# Harmonized Web API for real-time data

## A unified Smart City API

Peter Waher

Trust Anchor Group AB (TAG)
peter.waher@trustanchorgroup.com

**Abstract.** This paper presents a method to combine two apparently incompatibly types of data sources into one whole fused distributed data set, accessible via well-known and standardized interfaces. It shows how decentralized and distributed queries can be made to the Internet of Things as a whole, able to join, collect, consolidate and process information from such seemingly incompatible types of distributed data, such as *persistent* and *transient* data. Persistent data is typically available in traditional data sources such as databases, files, web resources and open data sets, including data sets available via APIs. Transient data, on the other hand, is not stored or persisted, as it loses its real-time value quickly. Instead, such data is received and acted upon live in real-time. Instead of reading from a database, different communication protocols are used to communicate with remote connected devices such as sensors and actuators directly. This new type of API presented in this paper fuses these two types of data into one, making it possible to use a single technology for accessing and querying all types of data, establishing a Unified API for Smart Cities.

## 1 Introduction

Traditional data sources using databases and advanced querying tools have been used by developers and data scientists for many decades for insights and business logic. The first computers had very limited internal memory. To be able to perform any useful task, all input and output of any process was assumed to be persisted in central storage in large computer halls. Very little was distributed or even maintained or processed in transit. As computer performance and demand have increased, processing of data has been able to be distributed, and processed elsewhere, closer to where and when it is needed. More of the data can also be processed in transit, without the need for intermediate persistence.

Due to the historical development of computers, different branches of computer science produced engineers with very different skillsets. On one hand, data, statistical and informatics engineers were trained in all different kinds of database operations and data processing related to such operations. The main goal was to efficiently process business logic, which drove the development. Most of the systems developed under this branch of engineering resulted in similar architectures: Centralized approaches, always with a huge database at the center, collecting all information used,

or potentially to be used. Business logic and client logic could then be distributed to wherever users require them.

Electronic engineers, however, concerned with the creation of sensors and other devices, often used in industrial applications, automation, control, monitoring applications, or in general interaction with the physical world (so called *cyber-physical systems*), were more concerned with the real-time aspects of information. Databases, while useful for archiving purposes, implied a performance degradation in the real-time operation. Focus on this branch of engineering is the real-time communication protocols that make interoperation possible. Devices don't work in a vacuum and require interconnection to be able to perform tasks. Such systems perform poorly if they run on historical data stored in databases. Such data is often valuable in the instant and quickly becomes stale and quickly replaced. Devices need to respond quickly, and on the most recent data available, preferably the actual current value. This focus on real-time processing (or near real-time processing) is the principal concern.

The purpose of this paper is to present a unified method and technology that allows these two different types of data to co-exist and be joined together as a whole and be queried as a single, albeit *federated*, unit.

## 2 Unified Information Model

A unified technology, or "API", requires a unified information model. Such a model exists. It was developed under the umbrella of the "*Semantic Web*" by the W3C. The focus was to create an information model that can be distributed across the World Wide Web, where every piece of information can be linked to any other piece of information on the web (so called *Linked Data*), and a model that could express any kind of information any human could be able to express in writing [1].

The name *semantic* comes from the realization made in the study of languages (*semantics*), that basic information could be expressed by categorizing words used into *subjects*, *predicates* and *objects*. A subject is someone or something being described. A predicate is a property, an action, etc., describing the subject. And the object is the description. The object in turn can also be a subject in new statements, creating thusly a huge graph of information described only using sets of (Subject, Predicate, Object) triples, often referred to as *semantic triples*. These can also be shortened to (S, P, O) or SPO.

To create a new information model for the semantic web, the Subject, Predicate and Object must be conveniently described, to create an *extensible*, *linkable* and *distributed* information model. The best technology for this purpose was to reuse the concept of the Uniform Resource Locator (or URL) used on the World Wide Web, but in a more generalized form: The *Uniform Resource Identifier*, or the URI. A URL is supposed to be resolved into a resource that can be accessed over the Internet. A URI looks like a URL is not in the general case resolvable. It is only used to identify something. A URI (just like a URL) is distributed by its nature and linkable. Anyone owning a domain can create URIs under its domain. By requiring all Subjects in the

model to be URIs, in one stroke, everything being described by the information model is intrinsically distributed.

Since not everyone can have access to all domains, the semantic web also defined the distributed information model to comply with the *Open World Assumption*. This means that you may never know everything there is to know about something. Lack of information is not the same as the information does not exist. The latter assumption is called *Closed World Assumption*, which is the traditional model used when using centralized databases [2].

Apart from Subjects being URIs, Predicates are also defined as URIs. By requiring them to be URIs you create a federated model for the descriptions being made. The Predicates are often grouped together into *Ontologies*, based on the base URI used when creating the predicate URIs. These sets of Ontologies form languages used to describe subjects and are the basis for interoperation with regards to *meaning* expressed by the information model [3].

Objects can also be URIs, allowing for the creation of huge graphs of information. They can also be *Literals*. Literals are *values*. There are many different types of values. They can be typed, and they can be localized into different languages. Each Literal Type is also expressed as a URI. This makes it possible for anyone controlling a domain to create new data types for literal values.

## 3 Representing Semantic Information

Once you can express information into a set of Semantic Triples, you need to represent this set somehow, to communicate it. You can represent a semantic model, or graph, in many ways. The Semantic Web initiative, and later the W3C Data Activity, developed multiple standards for representing semantic triples. One of the first mechanisms was called Notation3 (or N3), which simply listed the triples in a text document [4]. A more efficient mechanism called *Turtle* was developed, avoiding repeating subjects and predicates needlessly [5]. While easy to write by humans, there are few formal validation tools and editors available for plain text documents. A more formally stringent mechanism was developed based on XML, the *Resource Description Framework* (or RDF) [6]. A JSON alternative called JSON-LD was also developed for languages that do not support XML [7].

## 4 Graph Stores

Once you can represent semantic information, you may want to persist it. W3C developed a protocol for storing semantic information (or graphs) in Graph Stores [8]. The semantic information can be stored either directly as files, using the representation formats available, or stored in *Graph Databases*. Such Graph Databases are highly specialized databases that only store semantic triples. Since it specializes in Semantic Information, it only needs to develop a few indices, and therefore optimize its processing for that purpose.

## 5 Querying Semantic Information

Now we have a Unified Information Mode, standardized mechanism to represent, communicate and store, semantic information, we need a mechanism to query this semantic information. Such a mechanism was also developed by the W3C and is recursively called the ***S**PARQL **P**rotocol and **R**DF **Q**uery **L**anguage* (or SPARQL) [9]. This query language is based on pattern matching graphs, and returns matches found. It can also be distributed in a federated manner, meaning SPARQL can either fetch remote semantic information for processing, or distribute subqueries to remote SPARQP endpoints for remote processing, and then consolidate the responses returned.

## 6 Extending the Semantic Web to Transient Information

We have a mechanism to represent any information we can express we can communicate it, store it and query and process it, all using standardized technologies. But the standards assume persistence of information, and do not delve into the complications of real-time information, where persistence is not an option. To be able to create a Unified API for Smart Cities, we need to provide this aspect. We do this by providing a SPARQL endpoint on-top of an interoperable and distributed infrastructure that harmonizes access to devices. Such an interoperable and distributed infrastructure is provided by the *Neuro-Ledger*® [10], with interoperability interfaces defined by the *Neuro-Foundation* [11].

Each Neuro-Ledger node hosts a SPARQL endpoint. Apart from being able to process information in graph stores and online resources, it also publishes a new type of semantic graph that *dynamically generates* semantic information based on real-time *transient* information provided by connected devices. It can also convert semantic operations into actions and control commands in real-time.

## 7 Example

The following example is taken from the presentation titled "*Harmonized Web API for real-time data*" [12], which contains a series of examples [13] going from simple to more advanced showing how SPARQL can be used to process both persistent and transient information. The example begins by using a simple source of semantic information, expressed here using Turtle, that points out sensors available [14]:

```
@prefix demo:   <https://lab.tagroot.io/Demo/> .
@prefix lab:
    <https://lab.tagroot.io/MeteringTopology/> .

demo:SensorReferences demo:references [
    demo:node   [
```

```
            demo:nodeGraph lab:concon.weather;
            demo:country "Chile";
            demo:tech "Web API, Open Weather Maps"];
      demo:node   [
            demo:nodeGraph
<https://lab.tagroot.io/MeteringTopology/Channel-
86258a0b72f612d68707e8054911dcf0%231>;
            demo:country "Portugal";
            demo:tech "MQTT, IEEE 1451.1.6, IEEE
1451.0"];
      demo:node   [
            demo:nodeGraph lab:stockholm.openweathermap;
            demo:country "Netherlands";
            demo:tech "Web API, Open Weather Maps"];
      demo:node   [
            demo:nodeGraph
<https://lab.tagroot.io/MeteringTopology/ProcTemp>;
            demo:country "Italy";
            demo:tech "ModBus"];
      demo:node   [
            demo:nodeGraph lab:azure-sensor-interop;
            demo:country "Chile";
            demo:tech "XMPP, IEEE P1451.99"]
  ] .
```

The series of examples conclude with a query that reads all momentary values from these nodes, filters out Temperature values, and outputs the nodes and temperatures that are above 290 K. Implicit unit conversion is done automatically by the harmonization layer. The SPARQL endpoint used was [15].

```
  PREFIX demo:    <https://lab.tagroot.io/Demo/>
  PREFIX conc:    <urn:nf:iot:concentrator:1.0:>
  PREFIX sd:      <urn:nf:iot:sd:1.0:>
  PREFIX unit:    <urn:nf:iot:sd:1.0:unit:>
  PREFIX rdfs:    <http://www.w3.org/2000/01/rdf-schema#>

  SELECT
      ?nodeName ?country ?tech ?timestamp ?fieldName
?fieldType ?value ?qos
  FROM
      demo:SensorReferences.ttl
  WHERE
  {
      ?s demo:node [
            demo:nodeGraph ?node;
```

```
            demo:country ?country;
            demo:tech ?tech] .
      GRAPH ?node
      {
            ?node conc:operations [
                  ?p [conc:read ?read]].
            FILTER(STRENDS(STR(?read), "/momentary"))
      }
      GRAPH ?read
      {
            ?s2 sd:timestamps [
                  ?timestampNr [
                        sd:timestamp ?timestamp;
                        sd:fields [
                              ?fieldNr [
                                    rdfs:label ?fieldName;
                                    a ?fieldType;
                                    sd:value ?value;
                                    sd:qos ?qos]]]].
            FILTER(?fieldName = "Temperature")
            FILTER(?value > "290"^^unit:K)
      }
      BIND(STRAFTER(STR(?node), "MeteringTopology/") as
?nodeName)
  }
```

Depending on the current temperatures, a response could look as follows:

| nodeName | country | tech | timestamp | fieldName | fieldType | value | qos |
|---|---|---|---|---|---|---|---|
| Channel-86258a0b72f612d68707e8054911dcf0%231 | Portugal | MQTT, IEEE 1451.1.6, IEEE 1451.0 | 10/23/2025 5:18:40 PM +00:00 | Temperature | Momentary | 303.15 K | AutomaticReadout |
| ProcTemp | Italy | ModBus | 10/23/2025 5:19:05 PM +00:00 | Temperature | Momentary | 20.8 °C | AutomaticReadout |

# 8 Cybersecurity

The SPARQL and Graph Store protocols are HTTP-based protocols. The Neuro-Ledger supports different forms of authenticating users and providing access to these resources. Internally, the Neuro-Ledger supports defining users with different roles and privileges. The Neuro-Ledger authenticates users using either WWW-Authentication, JWT bearer tokens, or Mutual TLS (mTLS). The later is preferred for distributed queries, as they conform well with semantic web standards. The underlying harmonization layer for Internet of Things runs on XMPP with strong identities, authentication and spoofing protection. To this, the Neuro-Foundation [11] adds decision support, digital identities, provisioning and smart contracts, as well as distributed tokens, peer-to-peer communication and end-to-end encryption to protect identities, authorizations and access and integrity of information.

## 9 Summary

The SPARQL endpoint provided by the Neuro-Ledger has access to both persisted data via traditional graph stores, and dynamic and transient data via the harmonized IoT layer. The Neuro-Ledger is also distributed, and therefore provides an infrastructure component with a Unified API that can access all types of data in a Smart City and Smart Society.

## 10 References


[1] W3C Data Activity – Building the Web of Data, Available Online 2025-10-23: https://www.w3.org/2013/data/

[2] Open World Assumption vs. Closed World Assumption, Juan Sequeda, 2012-11-30, Available Online: https://www.dataversity.net/articles/introduction-to-open-world-assumption-vs-closed-world-assumption/

[3] Linked Open Vocabularies (LOV), for searching ontologies. Available Online: https://lov.linkeddata.es/dataset/lov/

[4] Notation3 (N3), W3C, 2011-03-28, Available Online: https://www.w3.org/TeamSubmission/n3/

[5] RDF Turtle 1.2, Working Draft, W3C, 2025-09-28, Available Online: https://www.w3.org/TR/rdf12-turtle/

[6] RDF 1.2 Primer, Group Draft Node, W3C, 2025-04-03, Available Online: https://www.w3.org/TR/rdf12-primer/

[7] JSON for Linking Data, Availanble Online: https://json-ld.org/

[8] SPARQL 1.2 Graph Store Protocol, Working Draft, W3C, 2024-12-19, Available Online: https://www.w3.org/TR/sparql12-graph-store-protocol/

[9] SPARQL 1.2 Protocol, Working Draft, W3C, 2025-08-14, Available Online: https://www.w3.org/TR/sparql12-protocol/

[10] Neuro-Ledger™, Executive Summary, 2019-10-11, Available Online 2025-07-22: https://neuro-foundation.io/Papers/Neuro-Ledger,%20Executive%20Summary.pdf.

[11] The Neuro-Foundation, Available Online 2025-07-22: https://neuro-foundation.io/

[12] Harmonized Web API for real-time data, Neuro-Foundation, 2025-08-23, Available Online: https://neuro-foundation.io/Presentations/Harmonized%20Web%20API%20for%20real-time%20data.pptx

[13] SPARQL Examples querying real-time information from sensors, 2025-10-23, Available Online: https://lab.tagroot.io/Demo/SparqlExamples.txt

[14] Sensor references used in SPARQL demo, 2025-10-23, Available Online: https://lab.tagroot.io/Demo/SensorReferences.ttl

[15] SPARQL Endpoint for demo, 2025-10-23, Available Online: https://lab.tagroot.io/Sparql.md