Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Project Acronym: CitySDK

Grant Agreement number:
297220

Project Title:

Smart City Service Development Kit and its application pilots

D3.1 Participation Pilot Application and it's SDK components

Revision: Final

Authors:

Hanna Niemi-Hugaerts (Forum Virium Helsinki)
Henri Salomaa (Sanoma)
Jaakko Rajaniemi (City of Helsinki)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	x
C	Confidential, only for members of the consortium and the Commission Services	

REVISION HISTORY AND STATEMENT OF ORIGINALITY

Revision History

Revision	Date	Author	Organisation	Description

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Work Package 3: Smart Participation

D3.1 Participation Pilot Application and it's SDK components

Hanna Niemi-Hugaerts, Forum Virium Helsinki
Henri Salomaa, Sanoma
Jaakko Rajaniemi, City of Helsinki

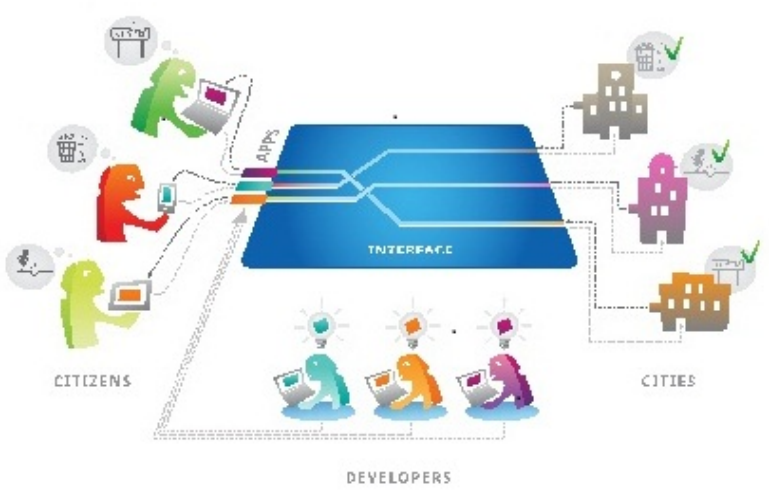
1 Purpose of this document 5
2 Introduction to WP3: Smart Participation 6
3 Requirements for Smart Participation 8
3.1 Terminology 9
3.2 Requirements for Smart Participation through User Scenarios 9
3.2.1 Listing request types and definitions 10
3.2.2 Quering several requests 12
3.2.3 Adding a picture and writing a description 13
3.2.4 Submitting the request and following up on it 14
3.3 Future enhancements to the interface 15
4 Standards for Smart Participation 16
4.1 Open311 (Geo Report v2) 16
4.2 Other available standards 16
5 CitySDK Smart Participation interface specification 17
5.1 General 17
5.2 Architecture 17
5.3 Formats 18
5.4 Service Discovery 18
5.5 API methods 20
5.5.1 GET Service List 20
5.6 GET Service Definition 22
5.6.1 Possible errors 24
5.6.2 POST Service Request 26
5.6.3 GET Service Requests 29
5.6.4 GET Service Request 34
5.6.5 Error messages 37
5.7 New parameter definitions 38
5.7.1 Extensions and city specific parameters 38
5.7.2 Status of service requests (detailed_status) 39
5.7.3 Service Objects (service_object_type,service_object_id, service_object_base) 39
5.7.4 Language support (locale) 40
5.7.5 New query options 40
5.7.6 Media handling 41
5.7.7 Other changes to Open311 v2 specification 41
5.8 Examples on API messages and usage 41
5.8.1 Submitting a service request with an image 41
5.8.2 Service request with image and attributes 42
5.8.3 Quering service request with extensions 43
5.8.4 Listing service definition with language set to Finnish language 44
6 Smart Participation Lead Pilot 47
6.1 Metro.fi 48
6.2 City of Helsinki's centralized feedback management system 50
6.3 City of Helsinki's Public Works department 51
6.4 Developer engagement in the Lead Pilot 53
7 Conclusions 54

1 Purpose of this document

This document introduces the possibilities that Participation Lead Pilot Application and it's SDK components can offer to developers, municipalities and citizens. Document walks the reader through the decision-making process behind Smart Participation interface specification and choices related to it (see Chapters Requirements for Smart Participation and Standards for Smart Participation). The user perspective is showcased through user scenarios whilst making a strong connection to the technical requirements.

Smart Participation Work Package's SDK Component, the issue-reporting interface, is carefully specified in the section CitySDK Smart Participation interface specification.
In the last part the Lead Pilot is presented providing readers with a concrete pilot case of Smart Participation interface.

2 Introduction to WP3: Smart Participation



The purpose of the Smart Participation work package is to create an open interface that acts as an issue-reporting channel between the citizens and the civil servants. The work is based on the Open311 technology, which is a standardized protocol for location-based collaborative issue tracking. Further work on the interface was needed, as it didn't fulfill all needs of the Lead Pilot and Replication Pilots.
The goal of this work package is to a) provide cities with a specification for issue reporting interface, b) support developers building apps that make citizen feedback easier and c) allow citizens to give feedback via commonly used virtual platforms or applications that are not necessarily owned or maintained by the city. To make the work manageable Smart Participation focuses mainly on location related issue reports, leaving topics like participatory budgeting out of scope.

Figure 1: Open311 based Citizens Connect from Boston

Within the Smart Participation domain interfaces are built between the feedback systems and other platforms to enable a direct flow of citizen feedback to relevant recipients in the City Hall. For the citizens, different pilots give an opportunity not only to give feedback but also to follow it by applying commonly used digital services.

Figure 2: Chicago utilisizing Open311 in addition to their other services

The open interface used in the Smart Participation pilot will be shared with the developers. By using the Open113 -based technology the developers are able to create new applications that can easily travel across borders.

Figure 3: Screenshot from Open311 based Verbeeter de buurt from Amsterdam

3 Requirements for Smart Participation

The requirements for the Smart Participation interface were collected both from existing services and through collaboration with active partners in Smart Participation. This resulted in the Smart Participation requirements document. [1]

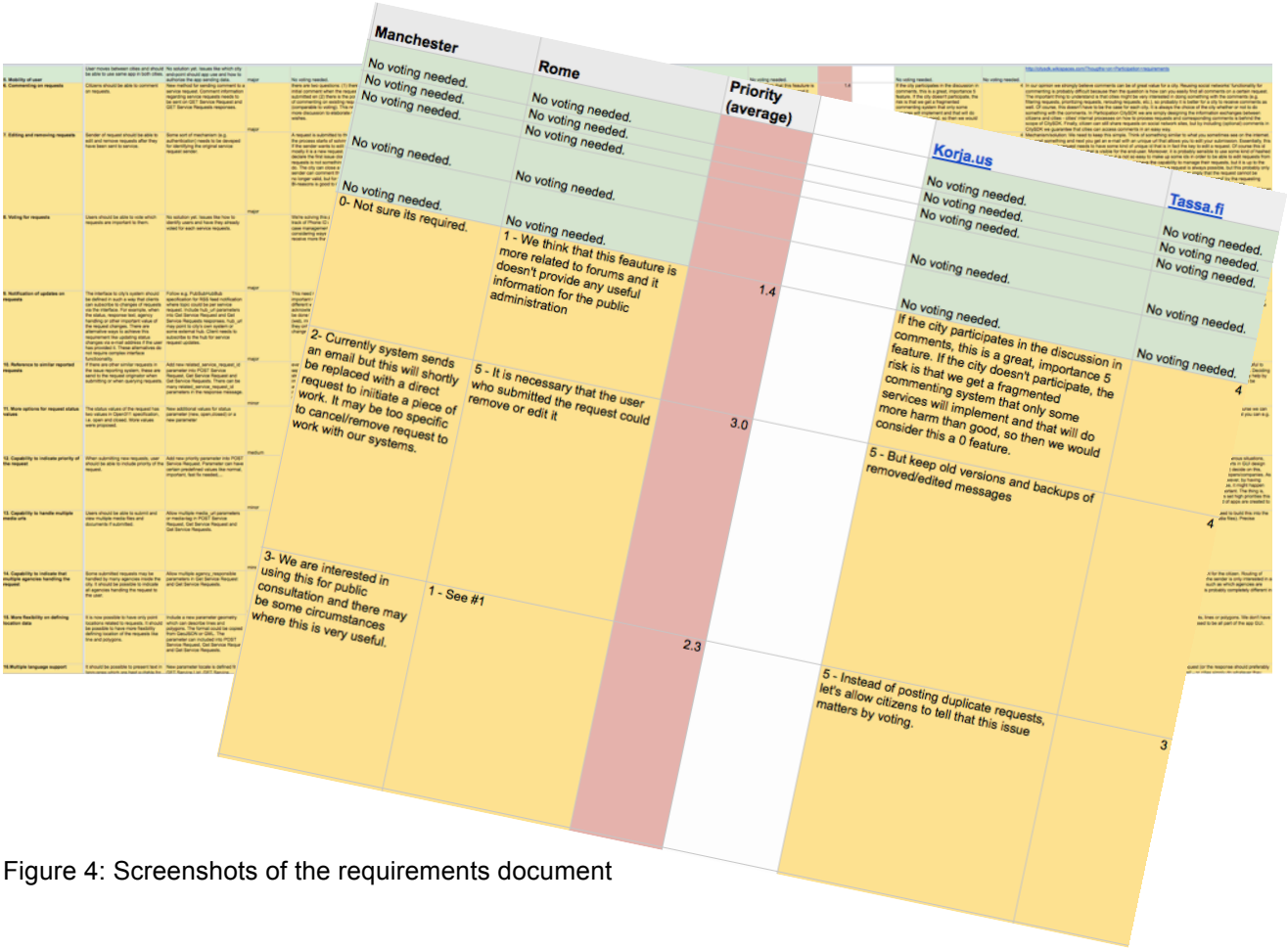


Figure 4: Screenshots of the requirements document

Partners collaborated on prioritizing the suggested extensions to interface specification using a collaborative document. [2]

In addition to the 7 active city partners and the WP2 leader, also third parties were heard. These included Finnish app developer group Korja.us, SME running Tassa.fi service and Dutch GovUnited[3].

Local developer community and SMEs have also been heard during the process both in public developer meet-ups and in other meetings and via social media, eg. CitySDK Helsinki Facebook page.[4]

---

1 https://docs.google.com/a/forumvirium.fi/spreadsheet/ccc?key=0AqvnokBCNLe5dGE5ak5yd21tajNHaTRVSU0tZ2RROHc#gid=0

2 https://docs.google.com/a/forumvirium.fi/spreadsheet/ccc?key=0AqvnokBCNLe5dFJfNXNKUjlkdHE2S2pJTzNPZ1M5eGc#gid=0

3 http://www.govunited.nl/

4 https://www.facebook.com/CitySDKHelsinki

3.1 Terminology

Request = *issue report or feedback which is sent to the city's feedback system.*
- *Example: request on pothole or traffic sign*

Request type = *different categories of issue reports which city will be addressing and fixing*
- *Example: request type on street conditions*

City's feedback system = *issue management system used by city personnel*

3.2 Requirements for Smart Participation through User Scenarios

In the following pages Smart Participation requirements are introduced from the point of view of the user but bearing in mind the technical aspects and possibilities of each requirement. Illustrations are included together with the use cases to show how and what information travels through the interface from the citizen to the city and vice versa. First are the scenarios that will be supported by the Lead Pilot and the latter part goes through requirements that will be supported in the future versions of the interface.

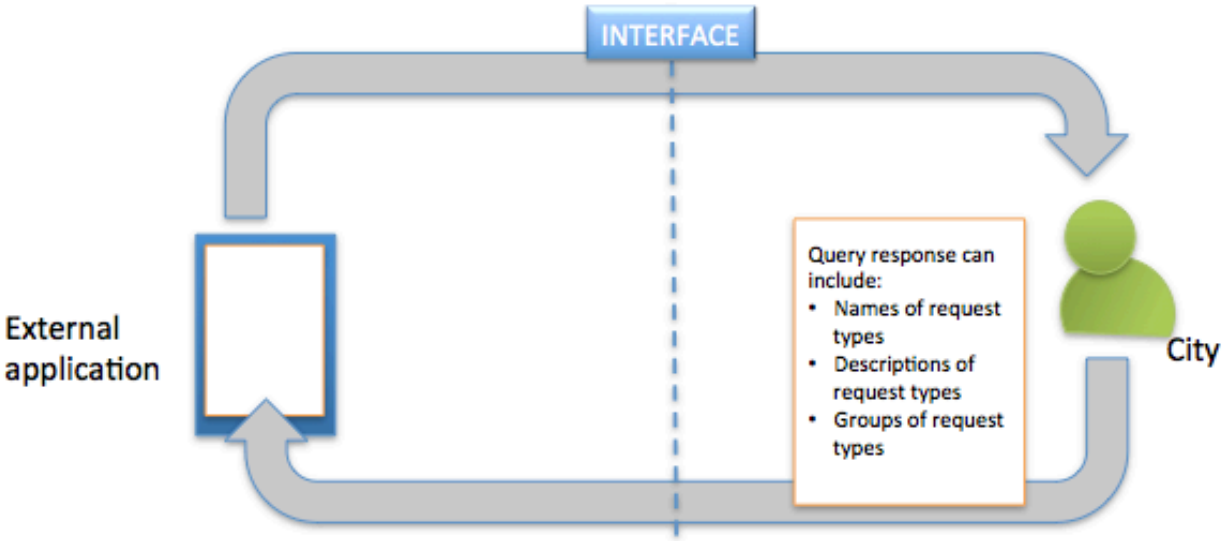
As a starting point we took Tim, who faces something in the urban environment that he would like the city to fix.

- Tim is a 30 year old office worker living in Helsinki, Finland
- Tim uses smart phone apps in his everyday life
  - To keep in contact with friends and family
  - To manage work related e-mail and personal bank issues
- Tim feels distant from city's decision making. He wants more direct ways to contribute to his community and neighborhood

3.2.1 Listing request types and definitions

1. On his way home Tim sees a dangerous hole in the street; he wants to inform the city administrator about it
2. He uses a smart phone app to report issues. App shows on which issues Tim can report and which information is required.

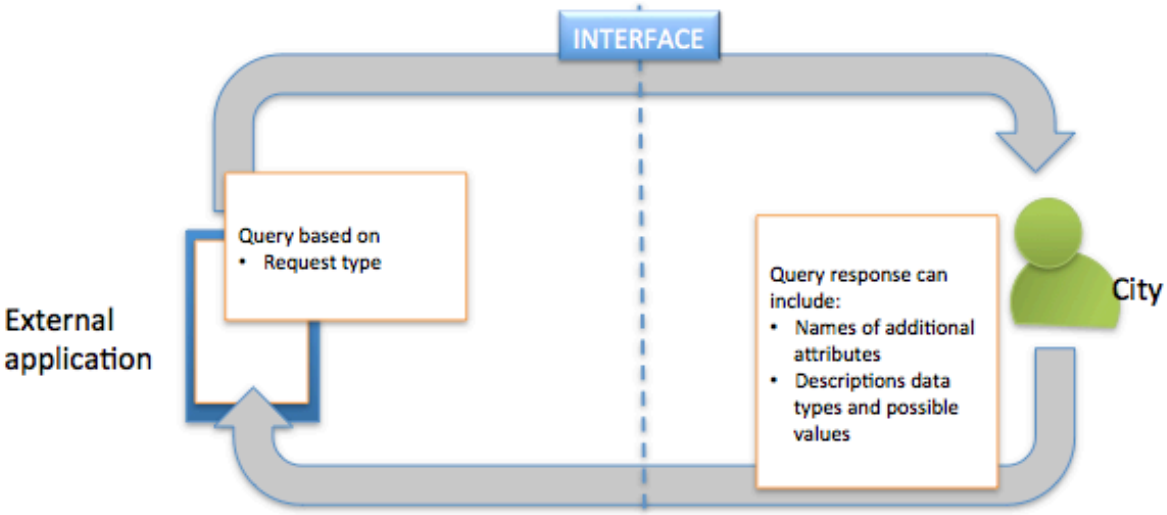
Listing request types and definitions



Query response can include:

- Names of request types
- Descriptions of request types
- Groups of request types

Listing request types



Query response with attributes can include:

- Names of additional attributes
- Descriptions data types and possible values

3.2.2 Quering several requests

Tim checks if somebody has already made a report about the hole

5

Smart Participation interface needs to support quering of submitted issue reports. The interface allows queries based on

- Submission date and time (start and endtime)
- Location (lat/long+radius)
- Status Service request type

Interface supports City's response, which can include:

- Description
- Location
- State
- Response text
- Submission date and time
- Update date and time
- Expected date and time when fixed
- Government agency responsible for the service request
- Service request type
- Photo

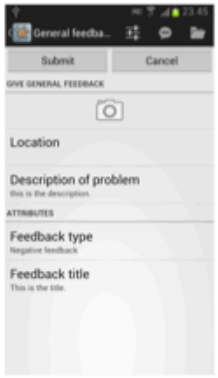
[5] Screenshot of Citizens connect Boston

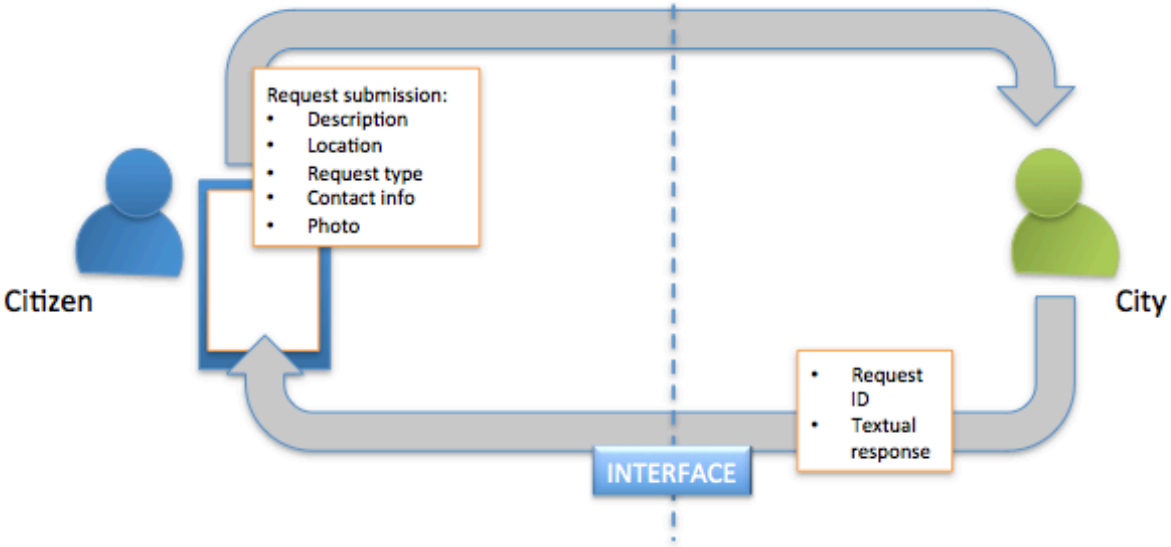
3.2.3 Adding a picture and writing a description

3. Tim takes a **picture** of the hole with his phone...

4. and **finishes** the report by filling in information:

- Title of the feedback
- Description (the problem explained in words)
- Location (point on Google map)
- Request type (category)
- Contact info (name, e-mail, phone)
- Photo of the issue





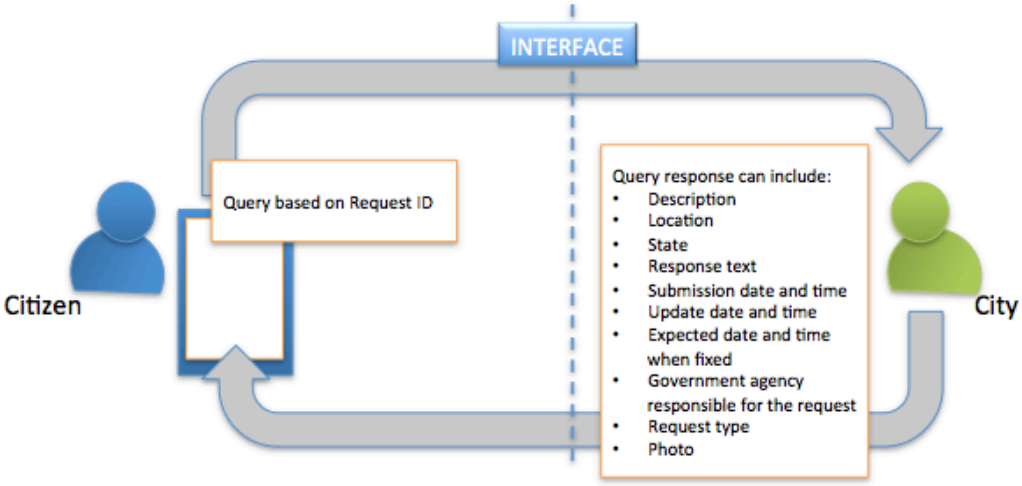
Smart Participation interface sets constraints to the issue reporting interface and form for some parts. Request types (in the image “Category”) are fetched from the interface and are defined separately for each interface instance by the interface provider.
Content required from the user submitting the request often includes (mandatory ones marked with *):

- Description*
- Location
- Request type*
- Contact info*
- Photo

3.2.4 Submitting the request and following up on it

5. Tim's report is **automatically forwarded** to city's feedback system
- City will respond to Tim with
  - Personal request ID for the report
  - Textual response: "The city of Helsinki has received your report – thank you"

6. Tim gets an automatic answer when the report is
  - Processed to another department or civil servant
  - Fixed
  - Will not be fixed



The Smart Participation interface allows users to submit their issue reports in the given format. After issue report is submitted, the interface provides the user with:
- Request ID
- Textual response

With request IDs users are able to follow the status of their issue report (request). User is also able to make queries using the ID. The response to this query can include:
- Description
- Location
- State
- Response text
- Submission date and time
- Update date and time
- Expected date and time when fixed
- Government agency responsible for the request
- Request type
- Photo

3.3 Future enhancements to the interface

CitySDK project partners voted on the new features, which they would like to see implemented in the service. Defining solutions for this is a work in progress and will be lead by the Replication Pilot in need for the feature in question.

Mobility of the user between cities

It should be possible that the user of one application can use the same application when moving to other city. This is especially useful on metropolitan areas where people move a lot between cities. Users don't even always know where they are at the point they reporting an issue and using the applications. No solution is yet defined, but there are efforts like GeoWeb DNS, which may provide the solution.

Commenting on an issue

During the requirement specification work it became clear that for some pilots comments or other updates are seen crucial for enabling collaboration around issue reporting without generating duplicate reports. Handling comments would require new method for sending comments and linking them to service requests.

Giving votes for issues user finds important

User should be able to vote for issues that he/she finds important and would like the city to fix. Issues like identifying users and checking whether they have already voted for each service requests should be addressed.

Notification of updates on request

The interface to city's system should be defined in such a way that clients can subscribe to changes of requests via the interface. For example, when the status, response text, agency handling or other important value of the request changes. There are alternative ways to achieve this requirement like updating status changes via e-mail address if the user has provided it. These alternatives do not require complex interface functionality.

Editing and removing submitted requests

Sender of request should be able to edit and remove requests after they have been sent to service.

More flebility on defining locations

It is now possible to have only point locations related to requests. It should be possible to have more flexibility defining location of the requests like line and polygons.

Integrating the feedback channels to users' city accounts

User should be able to submit request and be able to authenticate with an account provided by the city.

4 Standards for Smart Participation

4.1 Open311 (Geo Report v2)

GeoReport version 2 or better known as Open311 specification is mostly used in cities in USA. Open311 standard is especially used for reporting issues to city's feedback systems. It is implemented by 35 cities so far. The latest specification version 2 was released 2011.

Reasons for choosing Open311

The number of implementations and cities involved has shown that the interface is mature. There are several service providers and applications, which guarantee that there are alternatives for cities to choose from if they plan to have Open311 endpoint. Existing open source libraries and developer community is also a good building ground for new solutions and innovations.

Possible risks related to Open311

CitySDK members try collaborate and develop the interface together with the Open311 community. When more parties are involved in developing an interface standard, it may become slower. It's not always clear how the rough consensus can work and can achieve goals especially in a community, which lacks clear model of operation and leadership.

4.2 Other available standards

There are other standards for reporting problems and tracking them, e.g., Trouble Ticket standard. These can be quite general standards and therefore more complex than Open311. There are also open source reporting platforms like Ushadidi platform, which are used on similar things like reporting issues on a certain location. These are mostly focusing on the platform part and not so much on defining an interface to others to implement on.

5 CitySDK Smart Participation interface specification

5.1 General

This document defines the interface for CitySDK Smart Participation Work Package WP3. The interface is specified based on commonly used GeoReporting version 2, which is better known as Open311 specification [http://open311.org/]. The interface is designed in such a way that any GeoReporting version 2 compatible client is able to use the interface.

In addition to the version standard new fields have been added to API methods which have been indicated in specification CitySDK specific. CitySDK is collaborating with Open311 community with the aim to identify if the currently non-standard features could be included into the future version of the Open311 standard.

5.2 Architecture

Each API endpoint may be connected to several jurisdictions. In order to distinguish multiple jurisdictions within one API endpoint, "jurisdiction_id" is a unique identifier included into every request.

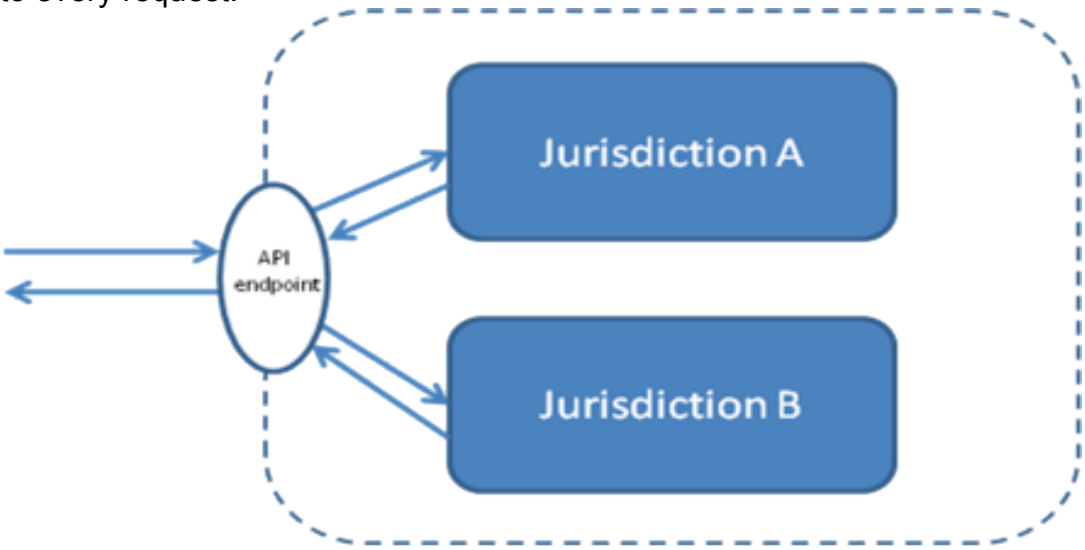


Image 1. Overall architecture

It has been recommended that the "jurisdiction_id“ is the jurisdiction's main website root URL without the www. For Helsinki, the jurisdiction_id is “hel.fi”. Implementations can ignore this parameter and treat it as an "Optional Argument" if the implementation only serves one jurisdiction.

In a typical scenario each jurisdiction, i.e. city, has one API endpoint, which provides all services.

Services provided by the interface are:

1. Listing of service request types and definitions of the meta information i.e. additional attributes
2. Submitting service requests[6]
3. Quering individual or multiple service requests and their descriptions and status

[6] Service request is an issue such as pothole which is reported to the municipality

5.3 Formats

XML is a required format. JSON can be also provided the API provider. The output formats supported by the provider are indicated through the Service Discovery formats field for the API endpoint being used.

5.4 Service Discovery

Service Discovery as defined in Open311 (http://wiki.open311.org/Service_Discovery) contains following required fields:

- **changeset** - String - Sortable field that specifies the last time this document was updated.
- **contact** - String - Human readable information on how to get more information on this provider.
- **key_service** - String - Human readable information on how to get an API key.
- **endpoints** - Array - Data structure holding the endpoints supported by this provider
- **endpoints.endpoint** - Hash - Data structure holding the metadata for each endpoint supported by this provider
- **endpoints.endpoint.specification** - String - The token of the service specification that is supported. This token will be defined by each spec. In general the format is a URL that identifies the specification and version number much like an XMLNS declaration. (eg http://wiki.open311.org/GeoReport_v2)
- **endpoints.endpoint.url** - String - URL of the endpoint provider
- **endpoints.endpoint.changeset** - String - Sortable field that specifies the last time this document was updated.
- **endpoints.endpoint.type** - String - Either "production" or "test" defines whether the information is live and will be acted upon.
- **endpoints.endpoint.formats** - Array - Data structure of supported MIME types.
- **endpoints.endpoint.formats.format** - String - Supported MIME type for this endpoint.

In addition to this, new fields for language support is added:

- **endpoints.endpoint.locales** - Array - Data structure of supported language.
- **endpoints.endpoint.locales.locale** - String - Supported language as locale.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<discovery>
  <changeset>2011-02-03 14:18</changeset>
  <contact>Email or call for assistance api@mycity.eu</contact>
  <key_service>Request a key: http://api.mycity.eu/api_key/request</key_service>
    <endpoints>
        <endpoint>
          <specification>http://wiki.open311.org/GeoReport_v2</specification>
          <url>http://open311.mycity.gov/v2</url>
          <changeset>2010-11-23 09:01</changeset>
          <type>production</type>
          <formats>
            <format>text/xml</format>
          </formats>
```

```
            <locales>
              <locale>fi_FI</format>
                      <locale>sv_FI</format>
                      <locale>sv_SE</format>
                      <locale>en_US</format>
                     </locales>
        </endpoint>
      <endpoints>
</discovery>
```

5.5 API methods

5.5.1 GET Service List

Purpose	Provides a list of acceptable service request types and their associated service codes. These request types can be unique to the city/jurisdiction.
URL	https://[API endpoint]/services.[format]
Sample URL	https://api.hel.fi/services.xml
Formats	XML and JSON
Requires API key	No
HTTP Method	GET

Required Parameters

Parameter name	Description	Required	Notes
jurisdiction_id	Unique id for the jurisdiction, i.e. city. For example jurisdiction_id for Helsinki is “hel.fi”.	No	This is only required if the endpoint serves multiple jurisdictions.
locale CitySDK specific	Preferred language	No	Locales can be, for example, fi_FI, or en_US. Default values depends on the endpoint. See also chapter on Language support.

Response

Parameter name	Description	Required	Notes
service_code	The unique identifier for the service request type	Yes	
service_name	The human readable name of the service request type	Yes	
description	A brief description of the service request type.	Yes	
metadata	Determines whether their are additional form fields for this service type. • true: This service request type requires additional metadata so the client will need to make a call to the Service Definition method. • false: No additional information is required and a call to the Service Definition method is not needed.	Yes	
type	• realtime: The service request ID will be returned immediately after	Yes	

	the service request is submitted. • batch: A token will be returned immediately after the service request is submitted. This token can then be later used to return the service request ID. • blackbox: No service request ID will be returned after the service request is submitted		
keywords	A comma separated list of tags or keywords to help users identify the request type. This can provide synonyms of the service_name and group.	Yes	
group	A category to group this service type within. This provides a way to group several service request types under one category such as "sanitation"	Yes	

Possible errors

The numbers represent the HTTP status code returned for each error type:

- 404 - service_code or jurisdiction_id was not found (specified in error response)
- 400 - service_code or jurisdiction_id was not provided (specified in error response)
- 400 - General Service Error (Any failure during create request processing, eg CRM is down. Client will need to notify us)

Example Request

https://api.hel.fi/services.xml

Example Response

```
<?xml version="1.0" encoding="utf-8"?>
<services>
        <service>
                <service_code>001</service_code>
                <service_name>Street lights</service_name>
                <description> Report if street lights are out of order.</description>
                <metadata>false</metadata>
                <type>realtime</type>
                <keywords>streetlight</keywords>
                <group>Streets and Sanitation</group>
        </service>
        <service>
                <service_code>002</service_code>
                <service_name>Potholes in street</service_name>
                <description> Report Potholes in public streets</description>
                <metadata>false</metadata>
                <type>realtime</type>
```

```
                <keywords>pothole</keywords>
                <group>Transportation</group>
        </service>
        <service>
                <service_code>003</service_code>
                <service_name>General feedback</service_name>
                <description>Give general feedback</description>
                <metadata>true</metadata>
                <type>realtime</type>
                <keywords>general, feedback</keywords>
                <group>General</group>
        </service>
</services>
```

5.6 GET Service Definition

Purpose	Define attributes associated with a service code. These attributes can be unique to the city/jurisdiction.
URL	https://[API endpoint]/services/[service_code].[format]
Sample URL	https://api.hel.fi/services/033.xml
Formats	XML and JSON
Requires API key	No
HTTP Method	GET

Required Parameters

Parameter name	Description	Required	Notes
jurisdiction_id	Unique id for the jurisdiction, i.e. city. For example jurisdiction_id for Helsinki is “hel.fi”.	No	This is only required if the endpoint serves multiple jurisdictions.
service_code	The unique identifier for the service request type.	Yes	The service_code is specified in the main URL path rather than an added query string parameter.
locale CitySDK specific	Preferred language	No	Locales can be, for example, fi_FI, sv_SE, en_US tai en_GB. Default values depends on the endpoint. See also Language support.

Response

Parameter name	Description	Required	Notes
service_definition ⤵			
service_code	Returns the service_code associated with the definition, the same one submitted for this call.	Yes	
attributes ⇊			

attribute ⤵			
variable	• **true** denotes that user input is needed • **false** means the attribute is only used to present information to the user within the description field	Yes	
code	A unique identifier for the attribute	Yes	
datatype	Denotes the type of field used for user input. • **string**: A string of characters without line breaks. Represented in an HTML from using an <input> tag • **number**: A numeric value. Represented in an HTML from using an <input> tag • **datetime**: The input generated must be able to transform into a valid ISO 8601 date. Represented in an HTML from using <input> tags • **text**: A string of characters that may contain line breaks. Represented in an HTML from using an <textarea> tag • **singlevaluelist**: A set of predefined values (specified in this response) where only one value may be selected. Represented in an HTML from using the <select> and <option> tags • **multivaluelist**: A set of predefined values (specified in this response) where several values may be selected. Represented in an HTML from using the <select multiple="multiple"> and <option> tags	Yes	
required	• **true** means that the value is required to submit service request • **false** means that the value	Yes	

	not required.		
data_description	A description of the datatype which helps the user provide their input	Yes	
order	The sort order that the attributes will be presented to the user. 1 is shown first in the list.	No	
description	An description of the attribute field with instructions for the user to find and identify the requested information	Yes	
values ⇊			
value ⤵			
key	The unique identifier associated with an option for singlevaluelist or multivaluelist. This is analogous to the value attribute in an html option tag.	Yes	
name	The human readable title of an option for singlevaluelist or multivaluelist. This is analogous to the innerhtml text node of an html option tag.	Yes	

5.6.1 Possible errors

The numbers represent the HTTP status code returned for each error type:

- 404 - service_code or jurisdiction_id was not found (specified in error response)
- 400 - service_code or jurisdiction_id was not provided (specified in error response)
- 400 - General Service Error (Any failure during create request processing, eg CRM is down. Client will need to notify us)

Example Request

https://api.hel.fi/services/003.xml

Example Response

```
<service_definition>
	<service_code>003</service_code>
	<attributes>
		<attribute>
			<variable>true</variable>
			<code>service_request_type</code>
			<datatype>singlevaluelist</datatype>
			<required>true</required>
			<datatype_description></datatype_description>
			<order>1</order>
			<description>Feedback type</description>
			<values>
				<value>
					<key>IDEA</key>
					<name>Idea</name>
				</value>
				<value>
					<key>THANK</key>
					<name>Positive feedback</name>
				</value>
				<value>
					<key>BLAME</key>
					<name>Negative feedback</name>
				</value>
				<value>
					<key>QUESTION</key>
					<name>Question</name>
				</value>
				<value>
					<key>OTHER</key>
					<name>Other</name>
				</value>

			</values>
		</attribute>
		<attribute>
			<variable>true</variable>
			<code>title</code>
			<datatype>string</datatype>
			<required>false</required>
			<datatype_description></datatype_description>
			<order>2</order>
			<description>Feedback title</description>
		</attribute>
	</attributes>
</service_definition>
```

5.6.2 POST Service Request

Purpose	Create service requests
URL	https://[API endpoint]/requests.[format]
Sample URL	https://api.hel.fi/requests.json
Posting Formats	Content-Type: application/x-www-form-urlencoded tai multipart/form-data (ks. media parametri)
Formats	XML and JSON
Requires API key	Yes
HTTP Method	POST

Required Parameters

Parameter name	Description	Required	Notes
api_key		Yes	
jurisdiction_id	Unique id for the jurisdiction, i.e. city. For example jurisdiction_id for Helsinki is “hel.fi”.	No	This is only required if the endpoint serves multiple jurisdictions.
locale CitySDK specific	Preferred language	No	Locales can be, for example, fi_FI, sv_SE, en_US tai en_GB. Default values depends on the endpoint. See also Language support.
service_code	The unique identifier for the service request type	Yes	
attribute	An array of key/value responses based on Service Definitions.	Yes	Yes, if defined in the Service Definitions
lat CitySDK specific	Latitude using the (WGS84) projection.	No	It is not mandatory to have location, which differs from Open v2. Each city must define in their specification if it is possible to send request without for certain service request types (see Extensions and city specific parameters).
long CitySDK specific	Longitude using the (WGS84) projection.	No	It is not mandatory to have location, which differs from Open v2.
address_string	Human entered address or description of location.	No	
address_id	he internal address ID used by a jurisdiction's master address repository or other addressing system.	No	
email	The email address of the person	No	

	submitting the request		
device_id	The unique device ID of the device submitting the request. This is usually only used for mobile devices.	No	
account_id	The unique ID for the user account of the person submitting the request	No	
first_name	The given name of the person submitting the request	No	
last_name	The family name of the person submitting the request	No	
phone	The phone number of the person submitting the request	No	
description	A full description of the request or report being submitted.	No	This may contain line breaks, but not html or code. Otherwise, this is free form text limited to 4,000 characters.
media_url	A URL to media associated with the request, e.g. an image.	No	
media CitySDK specific	An array of file uploads	No	A client may POST multiple files as multipart/form-data. This is the equivalent of having multiple <input type="file" name="media[]" /> inputs. Subsequent calls for GET Service Requests should return the URL for this file via the media_urls field. This means that a client can not simultaneously post a URL for media_url and post files via media. If this is done, the files uploaded via the media field will take precedence and be returned as URLs in the media_url field. This planned for Open311 version 2.1.
service_object_type CitySDK specific	Describes the point of interest reference which is used for identifying the request object.	No	See Chapter Service Objects.
service_object_id CitySDK specific	Describes the ID of the service object	No	See Chapter Service Objects. If service_object_id is included in the request, then service_object_type must be included.

service_object_base CitySDK specific	Describes the endpoint base of the service object data source.	No	See Chapter Service Objects. If service object is defined then, then it is recommended the service_object_base is included in the message.

Response

Parameter name	Description	Required	Notes
service_request_id	The unique ID of the service request created.	No	This should not be returned if token is returned.
token_id	If returned, use this to call GET request_id from a token.	No	This should not be returned if service_request_id is returned
service_notice	Information about the action expected to fulfill the request or otherwise address the information reported.	No	
account_id	The unique ID for the user account of the person submitting the request.	No	

Possible errors

The numbers represent the HTTP status code returned for each error type:

- 404 - service_code or jurisdiction_id was not found (specified in error response)
- 400 - service_code or jurisdiction_id was not provided (specified in error response)
- 400 - General Service Error (Any failure during create request processing, eg CRM is down. Client will need to notify us)

Example Request

```
POST /dev/v2/requests.xml
Host: api.hel.fi
Content-Type: application/x-www-form-urlencoded; charset=utf-8

api_key=xyz&service_code=001&lat=37.76524078&long=122.4212043&address_string=1234+5th+street&email=smit333%40sfgov.edu&device_id=tt222111&account_id=123456&first_name=john&last_name=smith&phone=111111111&description=A+large+sinkhole+is+destroying+the+street&media_url=http%3A%2F%2Ffarm3.static.flickr.com%2F2002%2F2212426634_5ed477a060.jpg
```

Example Response

```
<?xml version="1.0" encoding="utf-8"?>
<service_requests>
        <request>
                <service_request_id>293944</service_request_id>
                <service_notice>Thank you for your issue-report!</service_notice>
                <account_id/>
        </request>
```

</service_requests>

5.6.3 GET Service Requests

Purpose	Query the current status of multiple requests.
URL	https://[API endpoint]/requests.[format]
Sample URL	https://api.hel.fi/requests.xml?start_date=2010-05-24T00:00:00Z&end_date=2010-06-24T00:00:00Z&status=open
Formats	XML and JSON
Requires API key	No
HTTP Method	GET

Required Parameters

Parameter name	Description	Required	Notes
jurisdiction_id	Unique id for the jurisdiction, i.e. city. For example jurisdiction_id for Helsinki is “hel.fi”.	No	This is only required if the endpoint serves multiple jurisdictions.
locale CitySDK specific	Preferred language	No	Locales can be, for example, fi_FI, sv_SE, en_US tai en_GB. Default values depends on the endpoint. See also Language support.
service_request_id	To call multiple Service Requests at once, multiple service_request_id can be declared; comma delimited.	No	This overrides all other arguments.
service_code	Specify the service type by calling the unique ID of the service_code.	No	This defaults to all service codes when not declared; can be declared multiple times, comma delimited
start_date	Earliest requested_datetime to include in search. When provided with end_date, allows one to search for requests which have a requested_datetime that matches a given range, but may not span more than 90 days..	No	Must use w3 format, e.g 2010-01-01T00:00:00Z.
end_date	Latest requested_datetime to include in search. When provided with start_date, allows one to search for requests which have a requested_datetime that matches	No	Must use w3 format, e.g 2010-01-01T00:00:00Z.

	a given range, but may not span more than 90 days.		
updated_after CitySDK specific	Earliest updated_datetime to include in search. Allows one to search for requests which have an updated_datetime between the updated_after time and updated_before time (or now). This is useful for downloading a changeset that includes changes to older requests or to just query very recent changes.		Must use w3 format, e.g 2010-01-01T00:00:00Z.
updated_before CitySDK specific	Latest updated_datetime to include in search. Allows one to search for requests which have an updated_datetime between the updated_after time and the updated_before time. This is useful for downloading a changeset that includes changes to older requests or to just query very recent changes.		When not specified (updated_after is used without updated_before) then updated_before is assumed to be now. Must use w3 format, eg 2010-01-01T00:00:00Z.
status	Allows one to search for requests which have a specific status. This defaults to all statuses; can be declared multiple times, comma delimited.	No	
extensions CitySDK specific	The endpoint provides supplemental details about service requests that are in addition to the ones described in the standard specification. These data are nested in the 'extended_attributes' field in the Service Request response. In order to retrieve the new supplemental details, add the query parameter “extensions=true” to the request	No	
page_size CitySDK specific	Controls the maximum amount of results a single call will return. The default value is 50. The maximum value is 500.	No	
page CitySDK specific	For calls that logically include more records than the page size, the page parameter can be use to page through the results. Use in combination with page_size and with multiple calls to download all data in a logical set of records.	No	
lat	Defines latitude of the location	No	When search is performed on

CitySDK specific	where search is performed.		certain location then both lat and long must be present and radius is optional.
long CitySDK specific	Defines longitude of the location where search is performed.	No	When search is performed on certain location then at least lat and long must be present and radius is optional.
radius CitySDK specific	Defines radius of the location where search is performed.	No	radius is optional if lat and long parameters are given. However, it is recommended that value is give to radiusgiven, if not then server sends
service_object_type CitySDK specific	Describes the point of interest reference which is used for identifying the request object.	No	See Chapter Service Objects.
service_object_id CitySDK specific	Describes the ID of the service object	No	See Chapter Service Objects. If service_object_id is included in the request, then service_object_type must be included.
service_object_base CitySDK specific	Describes the endpoint base of the service object data source.	No	See Chapter Service Objects. If service object is defined then, then it is recommended the service_object_base is included in the response.

Response

Parameter name	Description	Required	Notes
service_request_id	The unique ID of the service request created.	Yes	
status	The current status of the service request. • open: it has been reported. • closed: it has been resolved.	Yes	
status_notes	Explanation of why status was changed to current state or more details on current status than conveyed with status alone.	No	
service_name	The human readable name of	Yes	

	the service request type		
service_code	The unique identifier for the service request type	Yes	
description	A full description of the request or report submitted.	Yes	This may contain line breaks, but not html or code. Otherwise, this is free form text limited to 4,000 characters.
agency_responsible	The agency responsible for fulfilling or otherwise addressing the service request.	No	
service_notice	Information about the action expected to fulfill the request or otherwise address the information reported.	No	
requested_datetime	The date and time when the service request was made.	Yes	Returned in w3 format, eg 2010-01-01T00:00:00Z
updated_datetime	The date and time when the service request was last modified. For requests with status=closed, this will be the date the request was closed.	No	Returned in w3 format, eg 2010-01-01T00:00:00Z
expected_datetime	The date and time when the service request can be expected to be fulfilled. This may be based on a service-specific service level agreement.	No	Returned in w3 format, eg 2010-01-01T00:00:00Z May not be returned
address	Human readable address or description of location.	No	
address_id	The internal address ID used by a jurisdictions master address repository or other addressing system.	No	
zipcode	The postal code for the location of the service request.	No	
lat CitySDK specific	latitude using the (WGS84) projection.	No	It is not mandatory to have location, which differs from Open v2.
long CitySDK specific	longitude using the (WGS84) projection.	No	It is not mandatory to have location, which differs from Open v2
media_url CitySDK specific	A URL to media associated with the request, eg an image.	No	May contain more than one of these elements
service_object_type	Nested in extended_attributes field	No	See Chapter Service Objects.

CitySDK specific	(extended_attributes.service_object_type). Describes the point of interest reference which is used for identifying the request object.		
service_object_id CitySDK specific	Nested in extended_attributes field (extended_attributes service_object_id). Describes the ID of the service object.	No	See Chapter Service Objects. If service_object_id is included in the response, then service_object_type must be included.
service_object_base CitySDK specific	Describes the endpoint base of the service object data source.	No	See Chapter Service Objects. If service object is defined then, then it is recommended the service_object_base is included in the response.

Response Volume

Default query limit is a span of 90 days or first 1000 requests returned, whichever is smallest.

Possible errors

The numbers represent the HTTP status code returned for each error type:

- 404 - service_code or jurisdiction_id was not found (specified in error response)
- 400 - service_code or jurisdiction_id was not provided (specified in error response)
- 400 - General Service Error (Any failure during create request processing, eg CRM is down. Client will need to notify us)

Example Request

https://api.hel.fi/dev/v2/requests.xml?start_date=2012-05-24T00:00:00Z&end_date=2012-06-24T00:00:00Z&status=open,close

Example Response

```
<?xml version="1.0" encoding="utf-8"?>
<service_requests>
        <request>
                <service_request_id>638344</service_request_id>
                <status>open</status>
                <status_notes>IN_PROCESS</status_notes>
                <service_name>Vandalism</service_name>
                <service_code>001</service_code>
                <description>Trash bin broken.</description>
                <agency_responsible>Public Works Department</agency_responsible>
                <service_notice>Repair has been ordered.</service_notice>
                <requested_datetime>2012-05-26T06:37:38-08:00</requested_datetime>
```

```
                <updated_datetime>2012-05-26T06:37:38-08:00</updated_datetime>
                <expected_datetime>2012-05-28T06:37:38-08:00</expected_datetime>
                <address>Aleksanterinkatu 16-18</address>
                <zipcode>00100</zipcode>
                <lat>60.168569</lat>
                <long24.950627</long>
                <media_url>http://images.hel.fi/requests/media/638344.jpg</media_url>
        </request>
        <request>
                <service_request_id>638349</service_request_id>
                <status>open</status>
                <status_notes>RECEIVED</status_notes>
                <service_name>Street conditions</service_name>
                <service_code>003</service_code>
                <description>There is a pothole.</description>
                <agency_responsible>Public Works Department</agency_responsible>
                <service_notice></service_notice>
                <requested_datetime>2012-06-12T06:37:38-08:00</requested_datetime>
                <updated_datetime>2012-06-12T06:37:38-08:00</updated_datetime>
                <expected_datetime>2012-06-12T06:37:38-08:00</expected_datetime>
                <address>Unioninkatu 8</address>
                <zipcode>00100</zipcode>
                <lat>60.168569</lat>
                <long>24.950627</long>
                <media_url>http://images.hel.fi/requests/media/638349.jpg</media_url>
        </request>
</service_requests>
```

5.6.4 GET Service Request

Purpose	Query the current status of an individual request.
URL	https://[API endpoint]/requests/[service_request_id].[format]
Sample URL	https://api.hel.fi/requests/123456.xml?jurisdiction_id=hel.fi
Formats	XML and JSON
Requires API key	No
HTTP Method	GET

Required Parameters

Parameter name	Description	Required	Notes
jurisdiction_id	Unique id for the jurisdiction, i.e. city. For example jurisdiction_id for Helsinki is “hel.fi”.	No	This is only required if the endpoint serves multiple jurisdictions.
service_request_id		Yes	The service_request_id is specified in the main URL path rather than an added query string parameter.
locale	Preferred language	No	Locales can be, for example, fi_FI, sv_SE,

CitySDK specific			en_US tai en_GB. Default values depends on the endpoint. See also Language support.
extensions CitySDK specific	The endpoint provides supplemental details about service requests that are in addition to the ones described in the standard specification. These data are nested in the 'extended_attributes' field in the Service Request response. In order to retrieve the new supplemental details, add the query parameter “extensions=true” to any Open 311 API request.	No	Options: **true**, **false**.

Response

Parameter name	Description	Required	Notes
service_request_id		Yes	
status	The current status of the service request. • open: it has been reported. • closed: it has been resolved.	Yes	See also Status of service request
status_notes	Explanation of why status was changed to current state or more details on current status than conveyed with status alone.	Yes	
service_name	The human readable name of the service request type	Yes	
service_code	The unique identifier for the service request type	Yes	
description	A full description of the request or report submitted.	Yes	This may contain line breaks, but not html or code. Otherwise, this is free form text limited to 4,000 characters.
agency_responsible	The agency responsible for fulfilling or otherwise addressing the service request.	Yes	
service_notice	Information about the action expected to fulfill the request or otherwise address the	Yes	

	information reported.		
requested_datetime	The date and time when the service request was made.	Yes	Returned in w3 format, eg 2010-01-01T00:00:00Z
updated_datetime	The date and time when the service request was last modified. For requests with status=closed, this will be the date the request was closed.	Yes	Returned in w3 format, eg 2010-01-01T00:00:00Z
expected_datetime	The date and time when the service request can be expected to be fulfilled. This may be based on a service-specific service level agreement.	No	Returned in w3 format, eg 2010-01-01T00:00:00Z May not be returned
address	Human readable address or description of location.	No	
address_id	The internal address ID used by a jurisdictions master address repository or other addressing system.	No	
zipcode	The postal code for the location of the service request.	No	
lat	latitude using the (WGS84) projection.	Yes	
long	longitude using the (WGS84) projection.	Yes	
media_url	A URL to media associated with the request, eg an image.	No	May contain more than one of these elements
service_object_type CitySDK specific	Nested in extended_attributes field (extended_attributes.service_object_type). Describes the object or point of interest reference which is used e.g. ”http://www.hel.fi/palvelukarttaws/rest/ver2_en.html”	No	See Chapter Service Objects.
service_object_id CitySDK specific	Nested in extended_attributes field (extended_attributes.service_object_id) describes the ID of the object e.g. public toiled in Helsinki Esplanade park ID = 10844	No	If service_object_id is included, then service_object_type must be also included.
service_object_base CitySDK specific	Describes the endpoint base of the service object data source.	No	See Chapter Service Objects. If service object is defined then, then it is recommended the service_object_base is included in the response.

detailed_status CitySDK specific	Nested in extended_attributes field (extended_attributes. detailed_status) describes detailed status of the request status.	No	This can contain multiple status values; comma delimited. See more on Status of service requests.

Possible errors

The numbers represent the HTTP status code returned for each error type:

- 404 - service_code or jurisdiction_id was not found (specified in error response)
- 400 - service_code or jurisdiction_id was not provided (specified in error response)
- 400 - General Service Error (Any failure during create request processing, eg CRM is down. Client will need to notify us)

Example Request

https://api.city.gov/dev/v2/requests/638344.xml

Example Response

```
<?xml version="1.0" encoding="utf-8"?>
<service_requests>
        <request>
                <service_request_id>638344</service_request_id>
                <status>closed</status>
                <status_notes>Duplicate request.</status_notes>
                <service_name>Sidewalk and Curb Issues</service_name>
                <service_code>001</service_code>
                <description></description>
                <agency_responsible></agency_responsible>
                <service_notice></service_notice>
                <requested_datetime>2010-04-14T06:37:38-08:00</requested_datetime>
                <updated_datetime>2010-04-14T06:37:38-08:00</updated_datetime>
                <expected_datetime>2010-04-15T06:37:38-08:00</expected_datetime>
                <address>8TH AVE and JUDAH ST</address>
                <zipcode>94122</zipcode>
                <lat>60.168569</lat>
                <long>24.950627</long>
                <media_url>http://images.hel.fi/requests/media/638344.jpg</media_url>
        </request>
</service_requests>
```

5.6.5 Error messages

Parameter name	Description	Required	Notes
code	The error code representing the type	Yes	

	of error. In most cases, this should match the HTTP status code returned in the HTTP header.		
description	A human readable description of the error that occurred. This is meant to be seen by the user.	Yes	

General error:

- 403 – Missing or Invalid API Key (specify in error message)
- 400 – The URL request is invalid or open311 service is not running or reachable. Client should notify us after checking URL

Example Error

HTTP/1.1 403 Forbidden

```
<?xml version="1.0" encoding="utf-8"?>
<errors>
  <error>
    <code>403</code>
    <description>Missing API key, the request cannot be stored.</description>
  </error>
</errors>
```

5.7 New parameter definitions

5.7.1 Extensions and city specific parameters

The endpoint may provide supplemental details about service requests that are in addition to the ones described in the Open311 version 2. These parameters are nested in the 'extended_attributes' field in the service request response. In order to retrieve the new supplemental details, the query parameter “extensions=true” must be set on the request.

This specification specifies new parameters service_object_type and service_object_type in chapter Service objects and detailed_status in chapter Status of service requests, which may be provided by the endpoint. These parameters are returned only if extensions is set “true” and nested in the 'extended_attributes' field.

Cities can specify other parameters using the extensions mechanism. These parameters are not part of this specification, but city’s own specification. For example in Helsinki these parameters are likely specified in addition.

- attributes defined for some service request types may be returned in response. For example service_request_type,title attributes
- service_request_url - URL address of the service request in city’s web page, e.g. ”http://hel.fi/request/1234567”
- external_url – URL address of the service request in the external service if it exists, e.g. ” http://metro.fi/feedback/0987654”

The extension parameter is already used in Chicago API endpoint.

5.7.2 Status of service requests (detailed_status)

The status of each service is transferred in status parameter. Status can have only values “open” or “closed”. If more detailed level of status information needs to be transferred detailed_status parameter is defined via the extension mechanism (see more Extensions).

For interoperability purpose set of detailed status values are specified:

- RECEIVED – service request is received but nothing is done it. Status field value is open.
- IN_PROCESS - service request is received and its handling has been started. Status field value is open.
- PROCESSED - service request has been resolved. Status field value is closed.
- ARCHIVED - service request has been resolved and archived. Status field value is closed.
- REJECTED -service request has been rejected. Status field value is closed.

Different city systems may be able to provide more detailed and city specific status values. Therefore, detailed_status extension parameter or status_notes field can have multiple status values, which are comma separated. For example, detailed_status is “IN_PROCESS,PUBLIC_WORKS_HANDLING”.

In Helsinki pilot, there will be 3 additional values ASPA_AVOIN, ASPA_KÄSITTELYSSÄ and ASPA_VALMIS, which are directly auto-generated from the public works department system states.

5.7.3 Service Objects (service_object_type,service_object_id, service_object_base)

Service request may contain reference to a service object. Service object is typically an item from a data source like a city’s point-of-interest database . Service object can be, for example, a specific bus stop, toilet, museum, streetlight, which the feedback sent on the service request is about.

In order to separate different data sources, service_object_type must be present is if service object is defined, For example in Helsinki, it is possible to use City of Helsinki service map as the data source for the service objects, which is identified by service_object_type=http://www.hel.fi/servicemap/v2/unit/. Other service object types identified so far are “citysdk.eu/v1/poi” and “citysdk.eu/v1/mobility”, which refer to WP4 Mobility or WP4 Tourism specifications.

The actual service object is identified by using service_object_id parameter. It must be always present if service object is defined. For example in case of Helsinki Service map, service_object_id=10844 identifies the public toilet in Esplanade park.

It is recommended that a base URL of the endpoint is included into the service object definition. The service_object_base parameter is used for that. In case of CitySDK interfaces, it is possible that applications implementing WP4 Mobility or WP4 Tourism

specifications can fetch service object data from any CitySDK endpoint typically by appending service_object_id to service_object_base. For example, if service_object_base=http://api.citysdk.amsterdam.nl/cdk-node/ and service_object_id=cdk-n881955629 then URL address for fetching data is http://api.citysdk.amsterdam.nl/cdk-node/ cdk-n881955629.

Examples of usage

Public toilet in Helsinki service map:

- service_object_id=10844
- service_object_type=http://www.hel.fi/servicemap/v2/unit/

Point of interest as defined in Tourism Work Package:

- service_object_type=citysdk.eu/v1/poi
- service_object_base=http://www.lisboa.pt/citysdk_poi/
- service_object_id=1234

5.7.4 Language support (locale)

Language support is implemented by having an optional local – parameter into every request message. Supported languages are listed in the Service Discovery.

The locale –parameter may affect the language used inside following parameters:

API Method	Parameters which content can change depending on language
GET Service List	service_name description keywords group
GET Service Definition	data_description description name
POST Service Request	service_notice
Get Service Requests	status_notes service_name agency_responsible service_notice address
Get Service Request	status_notes service_name agency_responsible service_notice address
Errors – error message	Description

5.7.5 New query options

Quering based on time (updated_after,updated_before)

Allows one to search for requests which have an updated_datetime between the updated_after time and the updated_before time. This is useful for downloading a change set that includes changes to older requests or to just query very recent changes.

These parameters are already included in the draft Open311 v2.1.

Limiting query results (page,page_size)

These parameters allows to page through the results. Use in combination with page_size and with multiple calls to download all data in a logical set of records.

These parameters are already in use in some cities.

Quering based on location (lat, long, radius)

Allows one to search for requests submitted on certain area.

5.7.6 Media handling

Allows defining a standard way to send media to API endpoint.

This is already defined in the draft Open311 v2.1 and used in some cities.

5.7.7 Other changes to Open311 v2 specification

Location

It is possible to send service request without location. Developer should read city's specifications for guidance on, which service request types allows request without location. If there is no mention then standard Open311 v2 operation is assumed, i.e. location is mandatory.

5.8 Examples on API messages and usage

5.8.1 Submitting a service request with an image

```
POST /dev/v2/requests.json
Host: api.hel.fi
Content-Type: multipart/form-data; boundary=AaB03x

--AaB03x
Content-Disposition: form-data; name="api_key"

xyz
--AaB03x
Content-Disposition: form-data; name="service_code "

001
--AaB03x
Content-Disposition: form-data; name="lat"

60.168569
--AaB03x
Content-Disposition: form-data; name="lon"
```

```
24.950627
--AaB03x
Content-Disposition: form-data; name="description"

There is a huge pothole
--AaB03x
Content-Disposition: form-data; name="first_name "

John
--AaB03x
Content-Disposition: form-data; name="last_name "

Smith
--AaB03x
Content-Disposition: form-data; name="phone"

0503391387
--AaB03x
Content-Disposition: form-data; name="email"

john.smith@gmail.com
--AaB03x
Content-Disposition: file; filename="image.gif"
Content-Type: image/gif
Content-Transfer-Encoding: binary

...contents of image.gif...
--AaB03x
```

Response

```
[
  {
    "service_request_id":293944,
    "service_notice":"Thank you for the feedback!",
    "account_id":null
  }
]
```

5.8.2 Service request with image and attributes

```
POST /dev/v2/requests.json
Host: api.hel.fi
Content-Type: multipart/form-data; boundary=AaB03x

--AaB03x
Content-Disposition: form-data; name="api_key"

xyz
--AaB03x
Content-Disposition: form-data; name="service_code "

003
--AaB03x
Content-Disposition: form-data; name="attribute[service_request_type]"

THANK
```

```
--AaB03x
Content-Disposition: form-data; name="attribute[title]"

Thanks you for fabulous day!
--AaB03x
Content-Disposition: form-data; name="description"

Thank you for organizing a fabulous great Helsinki day on 12.6.!
--AaB03x
Content-Disposition: form-data; name="first_name"

Joan
--AaB03x
Content-Disposition: form-data; name="last_name"

Smith
--AaB03x
Content-Disposition: form-data; name="phone"

0503391387
--AaB03x
Content-Disposition: form-data; name="email"

joan.smith@gmail.com
--AaB03x
Content-Disposition: file; filename="image.gif"
Content-Type: image/gif
Content-Transfer-Encoding: binary

...contents of image.gif...
--AaB03x
```

Response

```
[
  {
    "service_request_id":293944,
    "service_notice":"Thanks for your feedback!"
  }
]
```

5.8.3 Quering service request with extensions

https://api.city.gov/dev/v2/requests/638344.xml?extensions=true

```
<?xml version="1.0" encoding="utf-8"?>
<service_requests>
  <request>
            <service_request_id>638344</service_request_id>
            <status>open</status>
            <status_notes></status_notes>
            <service_name>Public toilets</service_name>
            <service_code>001</service_code>
            <description>Toilet is a mess.</description>
            <agency_responsible></agency_responsible>
            <service_notice></service_notice>
```

```
        <requested_datetime>2010-04-14T06:37:38-08:00</requested_datetime>
        <updated_datetime>2010-04-14T06:37:38-08:00</updated_datetime>
        <expected_datetime>2010-04-15T06:37:38-08:00</expected_datetime>
        <address>8TH AVE and JUDAH ST</address>
        <zipcode>94122</zipcode>
        <lat>60.168569</lat>
        <long>24.950627</long>
        <media_url>http://images.hel.fi/requests/media/638344.jpg</media_url>
        <extended_attributes>
                <service_object_type>http://www.hel.fi/servicemap/v2 </service_object_type>
                <service_object_id>10844</service_object_id>
                <detailed_status>IN_PROCESS,PUBLIC_WORKS_OPEN</detailed_status>
        </extended_attributes>
  </request>
</service_requests>
```

5.8.4 Listing service definition with language set to Finnish language

This is shows example of how the Service Definition and language option affects the UI of the application. Image below shows the UI if Service List and Service Definition examples were used.

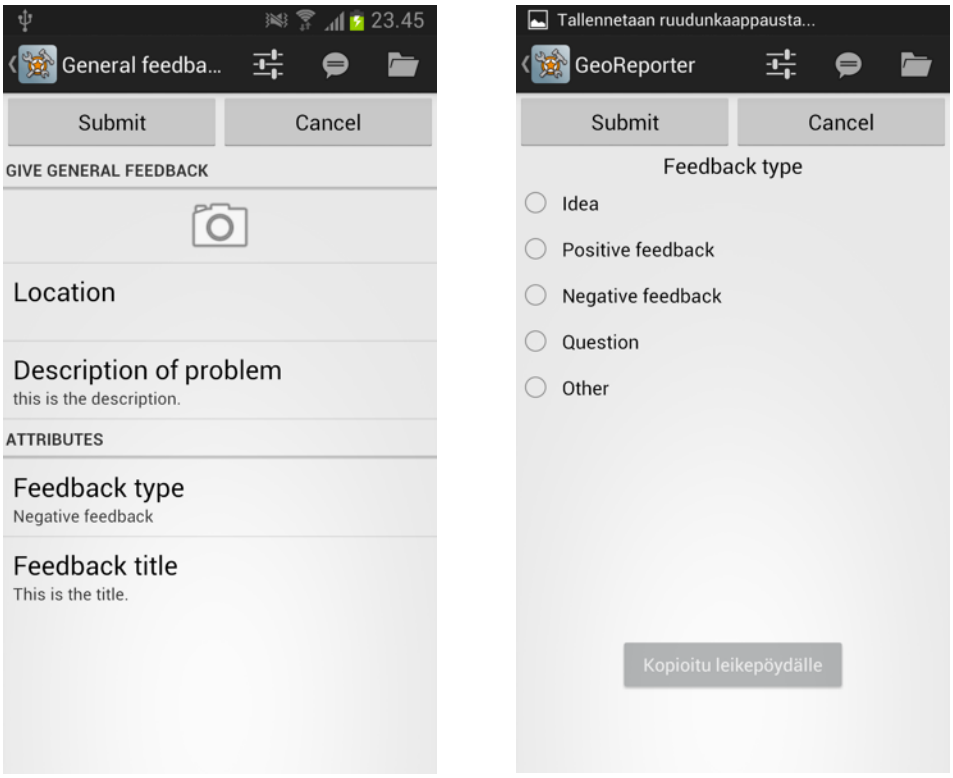


Image 2. Application UI

If the Service Definition example is loaded with language option set to Finnish the multivalue list option view would like.

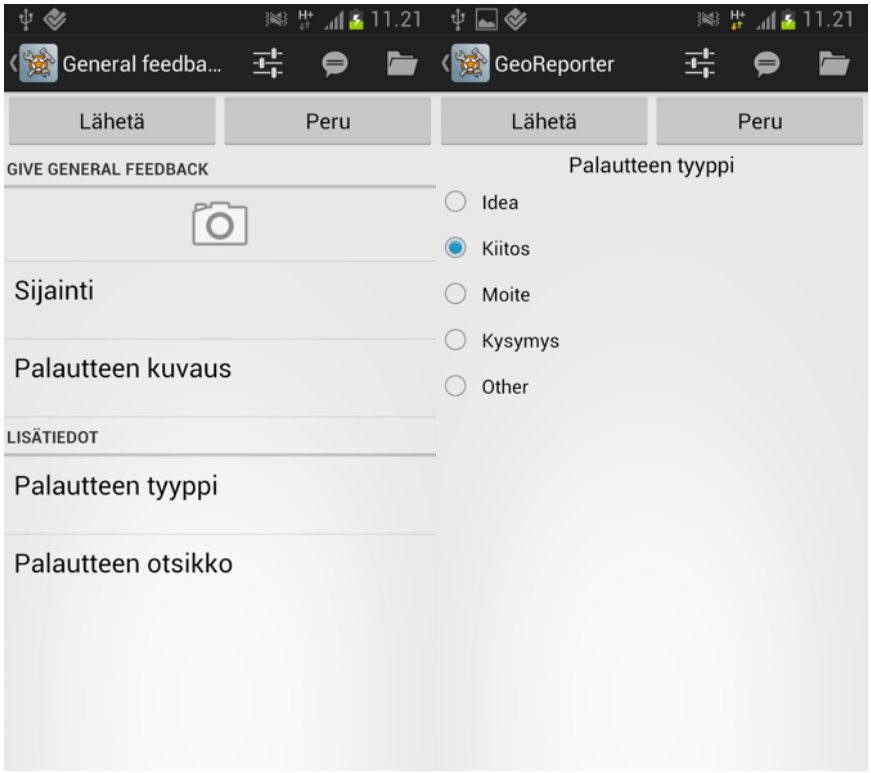


Image 3. UI with Finnish service definition

.

The Service Definition message with Finnish language option is below.

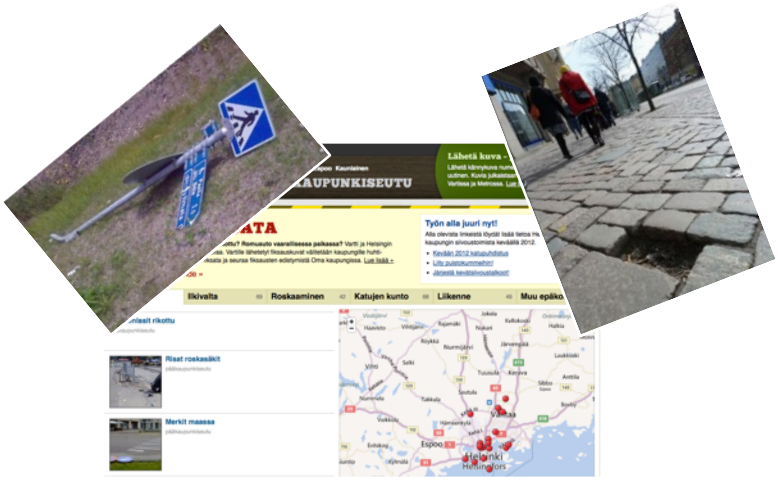
https://api.hel.fi/services/003.xml?locale=fi_FI

```
<service_definition>
        <service_code>003</service_code>
        <attributes>
                <attribute>
                        <variable>true</variable>
                        <code>service_request_type</code>
                        <datatype>singlevaluelist</datatype>
                        <required>true</required>
                        <datatype_description></datatype_description>
                        <order>1</order>
                        <description>Palautteen tyyppi</description>
                        <values>
                                <value>
                                        <key>IDEA</key>
                                        <name>Idea</name>
                                </value>
                                <value>
                                        <key>THANK</key>
                                        <name>Kiitos</name>
                                </value>
                                <value>
                                        <key>BLAME</key>
                                        <name>Moite</name>
                                </value>
                                <value>
                                        <key>QUESTION</key>
                                        <name>Kysymys</name>
```

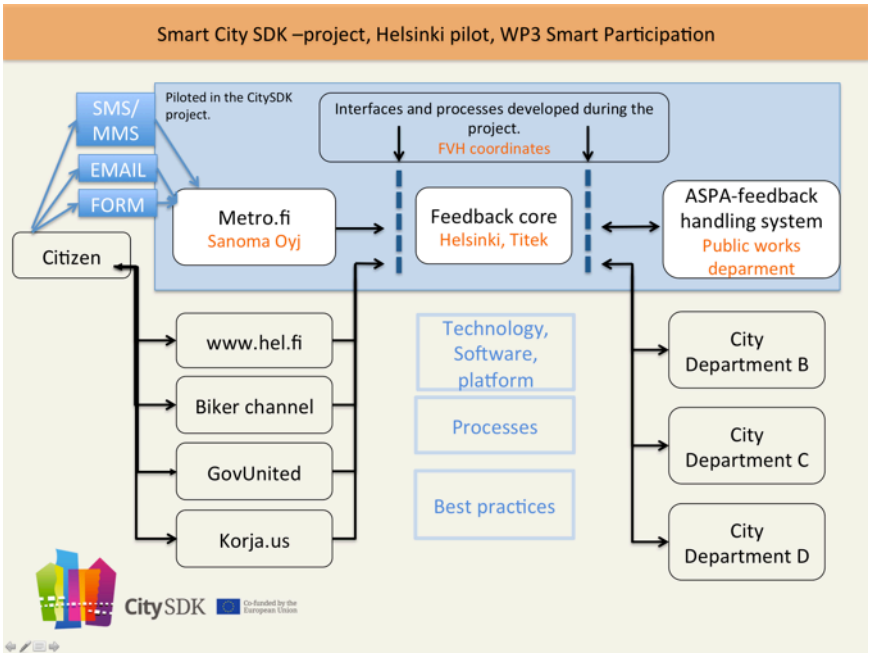
```
                                        </value>
                                        <value>
                                                <key>OTHER</key>
                                                <name>Muu</name>
                                        </value>

                                </values>
                        </attribute>
                        <attribute>
                                <variable>true</variable>
                                <code>title</code>
                                <datatype>string</datatype>
                                <required>false</required>
                                <datatype_description></datatype_description>
                                <order>2</order>
                                <description>Palautteen otsikko</description>
                        </attribute>
                </attributes>
</service_definition>
```

6 Smart Participation Lead Pilot



Smart Participation Lead Pilot in Helsinki demonstrates the possibility of innovative public-private collaboration where the technical feedback mechanisms are automated end-to-end, and the citizens interact with the city through a private high-reach news/media internet service – the site they already are visiting several times a day.



Smart Participation service will be piloted on local news site Metro.fi but other developers and businesses are welcome to utilize the interface as well.

6.1 Metro.fi

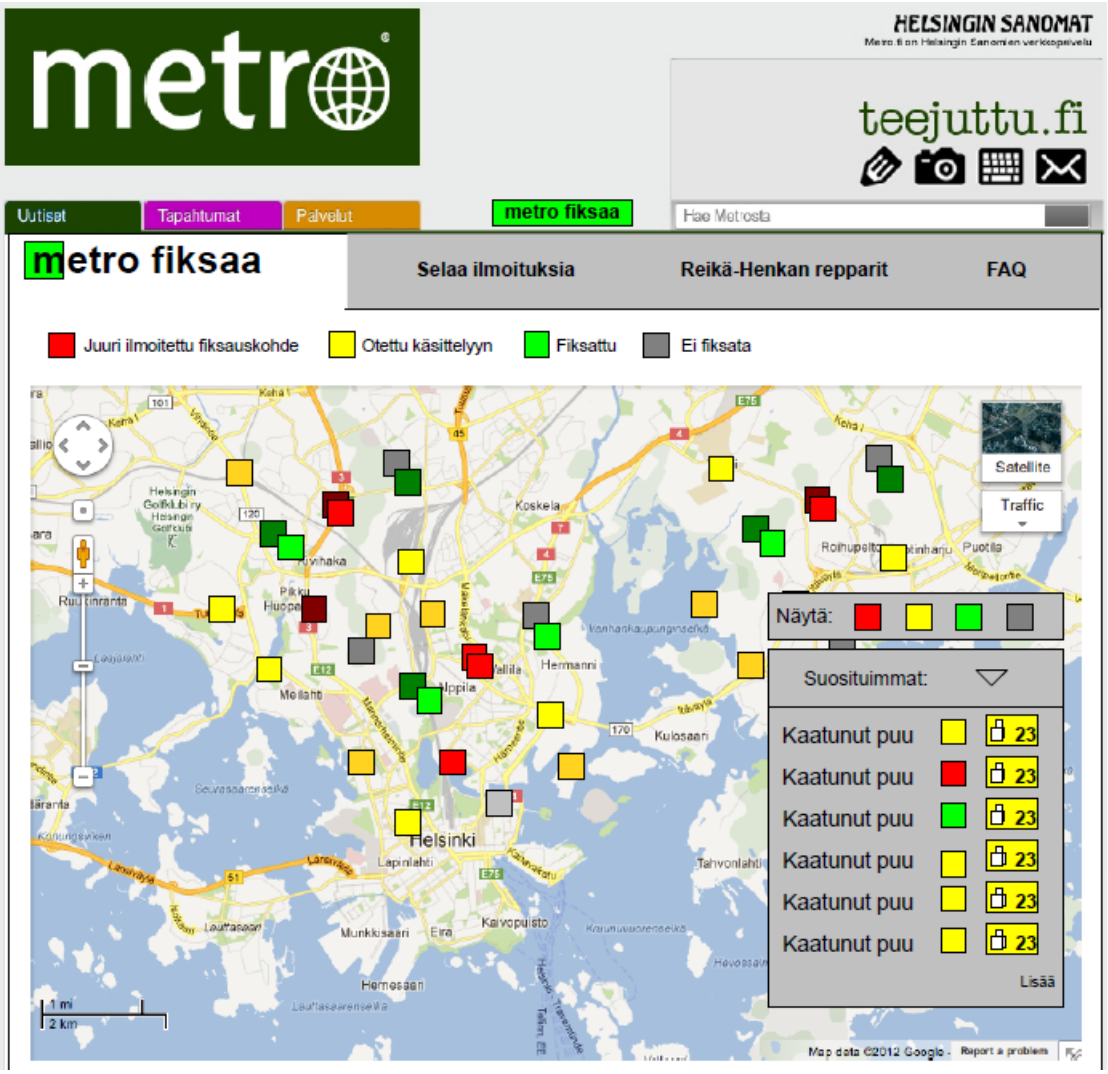


Figure 5: Wireframe for the Lead Pilot service

The Lead Pilot's service in Metro.fi provides a broad view on the reports submitted from the capital area of Helsinki. The reports made by fellow citizens are shown on the map and the colours indicate the status of the report. The users can check existing reports with one glimpse and get further information on interesting reports with just one click on the map. The issue submission form of the new service is compliant with the interface specification.

City's activity on handling the reports is also one of the first things the user will see when entering the service (the right hand box on the map). This is how the user can get a good view on how the city is managing the reports.

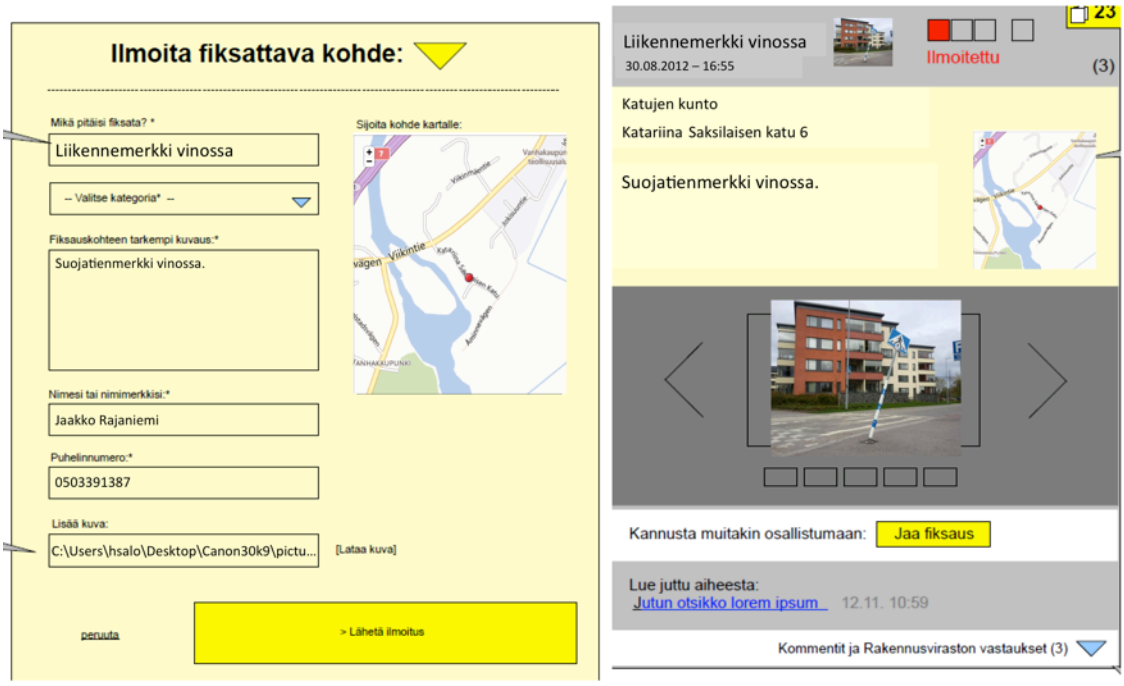
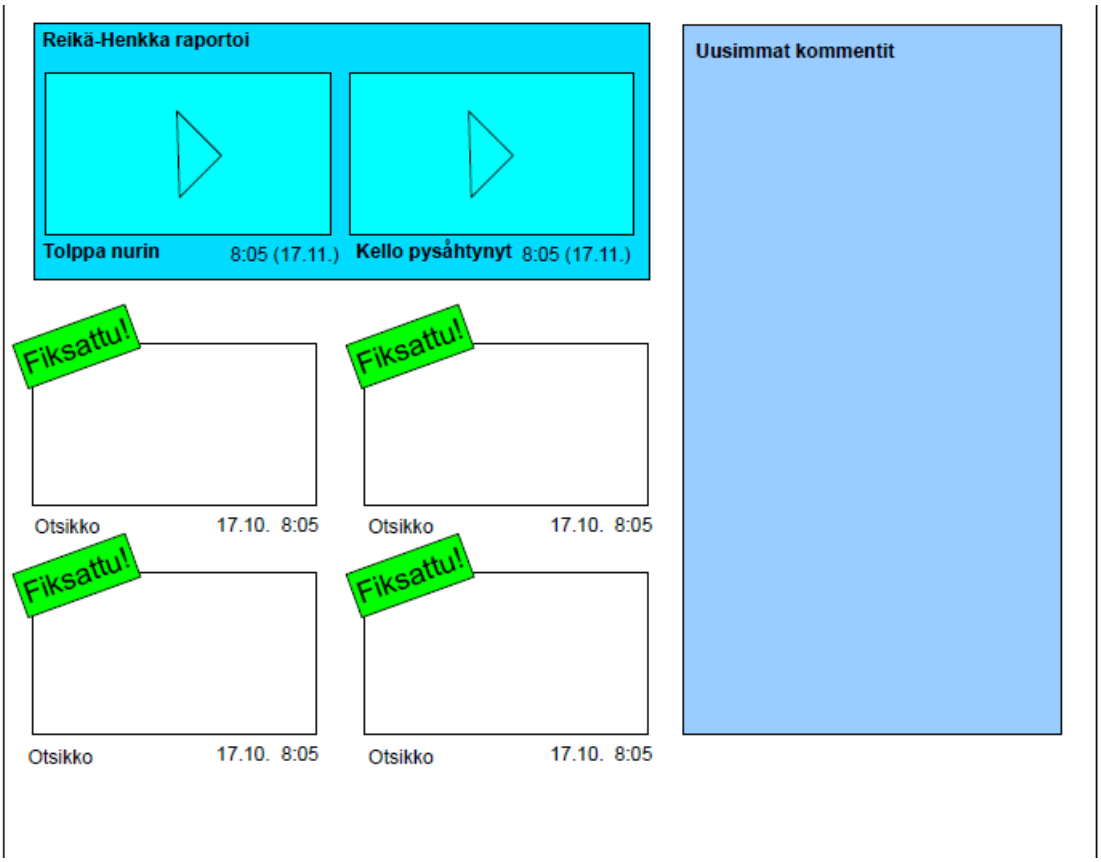


Figure 6. Users can submit their report and follow its status



We will provide articles and videos explaining and showing how the issues get actually fixed. Metro's FixMyStreet service has utilized user generated content by writing interviewing local people as well as city's administrative people handling the actual issues. this provides a rewarding private-public-media collaboration and enhances the citys policy making transparency.

6.2 City of Helsinki's centralized feedback management system

City of Helsinki has been developing a centralized feedback system, which is planned to be taken into use by all city's departments. This will happen in phases starting from the beginning of 2013.

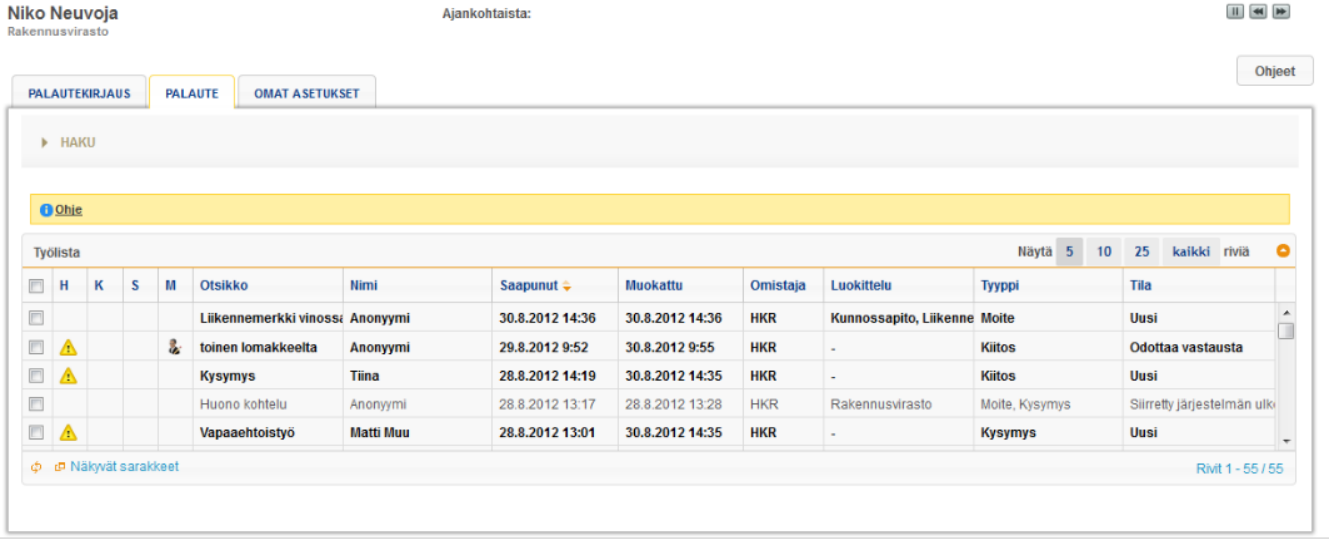


Image 4. Work list view of the centralized feedback system

The feedback given by citizens via web pages, phone calls or e-mails is all stored in the centralized feedback system. From there, feedback handlers will answer or forward receive feedback to the correct city's personnel for answering.

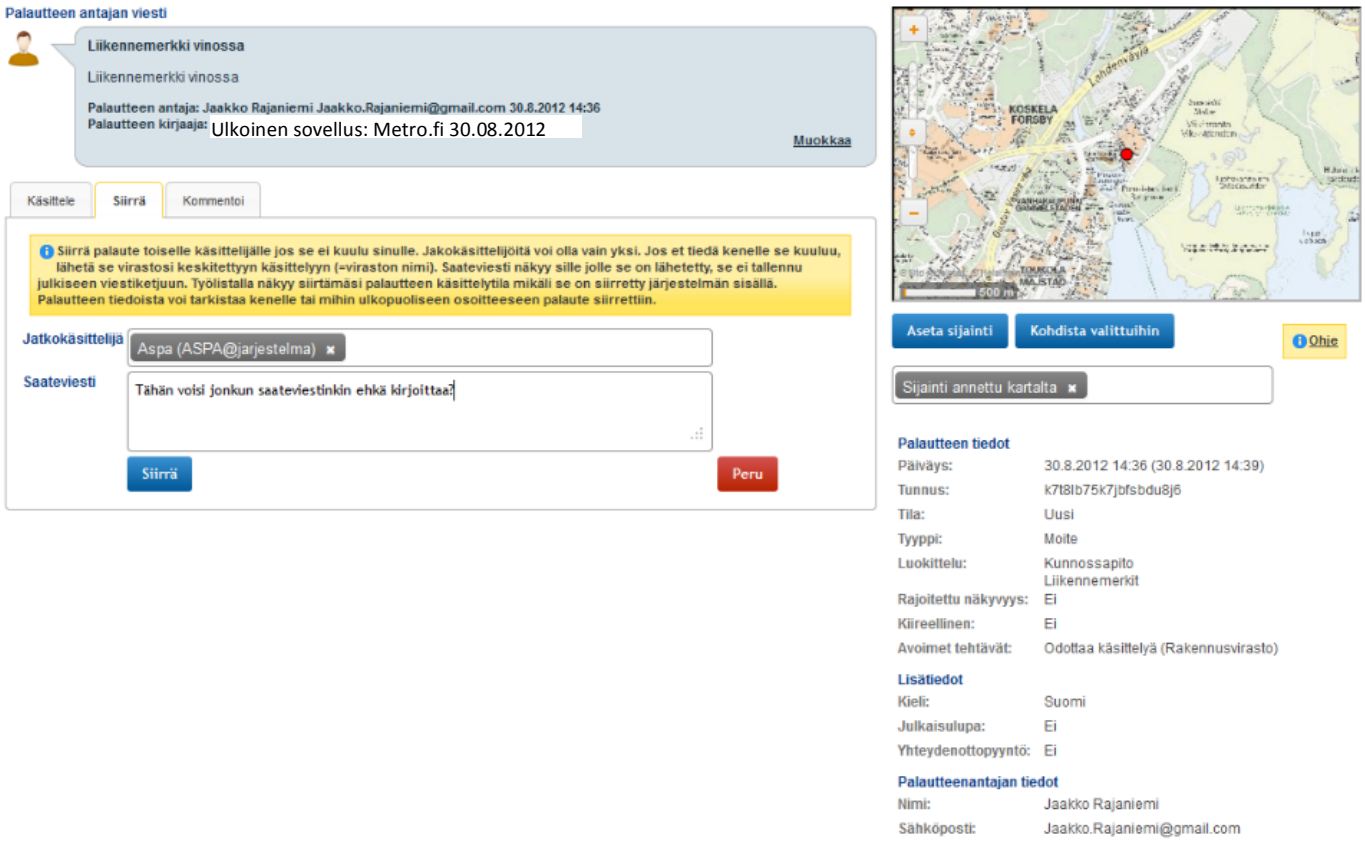


Image 5. View of one feedback from the work list

As it will be used by all city's departments, it can handle different kinds of feedback like questions, positive or negative feedback, ideas and any other kinds of feedback coming from citizens or other external sources. It will be the first time that City of Helsinki can have a comprehensive view on the feedback it receives from the citizens.

The CitySDK Smart Participation interface is implemented into the new centralized feedback system, where all service requests sent via the interface will arrive. From there, they are handled as any other feedback.

6.3 City of Helsinki's Public Works department

Even with the new centralized feedback management system Helsinki's Public Works department will keep its own case management system called Aspa. It is used only for reported issues, which require action from Public Works department. For example, if there is a pothole, which needs fixing, the issue is stored in the Aspa system. Aspa system contains information on on responsible personnel who needs to take action on fixing different issues. Aspa is used by the actual people who will organize the actual repair work.

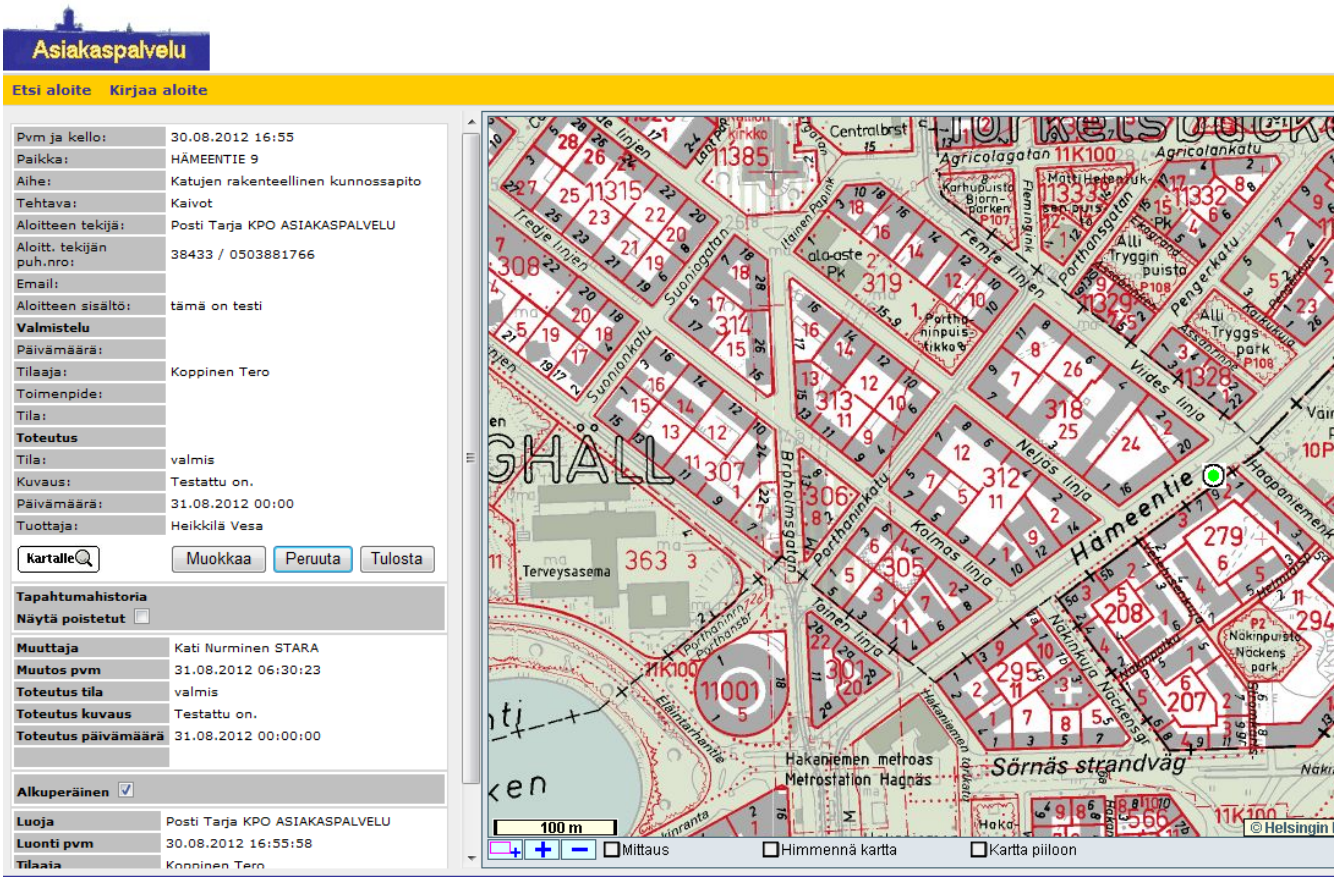


Image 6. Aspa case management system

Part of the Helsinki WP3 pilot is to integrate these systems in such a way that when citizens send service requests via the CitySDK interface into the city's centralized feedback system, it can be forwarded to the Public Works department's Aspa system. And also if issues were first reported into Aspa they are forwarded to the centralized feedback system and are also

visible to the citizens via the CitySDK Smart Participation interface. The status of each issue report is kept synchronized in the central feedback system.

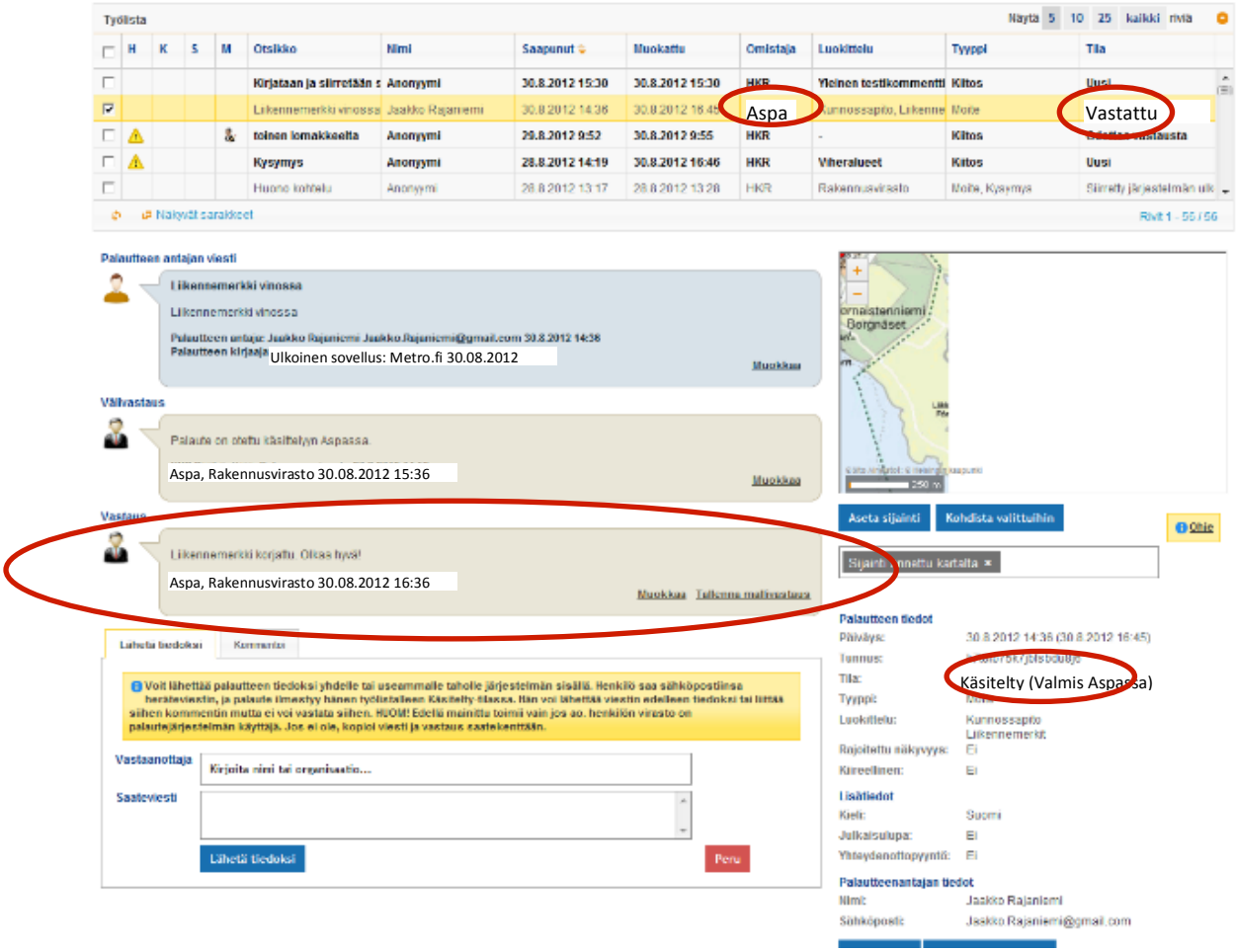


Image 7. Centralized feedback system shows status of the issue in Aspa

6.4 Developer engagement in the Lead Pilot

The interface specification, API keys and instructions for using the interface will be provided to the developer community as soon as the interface is implemented and tested. This information will be provided through CitySDK Developer pages as well.[7] Additionally, all the earlier developer and SME contacts Lead Pilot has made will be informed on the progress. Local Facebook page will be used in promoting the Smart Participation Lead Pilot components in Finland.

[7] http://www.citysdk.eu/developers/

7 Conclusions

Smart Participation SDK component, the interface, has been specified through the collaborative requirement definition process. The Lead Pilot will be using this interface starting from the launch of the pilot in January 2013.

After the launch of the Lead Pilot in the beginning of 2013, pilot and apps utilizing the interface will be promoted locally. Developer engagement and national efforts for harmonized issue reporting interfaces will be continued.

During the summer 2013, the Smart Participation concept will be expanded to Amsterdam, Barcelona, Lamia, Lisbon, Manchester and Rome. These partner cities will enable the two-way issue-reporting channel for their citizens and an opportunity to develop the interface according to their special needs. The work on the replication pilots has already started in many cities, as some already have existing feedback handling systems, apps or even national efforts to harmonize the interfaces, like in the Netherlands.

[8] Screenshots from Lamia's, Barcelona's and Amsterdam's existing services