Basic function *Authorizing Actor*

Introduction

This document elaborates on the basic function of *Authorising an Actor* to represent another *Actor*.

The functionality is described in below in the following categories.

Description

In the Netherlands (as in other EU-member states), citizens have the right to allow others to represent them. An *Authorisation* is a way to organise that. An *Authorisation* is a statement indicating that someone else has the authority to perform certain actions for an *Actor*. It also includes a description of the process to make the statement, in which the representing *Actor* and the scope are clear.

Non-natural persons can only act through an *Authorisation*. It is therefore an essential function to support use cases for businesses and organisations.

In certain cases, an *Authorisation* contains a chain of other authorisations underlying the final *Authorisation*.

Note: This document is published for consultation purposes and can be updated to a 1.0 version after implementation by TIP Partners. Comments on this document are appreciated via an email at info@trustedinformationpartners.nl .

Applicability

A digital *Authorisation* offers many advantages over paper. It is more efficient, offers opportunities to maintain control over the *Authorisation*, provides good security options and offers lower administrative burden. These benefits depend on implementation.

The digital *Authorisation* process MUST meet the following conditions:

1. The identity of the authorising *Actor* (usually called principal, but to match the description of the authorisation format later in this document hereinafter called Issuer) and the *Actor* being authorised (usually called proxy-holder, but for the same reason hereinafter called Subject) is established at a sufficient level of reliability.
2. The *Authorisation* is valid: it contains the explicit acceptance of the Issuer and the Subject of the content and scope of the agreement, including the lifespan of the *Authorisation* (think of the description of the activities to be performed and agreements on validity periods).
3. The Issuer must be able to implement changes that have a direct effect on the parties and/or objects involved, such as a withdrawal when a revocation method is used.
4. The solution must be able to deal with the situation that multiple *Authorisations* are possible at the same time and that the Issuer may also perform the action himself.
5. The *Authorisation* must be human readable and accessible by both Issuer and Subject.

The following types of *Authorisations* are described in the Dutch civil code:

- General power of attorney: According to the Dutch Civil Code, a general power of attorney is defined as: the authority granted by the Issuer to another, the Subject, to perform legal acts on his behalf." (3:60, lid 1 Burgerlijk Wetboek).
- Special power of attorney: the Subject may only perform certain acts related to a purpose or domain on behalf of the Issuer (see 3:62, lid 2 Burgerlijk Wetboek)

There are several ways in which an *Authorisation* can be established: the Issuer can take the initiative to record a certain type of *Authorisation* or can be requested to agree to a proposed authorization. An *Authorisation* can have several manifestations. This document only applies to electronically signed *Authorisations* in digitally structured form (i.e. not to oral *Authorisations* or written down on paper). Electronically signed *Authorisations* can be recorded with or without the registration by a notary. In addition, the court (=Issuer) can in some cases appoint a Subject (e.g. curator), without the represented *Actor* having a say in it.

Contextual Clarification for Authorizations

In the realm of authorizations, the context in which an authorization is granted significantly influences its technical implementation. Decentralized authorizations typically have a short lifespan and are often intended for single-use processes. The lifespan may increase in the future as standardized methods for revocation become common practice. Decentral authorizations are created using qualified electronic signatures and are not recorded in a central registry, resembling the traditional paper-based agreements where both parties sign digitally. With the basic function “Publishing chain and service specifications” TIP offers the ability for domain governance to specify domain-specific rules for authorizations in a given context, introducing guardrails in otherwise free-format decentral authorizations.

On the other hand, centralized authorizations are employed when an authorization needs to have a longer lifespan. These authorizations are stored in specific registers designed for this purpose, such as the Chamber of Commerce register, eHerkenning, DigiD Machtigen, or registers maintained by service providers like the Dutch Tax Authority. Centralized authorizations offer advantages such as better control, oversight, and lower administrative burden, making them suitable for ongoing processes and business operations. A drawback in the current state of affairs is that these central solutions are often domain or region specific, and differ significantly in functionality and technology. Interoperability is currently lacking, even when using the same technology stack the implementations are not interoperable where they could be on the implementation level.

The choice between decentralized and centralized authorizations depends on the specific requirements of the authorization's lifespan and the nature of the processes involved. It is important to keep this distinction in mind when reviewing the technical specifications in this document, as it will help in understanding the implementation details and the rationale behind different authorization methods.

Horizontal, vertical and chain Authorisations

Horizontal, vertical and chain *Authorisations* are terms to describe the relationship between issuer and subject.

An *Authorisation* between parties of equal standing is called a **horizontal *Authorisation*** (e.g. citizen authorises citizen or organisation authorises organisation).

An *Authorisation* between parties with an unequal position is a **vertical *Authorisation*** (e.g. when (the authorised *Actors* of) an organisation authorise(s) an employee or contractor to act on behalf of the organisation.

A **chain *Authorisation*** consists of multiple *Authorisations* that combined allow a Subject to act on behalf of the Issuer. For example: An enterprise authorises an accountancy agency to do their tax returns (part 1 of the chain, a horizontal *Authorisation*); The accountancy agency authorises an employee to do tax returns on behalf of customers (part 2 of the chain, a vertical *Authorisation*). Combined, these two *Authorisations* form an effective chain *Authorisation*, where the employee can do tax returns for the enterprise (but not for the tax obligations of the accountancy agency itself).

Revoking an Authorisation

An *Authorisation* SHOULD be revocable, and ideally, that SHOULD take immediate effect. The revocation status of an *Authorisation* MUST be checked by the service provider at the authoritative source before settling/committing a transaction. For instance, if an employee is summarily discharged or changes roles, he should no longer be able to act on behalf of the company or exercise his old authorisations.

A central *Authorisation* register MUST have functionality to handle this; an *Authorisation* manager revokes the employee's *Authorisation* with immediate effect. In the case of a decentralised *Authorisation,* if the Issuer hands over an *Authorisation* to the Subject, the Issuer cannot remotely undo the hand over. Therefor arrangements for revoking a decentralised Authorisation MUST also need be made.

Several revocation methods for revoking decentralised authorisations are available:

- Revocation List. This revocation status of an *Authorisation* is looked up in a revocation list, which is stored centrally, only contains a list of *Authorisation* IDs (thus not the *Authorisations* themselves) and their status as active or revoked (including revocation date). The certainty of whether or not *Authorisations* have been revoked depends on the frequency with which the revocation list is refreshed.
- Short time to live: In this method the *Authorisation*'s revocation status is checked repeatedly every certain period of time (e.g. every 24 hours) by the Subject's *Wallet* used to store the *Authorisation*. In offline situations service providers trusting this *Wallet* can be confident that the *Authorisation* was still valid when last checked. The appropriate period of time is determined by the Issuer or *Chain specifications*. In offline situations this method might not allow for immediate revocation by the Issuer.
- Non revocable: an explicit statement that no revocation method is available for this authorisation. This MUST be used in cases where the consent to create an *Authorisation* may not be altered afterwards (e.g. an Issuer with dementia who may be deemed unable to revoke or re-authorise at a later date).

Since different domains, *Customer process chains* or *Information chains* can have different requirements for *Authorisations* and revocations and the service provider takes the risk of providing the actual service, the service provider (relying party) determines which revocation method is required for his service.

Lifecycle of an *Authorisation*

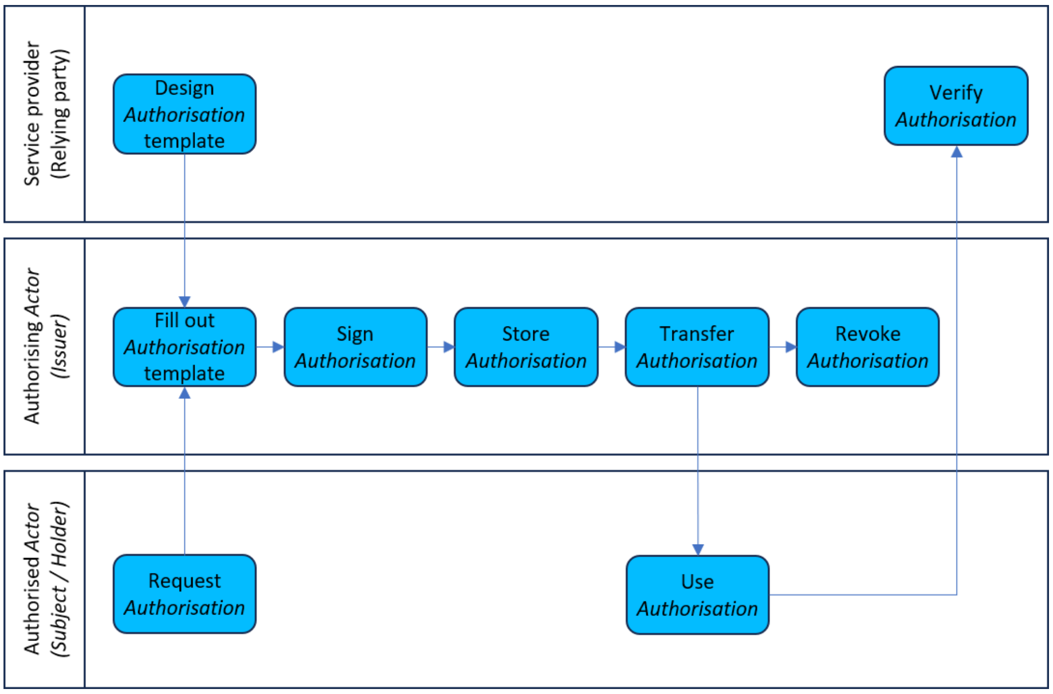
Service provider (Relying party)
Design *Authorisation* template
Verify *Authorisation*
Authorising *Actor* (*Issuer*)
Fill out *Authorisation* template
Sign *Authorisation*
Store *Authorisation*
Transfer *Authorisation*
Revoke *Authorisation*
Authorised *Actor* (*Subject / Holder*)
Request *Authorisation*
Use *Authorisation*

Figure 1: Generic lifecycle of an *Authorisation*

Standardisation and interoperability

Without standardisation of digital *Authorisations*, an unmanageable multitude of central authorisation registers and decentralised authorisation types will arise. This will lead to *Authorisations* that cannot be used, exchanged or reused in chain *Authorisations* due to definitional differences or format incompatibilities. Therefor *Authorisation* standardisation is required to prevent confusion and administrative burdens for the Issuer, the Subject and the service providers. This will have an impact on the interfaces of existing *Authorisation* registers. Note: this applies not only to the Dutch market but across all EU markets.

Additionally, consideration must be given by all central registers to the need to refer to a single *Authorisation* by using a URI. This is necessary to enable the use of chain *Authorisations* consisting of one *Authorisation* in a central register, and another in a decentralised *Authorisation* document.

Standards

The standards upon which we build to standardise digital *Authorisations* are described here. For signing the *Authorisations*, the basic functions *Signing data and* V*alidating signatures MUST* be used.

There are standards to define/record *Authorisations*, define (executable) policies for those *Authorisations* and exchange/operationalize those *Authorisations*:

Recording/defining *Authorisations*

- Verifiable Credentials enable the issuance, holding, and verification of digital credentials that are portable, tamper-evident, and privacy-respecting. Verifiable credentials can be used to prove *Authorisation*. OpenID4VC can be extended to support decentralised *Authorisation*. By using verifiable credentials, OpenID4VC can provide a way for actors to prove their *Authorisation* in a decentralised manner.
- ZCAP-LD (Z-Capabilities for Linked Data) is a framework that combines the concepts of capabilities-based security with decentralised identifiers (DIDs) and linked data to express authorisations in a secure, decentralised environment.

For decentralised *Authorisations*, standards as mandated in the Architecture Reference Framework (ARF) of the EUDI Wallet initiative should be used. This is elaborated in the basic function *Attestation of Attributes.*

Policies

XACML can be used to define policies for authorisations. (eXtensible Access Control Markup Language) is a standard that defines a language for writing access control policies and a processing model for making authorisation decisions based on those policies. XACML supports fine-grained control and is suitable for complex authorisation scenarios involving multiple factors and conditions. A minimal form of XACML is currently being used in Operationalisation/Exchange[1]

The basic function *Attestation of Attributes* is used for issuing and presenting *Authorizations*. The basic function *Delivering messages* is used for exchanging the *Authorizations*.

Best practices

Authorisation model

For *Authorizing Actors* there is no common internationally accepted format to capture the authorization. TIP partners have drafted a format flexible enough to capture a wide variety of use cases collected by Dutch government agencies. This format was designed to be forwards compatible

[1] https://afsprakenstelsel.etoegang.nl/Startpagina/as/interface-specifications-hm-mr en Interface specifications HM-MR chain authorization

as an attestation of attribute within an EU Digital Identity wallet. This format will be tested in an international setting in a WE BUILD cross-border large scale pilot.

The table below shows the properties of an *Authorisation* and describes their meaning. In a following phase many of the defined properties will be described in the basic function *Attestation of Attributes.*

Property		
Issuer	Description:	Specifies the issuer (*Actor* that created and signed the *Authorisation*). Contains an ETSI EN 219 412-1 V1.5.1. compliant ecosystem wide unique identifier of the *Actor* providing the *Authorisation.*
	Type	String
	Cardinality:	1-1 (mandatory, may appear only once)
	JSON:	iss
	Example values:	"PASNL-NW863B5X", "IDCDE-590082394654", "PNONL-170555873", "EI:SE-200007292386", "NTRNL-87654321", "VATNL-861050198B01", "LEIXG-506700GE1G29325QX363", "CR:NL-1028549"
Subject	Description:	Refers to the Subject (Actor using the *Authorisation* to prove the capacity of acting on behalf of the Issuer). Contains an ETSI EN 219 412-1 V1.5.1. compliant ecosystem wide unique identifier of Subject.
	Type:	String
	Cardinality:	1-1 (mandatory, may appear only once)
	JSON:	sub
	Example values:	"PASNL-NW863B5X", "IDCDE-590082394654", "PNONL-170555873", "EI:SE-200007292386", "NTRNL-87654321", "VATNL-861050198B01", "LEIXG-506700GE1G29325QX363", "CR:NL-1028549"
Audience	Description:	URI that specifies for which audience the *Authorisation* is intended.
	Type:	String
	Cardinality:	0-1 (optional, may appear only once)
	JSON:	aud

	Example values:	"TIP:NL:belastingaangifte:VIA", "https://services.belastingdienst.nl/2024/IH/VIA"
Expiry	Description:	The time stamp of the end date and time at which the *Authorisation* becomes invalid. Optional because of certain cases of legally incapacitated *Actors.*
	Type:	Number
	Cardinality:	0-1 (optional, may appear only once)
	JSON:	exp
	Example values:	1475878357
Not before	Description:	The time stamp of the date and time at which the *Authorisation* becomes valid (if all required *Signatures* have been provided, the correct consent policy and *Authorisation* chain have been provided, etc.).
	Type:	Number
	Cardinality:	1-1 (mandatory, may appear only once)
	JSON:	nbf
	Example values:	1475876457
Issued at	Description:	Time stamp indicating the date and time at which the *Authorisation* was issued.
	Type:	Number
	Cardinality:	1-1 (mandatory, may appear only once)
	JSON:	iat
	Example values:	1475874457
Token identifier	Description	Globally unique ID of the *Authorisation*
	Type:	String
	Cardinality:	1-1 (mandatory, may appear only once)
	JSON:	jti

	Example values:	"uuid:1D44068C-7CFE-4538-8341-C7317C1FC49F"
Represented *Actor*	Description:	An ecosystem wide unique identifier of the *Actor* being represented through the *Authorisation*. Might be different from the Subject (e.g. when a judge mandates a curator)
	Type:	String
	Cardinality:	1-1 (mandatory, may appear only once)
	JSON:	nl.trustedinformationpartners.authorization.represented_acto r
	Example values:	“nl:BSN:123456789”, “nl:RSIN:87654321”, “nl:tel:0301234567”, “+31678765432”, ”peter.de_vries@domeinnaam.eu”
Revocation method	Description:	Type of revocation method on the *Authorisation.*
	Type:	String
	Cardinality:	1-1 (mandatory, may appear only once)
	JSON:	nl.trustedinformationpartners.authorization.revocation_metho d
	Example values:	"Bitstring Status List v1.0", "central register", "Revocation List", "mDOC proprietary", "non revocable"
Revocation value	Description:	Value is based on the revocation method used. This element is not present when empty.
	Type:	String
	Cardinality:	0-1 (optional, may appear only once)
	JSON:	nl.trustedinformationpartners.authorization.revocation_value
	Example values:	"Bitstring:297", "nl:eID:DigiDMachtigen", "https://vcrl.qtsp1.nl/revocationListService/".
Credential chain	Description:	Credentials proving claims needed to complete a full *Authorisation* (e.g. proof of being a judge, proof of being director of a company, an *Authorisation* signed by another non-independent major shareholder, etc.). Multiple credential chains might be present (e.g. an employee of a notary office,

		proving to be a notary and proving to be authorised to act on behalf of the notary office). The credential chain might also contain an *Authorisation* containing a credential chain (e.g. an employee proving to be authorised by the managing director and a managing director proving to be managing director of the organisation).
	Type:	Object
	Cardinality:	0-many (optional, may appear more than once)
	JSON:	nl.trustedinformationpartners.authorization.credential_chain
Conshttps://en.wikipedia.org/wiki/XACML#/media/File:XACML_Architecture_&_Flow.png **ent policy**	Description:	Specifies which consent is provided by whom (see below). One consent policy is required, more may be added.
	Type:	Object
	Cardinality:	1-many (mandatory, may appear more than once)
	JSON:	nl.trustedinformationpartners.authorization.iss_consent_policy

	Example values:	{
		"operation": "operationA",
		"resource": "resource1"
		}

Transferable	Description:	Specifies if the authorization is transferable to another actor. 0 means not transferable, 1 means 1 time transferable, 999 means 999 times transferable, etc.
	Type:	Number
	Cardinality:	1-1 (mandatory, may appear only once)
	JSON:	nl.trustedinformationpartners.transferable
	Example values:	0 1

		999

Consent policy model

The table below shows the properties of a consent policy and describes their meaning.

Property		
Operation	Description:	An ecosystem wide unique consent type ID in the form of a URN, indicating the kind of consent provided by the *Actor.*
	Type:	String
	Cardinality:	1-1 (mandatory, may appear only once)
	JSON:	operation
	Example values:	'nl:minfin:belastingdienst:service', indicating that the consent is concerning a *Service* for which the *Authorisation* is intended or 'nl:minjus:notriaat:volmacht:identificatie', indicating that the consent is concerning a notary witnessing the creation of a power of attorney for which all participants have been identified
Resource	Description:	The value of the provided consent type.
	Type:	String
	Cardinality:	1-1 (mandatory, may appear only once)
	JSON:	resource
	Example values:	"http://services.belastingdienst.nl/2023/IB/IH-Winst" for a consent type "service", or "true" for a consent type "volmachtidentificatie"

Mapping to ETD

In ETD [2]a Service (NL:"Dienst") is the granularity at which an authorisation can be given. There is one service catalogue[3], and each service is identified by a GUID. There are multiple authorisation registers and it can be assumed that internally authorisations are registered in records that have GUIDS. These are however not coordinated and/or persistent.

[2] https://afsprakenstelsel.etoegang.nl/?l=nl

[3] https://afsprakenstelsel.etoegang.nl/Startpagina/v1/service-catalog

ETD has the concept of an authorisation manager that can be appointed by the legal representatives of an organisation, and who can be given a specific scope. Furthermore, with this special authorisation manager authorisation, the authorisation manager MAY register a “normal” authorisation for himself, thus use the services himself within the scope of his authorisation manager authorisation.

Publishing authorisation types by relying party

Multiple relying parties will use the basic functionality of TIP for authorisations. These relying parties have different requirements for an authorisation. Therefore, relying parties must specify which elements are required for a specific authorisation. Also the possible parameters for the elements have to be defined by the relying party.

Uniformity on specifying the authorisation types which are accepted is necessary. Therefore, relying parties must use the basic function “*Publishing Service or Chain specifications”* for specifying the elements and corresponding parameters. A relying party must define a schema in which all requirements are set (elements and parameters). When a relying party needs more elements than defined in **Fout! Verwijzingsbron niet gevonden.**, the basic function needs to be adjusted by the workgroup.

Best practices for verification by the relying party

Upon receiving an authorisation, a relying party MUST perform a number of checks before proceeding with the workflow. An authorisation must meet both technical requirements and context specific requirements to be valid. These context specific requirements can be domain or service specific. The basic function *Publishing chain and service specifications* allows these domain specific requirements to be captured and published digitally.

In general, the following checks are to be performed by the relying party:

- The authorisation must meet the specified technical format. Only then subsequent checks can be carried out in good order.
- The specified context in the authorisation must match the service or legal action that is being invoked by the authorized actor.
- Schema checks must include the additional requirements set in the specified context. An example is an optional element that must be present in a certain context.
- If a revocation method is specified, then the revocation status must be checked.
- The time at which a service or legal action is being invoked must be within the validity timeframe of the authorisation.
- The signature(s) must be checked for validity, including all checks specified in the TIP basic function *Validating signatures*.
- The identity of the *Actor* that invokes a service or legal action must match the identity of the authorized actor as stated in the authorisation.
- The presence of evidence must be checked, including completeness and integrity of the chain of evidence.

- The presence of data that requires human assessment must be checked, and if present this must be assessed.

Reference to a "Context" for the authorisation

The TIP-specification for authorisation introduces the concept of referring to a 'Context' to increase the comprehensibility and practical applicability of the authorisation. The schema for the authorisation was designed to support many use cases, including complex use cases that many existing authorisation schemes do not support (yet). The convenience of both drafting and verification of the authorisation is significantly enhanced by using domain-specific constraints and guidance.

This approach is very similar to the 'Signature policy' which is referred to in digital signatures adhering to the European standards for digital signatures (e.g. XAdES, PAdES, etc). The 'Context' is a digital document which is under the stewardship of a domain authority. This domain authority is responsible for the procedure of drafting the 'context' document, publication and maintenance. Such a domain authority can be a formal authority (government agency or industry association), or can be a company that is structuring its own interactions or a contract-specific clause.

The 'Context' document is to be published on a URL or is alternatively referred to be URI. The context document is both human readable and machine readable. The context document is digitally signed to proof its authenticity and integrity. It can contain the following topics:

- Practical information for the authorisation, such as a clear explanation of legal consequences for all parties involved, or a reference to articles in the law.
- List an enumeration that must be applied, for instance a list of consent policy types that are supported in that domain.
- Impose additional constraints on the schema. For instance, limiting the cardinality to keep an authorisation simple. Or to enforce the presence of optional elements in specific circumstances.
- Reference to a methodology or criteria for the verifier of the authorisation.

The benefits of a 'Context' are the following:

- More clarity in advance for all parties involved to facilitate the process of an authorisation coming into being. The practical and legal value of the expression of will laid down in the authorisation is increased.
- A source of "truth" about the legal implications that can help settle a legal dispute.
- Allowing for higher levels of automation for domain specific aspects of the authorisation. This applies to both creation and verification of an authorisation.
- Performing context specific checks when drafting the authorisation reduces the rejection rate upon verification. When known in advance certain elements are mandatory in a certain context, this can be enforced upon creation, resulting in a lower rejection rate at the moment an authorized actor wants to use the authorisation.
- Simplicity of the authorisation can be enforced in situations where this is applicable.

- For more complex types of authorisations the exact scheme can be determined.

Supplier(s)

Multiple suppliers are involved in the lifecycle (see Figure 1) of an *Authorisation*. Some of these functions are already described in other basic functions, for which the suppliers described in those basic functions can be used.

For the process of Signing the suppliers of *Signing data* will be used. For Storing an Authorisation the suppliers of *Archiving data* can be used. Transfer and use Authorisation is an output of *Attestation of Attributes.*

For all other functions the suppliers are not prescribed.

There most likely will be an ecosystem where service providers will offer integrated solutions for key segments of the authorisation lifecycle. These will embed the TIP basic functions in an integral user-friendly workflow. For example, these offerings could include:

- A workflow for *Actors* to request an authorisation from a subject.
- A workflow that allows authorised *Actors* to use the authorisations within business processes with other stakeholders.
- A workflow catered to relying parties to carry out all mandatory and context specific checks on an authorisation.

Costs

Pricing for the use of the basic function *Authorizing Actor* are established on the basis of agreements between *Value-added service* providers, *Actors* and *Relying parties* who use Authorisations in their processes. Payment is made through the basic function *Making payments*.

Domain authorities or individual *Actors* may choose to cover the costs of *Authorizing Actors* in order to lower the barrier to interact digitally in a safe manner.

Annex

The following example is based on the authorisation model in the best practices section.

Simple example

```
{
  "iss": "PNONL-123456789",
  "sub": "NTRNL-00000003302174880000",
  "aud": "https://services.belastingdienst.nl/2024/IB/VIA",
  "exp": 1727949059,
  "nbf": 1725357059,
  "iat": 1725357059,
  "jti": "130018c9-e9f9-4470-9b11-b1e0021d0b12",
  "nl.trustedinformationpartners.authorization.represented_actor": "PNONL-123456789",
  "nl.trustedinformationpartners.authorization.revocation_method": "non revocable",
  "nl.trustedinformationpartners.authorization.iss_consent_policy": {
       "operation": "nl:minfin:belastingdienst:service",
      "resource": "https://services.belastingdienst.nl/2024/IB/VIA"
  },
  "nl.trustedinformationpartners.authorization.transferable":0
}
```

Credentials are transformed into base64:

“ew0KICAiaXNzIjogIlBOT05MLTEyMzQ1Njc4OSIsDQogICJzdWIiOiAiTlRSTkwtMDAwMDAwMDMzMDIx
NzQ4ODAwMDAiLA0KICAiYXVkIjogImh0dHBzOi8vc2VydmljZXMuYmVsYXN0aW5nZGllbnN0Lm5sLzIw
MjQvSUIvVklBIiwNCiAgImV4cCI6IDE3Mjc5NDkwNTksDQogICJuYmYiOiAxNzI1MzU3MDU5LA0KICAiaW
F0IjogMTcyNTM1NzA1OSwNCiAgImp0aSI6ICIxMzAwMThjOS1lOWY5LTQ0NzAtOWIxMS1iMWUwMDI
xZDBiMTIiLA0KICAibmwudHJ1c3RlZGluZm9ybWF0aW9ucGFydG5lcnMuYXV0aG9yaXphdGlvbi5yZXByZ
XNlbnRlZF9hY3RvciI6ICJQTk9OTC0xMjM0NTY3ODkiLA0KICAibmwudHJ1c3RlZGluZm9ybWF0aW9ucGF
ydG5lcnMuYXV0aG9yaXphdGlvbi5yZXZvY2F0aW9uX21ldGhvZCI6ICJub24gcmV2b2NhYmxlIiwNCiAgIm
5sLnRydXN0ZWRpbmZvcm1hdGlvbnBhcnRuZXJzLmF1dGhvcml6YXRpb24uaXNzX2NvbnNlbnRfcG9sa
WN5Ijogew0KIm9wZXJhdGlvbiI6ICJubDptaW5maW46YmVsYXN0aW5nZGllbnN0OnNlcnZpY2UiLA0KIC
AgICAgICJyZXNvdXJjZSI6ICJodHRwczovL3NlcnZpY2VzLmJlbGFzdGluZ2RpZW5zdC5ubC8yMDI0L0lCL1ZJ
QSINCiAgfSwNCiAgIm5sLnRydXN0ZWRpbmZvcm1hdGlvbnBhcnRuZXJzLmF1dGhvcml6YXRpb24udHJh
bnNmZXJhYmxlIjowDQp9”

Authorisation with signature:

{

 "payload":

"ew0KICAiaXNzIjogIlBOT05MLTEyMzQ1Njc4OSIsDQogICJzdWIiOiAiTlRSTkwtMDAwMDAwMDMzMDIx
NzQ4ODAwMDAiLA0KICAiYXVkIjogImh0dHBzOi8vc2VydmljZXMuYmVsYXN0aW5nZGllbnN0Lm5sLzIw
MjQvSUIvVklBIiwNCiAgImV4cCI6IDE3Mjc5NDkwNTksDQogICJuYmYiOiAxNzI1MzU3MDU5LA0KICAiaW
F0IjogMTcyNTM1NzA1OSwNCiAgImp0aSI6ICIxMzAwMThjOS1lOWY5LTQ0NzAtOWIxMS1iMWUwMDI
xZDBiMTIiLA0KICAibmwudHJ1c3RlZGluZm9ybWF0aW9ucGFydG5lcnMuYXV0aG9yaXphdGlvbi5yZXByZ
XNlbnRlZF9hY3RvciI6ICJQTk9OTC0xMjM0NTY3ODkiLA0KICAibmwudHJ1c3RlZGluZm9ybWF0aW9ucGF

ydG5lcnMuYXV0aG9yaXphdGlvbi5yZXZvY2F0aW9uX21ldGhvZCI6ICJub24gcmV2b2NhYmxlIiwNCiAgIm5sLnRydXN0ZWRpbmZvcm1hdGlvbnBhcnRuZXJzLmF1dGhvcml6YXRpb24uaXNzX2NvbnNlbnRfcG9saWN5Ijogew0KIm9wZXJhdGlvbiI6ICJubDptaW5maW46YmVsYXN0aW5nZGllbnN0OnNlcnZpY2UiLA0KICAgICAgICJyZXNvdXJjZSI6ICJodHRwczovL3NlcnZpY2VzLmJlbGFzdGluZ2RpZW5zdC5ubC8yMDI0L0lCL1ZJQSINCiAgfSwNCiAgIm5sLnRydXN0ZWRpbmZvcm1hdGlvbnBhcnRuZXJzLmF1dGhvcml6YXRpb24udHJhbnNmZXJhYmxlIjowDQp9",

"signatures": [

{

"protected":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpPU0UiLCJzaWdUIjoi...VWZTh2K2lLWDE1Vk4wST0ifQ",

"signature": "N98p1Y-
cYC6ewHtSp3DehMzdjpynGMAlYL3Bz5lyTGyFXKa_...TKqKOeXo0mKG8g7Hz6EQl5KoGBQ5ZCNLA"

}

]

}