



API toolkit

6Aika

Leverage from
the EU
2014–2020



European Union
European Regional
Development Fund

Publisher

The 6Aika Open Data and Interfaces
Spearhead Project
Helsinki
Espoo
Vantaa
Tampere
Turku
Oulu

Editor-in-chief

Annukka Varteva

Text

© Forum Virium Helsinki, 2017
v1.0, 2017
Writer: Petja Partanen
User license: CC BY-SA 4.0

Illustrations and layout


© Paper Planes Oy, 2016
User license: CC BY-ND 4.0

These materials can be used as long as the publisher is credited. The attribution must be removed if so requested by the publisher of the material.

The aim of the 6Aika Open Data and Interfaces Spearhead Project is to bring the opening of data as part of the cities' normal operations and to help the cities facilitate the creation of data-based services and business.

Content

Basic concepts	5
Open data	5
Application programming interface, API	6
The anatomy of an API	8
When is an API needed?	8
What does an API do?	9
How to create an API	9
Ways of creating an API	11
Data format	12
Data model	13
Building an API	15
Service level agreement	15
Make your API modular	16
Data licenses	16
Terms of API use	16
Open source license types	17
Maintaining and managing an API	19
The life cycle of an API	19
Managing an API	20
The discoverability of APIs	21
Documentation and instructions	21
International and national networks	22



This brochure is intended to be read in conjunction with *the Open API recommendations for cities* publication, which explores APIs, their importance as part of the services offered by cities and the objectives and measures related to APIs on a general level.

This brochure takes a more detailed look at the most important issues that should be taken into consideration in the implementation of an open API. The brochure is primarily meant to provide help with API procurement, due to which it does not explore the technical intricacies of APIs. The brochure also includes links to additional information.

Unless otherwise stated, in this publication API refers to an open application programming interface (API).

Basic concepts



Open data

5

Open data is information that has been made available to the public in a machine-readable format and licensed in a way that allows the data to be utilised freely and free of charge. Open data is also characterised by the fact that users do not need to ask permission to utilise it. Open data may consist of statistics, financial information, maps, images, video recordings or 3D models, for example.

Open data can be used free of charge by anyone – even for commercial purposes. Most open data is published by public sector organisations, but companies, other organisations and private individuals can also open up their data for use by the public.

Open data serves as digital raw material. It can be refined and combined with other data, opening up entirely new possibilities for digital services and business.

Not all public information can be opened up due to privacy and licensing issues, for example. However, most of the information produced in the public sector is already public by law and available to anyone who requests access to it.

Publishing data online in a machine-readable format so that it is easily accessible to everyone around the clock not only facilitates the utilisation of said data, but also frees up resources that would be otherwise needed to process information requests. Open data can be published in the form of individual files or through an open API.

Here are some useful instructions for opening up public data:

- The Ministry of Transport and Communications' *Public data – an introduction to opening information resources* (pdf) publication, published in 2011, is still a topical guide for opening up the public sector's information resources.
- The open data distribution platform *avoindata.fi* maintained by the Population Register Centre includes a *Guide to open data*, which provides process flow charts for public sector organisations seeking to open up public data.
- The Helsinki metropolitan area's *Helsinki Region Infoshare* service provides instructions on how to open up data related to the city.

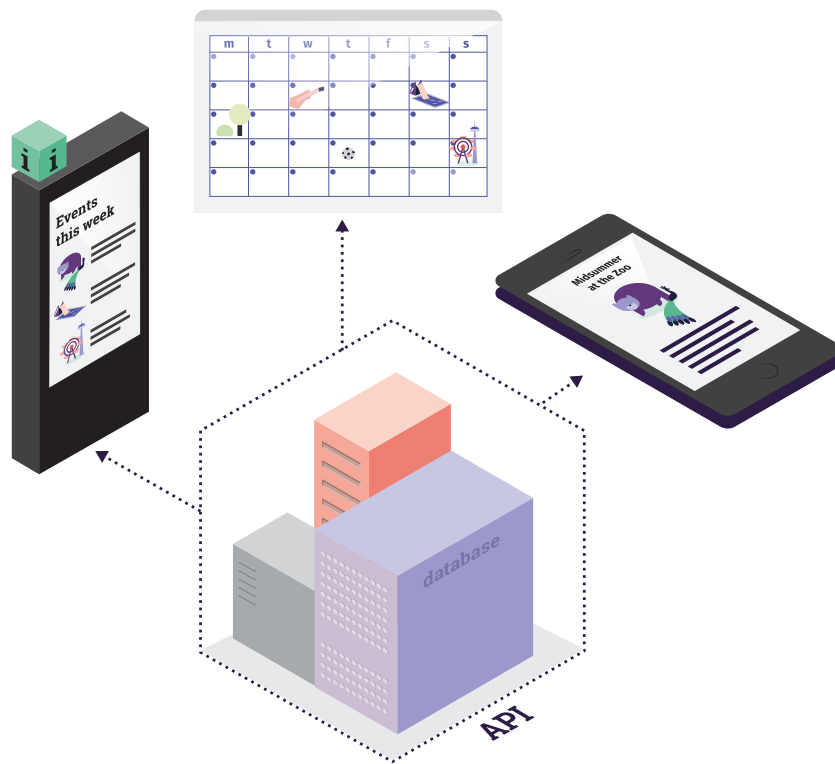
Application programming interface, API

An application programming interface (API) is a way of building connections between systems, devices and applications. An API includes commands that can be used, for instance, to retrieve data or use the functions of a back-end system without having to provide third-party access to the system itself.

For example, the 6Aika cities' Linked Events API provides access to data on events held in the 6Aika cities in a unified and machine-readable format, allowing the data to be used in various services. Thanks to the API, the event data is not confined to the cities' own systems and user interfaces, but can instead be freely utilised in various applications, from smartphones to information displays.

Publishing data in a machine-readable format makes it easier to utilise.

APIs intended for the distribution of open data consist almost exclusively of online *Web Services*. An API can be a **data API** (one example being the 6Aika Open Decision API,



An API provides access to data contained in back-end systems in a unified format. Different applications can utilise the data provided through the API in various ways.

which provides access to the 6Aika cities' decision-making data in a machine-readable format) or a functional API (one example being the 6Aika Issue Reporting API, which can also be used to submit issue reports).

Being open means that all the properties of the API are public and that the API can be used without any restricting terms and conditions.

An open API is public and can be used without restrictions.

Further information:

The definition of an open API: *Open API recommendations for cities*, pp. 14–17.

The anatomy of an API



8

When is an API needed?

The need for an API is always resolved on a case-by-case basis. An API is typically needed when the data being utilised is real-time or updated regularly, for example. However, publishing the data through an API may not be necessary if the data is updated infrequently, for example only once a year, or if the amount of data is small.

The best way to start opening up data is to first publish all the data as a static file, which should be updated at regular intervals. If keeping the data up-to-date proves time-consuming or if the users of the data would like an API, the next step is to consider the creation of an open API. Compared to an API, the main drawback of a static file is that when the data contained in it is updated, all the services that utilise the data need to be updated manually. While building an API takes some work, doing so may end up saving a considerable amount time and effort in the future, for both the organisation opening up the data and the parties utilising said data.

The key benefit of an API is that the data stays up-to-date.

The key benefit of an API is that the data provided through it is always up-to-date, which can be vitally important for the parties utilising the data. A real-time API enables the creation of real-time applications, such as services that show the movements of buses or snowploughs on a map, for example.

When data is published through an open API, the parties looking to utilise the data do not need access to the back-end system in order to make use of the data. An API can be integrated directly into the operative system, in which case access can be restricted to data that is open. Alternatively, an API can be separated from the operative system and made to retrieve and add data to its own database at regular intervals.

An API enables the creation of real-time services.

What does an API do?

An API provides data in machine-readable format. For example, the 6Aika cities' Linked Events API enables the retrieval of data on events organised in the cities, which can then be utilised in various digital services. When a service, such as a simple smartphone application, sends a request to a server running a REST API (such as `http GET api.hel.fi/linkedevents/ v1/event/?start=today`), the server retrieves all the events from the event database that are set to start today or in the future. The server then confirms the completion of the HTTP request it received and returns a response to the client software in JSON format, which contains data on thousands of events.

This communication process is not that different from the communication between a web browser and a web server. Regardless of whether the sender of a request is a person browsing the web or a piece of client software, the client sends a request complying with the HTTP protocol to a network location on the server in accordance with the URI protocol. The only difference is the message that the server sends back in response: the response sent by a REST API is stripped of all the elements intended for human users, such as HTML and CSS codes. What remains is raw data, which can be easily processed by systems and applications.

How to create an API

One of the first choices that the creator of an API needs to make is the choice of architectural style. Before REST APIs became common, web APIs were typically based on SOAP (Simple Object Access Protocol), a standardised information exchange protocol developed by Microsoft and based on the XML standard. The lighter and simpler REST architecture style was introduced in 2000.

REST-architecture

REST (*Representational State Transfer*) is an architectural model for creating programming interfaces based on the HTTP protocol. Due to its simplicity and reliability, REST has quickly become the most important software architecture for creating APIs. It is characterised by statelessness (all the data related to a request is transferred in each request), the server-client model and the fact that it's based on the HTTP protocol.

While SOAP is a standardised protocol for information exchange, REST is not a strict standard, but rather a set of architectural principles. Because of this, REST provides a greater degree of freedom for API designers.

The differences between REST and SOAP are effectively illustrated in infographics created by Nordic APIs: nordicapis.com/rest-vs-soap-nordic-apis-infographic-comparison

The popularity of lightweight REST APIs has been growing steadily relative to SOAP APIs, which are generally considered more complex. However, API technologies are in a state of constant development, and in the future REST may very well be supplanted by new query languages, such as GraphQL. As such, you should always check how up-to-date your chosen technology is before creating an API.

The user interface is built separately on top of the API.

The second choice that needs to be made is **how to create the API**. The development of an API can be outsourced to the provider of the back-end system or to a service provider specialising in the development of APIs, or the API can be developed in-house. One thing you should keep in mind is that an API is not the same thing as a user interface. The user interface intended for regular users needs to be built separately on top of the API. The user interface can be created independently from the API, either internally or by an external service provider. The table below illustrates the benefits and challenges of three different ways of creating an API.

Ways of creating an API		
1.	2.	3.
The API is provided by the provider of the back-end system	The provider of the back-end system conducts minimum integration, another system provider creates the API	The provider of the back-end system conducts minimum integration, the city creates the API in-house (requires the city to have some coding expertise of its own)
Benefits		
<ul style="list-style-type: none"> » Utilisation of existing procurement contracts » Ensures the highest competence in regard to the back-end system » A single provider is responsible for the entire system 	<ul style="list-style-type: none"> » Procurement can be based on the best API expertise and developer support » A more agile development model for an external API (independent from the back-end system) » The back-end system is effectively isolated from the public API 	<ul style="list-style-type: none"> » A more agile development model for an external API (independent from the back-end system) » The back-end system is effectively isolated from the public API » The most agile procurement model (in-house work instead of a procurement process)
Challenges		
<ul style="list-style-type: none"> » The API competence of different providers varies » Limited readiness for developer support » Dependence on the back-end system: if the back-end system is replaced, a new API has to be created » Difficult to replicate in other cities unless they are using the same system 	<ul style="list-style-type: none"> » Integration between two different systems and suppliers poses risks in regard to reliability » The back-end system provider may have limited interest in developing and supporting APIs » Higher cost (two providers, no synergy benefits for the development of the back-end system) » Easier to replicate, but still requires system-specific integration for different back-end systems 	<ul style="list-style-type: none"> » Limited resources » Continuity and maintenance over the long term » Integration between two different systems and suppliers poses risks in regard to reliability » The back-end system provider may have limited interest in developing and supporting APIs » Easier to replicate, but still requires system-specific integration for different back-end systems

Data format

It is also important to choose the right format for presenting the data provided by the API. The choice should preferably be based on the potential users of the data: which data format would be the easiest to utilise for users? The two most popular API data formats are **JSON** (JavaScript Object Notation) and **XML** (Extensible Markup Language).

JSON is a simple open-standard data-interchange format. Despite its name and the fact that it is derived from JavaScript, JSON is completely language-independent. JSON can be easily parsed by both people and computers.

JSON

```
{ "employees": [
  { "firstName": "John", "lastName": "Doe" },
  { "firstName": "Anna", "lastName": "Smith" },
  { "firstName": "Peter", "lastName": "Jones" }
]}
```

XML

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

The same data in two different languages. JSON presents the data in a more compact format and can be directly utilised by the objects of different programming languages.

Source: www.w3schools.com/js/js_json_xml.asp

XML is a metalanguage similar to HTML developed in the 1990s by the World Wide Web Consortium. It is used to describe the structure of data without predefined codes. XML can be used to form new codes, with the help of which documents can be adapted for a wide range of different purposes.

JSON presents data in a more compact format than XML.

The popularity of the structurally simple JSON has grown rapidly, making it the most popular data format solution for new APIs.

Definition of the JSON language: www.json.org

Definition of the XML language: www.w3.org/TR/REC-xml

If possible, the data format should be harmonised with other actors that are opening up similar data. You should also consider providing the data in multiple formats (for example Excel, CSV, XML, JSON) if the back-end system allows for this.

Choose a data format that is easy to use.

Data model

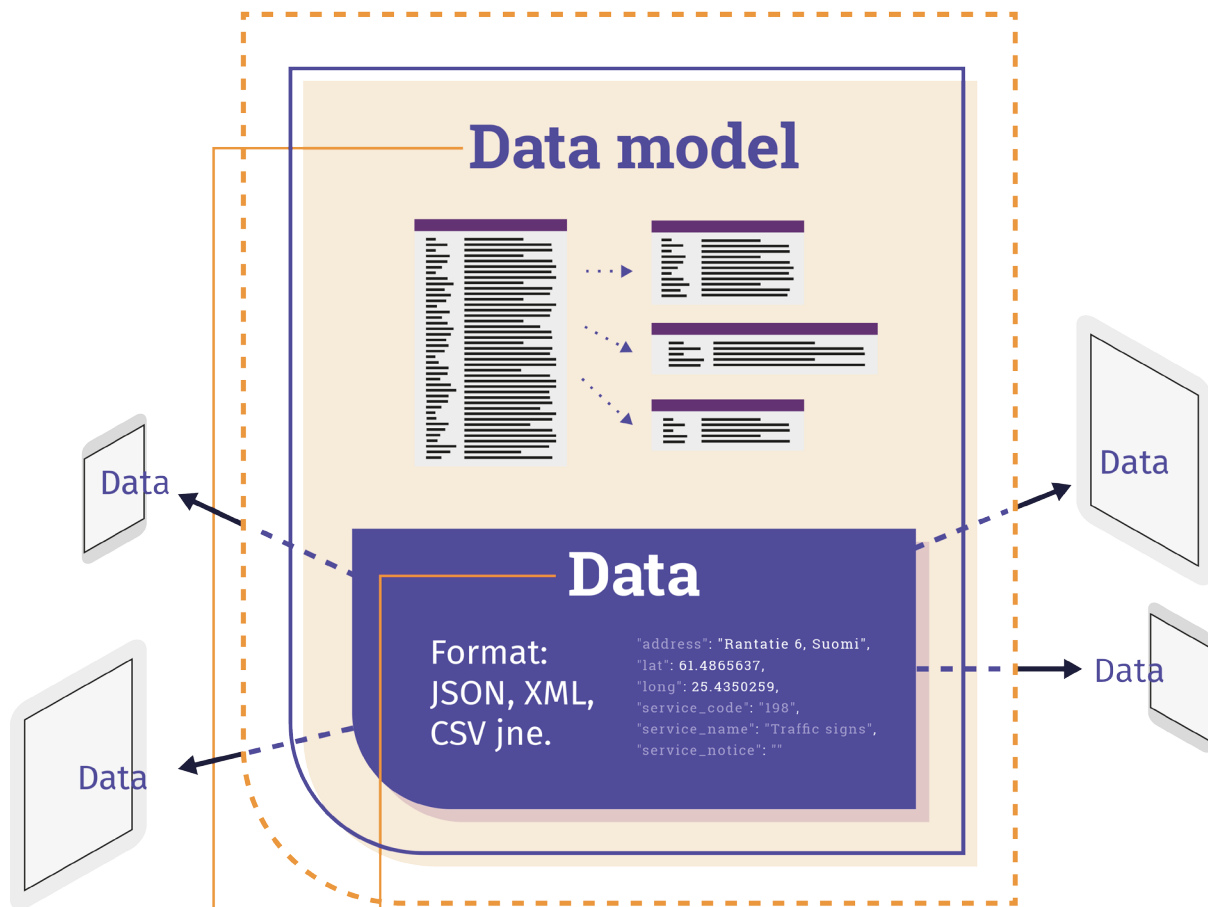
13

The data model describes the data fields contained in the API and how they relate to one another. Design a logical data model for your API, preferably international open data models, such as schema.org or Popolo. From a long-term perspective, using standardised data models improves the interoperability of information systems.

- The international Popolo standard describes relationships between organisations and people: www.popoloproject.com
- Schema.org's vocabularies provide models for structured data on the Internet: schema.org

Standardised data models improve interoperability.

API



● **Data** consists of zeroes and ones.

Format is the technical representation of the data.

● **Data model** is the definition of how the data is structured. It also describes what data fields the API contains and how they relate to one another.

Building an API

Service level agreement

The process of outsourcing services often includes the preparation of a service level agreement (SLA) between the client and the service provider, which defines the quality of service to be provided. Drawing up a service level agreement can also increase the attractiveness of a free API. If you decide to draw up a service level agreement, be sure to also make it available to the public.

The aim of the SLA is a sufficient service level for the API, so that users inside and outside the organisation can rely on its operability. In case of an error, the users need to have access to a version of the data that is as up-to-date as possible, even under extraordinary circumstances. Be sure to set a sufficiently high target service level. If you wish to make your API attractive to commercial actors, it should be usable without interruptions and response times should be short.

Developers need to be convinced that their contributions won't just suddenly disappear. Do so by publishing a development plan for your API. Demonstrate to them that the API will remain available far into the future and that its development will continue.

Reliable APIs
are attractive for
businesses

Make sure that your API can accommodate extensions without the need to change the core of the software.

Make your API modular

When designing a new API, be sure to take expandability into consideration from the get-go. Even a well-designed API often ends up missing some features that are important for users, such as specific data fields or search parameters.

The structure of an API's source code should be designed in a way that enables the creation of plug-ins and extensions without changing the core of the software. A poorly designed structure may result in the need to make changes

to the API's source code, branching the development of the API. Synchronising changes between different branches may prove difficult and require additional work.

License the data that you are opening up under an open license.

Data licenses

Provide users as much freedom to utilise the data as possible. An open data license provides the parties utilising the data with the assurance that the licensed data can be freely used in the ways permitted by the license.

The license recommended by the JHS Public Administration Recommendations is *Creative Commons Attribution 4.0*. Material licensed under this license can be copied, shared, shown and presented as well as used as part of another work. The material can be used for both non-commercial and commercial purposes. The material must be attributed to its creator.

Terms of API use

The terms of API use define who is allowed to use the API and how. Keep the terms simple: describe practices related to connecting to the API and whether there are any limits on requests, for example. Try to make access to the API as open as possible. If access to the API is restricted, define and document the restrictions clearly. If necessary, the use of an API can be restricted so that users are permitted a specific number of requests free of charge, after which they can increase their maximum for a fee.

If using the API requires an API key or some other form of authentication, make sure that automatic authentication is possible even outside of office hours. The very definition of openness requires the API to be usable without any input from an administrator. However, requiring users to log in may be useful for reaching developers and other users of the API in exceptional situations.

Examples of the terms of API use:

- *The Finna.fi service of Finnish archives, libraries and museums*
- *The City of Helsinki's feedback system*

Keep the terms of API use simple.

Open source license types

If the API includes source code, license the code under an open source license, if possible. When choosing an open source license, make sure the check its compatibility with other licenses, as this affects the range of tools that can be used with the API.

There are three types of open source licenses:

1. Strong copyleft licenses (such as GPL)

If the original software is modified or new elements are added to it by linking, for example, the source code must be published under the same license terms as the original software. This way derivative works also stay open and subject to the same license terms.

2. Copyleft licenses (such as AGPL, EPL)

If the software is modified, the changes must be published. Published software can be freely combined with software made under different licenses. This allows for the incorporation of closed components that have their own licenses.

3. **Permissive licenses** (such as MIT, BSD, Apache)

If the software is modified, the source code can be included in the derivative work, but this is not required. Derivative works are not required to be open. In other words, the software can be incorporated into closed software.

Further information:

JHS 169 Use of Open Source software in Public

Administration: www.jhs-suositukset.fi/suomi/jhs169

Maintaining and managing an API

The life cycle of an API

The life cycle of an API should be planned in the same way as that of any other system. It should be noted, however, that the life cycle of an API may differ from that of its back-end system. Publish the planned life cycle of your API (future updates, versions, etc.) in its documentation and be sure to update the plan regularly.

Do not make any changes to the version of the API currently in use that would introduce incompatibility issues. When such changes are necessary, publish a new version of the API and make sure that the old version stays functional for a sufficient period of time alongside the new version. Doing so gives developers time to modify their client software to function with the new version of the API.

When an API's back-end system is nearing the end of its life, plans must be made on how to carry out a controlled ramp down of the API in a way that inflicts minimal harm to other systems and external parties.

Make sure to provide users with a sufficient transition period when modifying your API.

Managing an API

An open API is just like any other service provided by a city. As such, an API needs clearly defined processes, roles and persons responsible in order to ensure that it remains operational even in exceptional situations.

The management model of an API needs to take both technological and organisational issues into consideration. The best way to keep things clear is to draw-up separate plans for both categories. **The technical specification of the API** describes the technology of the API: what types of information technology, software and techniques the API uses and what kind of data it relays.

The technical administration of the service must address at least the following issues or outsource them to a service provider:

Important questions related to the **management of an API** include who is responsible for the administration and further development of the API and how this development is funded.

Technical administration

- » Monitoring of the API's performance
- » Monitoring of server capacity
- » Monitoring and optimisation of database efficiency
- » Load balancing
- » Responding to error situations and service interruptions

Documentation and user support

- » Maintenance and updating of documentation
- » Monitoring and documentation of the API's usage statistics
- » Planning and providing notifications of service interruptions
- » Processing of technical support requests

Technical development

- » A development model that takes into account the users of the API and minimises disruptive changes
- » Iterative improvement of the API
- » Implementation of new functionality

The discoverability of APIs

Even a great API will remain unutilised if the developer community cannot find it. The first step to improving discoverability is adding the API to directory services.

International API directories

- ProgrammableWeb.com is an API directory and news site specialising in APIs: : www.programmableweb.com

Finnish open data services and directories

- The Population Register Centre's open data and interoperability service: www.avoindata.fi
- Helsinki Region Infoshare, the Helsinki metropolitan area's data directory: www.hri.fi
- City of Tampere data directory: data.tampere.fi
- City of Oulu data directory: avoindata.ouka.fi
- Southwest Finland data directory: www.lounaistieto.fi

Documentation and instructions

Create a website for your API that provides developers with easy access to the API's specifications and the chance to test the API on a browser.

Provide users who wish to test the API with sample requests, the functionality of which has been tested. Explain what the parameters included in the request do, what fields the retrieved data contains and what they mean.

Document the format of the response data and any exceptions to the format thoroughly. It is especially important to thoroughly document the parameters (filters) available for API requests.

Provide sample code that utilises the API in one or several generally used programming languages.

Good API documentation includes at least the following:

- A functional description of the API
- A technical description of the API
- A description of the data content of the API
- Instructions for implementing and using the API
- Documentation on the data security and privacy protection of the API
- Machine-readable API documentation (such as Swagger, OpenAPI definition, RAML, JSON Schema)

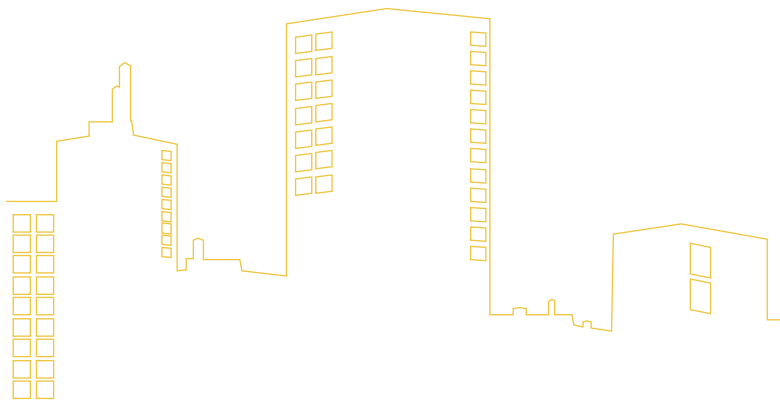
International and national networks

International organisations

- The Open & Agile Smart Cities initiative (OASC), a city network established in 2015: www.oascities.org
- Open311, an open source community developing the definition of a civic issue tracking API: www.open311.org
- Nordic APIs, a Nordic API community: nordicapis.com
- The CitySDK website, an international showcase of CitySDK and 6Aika API projects: www.citysdk.eu

Public sector organisations specialising in APIs in Finland

- 6Aika – Cooperation strategy for sustainable urban development: 6aika.fi
- The Association of Finnish Local and Regional Authorities: www.kuntaliitto.fi/asiasanat/avoin-data
- The Government ICT Centre Valtori provides independent ICT services for central government administration: www.valtori.fi
- The Population Register Centre administers services such as theSuomi.fi service, which provides citizens with easy access to public services, their personal information and electronic messages, all in one place. vrk.fi
- The Open Knowledge Finland developer community: fi.okfn.org
- The {API:Suomi} group on Facebook: www.facebook.com/groups/apisuomi



Contact us

6Aika API cooperation

Forum Virium Helsinki

info@forumvirium.fi

Helsinki

hri@hel.fi

Espoo

ict-palvelut@espoo.fi

Vantaa

ictpalvelut@vantaa.fi

Tampere

avoindata@tampere.fi

Turku

avoindata@turku.fi

Oulu

avoindata@ouka.fi

6Aika

Leverage from
the EU
2014–2020

