *TSA_HOME/db/**ddl_sqlserver_v2.sql*** to create the new tables and triggers for the client
2. Initialize the seed of the sequence for the journal ids

- SELECT IDENT_CURRENT('bufBehealthTSBag');
- DBCC CHECKIDENT ('bag', RESEED, [the value selected in upper line]);

3. Execute the dbcr TSA_HOME/db/**ddl_sqlserver_v2_dbcr_1.sql** that add missing indexes

7.1.3 Configuration

Fill-up the following configuration files:

1. In **application.properties**:
   - **java.mail.***: host and port number of your mail server
   - **mail.***: subject, destination and sender of a mail that is sent by the application
   - **buffer.datasource.***: url, username and password for your BUFFER database
   - **archive.datasource.***: url, username and password for your ARCHIVE database

2. **In be.ehealth.technicalconnector.properties of each environment:**
   - **timestamping.tsclientid**: your tsa client name
   - **KEYSTORE_DIR**: absolute path to your TSA_HOME/keystores directory
   - **http.***: host, port, user and password of your HTTP proxy (if you happen to have one)
   - **https.***: host, port, user and password of your HTTPS proxy (if you happen to have one)
   - **credentials.location**: the relative path to your certificate, starting from your TSA_HOME directory
   - **credentials.password**: the password of your certificate
   - **credentials.alias**: the alias of your certificate

3. In **logback.xml**:
   - Choose the log folder, by default it's **TSA_HOME/logs/[acc|prd].**
   - You can also adapt the policy retention, see ***https://logback.qos.ch/manual/configuration.html*** for more information

7.1.4 Validation of your configuration

For all environments, validate the runtime environment and the configuration.

From the **TSA_HOME** folder, run with a command-line tool the following command:

```
java -Dloader.path=[path_to_you_configuration_folder]\ -jar timestamping-client-validator-1.0.0-beta-1.jar
```

- Do not forget the '****' at the end of the environment specific config folder)
- Do not forget to set the JAVA_HOME var or use java executable full path)

It will display a report of the state of the configuration.

Pay attention that if you want to reduce the logs nuisances, you should remove the **logback.xml** file before running the configuration validation and put it back afterwards. It you do not remove it, the configuration validator will consider it and provide you with a fully detailed log result. This might be hard to read and understand, as it will display unnecessary information.

Once the configuration validated, you will have a report that looks like this:

```
16:52:23.388 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:23.389 | Logger:73 | |                                                                                     |
16:52:23.390 | Logger:73 | |                        eHealth TSA Client Configuration Validator                   |
16:52:23.390 | Logger:73 | |                                                                                     |
16:52:23.390 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:23.390 | Logger:73 |
16:52:23.392 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:23.393 | Logger:73 | | Verifying presence of configuration files                                         |
16:52:23.393 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:23.393 | Logger:73 |
16:52:23.394 | Logger:73 |             [OK] [application.properties] exists
16:52:23.394 | Logger:73 |             [OK] [be.ehealth.technicalconnector.properties] exists
16:52:23.395 | Logger:73 |
16:52:23.397 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:23.398 | Logger:73 | | Verifying [application.properties] configuration file                             |
16:52:23.398 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:23.398 | Logger:73 |
16:52:23.399 | Logger:73 |             [OK] [java.mail.host] set
16:52:23.399 | Logger:73 |             [OK] [java.mail.port] set
16:52:23.400 | Logger:73 |             [OK] [java.mail.transport.protocol] set
16:52:23.400 | Logger:73 |             [OK] [mail.subject] set
16:52:23.400 | Logger:73 |             [OK] [mail.from] set
16:52:23.400 | Logger:73 |             [OK] [mail.to] set
16:52:23.401 | Logger:73 |             [OK] [buffer.datasource.url] set
16:52:23.401 | Logger:73 |             [OK] [buffer.datasource.username] set
16:52:23.402 | Logger:73 |             [OK] [buffer.datasource.password] set
16:52:23.402 | Logger:73 |             [OK] [archive.datasource.url] set
16:52:23.403 | Logger:73 |             [OK] [archive.datasource.username] set
16:52:23.403 | Logger:73 |             [OK] [archive.datasource.password] set
```

```
16:52:23.404 | Logger:73 |
16:52:23.404 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:23.404 | Logger:73 | | Verifying [be.ehealth.technicalconnector.properties] configuration file           |
16:52:23.405 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:23.406 | Logger:73 |
16:52:23.407 | Logger:73 |             [OK] [timestamping.tsclientid] set
16:52:23.409 | Logger:73 |             [OK] [endpoint.ts.authority] set
16:52:23.410 | Logger:73 |             [OK] [endpoint.ts.consult] set
16:52:23.412 | Logger:73 |             [OK] [credentials.location] set
16:52:23.413 | Logger:73 |             [OK] [credentials.password] set
16:52:23.414 | Logger:73 |             [OK] [credentials.alias] set
16:52:23.418 | Logger:73 |             [OK] [credentials.type] set
16:52:23.420 | Logger:73 |             [OK] [timestamp.signature.keystore.path] set
16:52:23.422 | Logger:73 |             [OK] [timestamp.signature.keystore.pwd] set
16:52:23.424 | Logger:73 |             [OK] [KEYSTORE_DIR] set and is a directory
16:52:23.425 | Logger:73 |
16:52:23.425 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:23.426 | Logger:73 | | Verifying location of keystores and certificates                                  |
16:52:23.426 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:23.427 | Logger:73 |
16:52:23.733 | Logger:73 |             [OK] /NIHII-HOSPITAL=71089815, TESTINGTSACLIENT 20190207-163953.intrc-p12
16:52:23.746 | Logger:73 |             [OK] /tsacertificate.jks
16:52:23.750 | Logger:73 |             [OK] ${KEYSTORE_DIR}/tslostore.jks
16:52:23.775 | Logger:73 |             [OK] ${KEYSTORE_DIR}/truststore.jks
16:52:23.801 | Logger:73 |             [OK] /caCertificateKeystore.jks
16:52:23.874 | Logger:73 |
16:52:23.877 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:23.878 | Logger:73 | | Verifying V1 database [buffer]                                                    |
16:52:23.879 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:23.883 | Logger:73 |
16:52:24.847 | Logger:73 |             [OK] No problem found with table [journal]
16:52:24.850 | Logger:73 |             [OK] No problem found with table [bufbehealthtsbag]
16:52:24.853 | Logger:73 |             [OK] No problem found with table [bufjournal]
16:52:24.856 | Logger:73 |             [OK] No problem found with table [bufarchive]
```

```
16:52:24.857 | Logger:73 |
16:52:24.858 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:24.859 | Logger:73 | | Verifying V1 database [archive]                                                   |
16:52:24.860 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:24.861 | Logger:73 |
16:52:24.951 | Logger:73 |             [OK] No problem found with table [bags]
16:52:24.954 | Logger:73 |             [OK] No problem found with table [journal]
16:52:24.954 | Logger:73 |
16:52:24.960 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:24.963 | Logger:73 | | Verifying V2 database [buffer]                                                    |
16:52:24.963 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:24.965 | Logger:73 |
16:52:24.972 | Logger:73 |             [OK] No problem found with table [journal_staging]
16:52:24.978 | Logger:73 |
16:52:24.995 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:25.004 | Logger:73 | | Verifying V2 database [archive]                                                   |
16:52:25.017 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:25.018 | Logger:73 |
16:52:25.021 | Logger:73 |             [OK] No problem found with table [bag_archive]
16:52:25.027 | Logger:73 |             [OK] No problem found with table [journal_archive]
16:52:25.030 | Logger:73 |
16:52:25.030 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:25.032 | Logger:73 | | Verifying SSL connection to [services-acpt.ehealth.fgov.be] on port [443]         |
16:52:25.034 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:25.035 | Logger:73 |
16:52:25.542 | Logger:73 |             [OK] Successfully connected
16:52:25.543 | Logger:73 |
16:52:25.544 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:25.545 | Logger:73 | | Verifying SSL connection to [services.ehealth.fgov.be] on port [443]              |
16:52:25.545 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:25.546 | Logger:73 |
16:52:25.581 | Logger:73 |             [OK] Successfully connected
```

```
16:52:25.582 | Logger:73 |
16:52:25.582 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:25.584 | Logger:73 | | Verifying eHealth TimeStamp Consultation webservice                               |
16:52:25.584 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:25.585 | Logger:73 |
16:52:27.792 | Logger:73 |             [OK] Successfully contacted the webservice
16:52:27.794 | Logger:73 |                      -> Error type: none
16:52:27.797 | Logger:73 |                      -> Error number: none
16:52:27.797 | Logger:73 |
16:52:27.798 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:27.798 | Logger:73 | | Verifying eHealth TimeStamp Authority webservice                                  |
16:52:27.799 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:27.801 | Logger:73 |
16:52:29.464 | Logger:73 |             [OK] Successfully contacted the webservice
16:52:29.465 | Logger:73 |                      -> Result major: urn:oasis:names:tc:dss:1.0:resultmajor:Success
16:52:29.466 | Logger:73 |                      -> Result minor: urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:OnAllDocuments
16:52:29.466 | Logger:73 |                      -> Result message: none
16:52:29.467 | Logger:73 |
16:52:29.468 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:29.479 | Logger:73 | | Verifying the mailing system                                                      |
16:52:29.480 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:29.480 | Logger:73 |
16:52:29.740 | Logger:73 |             [OK] An email has been sent. Please, verify your mailbox.
16:52:29.742 | Logger:73 |
16:52:29.744 | Logger:73 | |-------------------------------------------------------------------------------------|
16:52:29.745 | Logger:73 | |                                                                                   |
16:52:29.745 | Logger:73 | |                                    SUCCESS                                        |
16:52:29.745 | Logger:73 | |                                                                                   |
16:52:29.746 | Logger:73 | |-------------------------------------------------------------------------------------|
```

If there is any error, you will see a **FAILED** at the end of the report and a **[FAIL]** in front of each problematic point as well as an explanation why it failed.

Note that if you installed the database V2 from scratch, you will see error messages telling you that the V1 tables are not found. These error messages can be ignored, as you will not use the V1 tables.

```
[main] INFO  b.f.e.t.u.Logger - | Verifying V1 database [buffer]                                                    |
[main] INFO  b.f.e.t.u.Logger - |---------------------------------------------------------------------------------------|
[main] INFO  b.f.e.t.u.Logger -
[main] INFO  c.z.h.HikariDataSource - HikariPool-1 - Starting...
[main] INFO  c.z.h.p.PoolBase - HikariPool-1 - Driver does not support get/set network timeout for connections. (null)
[main] INFO  c.z.h.HikariDataSource - HikariPool-1 - Start completed.
[main] INFO  b.f.e.t.u.Logger -            [FAIL] Problem found with table [journal]:
[main] INFO  b.f.e.t.u.Logger -                   -> StatementCallback; bad SQL grammar [SELECT count(*) FROM journal]; nest
is java.sql.SQLException: Invalid object name 'journal'.
[main] INFO  b.f.e.t.u.Logger -
[main] INFO  b.f.e.t.u.Logger -            [FAIL] Problem found with table [bufbehealthtsbag]:
[main] INFO  b.f.e.t.u.Logger -                   -> StatementCallback; bad SQL grammar [SELECT count(*) FROM bufbehealthtsb
xception is java.sql.SQLException: Invalid object name 'bufbehealthtsbag'.
[main] INFO  b.f.e.t.u.Logger -
[main] INFO  b.f.e.t.u.Logger -            [FAIL] Problem found with table [bufjournal]:
[main] INFO  b.f.e.t.u.Logger -                   -> StatementCallback; bad SQL grammar [SELECT count(*) FROM bufjournal]; n
on is java.sql.SQLException: Invalid object name 'bufjournal'.
[main] INFO  b.f.e.t.u.Logger -
[main] INFO  b.f.e.t.u.Logger -            [FAIL] Problem found with table [bufarchive]:
[main] INFO  b.f.e.t.u.Logger -                   -> StatementCallback; bad SQL grammar [SELECT count(*) FROM bufarchive]; n
on is java.sql.SQLException: Invalid object name 'bufarchive'.
[main] INFO  b.f.e.t.u.Logger -
[main] INFO  b.f.e.t.u.Logger - |---------------------------------------------------------------------------------------|
[main] INFO  b.f.e.t.u.Logger - | Verifying V1 database [archive]                                                     |
[main] INFO  b.f.e.t.u.Logger - |---------------------------------------------------------------------------------------|
[main] INFO  b.f.e.t.u.Logger -
[main] INFO  c.z.h.HikariDataSource - HikariPool-2 - Starting...
[main] INFO  c.z.h.p.PoolBase - HikariPool-2 - Driver does not support get/set network timeout for connections. (null)
[main] INFO  c.z.h.HikariDataSource - HikariPool-2 - Start completed.
[main] INFO  b.f.e.t.u.Logger -            [FAIL] Problem found with table [bags]:
[main] INFO  b.f.e.t.u.Logger -                   -> StatementCallback; bad SQL grammar [SELECT count(*) FROM bags]; nested
java.sql.SQLException: Invalid object name 'bags'.
[main] INFO  b.f.e.t.u.Logger -
[main] INFO  b.f.e.t.u.Logger -            [FAIL] Problem found with table [journal]:
[main] INFO  b.f.e.t.u.Logger -                   -> StatementCallback; bad SQL grammar [SELECT count(*) FROM journal]; nest
is java.sql.SQLException: Invalid object name 'journal'.
[main] INFO  b.f.e.t.u.Logger -
```

In the event you forgot to set the configuration files properly, the configuration's validation process will be aborted and you will get a report like following:

```
16:49:31.377 | Logger:73 | |---------------------------------------------------------------------------------------|
16:49:31.378 | Logger:73 | |                                                                                       |
16:49:31.379 | Logger:73 | |                      eHealth TSA Client Configuration Validator                       |
16:49:31.379 | Logger:73 | |                                                                                       |
16:49:31.380 | Logger:73 | |---------------------------------------------------------------------------------------|
16:49:31.380 | Logger:73 |
16:49:31.381 | Logger:73 | |---------------------------------------------------------------------------------------|
16:49:31.381 | Logger:73 | | Verifying presence of configuration files                                             |
16:49:31.382 | Logger:73 | |---------------------------------------------------------------------------------------|
16:49:31.382 | Logger:73 |
16:49:31.383 | Logger:73 |             [FAIL] [application.properties] is missing
16:49:31.383 | Logger:73 |             [FAIL] [be.ehealth.technicalconnector.properties] is missing
16:49:31.384 | Logger:69 |
16:49:31.384 | Logger:73 |             There are missing / problematic configuration files
16:49:31.385 | Logger:73 |             Please, make sure all configuration files are there and correct before proceeding
16:49:31.385 | Logger:73 |
16:49:31.385 | Logger:73 | |---------------------------------------------------------------------------------------|
16:49:31.409 | Logger:73 | |                                                                                       |
16:49:31.409 | Logger:73 | |                                      ABORTED                                          |
16:49:31.414 | Logger:73 | |                                                                                       |
16:49:31.415 | Logger:73 | |---------------------------------------------------------------------------------------|
```

In such cases, simply make sure you have set your configuration files properly.

Run the configuration validator as many time as you need (i.e. after your changes) until you get a **SUCCESS** at the end of the report.

You should have a **SUCCESS** before using the actual client.

7.2 Use

Once the TS Client is installed and configured and if you have a **SUCCESS** on the report of your configuration's validation, you are ready to use the different proposed methods.

7.2.1 SIGN

This method, which is the main one of the TS Client allows you to:

- Retrieve the journal entries inserted in table JOURNAL_STAGING of the BUFFER database
- Collect those journal entries into a TSBag
- Sign the TSBag by calling the eHealth TimeStamp Authority service
- Store the signed TSBag and the original journal entries into ARCHIVE database

Command:

```
java -Dloader.path=[path_to_you_acceptance_configuration_folder]/ -jar timestamping-client-boot-1.0.0.jar --sign
```

7.2.2 VERIFY

This method allows you to check the coherence of the timestamps in the hospital ARCHIVE database:

- Are all journal entries in a TSBag with the correct hash code ?
- Are all journal entries mentioned in a TSBag present ?
- Are the registered timestamps OK: digital signature OK, hash code corresponding to the hash code of the TSBag ?
- Does the list of timestamps correspond to what has been registered in the eHealth platform archive?

Command:

```
java -Dloader.path=[path_to_you_acceptance_configuration_folder]/ -jar timestamping-client-boot-1.0.0.jar --verify --start=[start_date] --end=[end_date]
```

- if options --start and --end are not specified, default it to today, otherwise, options --start and --end support the ISO-8601 format, for example: --start=2011-12-03T10:15:30. Note that the time part is optional.

7.2.3 DISPLAYBAG

This method allows displaying a bag details.

Command:

```
java -Dloader.path=[path_to_you_acceptance_configuration_folder]/ -jar timestamping-client-boot-1.0.0.jar --displayBag=[idOfTheBagToDisplay]
```

- idOfTheBagToDisplay can be retrieved in ARCHIVE DB: SELECT id FROM bag_archive
- Repeat it if you want several at once, by example: --displayBag=1 --displayBag=41

7.2.4 DISPLAYJOURNAL

This method allows displaying a journal details.

Command:

```
java -Dloader.path=[path_to_you_acceptance_configuration_folder]/ -jar timestamping-client-boot-1.0.0.jar --displayJournal=[idOfTheJournalToDisplay]
```

- idOfTheJournalToDisplay can be retrieved in ARCHIVE DB: SELECT id FROM journal_archive
- Repeat it if you want several at once, by example: --displayJournal=1 --displayJournal=41

It is possible to launch the commands sign and verify in the same run: java -Dloader.path=[path_to_you_acceptance_configuration_folder]/ -jar timestamping-client-boot-1.0.0-beta-1.jar --sign --verify

7.2.5 LIMITMAIL

This method allows you to disable all success mails. Error mails will still be sent.

Command:

```
java -Dloader.path=[path_to_you_acceptance_configuration_folder]/ -jar timestamping-client-boot-1.0.0.jar --sign --verify --limitMail
```

7.2.6 Recommandations

1. The signing should occur at least every 5 minutes.
2. The verification should occur at least once per day.
3. You should configure mails and alerts to ensure that the executions are effectively "running" and are successful.
4. You should launch the application only one instance at a time. For this, you have 3 possibilities:

   I. By default, the application will use a file lock mechanism in order to make sure you have only one instance running. Trying to launch other instances of the application will raise an error and cause the application to stop. Be aware you should make sure the user who launches the application has enough rights so the application can create, read and write files without problems. The lock file (**tsa.lock**) will be located in the temporary directory of your operating system (system propery **java.io.tmpdir** from Java) and, therefore, will be automatically deleted in case of an unexpected crash/reboot. If, for some reason, the lock file is not deleted, the application will detect it and override it if there are no other instances running.

   II. Launch the application with the argument **--lock=XXXXX** where **XXXXX** is an available network port number onto which the application can lock itself. If the port number is not specified, empty or invalid, the application will lock itself onto a default port number, which is **65534**. Trying to launch other instances of the application will raise an error. The port will be released once the process is killed (normal ending, unexpected crash, reboot, ...).

   ```
   java -Dloader.path=[path_to_you_acceptance_configuration_folder]/ -jar timestamping-client-boot-1.0.0.jar --sign --verify --lock=12345
   ```

   III. Launch the application with the argument **--noLock**. This will start the process without any lock mechanism what so ever. If you ever use this option, you will do it at your own risks. Also, when using this option, you should make sure the application is started only once.

   ```
   java -Dloader.path=[path_to_you_acceptance_configuration_folder]/ -jar timestamping-client-boot-1.0.0.jar --sign --verify --noLock
   ```

7.3 Maintenance

7.3.1 How to renew the timestamp server certificate

Download and copy in the config folder the certificates of the timestamping server from eHealth portal
See: (***https://www.ehealth.fgov.be/ehealthplatform/fr/service-datation-electronique-timestamping#docType3*** (French)

or:
https://www.ehealth.fgov.be/ehealthplatform/nl/service-elektronische-datering-timestamping#docType3 (Dutch)**.**

They allow the client to verify that the token has been produced by the eHealth timestamping server. Note that they are all stored into a single key store, it is easier than having multiple files in a folder

7.3.2 How to request/renew the bus certificate

In order to access the secured eHealth platform environment you have to obtain an eHealth platform certificate, used to identify the initiator of the request. In case you do not have one please consult the chapter about the eHealth Certificates on the portal of the eHealth platform

- ***https://www.ehealth.fgov.be/ehealthplatform/nl/ehealth-certificaten***
- ***https://www.ehealth.fgov.be/ehealthplatform/fr/certificats-ehealth***

For technical issues regarding eHealth platform certificates

- Acceptance***: acceptance-certificates@ehealth.fgov.be***
- Production***: support@ehealth.fgov.be***

8. Incident reporting

As explained previously, the TS Client generates incident report in case of issue encountered with the SIGN method. Indeed in case of issue with the bags signing, the hospital archive could contain a time gap in the sequence of TSBags. The incident reports allow to facilitate answering questions about these time gaps.

8.1 Concretely, how does it work?

Once a TSBag encounters a problem and cannot be signed by the eHealth platform and stored in the hospital archive, it is kept as it is in the BUFFER database while a new journal entry containing the incident report is inserted in the same database. In the next run of the TS Client, this journal entry will be packed in a new TSBag with the other journal entries and both TSBags will be timestamped. Therefore, an incident report in a TSBag concerns always the previous run of the TS Client.

If an error persists, the logical is the same: at each run, an incident report will be inserted in the BUFFER database and packed in a new TSBag as long as the issue remains unsolved. At this moment, all the TSBags kept in the BUFFER will sent to eHealth in order to be timestamped and then stored in the hospital archive.

In the table JOURNAL_ARCHIVE the incident reports have a different type value ("INCIDENT") in order to distinguish them.

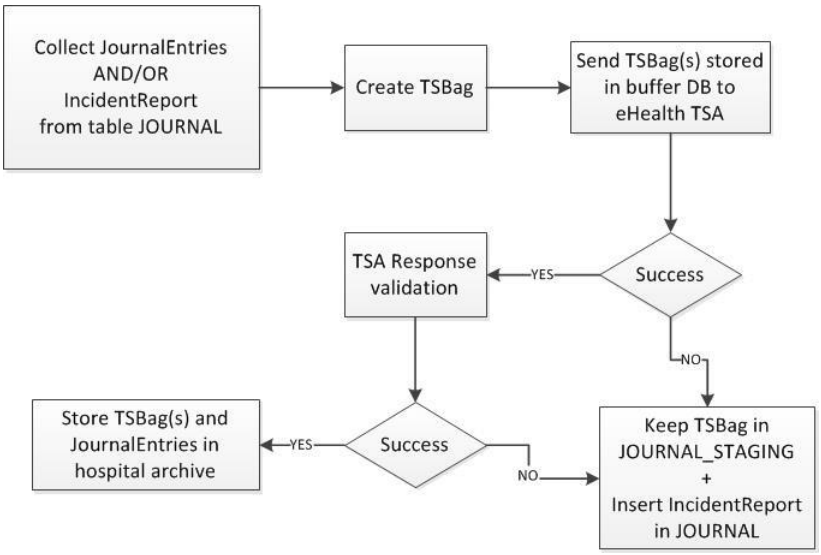

Fig 5

Example:

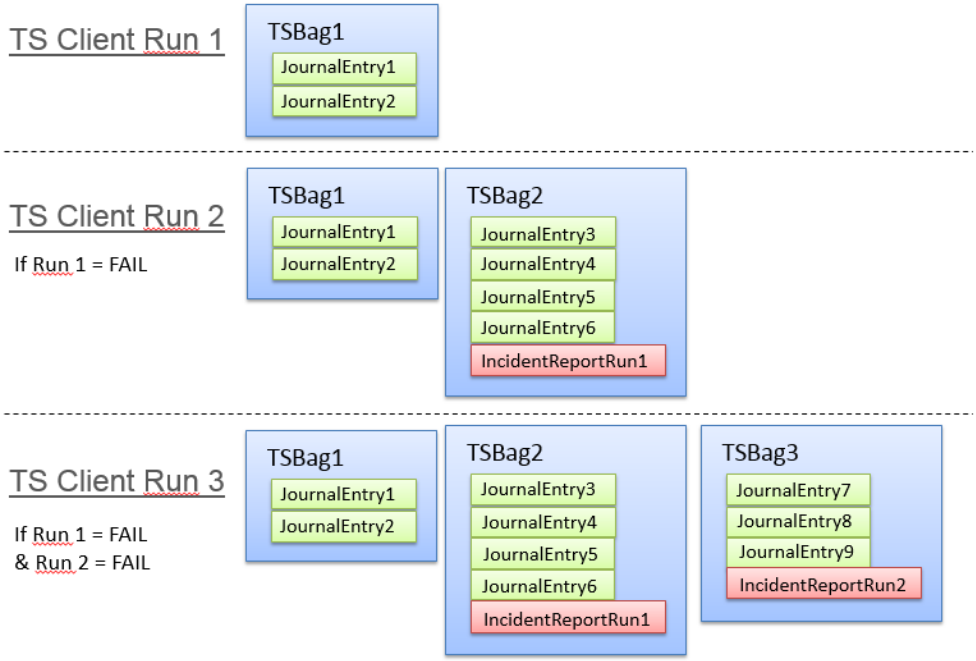

Fig 6

8.2 Format

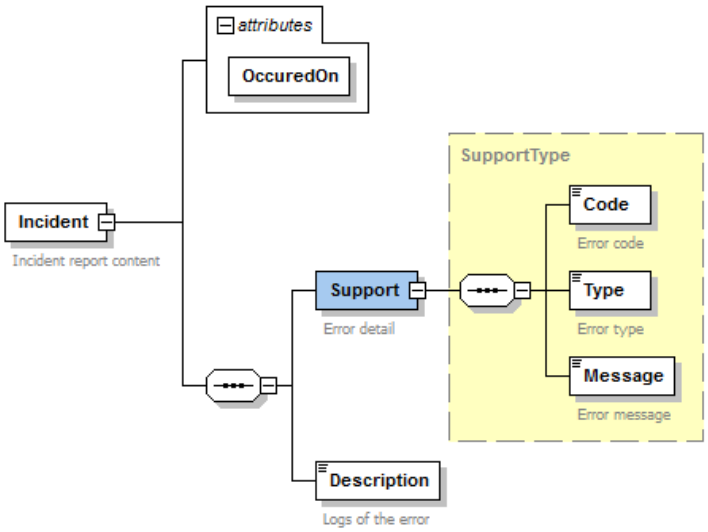

Fig 7

Element	Attribute	Description
	OccuredOn	Date time when the incident report was generated
Support/Code		Code of the error which allows to identify it
Support/Type		Type of the error which allows to identify the code layer impacted
Support/Message		Message explaining the reason of the error
Description		Complete error as it appears in the logs of the TS Client

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Incident occuredOn="2019-06-05 09:48:40">
  <Support>
    <Code>SER_404</Code>
    <Type>SERVICE</Type>
    <Message>404: The endpoint configured for the Timestamp Authority service is not
recognized</Message>
  </Support>
  <Description>be.fgov.ehealth.tsa.error.xcp.TimeStampException: 404: The endpoint configured for the
Timestamp Authority service is not recognized
        at
be.fgov.ehealth.tsa.service.support.AbstractTimestampService.handleConnectorException(AbstractTimestamp
Service.java:14)
        at
be.fgov.ehealth.tsa.service.authority.TimestampAuthorityServiceImpl.sendRequest(TimestampAuthorityServi
ceImpl.java:67)
        at be.fgov.ehealth.tsa.business.service.impl.SignerImpl.getResponse(SignerImpl.java:90)
        at be.fgov.ehealth.tsa.business.service.impl.SignerImpl.sign(SignerImpl.java:69)
        at be.fgov.ehealth.tsa.business.service.impl.SignerImpl.sign(SignerImpl.java:55)
        at
be.fgov.ehealth.tsa.orchestration.signer.SignerOrchestratorImpl.sign(SignerOrchestratorImpl.java:54)
        at be.fgov
        …
        …

</Description>
</Incident>
```