Towards a standardised preservation service for qualified electronic signatures and qualified electronic seals

Florian Otto[1], Tobias Wich[1], Tina Hühnlein[1], Mike Prechtl[1], Detlef Hühnlein[1]

Abstract: To preserve the legal validity and conclusiveness of qualified electronic signatures and qualified electronic seals over long periods of time it is necessary to apply appropriate preservation techniques. The present contribution provides an overview of the corresponding standards for long-term preservation of digital signatures, which are currently developed within ETSI TC ESI and outlines the design of a corresponding reference implementation, which is currently developed within the EU-funded FutureTrust project.

Keywords: Long-term preservation, qualified electronic signature, qualified electronic seal, time-stamp, evidence record, validation, eIDAS

1 Introduction

It is well-known, that electronic signatures, seals, time-stamps and similar signed data, need to be preserved over the long-term using adequate measures, which maintain the legal validity and conclusiveness of the signatures and signed data. Recital (61) [2] of the eIDAS-Regulation [EU14] explicitly stated the need for long-term preservation and Art. 34 of [EU14] introduced a specific type of trust service for this purpose: The qualified preservation service for qualified electronic signatures[3]. To standardise the policy requirements and pertinent technical aspects of preservation services, the Technical Committee (TC) for "Electronic Signatures and Infrastructures" (ESI)[4] within the European Telecommunications Standards Institute (ETSI) first conducted a scoping study [ET17] to establish a good foundation for the subsequent standard development process in which policy requirements [ET18a] and technical protocols for preservation services [ET18b] are currently developed. In close coordination with the still ongoing standardisation work within ETSI ESI, the EU-funded research project FutureTrust[5] is developing a reference implementation of a scalable preservation service according to [ET18a] and [ET18b], which may considerably ease the deployment of preservation

[1] ecsec GmbH, Sudetenstraße 16, 96247 Michelau, {florian.otto, tobias.wich, tina.huehnlein, mike.prechtl, detlef.huehnlein}@ecsec.de

[2] Recital (61) of [EU14] reads as follows: *"This Regulation should ensure the long-term preservation of information, in order to ensure the legal validity of electronic signatures and electronic seals over extended periods of time and guarantee that they can be validated irrespective of future technological changes."*

[3] As stated in Art. 40 of [EU14], Art. 34 applies mutatis mutandis to qualified electronic seals.

[4] See https://portal.etsi.org/TBSiteMap/esi/ESIActivities.aspx .

[5] See https://futuretrust.eu, G.A. No. 700542.

services across Europe and foster interoperability among different implementations.

For background, related work as well as an overview of pertinent standards we refer to [ET17]. The rest of the paper is organised as follows: Section 2 summarises the main aspects of standardised preservation services according to [ET18a] and [ET18b]. Section 3 provides an overview of the corresponding reference implementation of a preservation service according to [ET18b], which is currently developed within the FutureTrust project. Section 4 summarises the main aspects of the present paper and provides an outlook on further developments.

2 An overview of the ETSI preservation service standards

2.1 System Architecture

As depicted in Fig. 1 a preservation service according to [ET18b] provides a preservation interface, which can be used by a client to submit preservation objects, which are intended to be protected and preserved by the preservation service.

The preservation service may use an external time-stamping authority (TSA), which issues time-stamps (see [ET16d]), or a signature or seal creation service (SigS) which issues suitable digital signatures. It may optionally use a validation service (ValS) to collect revocation information[6] and validate digital signatures, if required, or directly gather certificate status information issued by a certificate status authority.

There are three main variants for a preservation service depending on the question whether it uses (a) a long-term storage, (b) a temporary storage or (c) no storage[7]. When it uses a storage, the preservation service may use an internal storage or an external storage under its control for preservation.

Furthermore, the preservation service may call back the client via the optional notification interface in order to inform it about relevant events[8].

[6] The collection of revocation information (e.g. OCSP-responses, CRLs) and possibly missing certificates up to applicable trust anchors is necessary, if the preservation goal is not limited to providing a proof of existence of the submitted data, but to extend the validity status of digital signatures over long periods of time.

[7] (a) `WithStorage` (WST), (b) `WithTemporaryStorage` (WTS), (c) `WithoutStorage` (WOS).

[8] An important type of event is that a previously applied cryptographic algorithm is expected to become weak and hence the client and/or the preservation service need to perform additional measures.

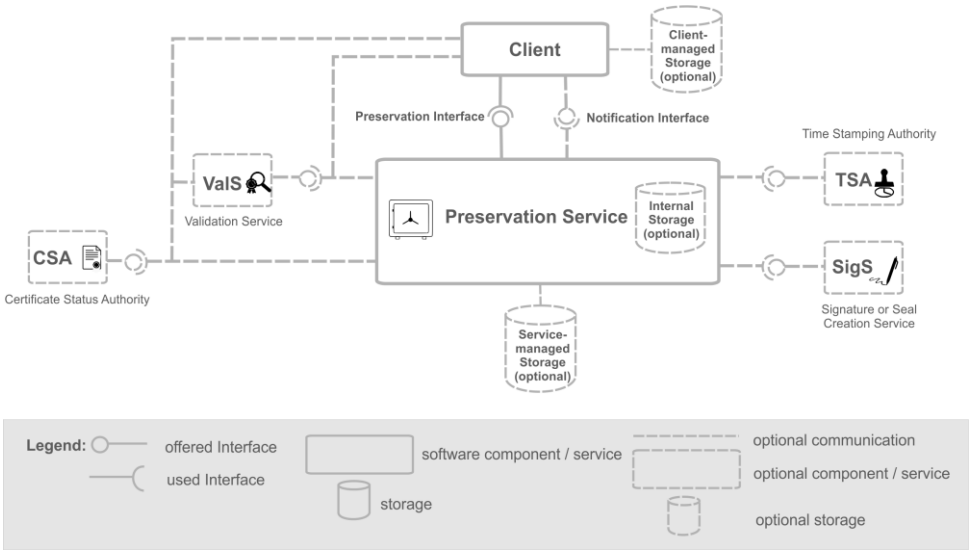

Fig. 1: System architecture with preservation service and related services[9]

2.2 Preservation schemes, profiles and policies

The ETSI preservation standards to [ET18a] and [ET18b] allow to implement different strategies for preservation, which are outlined in an abstract preservation scheme. A preservation service may implement one or more preservation profiles, which are derived from the abstract preservation scheme.

As outlined in Fig. 2, a preservation profile in particular specifies the applied storage model, the preservation goal (i.e. whether the status of digital signatures is to be preserved or not[10]), the supported operations (see Section 2.3), the supported input and output formats, the applicable policies, the expected evidence duration and, in case of a preservation service with temporary storage, the duration in which the client may pick up the asynchronously produced preservation evidence[11].

[9] See Figure 1 in [ET18b].

[10] For this purpose [ET18a] and [ET18b] distinguish between the two preservation goals: (1) "preservation of digital signatures" (PDS), which requires to collect validation material before a proof of existence (PoE) mechanism (e.g. a cryptographic time-stamp) is applied, and (2) "preservation of general data" (PGD), which immediately applies the PoE mechanism.

[11] This period is called "preservation retention period" in [ET18b].

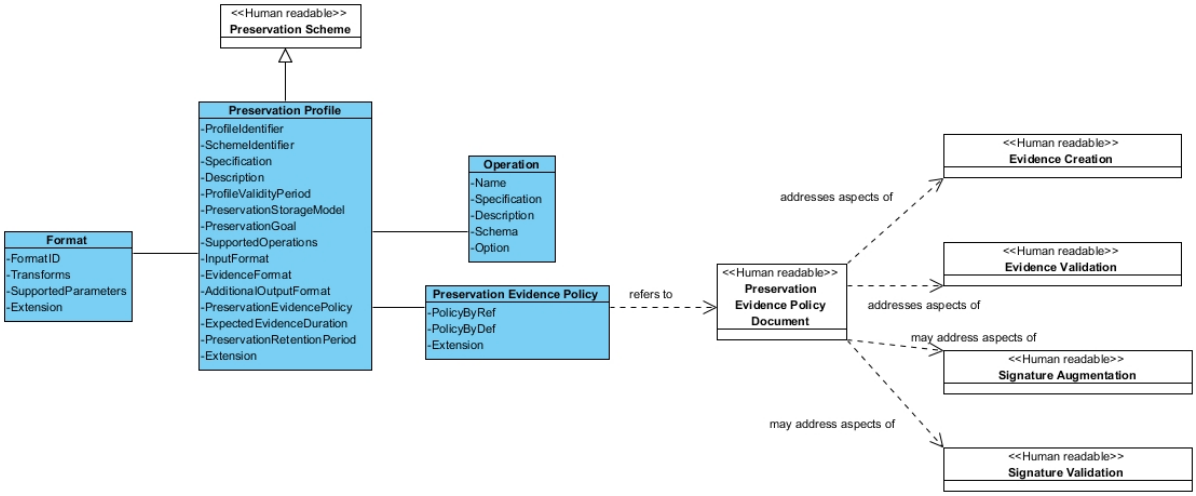

Fig. 2: Relationship between Preservation Scheme, Profile and Policy[12]

Annex F of [ET18b] defines four preservation schemes as outlined in Tab. 1.

Annex	Preservation Scheme[13]	Preservation Goal	Storage Model	Preservation Evidence
F.1	pds+pgd+wst+ers	PDS & PGD	WST	ERS[14]
F.2	pgd+wts+ers	PGD	WTS	ERS
F.3	pds+wst+aug	PDS	WST	ATS[15]
F.4	pds+wos+aug	PDS	WOS	ATS

Tab. 1: Preservation schemes defined in [ET18b](Annex F)

2.3 Preservation interface

[ET18b] first specifies the semantics of the different calls of the protocol for the preservation interface in a generic fashion and then specifies the concrete syntax of the conveyed data elements based on XML and JSON together with its binding to SOAP and REST respectively.

[12] See Figure 2 in [ET18b].
[13] The URIs for the preservation schemes defined in [ET18b] starts with http://uri.etsi.org/19512/scheme/ and are completed by the fragment as shown in the present column of Tab. 1.
[14] Evidence Records according to [GBP07] or [JSG11].
[15] Archive Time Stamps according to [ET16a] or [ET16b] or Document Time Stamps according to [ET16c].

The preservation interface specified in [ET18b] comprises the following operations:

Operation	Storage Model[16]			Description
	WST	WTS	WOS	
`RetrieveInfo`	**M**	**M**	**M**	Provides information about the preservation profiles (see Fig. 2) supported by a preservation service
`PreservePO`	**M**	**M**	**M**	Allows to submit preservation objects (PO) for preservation
`RetrievePO`	**M**	**M**	/	Allows to retrieve preservation objects (data objects and evidence)
`DeletePO`	**M**	/	/	Allows to delete stored preservation objects
`UpdatePOC`	O / C	/	/	Allows to update a preservation object container, which supports versioning
`RetrieveTrace`	O	O	O	Allows to retrieve a trace of operations related to a specific set of preservation objects
`ValidateEvidence`	O	O	O	Allows to validate the evidence created by a preservation service
`Search`	**M**	O	/	Allows to search for specific preservation objects within a preservation service with permanent storage.

Tab. 2: Overview of calls at the preservation interface according to [ET18b]

3 Towards a reference implementation of ETSI TS 119 512

Given the specification of the Preservation-API developed within ETSI ESI [ET18b], it is fairly straightforward to derive a design for a corresponding preservation service[17]. As can be seen in Fig. 1, the preservation service mainly combines existing services, like Validation Services or Time Stamping Services, in a way to reach the goals of long-term preservation. The main complexity lies within the aim to support arbitrary preservation profiles and in the long life-cycle of the preservation service.

The latter makes it particularly important to provide upgrade and migration paths for new

[16] WST=With Permanent Storage, WTS=With Temporary Storage, WOS=Without Storage, M=Mandatory, O=Optional, C=Conditional.

[17] See [FT17] for a corresponding design document, which reflects the state of the standardisation efforts in spring 2017.

and changed functionality.

This section describes methods, which have been applied to the service design in order to simplify

1. replacing components, which might need different properties for the anticipated usage scenario and
2. providing the flexibility to extend / modify the service for future changes of the standard.

Especially the second point stands out here, as it cannot be expected that given preservation periods of 100 years and more might pass without changes in the respective standards or general advances in technology.

3.1 Replaceability of Components

Considering the long life-cycle of the planned preservation service, it is necessary to be able to add and exchange single components of the service easily to address extensions or changes in technology, specification or the environment. This requirement has been formalized by a large number of design patterns, which provide the necessary abstractions to reach that goal. One principle that has to be considered is the separation of data and implementation. Data types carrying implementation details which are passed between components lead to strong coupling of these components. Strong coupling is the main reason to hinder reusability of software components, which is related to the case in which an entire component needs to be replaced.

Keeping that principle in mind all exchange data objects contain only data and no functionality like one would define data types in a functional programming language as opposed by an object-oriented approach, which would encapsulate the data as state in objects. Having well designed data definition decreases the effort needed to transform the data received via one of the public web interfaces which are based on JSON/REST or XML/SOAP.

Once the data is de-serialised and transformed to the internal data formats, the requested process uses various components to perform its actions. Each component can thereby perform further transformations on the data in order to reach a form suitable to fulfil an action, such as persist it in some data store, calculating hash values, building a hash-tree or adding a time-stamp to a particular hash value. Once an action is complete, it returns resulting data elements which are needed by further actions.

The design so far has improved the replaceability keeping the coupling of components low by separating state and functionality. A common pattern representing database transactions in Object Relational Mappers however introduces global state by hiding the transaction handling in object state during function invocation, meaning when entering and leaving a function. In order to reach the replaceability goals, database transactions

must be either completely local to a component or must be made explicit. Depending on the isolation level (ACID) of a transaction it is necessary for certain components to exchange a transaction state object to see changes made in a previously opened transaction. This problem is countered by a database design allowing partially complete results to be present in the database. The idea is to additionally save markers indicating which state the data set is in, so further transactions can further progress or complete the operation. This makes it possible to have completely local transactions per component. A failed or cancelled process can then easily identify unfinished data sets and perform suitable rollbacks.

The currently developed reference implementation uses the Contexts and Dependency Injection Framework of JavaEE (CDI)[18] to address the described requirement. CDI allows exchanging software components with low effort since the used implementation of interfaces can be chosen at deployment time without the need of altering the remaining software.

3.2 Profile Factory

As mentioned before, the main complexity of the preservation service specified in [ET18b] as considered here, lies within the composition of modules to allow the flexible usage of arbitrary profiles, that define how preservation workflows are performed, which is implemented by a "Profile Factory". Assuming the preservation service contains components and functionality to perform the tasks at hand, these parts can be seen as a flexible "construction kit". An implementation of a profile uses all the building blocks it needs and composes them into a profile-specific implementation of a function. Depending on the actual property of a part of the profile, different composition strategies are used. The profile interface resembles the profile-specific methods of the external interface.

Basic profiles reflect the main preservation storage models (WST, WTS, WOS) and consist of far reaching functionality spanning several building blocks, such as whether a preservation object is persisted or not. Functionality which further defines the steps in the general process can be provided by different components. The profile factory chooses the relevant parts according to the requested profile when the profile implementation is constructed. This can be seen as a dynamic variant of the Pipes and Filters pattern[19]. Additionally the implementation can further be adjusted by parameterization of the composable parts. As shown in Fig. 3, the profile factory uses basic profiles and further defines the main steps of those, by choosing appropriate implementations provided as components. Moreover it can adjust the process by setting parameters based on the requested profile, for example the specific hash algorithm to use.

[18] https://docs.oracle.com/javaee/6/tutorial/doc/giwhl.html
[19] https://docs.microsoft.com/en-us/azure/architecture/patterns/pipes-and-filters

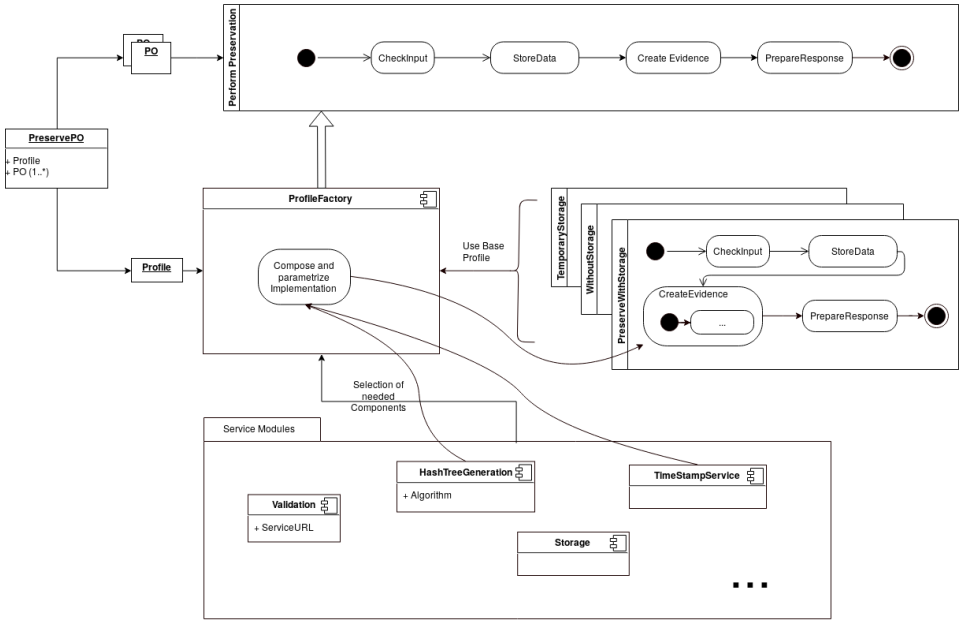

Fig. 3: Action of the Profile Factory using to the example of the `PreservePO` call

The availability of components and their provided functionality determines which specific preservation profiles can be used with the service. This gives enough room to create different, standard compliant profiles solely based on the configuration of the service. By adding components and implementations, new profiles can easily be integrated.

The simplified example in Fig. 4 illustrates how a profile is configured. The array notation in pre-PO-construction indicates that there can be a sequence of composable parts.

```
{
  id = "temporary-tsa-sha256"
  base = "org.ft.pres.profile.TemporaryStorageBase"
  config {
    hash = "SHA-256"
    tsa-url = "https://tsp.com/tsa"
  }
  pre-PO-construction = [
    {
      impl = "org.ft.pres.services.ValidationService"
      config {
        url = "https:/vals.futuretrust.eu/api/validate"
      }
    }
  ]
}
```

Fig. 4: Profile Configuration Example

3.3 Scalability Considerations

The reference implementation does not provide scalability out of the box. However given the previously described design decisions, the various levels of scalability can be achieved relatively easy. In order to identify the necessary changes, it makes sense to look at vertical scalability (scale up) and horizontal scalability (scale out) separately.

As demand grows one typically uses vertical scaling first, as it is easier to achieve. The first measure is to use better performing hardware, which is distinct from the design and is therefore not covered further. Once the performance limit of single host system is reached, the components of the preservation service can be separated from the core system and put into single standalone services (micro services). In the core application, the component is replaced by an implementation of the component interface relaying the data to the actual service. This is possible due to the loose coupling between the components and the well-defined exchange data types, which just have to be serialised to a format understandable by each service implementation.

Scaling out is considered to be the harder problem in case the system is going really large and the measures vary significantly on the anticipated usage numbers. When components work in parallel, they have to agree on common synchronization points to be sure to operate on a consistent state. The main problem is the RDBMS. Most modern systems provide replication and clustering support, but this has limits and the synchronization overhead grows larger than the performance gains of additional nodes at some point. In that case the only sensible option is to use client pinning to a specific node. The pinning can be performed via the preservation object identifier (POID) and depending on how the pinning is implemented reduces the synchronization to a single value (load balancer keeps track of ID) or removes it altogether (node address encoded into ID). Another distribution of functionality can be achieved by splitting up the hash-tree creation, which is usually performed in fixed intervals and thus has an upper runtime bound. Each subtree can then be merged into one larger tree, which is then finally time-stamped by an appropriate TSA.

4 Summary and Outlook

The present paper provides a current snapshot with respect to the ongoing standardisation efforts regarding long-term preservation of qualified electronic signatures and seals within ETSI ESI and discusses some design aspects of a corresponding reference implementation, which is currently developed within the EU-funded FutureTrust project. After a brief introduction in chapter 1 describing the underlying specifications, section 2 gives a general overview of the environment in which the preservation service lives and with which other services it interacts. Further the basic preservation strategies WST, WTS and WOS are introduced and it is described how those get configured through preservation profiles. Section 2 closes with a description of the preservation interface and an overview of available operations depending on the used preservation strategy. Section 3 examines considerations about the software architecture and how the requirements of extraordinary long life-cycle, changes in specifications and scalability can be addressed, which is mainly

achieved by strong decoupling of data and implementation and the use of interchangeable components. Additionally, the working principle of a profile factory is laid out which handles the high versatility of preservation profiles. The reference implementation described in this paper is planned to be used in forthcoming plug-tests to foster interoperability between different preservation solutions deployed across Europe. Stakeholders who would like to receive more information with respect to or use the forthcoming reference implementation are heartily invited to get in contact with the authors.

Bibliography

[ET16a] ETSI EN 319 122-1 (2016): Electronic Signatures and Infrastructures (ESI); V1.1.1. http://www.etsi.org/deliver/etsi_en/319100_319199/31912201/01.01.01_60/en_31912201v010101p.pdf.

[ET16b] ETSI EN 319 132-1 (2016): Electronic Signatures and Infrastructures (ESI); V1.1.1. https://www.etsi.org/deliver/etsi_en/319100_319199/31913201/01.01.01_60/en_31913201v010101p.pdf.

[ET16c] ETSI EN 319 142-1 (2016): Electronic Signatures and Infrastructures (ESI); V1.1.1. http://www.etsi.org/deliver/etsi_en/319100_319199/31914201/01.01.01_60/en_31914201v010101p.pdf.

[ET16d] ETSI EN 319 422 (2016): Electronic Signatures and Infrastructures (ESI); Time-stamping protocol and electronic time-stamp profiles. https://www.etsi.org/deliver/etsi_en/319400_319499/319422/01.01.01_60/en_319422v010101p.pdf.

[ET17] ETSI SR 019 510 (2017): Electronic Signatures and Infrastructures (ESI); Scoping study and framework for standardization of long-term data preservation services, including preservation of/with digital signatures. https://www.etsi.org/deliver/etsi_sr/019500_019599/019510/01.01.01_60/sr_019510v010101p.pdf.

[ET18a] ETSI TS 119 511 (2018): Electronic Signatures and Infrastructures (ESI); Policy and security requirements for trust service providers providing long-term preservation of digital signatures or unsigned data using signature techniques.

[ET18b] ETSI TS 119 512 (2018): Electronic Signatures and Infrastructures (ESI); Protocols for trust service providers providing long-term data preservation services.

[EU14] 2014/910/EU (2014): Regulation (EU) No 910/2014 of the European Parliament and of the council on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC. http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2014.257.01.0073.01.ENG.

[FT17] FutureTrust D3.4 (2017): Scalable Preservation Service.

[GBP07] RFC 4998 (2007): Gondrom, T.; Brandner, R.; Pordesch, U. https://tools.ietf.org/html/rfc4998.

[JSG11] RFC 6238 (2011): Jerman Blazic, A.; Saljic, A.; Gondrom, T. https://tools.ietf.org/html/rfc6283.