ROBERTO POLLI TEAM PER LA TRASFORMAZIONE DIGITALE

INTEROPERABILITY RULES FOR AN EUROPEAN API ECOSYSTEM: DO WE STILL NEED SOAP?

Agenda

The Italian Digital Team

Old SOAP Framework

SOAP & REST

The New Framework

Standardization & Reliability

Future ideas

Team Mission

Make **public services**
for citizens and businesses
accessible in an easy manner,

via a mobile first approach,

with **reliable**, scalable and
fault tolerant **architectures**,

based on clearly defined **APIs**.

Who am I

Roberto Polli - love writing in Python, C and Java

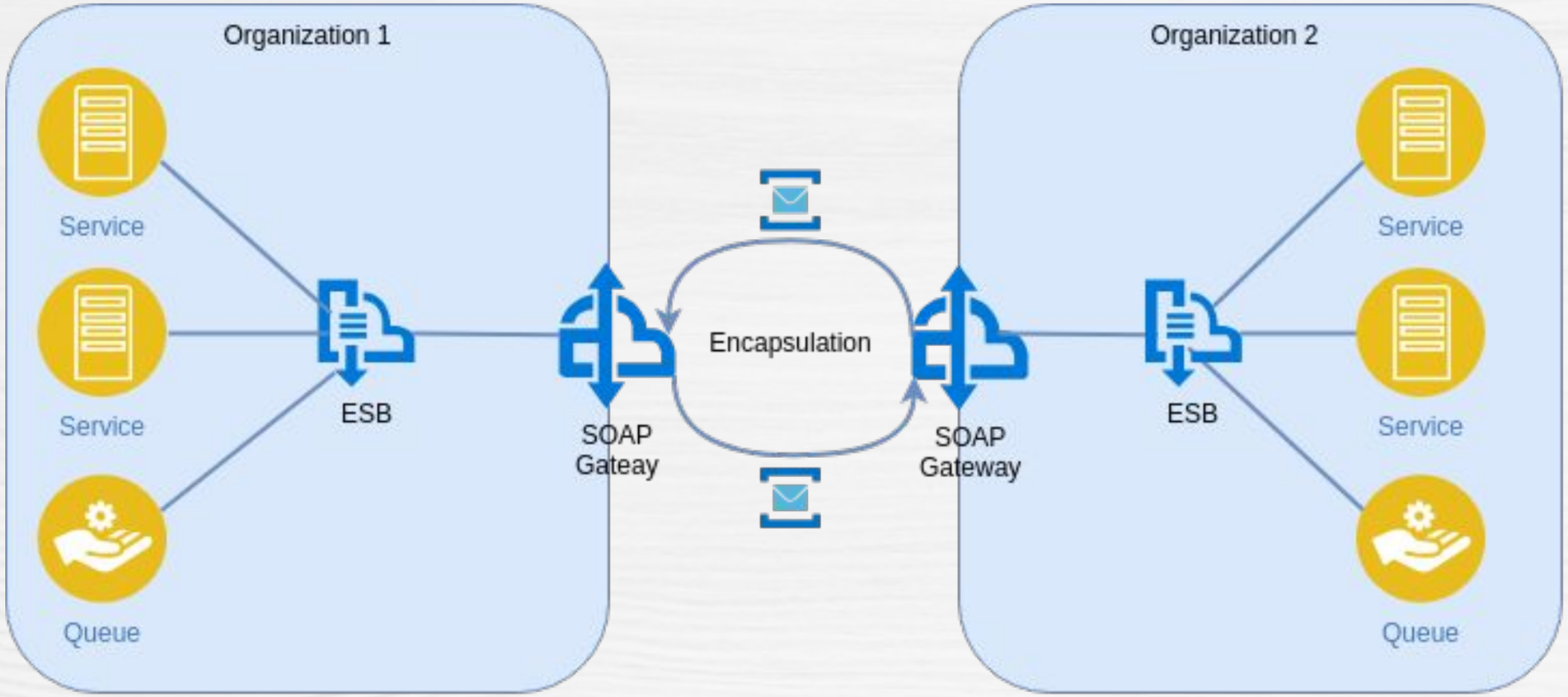
RHC{E,VA}, MySQL|MongoDB Certified DBA

API Ecosystem @ TeamDigitale

From Enterprise to The Web

The Old SOAP Framework

Ad-hoc encapsulation with a custom gateway

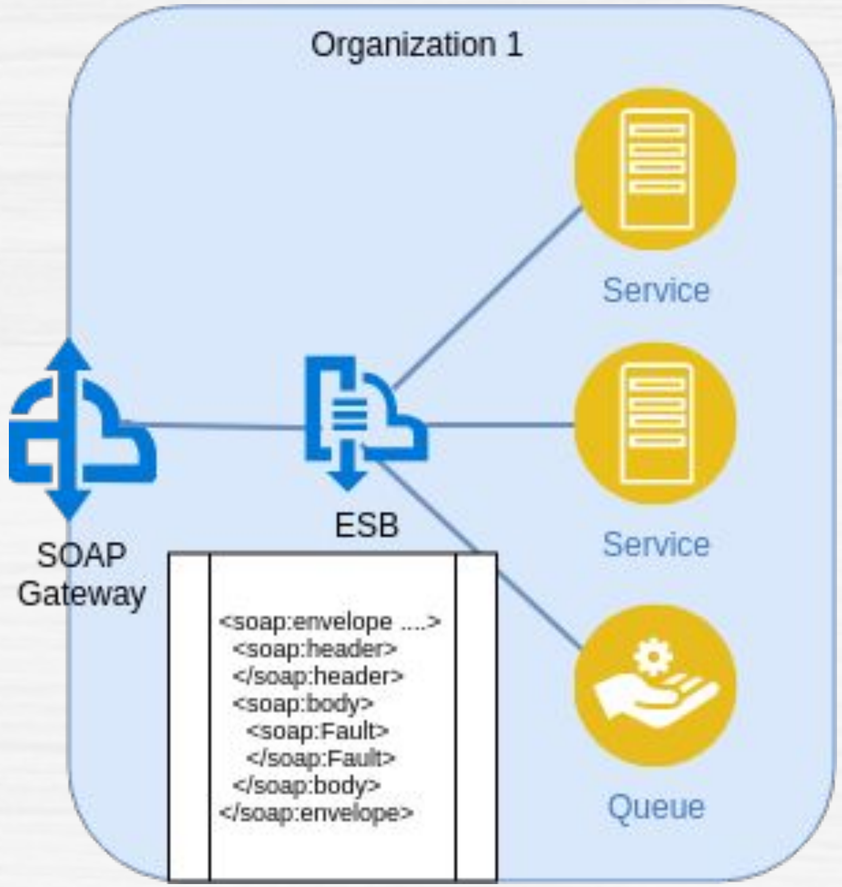


The Old SOAP Framework

Processing errors (SOAP Faults) required de/serialization of XML

No universal semantic for communicating service status (soap faults uses 500 for everything)

Errors at peak loads caused further thrashing



The Old SOAP Framework

Become a barrier for the creation of new services:

- Very expensive (both for setup and maintenance/operation)
- Complicates communication with non-governmental agencies
- The IT world was moving beyond SOAP

Beyond SOAP

SOAP was born in 1999:

- transfer-agnostic messaging protocol (HTTP, SMTP, ..)
- adds one layer, with computational and architectural costs
- virtually asynchronous exchanges (soap messages)

Today:

- new HTTP Semantics RFC 7230-7238 released in 2014
- services are inherently based on HTTP
- synchronous exchanges (eg. mail vs chat)

Beyond SOAP

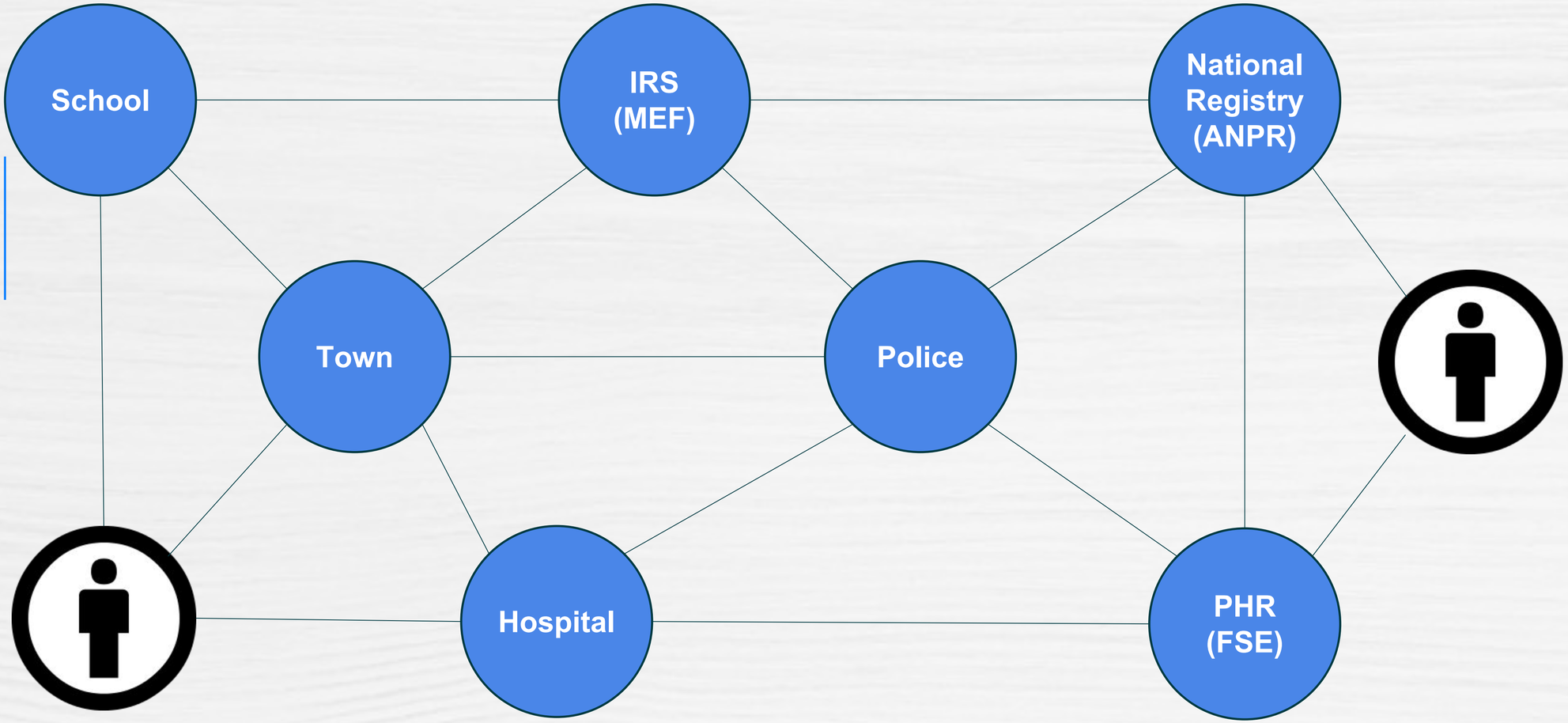
The new semantics allow to:

- route requests using Path and Method
  (Eg. idempotent vs non-idempotent)
- use Status and Headers for service management,
  don't have to process the body
- Caching, Conditional and Range Requests, ...

The New Framework

- Standardize HTTP APIs without SOAP
- API-first approach to REST APIs based on OpenAPI v3
- Scheme standardization based on national, European and industry standards
- Availability strategy based on a distributed circuit-breaker and throttling patterns

The New Ecosystem



Standardization

HTTPS

http

binary messages

Always HTTPS

Wrap queues (kafka, JMS, AMQP, ...) with HTTPS for authentication and authorization

Leverage STATUS, METHOD and PATH for auditing and routing

Logs, dates: RFC5424 / 3339

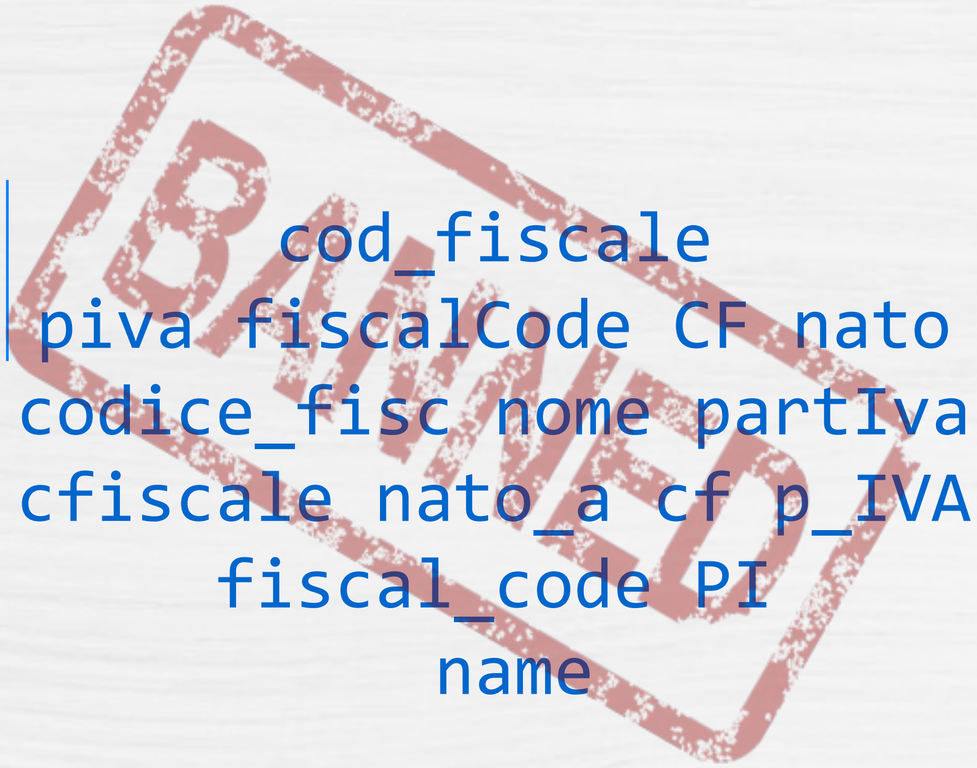

```
   ago  6 14:04:50
ago-06 18:58:50,000
 Aug 02 18:43:47.000
mer  9 ago 08:45:37 CEST
          2018
Fri May 05 08:45:37 IST
2018-May-08 10:06:25 AM

05/12/2018 2018/12/05
12-05-2018 05/12/2018
2018-12-05 12-05-2018
```

2018-05-08T10:06:25Z

2018-05-08T10:06:25.000Z

Ontology-based schemas



cod_fiscale
piva fiscalCode CF nato
codice_fisc nome partIva
cfiscale nato_a cf p_IVA
fiscal_code PI
name

tax_code
vat_number
given_name

(from w3id.org/italia)

Reliability

Reliability

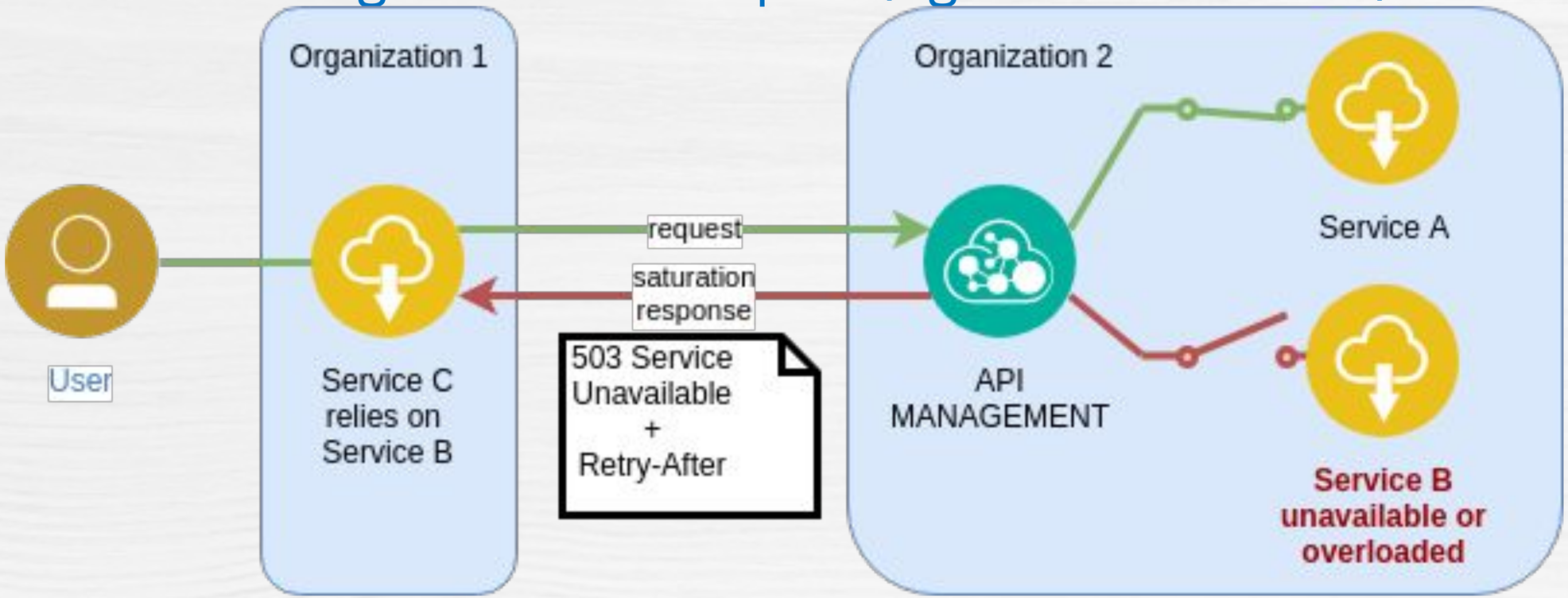
Business Continuity Plan (European Interoperability Framework)

Integrated management of load and failures

Avoid cascading failures

Reliability

Service management techniques (eg. circuit-breaker)



Service Management Headers

x-rate-limit-minute: 100

X-RateLimit-Retry-After: 11529485261

X-RateLimit-UserLimit: 1231513

X-RateLimit-UserRemaining

X-Rate-Limit-Limit:
name=rate-limit-1,1000

x-custom-retry-after-ms

X-Rate-Limit-Remaining-month

X-Rate-Limit-Reset: Wed, 21 Oct 2015 07:28:00 GMT

x-rate-limit-hour: 1000

Communicate service limits

X-RateLimit-Limit: #request

X-RateLimit-Remaining: #request

X-RateLimit-Reset: #seconds

Communicate service status

HTTP 503 (service unavailable)

HTTP 429 (too many requests)

Retry-After: #seconds

Errors: RFC7807

RFC 7807 is an **extensible** format for errors

```
{
    "type": "https://tools.ietf.org/html/rfc7231#section-6.6.4",
    "title": "Service Unavailable",
    "detail": "Service is active in forex hours",
    "status": 503,
     "instance": "/account/12345/msgs/abc",
}
```

Future steps

Standardized metrics

Readable indicators:

- use rates, not absolute values
- use basic units (eg. Bytes, seconds, ...)
- use increasing Service Level Indicators, the higher the better

Example:

- availability is 0-100%
- expose success rates, not error rates

Standardizes metrics

Set common and simple indicators:

- availability: eg. the service was up for 95% of the time
- success_rate: % of successful requests
- target_response_time: expected latency at 95p

Evaluating:

- or responsiveness: the service meet the target_response_time for 90% of the time
- or APDEX index: $Apdex_t = \frac{SatisfiedCount + \frac{ToleratingCount}{2}}{TotalSamples}$

Signatures and Encryption

Signing an exchange with a digital certificate is the **basis** for a non-repudiation framework.

SOAP has a well-established (and criticized) standard for Signing and Encryption

REST standards are Json Web Signatures|Encryption RFC7515 used by OpenID Connect (still criticized)

Signatures and Encryption

Possible choices:

- leave the signature to the application protocol (eg. json)
- sign just the body (a sort of ws-security built with JWS) extending the objects with claims or adding an Headers
- sign a `fingerprint(request,header,body)` via Headers

Current request/response fingerprint functions and Signature headers proposals (eg. amz, draft-cavage, signed-exchanges)

Further discussions

On digital certificates:

- RSA is considered a legacy
  https://github.com/WICG/webpackage/pull/181
- EC keys are easily embedded in claims and headers

On Headers

- evaluate Structured Headers

```
Example-DictHeader: en="Applepie", da=*w4ZibGV0w6ZydGUK=*
```

- deprecate or adopt Digest

References

New Italian Framework

https://forum.italia.it/c/piano-triennale/interoperabilita

http://lg-modellointeroperabilita.readthedocs.io/it/latest/

Roberto Polli
roberto@teamdigitale.governo.it

@ioggstream
@teamdigitaleIT

@team-per-la-trasformazione-digitale

teamdigitale.governo.it