LoST Javascript API

Yipeng Huang
Fall 2010

I. Overview

The LoSTJavascript API is an implementation of the client in the Location-to-Service Translation protocol described in RFC 5222. The API facilitates issuing LoST queries and interpreting LoST responses for use in Javascript web applications.

Of the four query types described in RFC 5222, the LoST Javascript API implements issuing findService requests and interprets findServiceResponse replies from the LoST server.

The API also implements additional features described in the LoST Extensions draft.

II. Instructions

A. Installation

The API requires installing a directory of five Javascript files, LoSTInterface.js, LoSTServiceInterface.js, QueryGenerator.js, ResponseParser.js, and xml2json.js, and one php program, LoST_proxy.php. Their functionality is described in the next section. These files must be kept in the same directory for the API to function correctly.

The demonstration webpage is in a file named demopage.html. The demonstration webpage requires the script Misc.js.

The server must be able to run PHP and have the cURL (client URL) library enabled for the API to function.

B. Code structure

To use the LoST Javascript API, the user should set up the following application code structure:

```
//Create a function that will be called when the LoST query is successful
var success_callback = function() {
	//Code for successful LoST query goes here.
	//Results from the query are available as public variables of the lostQuery object.
};

//Create a function that will be called when the LoST query encounters errors
var error_callback = function() {
	//Code for unsuccessful LoST query goes here.
	//A description of the error is available in lostQuery.exception.
};
```

```
//Issue the lostQuery.init call while passing in parameters for the query.
if (lostQuery.init({

    lostUrl: "http://128.59.22.81:8080/lost/LoSTServlet",
    service: "urn:service:sos",
    recursive: true,
    serviceBoundary: "value",
    locationId: "627b8bf819d0bad4d",
     civic: {
         country: "US",
         A1: "New York",
         A3: "New York",
         A6: "Broadway",
         HNO: 2920,
         PC: 10027
    },
    geodetic2D: {
       shape: "ArcBand",
       pos: {
              latitude: 40.807,
              longitude: -73.960
       },
       innerRadius: 50,
       outerRadius: 100,
       startAngle: 180,
       openingAngle: 30
     },
    validateLocation: true,
    region: true,
    maxDistance: 2000,
    limit: 100

})) {

       //If the init() step was successful, we can issue the query, and pass in to the query function references to the functions we
       created above.
       lostQuery.query(success_callback, error_callback);

}
```

C. Input specification

The options in the parameters associative array that is used in the init() call are described below:

Inputs that are features of RFC 5222 LoST:

Input name	Type	Description	Examples	Required?
lostUrl	url string	Pointer to the LoST server that will be queried.	“http://128.59.22.81:8080/lost/LoSTServlet”	Yes
service	urn string	The service type that will be looked up.	“urn:service:sos”, “urn:service:sos.police”	Yes
recursive	Boolean	Whether or not the queries should be resolved recursively.	true, false	No, default is false.
serviceBoundary	“value” or “reference”	Whether the descriptions of service boundaries should be returned as full values, or as a reference.	“value”, “reference”	No, default is reference.
locationId	string	An identifier for the current location profile that the API user would like to specify.	“627b8bf819d0bad4d”	No, default is no value.

civic	Javascript associative array	An associative array that describes the user’s civic address according to PIDF-LO format.	{
			country: ″DE″,
			A1: ″Bavaria″,
			A3: ″Munich″,
			A6: ″Otto-Hahn-Ring″,
			HNO: 6,
			PC: 81675
			}
				No, if no civic address is provided, the API will see if a geodetic2D location is provided. If neither geodetic2D nor civic location is provided, the API will attempt to acquire the user’s location through geolocation.

geodetic2D	Javascript associative array	An associative array that describes one of the five allowed shapes describing the query location. The shapes are described below.	See description below	No, if neither geodetic2D nor civic location is provided, the API will attempt to acquire the user’s location through geolocation.
validateLocation	boolean	Whether or not the server should return information about which address fields were recognized. Only meaningful when a civic address is used for query.	true, false	No, default is false.

Inputs that are features of LoST Extensions:

Input name	Type	Description	Examples	Required?
region	boolean	Whether or not the query should return only services that have service boundaries covering the user's location.	true, false	No, default is false.
maxDistance	int	The maximum distance the service should be from the user for it to be included in the findServiceResponse.	2000	No, default is no value set.
limit	int	The maximum number of mappings that should be included in the findServiceResponse.	1000	No, default is no value set.

Input format for civic addresses:

Civic addresses should be formatted as Javascript associative arrays. The arrays should have sufficient key value pairs for the LoST server to determine the user's intended location.

For example, both of these civic address formats are valid:

civic: {country: "DE", A1: "Bavaria", A3: "Munich", A6: "Otto-Hahn-Ring", HNO: 6, PC: 81675}

civic: {country: "US", A1: "New York", A3: "New York", A6: "Broadway", HNO: 2920, PC: 10027}

Input format for geodetic-2d locations:

Shape	Shape properties	Example
Point	Origin position	geodetic2D: {shape: "Point", pos: {latitude: 40.807, longitude: -73.960}}
Circle	Origin position Radius	geodetic2D: {shape: "Circle", pos: {latitude: 40.807, longitude: -73.960}, radius: 100}
Ellipse	Origin position Semi major axis Semi minor axis Orientation of major axis	geodetic2D: {shape: "Ellipse", pos: {latitude: 40.807, longitude: -73.960}, semiMajorAxis: 100, semiMinorAxis: 50, orientation: 180}
ArcBand	Origin position Inner radius Outer radius Start angle Opening angle	geodetic2D: {shape: "ArcBand", pos: {latitude: 40.807, longitude: -73.960}, innerRadius: 50, outerRadius: 100, startAngle: 180, openingAngle: 30}

Polygon	An array of positions. The last position must match the first one.	geodetic2D: {shape: "Polygon", posArray: [{latitude: 40.807, longitude: -73.960}, {latitude: 35, longitude: -73.960}, {latitude: 40.807, longitude: 70}, {latitude: 40.807, longitude: -73.960}]}

D. Output specification

Once the LoST query is executed with the query() function, the following public variables are available in the lostQuery object.

Output name	**Description**	**Sample output**
lostQuery.lat	The latitude of the user as determined by geolocation.	40.714
lostQuery.lon	The longitude of the user as determined by geolocation.	-74.006
lostQuery.lostQueryString	The XML query string that was sent to the LoST server. This is exposed to the user for debugging purposes.	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" serviceBoundary="value" recursive="true"> <location id="6020688f1ce1896d" profile="geodetic-2d"> <p2:Point id="point1" srsName="urn:ogc:def:crs:EPSG::4326"> <p2:pos>40.714 -74.006</p2:pos> </p2:Point> </location> <service> urn:service:sos.police </service> </findService>
lostQuery.lostResponseString	The XML response string that the API received from the LoST server. This is exposed to the user for debugging purposes.	--
lostQuery.mappings	An array of mappings that were obtained from the response. Each mapping is an associative array of attributes of that mapping. The attributes are described below.	--
lostQuery.mappings[i].expires	The absolute time at which the mapping becomes invalid.	"2009-01-01T00:00:00Z"

lostQuery.mappings[i].lastUpdated	The time at which this mapping last changed.	“2009-01-01T00:00:00Z”
lostQuery.mappings[i].source	A unique string identifying the authoritative generator of the mapping.	“lost.cs.columbia.edu”
lostQuery.mappings[i].sourceId	A token that identifies this particular mapping.	“4c594c32-1389-11de-bfdc-8da325ae7c2f”
lostQuery.mappings[i].displayName	A human readable name for the service.	“New York”
lostQuery.mappings[i].service	The urn of the service type that was used in the query.	“urn:service:sos”
lostQuery.mappings[i].uri[j]	One or more uri strings that are associated with this service.	“sip:psap_ny@irt.cs.columbia.edu”
lostQuery.mappings[i].serviceNumber[j]	One or more phone numbers that are associated with this service.	911
lostQuery.mappings[i].serviceBoundary	In the current implementation, this field accesses one of three possible responses from the LoST server that describes the service’s coverage: 1. The field may contain an associative array that has one source and one key attributes that allow the LoST client to look up the service boundary with a separate findServiceBoundary request. 2. The field may contain an associate array that has one associative array describing a civic address. 3. The field may contain an array of positions that draw a polygon.	1. {source: ″authoritative.example″, key:″7214148E0433AFE2F1172E″} 2. {xmlns:“urn:ietf:params:xml:ns: pidf:geopriv10:civicAddr”, country: “US”, a1: “NY”} 3. [{37.775 -122.4194}, {37.555 -122.4194}, {37.555 -122.4264}, {37.775 -122.4264}, {37.775 -122.4194}]
lostQuery.mappings[i].serviceLocation	The latitude and longitude of the service.	“33.665 -112.432”
lostQuery.mappings[i].civicAddress	The civic address of the service.	{country: ″US″, A1: ″New York″, A3: ″New York″, A6: ″Broadway″, HNO: 2920, PC: 10027}
lostQuery.locationValidation	An associative array that has the keys “valid”, “invalid”, and “unchecked”. Each key has a value denoting which civic address tags were recognized by the server.	{valid: “Country A1”}
lostQuery.exception	A string that describes the exception condition that caused the LoST query to fail.	“HTTP EXCEPTION: 404 not found”

E. Exception handling

When the LoST query encounters an exception condition and is forced to call the error_callback() function, the field lostQuery.exception contains text describing the exception condition. The possible conditions are described below:

Exception description	Example content of lostQuery.exception when this exception occurs
Insufficient parameters supplied to init() function.	“INIT EXCEPTION: query parameters not entered.”, “INIT EXCEPTION: LoST server URL missing.”, “INIT EXCEPTION: Service URN missing.”, ”INIT EXCEPTION: LoST query options incomplete.”
Geolocation fails during a getCurrentPosition() call	”GEOLOCATION EXCEPTION: ” + <error message from geolocator>
Geolocation init() call fails because browser does not support W3C geolocation API	”GEOLOCATION EXCEPTION: Browser does not support W3C geo.”
Generation of query string fails because of invalid geodetic-2d shape.	”QUERY EXCEPTION: Unknown shape profile.”
Generation of query string fails because location is in neither civic or geodetic-2d format.	”QUERY EXCEPTION: Unknown location profile.”
LoST findServiceResponse does not contain meaningful information because the server has no data concerning query, or because query string was incorrect.	”LoST EXCEPTION: ” + LoST server’s <errors> XML example follows “LoST EXCEPTION: HTTP/1.1 200 OK Server: Apache-Coyote/1.1 Server: CU-LoST-RI/1.1.3.20091220 Content-Type: application/lost+xml Content-Length: 215 Date: Fri, 17 Dec 2010 07:24:51 GMT <?xml version=”1.0” encoding=”UTF-8”?><errors xmlns=”urn:ietf:params:xml:ns:lost1” source=”ng911-lost1.irt.cs.columbia.edu”><internalError message=”ERROR: geometry contains non-closed rings” xml:lang=”en”/></errors>”
LoST server could not be reached or timed out.	”HTTP EXCEPTION: 404 not found”

III. Design

A. Required resources

The LoST Javascript API relies on the browser's implementation of the W3C Geolocation API to acquire location information of the user. The Geolocation API is supported only in the newest browsers. This API is confirmed to work well on Mozilla Firefox, Google Chrome, Apple Safari, Opera, and the Android web browser.

The API requires that the browser support XMLHttpRequests, also known as AJAX, for purposes of communicating with the LoST server.

The API relies on two scripts that are implemented by other authors and released on public licenses. One is a script to encode and decode GET method information from url strings, credited to www.webtoolkit.info. The other script is an XML to JSON parser written by Thomas Frank.

On the server side, the API requires that the server support PHP with the cURL library. This is necessary because browsers do not allow AJAX information to come from hosts other than the one hosting the current page. Due to this security constraint, browsers would reject information coming from LoST servers. The PHP script is necessary to act as a proxy for LoST responses coming from the LoST server, making it seem to browsers that the information is coming from the same source as the page provider.

B. Module description

The following table describes the architecture of the LoST Javascript API.

Module name	Module description
LoSTInterface.js	The main script for the API. This script obtains the parameters needed for the query, and maintains the variables that contain the results of the query. This script also maintains the program flow.
LoSTServiceInterface.js	The script that conducts the communication with the LoST server via AJAX. This script is also capable of parsing a small portion of the response message to determine whether or not the response contains usable information.
QueryGenerator.js	The script that maintains XML templates of LoST queries. Separate templates are kept for the different location profiles that can be used in LoST queries.
ResponseParser.js	The script that interprets the findServiceResponse XML and creates the mappings array.
xml2json.js	A script that contains an XML to JSON (Javascript Object Notation) parser.
LoST_proxy.php	A script that contains a proxy that forwards queries to the LoST server and returns responses to the LoST client.

IV. Test Cases

The following pages contain documentation regarding the test cases applied to the LoST Javascript API to ensure that it functions correctly. The tests are separated in input tests, output tests, and full integration tests. Because no existing LoST server implements the features in the LoST Extensions draft, the responses that contain tags found only in the LoST Extensions draft were tested with dummy responses fed to the parser as a static string.

A. Input Tests

Here, we test to make sure the LoST API can replicate the example queries in RFC 5222 and in the LoST Extensions draft.

Description	Test parameters	Expected query	Observed query
Base case using geolocation capabilities of the device	No civic address or geodetic location provided. API defaults to using geolocation	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" serviceBoundary="value" recursive="true"> <location id="6020688f1ce1896d" profile="geodetic-2d"> <p2:Point id="point1" srsName="urn:ogc:def:crs:EPSG::4326"> <p2:pos>37.775 -122.422</p2:pos> </p2:Point> </location> <service>urn:service:sos.police</service> </findService>	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" serviceBoundary="value" recursive="true"> <location id="6020688f1ce1896d" profile="geodetic-2d"> <p2:Point id="point1" srsName="urn:ogc:def:crs:EPSG::4326"> <p2:pos>40.714 -74.006</p2:pos> </p2:Point> </location> <service> urn:service:sos.police </service> </findService>
Validate location enabled	validateLocation: true,	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" recursive="true" validateLocation="true" serviceBoundary="value"> <location id="627b8bf819d0bad4d" profile="civic"> <civicAddress xmlns="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"> <country>DE</country> <A1>Bavaria</A1> <A3>Munich</A3> <A6>Otto-Hahn-Ring</A6> <HNO>6</HNO> <PC>81675</PC> </civicAddress> </location> <service>urn:service:sos.police</service> </findService>	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" recursive="true" validateLocation="true" serviceBoundary="value"> <location id="627b8bf819d0bad4d" profile="civic"> <civicAddress xmlns="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"> <country>DE</country> <A1>Bavaria</A1> <A3>Munich</A3> <A6>Otto-Hahn-Ring</A6> <HNO>6</HNO> <PC>81675</PC> </civicAddress> </location> <service> urn:service:sos.police </service> </findService>
Service boundary requested as reference	serviceBoundary: reference,	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" recursive="true"	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" serviceBoundary="reference"

		serviceBoundary="reference"> <location id="6020688f1ce1896d" profile="geodetic-2d"> <p2:Point id="point1" srsName="urn:ogc:def:crs:EPSG::4326"> <p2:pos>37.775 -122.422</p2:pos> </p2:Point> </location> <service>urn:service:sos.police</service> </findService>	recursive="true"> <location id="627b8bf819d0bad4d" profile="geodetic-2d"> <p2:Point id="point1" srsName="urn:ogc:def:crs:EPSG::4326"> <p2:pos>40.714 -74.006</p2:pos> </p2:Point> </location> <service> urn:service:sos.police </service> </findService>
Civic queries	civic: {country: "DE", A1: "Bavaria", A3: "Munich", A6: "Otto-Hahn-Ring", HNO: 6, PC: 81675},	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" recursive="true" serviceBoundary="value"> <location id="627b8bf819d0bad4d" profile="civic"> <civicAddress xmlns="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"> <country>DE</country> <A1>Bavaria</A1> <A3>Munich</A3> <A6>Otto-Hahn-Ring</A6> <HNO>6</HNO> <PC>81675</PC> </civicAddress> </location> <service>urn:service:sos.police</service> </findService>	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" recursive="true" validateLocation="false" serviceBoundary="value"> <location id="627b8bf819d0bad4d" profile="civic"> <civicAddress xmlns="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"> <country>DE</country> <A1>Bavaria</A1> <A3>Munich</A3> <A6>Otto-Hahn-Ring</A6> <HNO>6</HNO> <PC>81675</PC> </civicAddress> </location> <service> urn:service:sos.police </service> </findService>
Civic Address	civic: {country: "US", A1: "New York", A3: "New York", A6: "Broadway", HNO: 2920, PC: 10027},		<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" recursive="true" validateLocation="true" serviceBoundary="value"> <location id="627b8bf819d0bad4d" profile="civic"> <civicAddress xmlns="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"> <country>US</country> <A1>New York</A1> <A3>New York</A3>

			<A6>Broadway</A6> <HNO>2920</HNO> <PC>10027</PC> </civicAddress> </location> <service>urn:service:sos</service> </findService>
Geodetic 2D Point	{shape: "Point", pos: {latitude: 40.807, longitude: -73.960}}		<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" serviceBoundary="value" recursive="true"> <location id="627b8bf819d0bad4d" profile="geodetic-2d"> <p2:Point id="point1" srsName="urn:ogc:def:crs:EPSG::4326"> <p2:pos>40.807 -73.96</p2:pos> </p2:Point> </location> <service>urn:service:sos</service> </findService>
Geodetic 2D Circle	{shape: "Circle", pos: {latitude: 40.807, longitude: -73.960}, radius: 100}		<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" serviceBoundary="value" recursive="true"> <location id="627b8bf819d0bad4d" profile="geodetic-2d"> <p2:Circle srsName="urn:ogc:def:crs:EPSG::4326"> <p2:pos>40.807 -73.96</p2:pos> <p2:radius uom="urn:ogc:def:uom:EPSG::9001">100</p2:radius> </p2:Circle> </location> <service>urn:service:sos</service> </findService>
Geodetic 2D Ellipse	{shape: "Ellipse", pos: {latitude: 40.807, longitude: -73.960}, semiMajorAxis: 100, semiMinorAxis: 50, orientation: 180}		<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml"

			serviceBoundary="value" recursive="true"> <location id="627b8bf819d0bad4d" profile="geodetic-2d"> <p2:Ellipse srsName="urn:ogc:def:crs:EPSG::4326"> <p2:pos>40.807 -73.96</p2:pos> <p2:semiMajorAxis uom="urn:ogc:def:uom:EPSG::9001">100</p2:semiMajorAxis> <p2:semiMinorAxis uom="urn:ogc:def:uom:EPSG::9001">50</p2:semiMinorAxis> <p2:orientation uom="urn:ogc:def:uom:EPSG::9102">180</p2:orientation> </p2:Ellipse> </location> <service>urn:service:sos</service> </findService>
Geodetic 2D Polygon	{shape: "Polygon", posArray: [{latitude: 40.807, longitude: -73.960}, {latitude: 35, longitude: -73.960}, {latitude: 40.807, longitude: 70}, {latitude: 40.807, longitude: -73.960}]}		<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" serviceBoundary="value" recursive="true"> <location id="627b8bf819d0bad4d" profile="geodetic-2d"> <p2:Polygon srsName="urn:ogc:def:crs:EPSG::4326"> <p2:exterior> <p2:LinearRing> <p2:pos>40.807 -73.96</p2:pos> <p2:pos>35 -73.96</p2:pos> <p2:pos>40.807 70</p2:pos> <p2:pos>40.807 -73.96</p2:pos> </p2:LinearRing> </p2:exterior> </p2:Polygon> </location> <service>urn:service:sos</service> </findService>
Geodetic 2D ArcBand	{shape: "ArcBand", pos: {latitude: 40.807, longitude: -73.960}, innerRadius: 50, outerRadius: 100, startAngle: 180, openingAngle: 30}		<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" serviceBoundary="value"

			recursive="true"> <location id="627b8bf819d0bad4d" profile="geodetic-2d"> <p2:ArcBand srsName="urn:ogc:def:crs:EPSG::4326"> <p2:pos>40.807 -73.96</p2:pos> <p2:innerRadius uom="urn:ogc:def:uom:EPSG::9001">50</p2:innerRadius> <p2:outerRadius uom="urn:ogc:def:uom:EPSG::9001">100</p2:outerRadius> <p2:startAngle uom="urn:ogc:def:uom:EPSG::9102">180</p2:startAngle> <p2:openingAngle uom="urn:ogc:def:uom:EPSG::9102">30</p2:openingAngle> </p2:ArcBand> </location> <service>urn:service:sos</service> </findService>
Within distance X	maxDistance: 200,	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" serviceBoundary="value" recursive="true"> <location id="6020688f1ce1896d" profile="geodetic-2d"> <p2:Circle srsName="urn:ogc:def:crs:EPSG::4326"> <p2:pos>37.775 -122.422</p2:pos> <p2:radius uom="urn:ogc:def:uom:EPSG::9001"> 200 </p2:radius> </p2:Circle> </location> <service>urn:service:local.pizza</service> </findService>	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" serviceBoundary="value" recursive="true"> <location id="627b8bf819d0bad4d" profile="geodetic-2d"> <p2:Circle srsName="urn:ogc:def:crs:EPSG::4326"> <p2:pos>40.714 -74.006</p2:pos> <p2:radius uom="urn:ogc:def:uom:EPSG::9001">200</p2:radius> </p2:Circle> </location> <service>urn:service:sos</service> </findService>
Region	region: true,	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" xmlns:ext="http://www.thisisnotdoneyet.net/lostNe" serviceBoundary="value" recursive="true"> <ext:region>true</ext:region> <location id="6020688f1ce1896d" profile="geodetic-2d"> <p2:Circle srsName="urn:ogc:def:crs:EPSG::4326">	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" xmlns:ext="http://www.thisisnotdoneyet.net/lostNe" serviceBoundary="value" recursive="true"> <ext:region>true</ext:region> <location id="627b8bf819d0bad4d" profile="geodetic-2d">

		<p2:pos>37.775 -122.422</p2:pos> <p2:radius uom="urn:ogc:def:uom:EPSG::9001"> 200 </p2:radius> </p2:Circle> </location> <service>urn:service:local.pizza</service> </findService>	<p2:Point id="point1" srsName="urn:ogc:def:crs:EPSG::4326"> <p2:pos>40.714 -74.006</p2:pos> </p2:Point> </location> <service>urn:service:sos</service> </findService>
N nearest	limit: 20,	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" xmlns:ext="http://www.thisisnotdoneyet.net/lostNe" serviceBoundary="value" recursive="true"> <ext:limit>20</ext:limit> <location id="6020688f1ce1896d" profile="geodetic-2d"> <p2:Point id="point1" srsName="urn:ogc:def:crs:EPSG::4326"> <p2:pos>40.7128 -74.0092</p2:pos> </p2:Point> </location> <service>urn:service:food.pizza</service> </findService>	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" xmlns:ext="http://www.thisisnotdoneyet.net/lostNe" serviceBoundary="value" recursive="true"> <ext:limit>20</ext:limit> <location id="627b8bf819d0bad4d" profile="geodetic-2d"> <p2:Point id="point1" srsName="urn:ogc:def:crs:EPSG::4326"> <p2:pos>40.714 -74.006</p2:pos> </p2:Point> </location> <service>urn:service:sos</service> </findService>
N nearest and within distance X	limit: 20, maxDistance: 200	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" xmlns:ext="http://www.thisisnotdoneyet.net/lostNe" serviceBoundary="value" recursive="true"> <ext:region>false</ext:region> <ext:limit>20</ext:limit> <location id="6020688f1ce1896d" profile="geodetic-2d"> <p2:Circle srsName="urn:ogc:def:crs:EPSG::4326"> <p2:pos>37.775 -122.422</p2:pos> <p2:radius uom="urn:ogc:def:uom:EPSG::9001"> 200 </p2:radius> </p2:Circle> </location>	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml" xmlns:ext="http://www.thisisnotdoneyet.net/lostNe" serviceBoundary="value" recursive="true"> <ext:limit>20</ext:limit> <location id="627b8bf819d0bad4d" profile="geodetic-2d"> <p2:Circle srsName="urn:ogc:def:crs:EPSG::4326"> <p2:pos>40.714 -74.006</p2:pos> <p2:radius uom="urn:ogc:def:uom:EPSG::9001">200</p2:radius> </p2:Circle> </location> <service>urn:service:sos</service> </findService>

		<service>urn:service:local.pizza</service> </findService>	

B. Output Tests

In these tests, we check to make sure parsing is done accurately. The API must provide the correct outputs, and the fields that are displayed on the demo page must match the findServiceResponse XML.

Description	Response	Parse results
Base case	<?xml version="1.0" encoding="UTF-8"?> <findServiceResponse xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml"> <mapping expires="2007-01-01T01:44:33Z" lastUpdated="2006-11-01T01:00:00Z" source="authoritative.example" sourceId="7e3f40b098c711dbb6060800200c9a66"> <displayName xml:lang="en"> New York City Police Department </displayName> <service>urn:service:sos.police</service> <serviceBoundary profile="geodetic-2d"> <p2:Polygon srsName="urn:ogc:def::crs:EPSG::4326"> <p2:exterior> <p2:LinearRing> <p2:pos>37.775 -122.4194</p2:pos> <p2:pos>37.555 -122.4194</p2:pos> <p2:pos>37.555 -122.4264</p2:pos> <p2:pos>37.775 -122.4264</p2:pos> <p2:pos>37.775 -122.4194</p2:pos> </p2:LinearRing> </p2:exterior> </p2:Polygon> </serviceBoundary> <uri>sip:nypd@example.com</uri> <uri>xmpp:nypd@example.com</uri> <serviceNumber>911</serviceNumber> </mapping> <path>	Passed

	<via source="resolver.example"/> <via source="authoritative.example"/> </path> <locationUsed id="6020688f1ce1896d"/> </findServiceResponse>	
Civic format serviceBoundary	<?xml version="1.0" encoding="UTF-8"?> <findServiceResponse xmlns="urn:ietf:params:xml:ns:lost1"> <mapping expires="2007-01-01T01:44:33Z" lastUpdated="2006-11-01T01:00:00Z" source="authoritative.example" sourceId="4db898df52b84edfa9b6445ea8a0328e"> <displayName xml:lang="de"> Muenchen Polizei-Abteilung </displayName> <service>urn:service:sos.police</service> <serviceBoundary profile="civic"> <civicAddress xmlns="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"> <country>DE</country> <A1>Bavaria</A1> <A3>Munich</A3> <PC>81675</PC> </civicAddress> </serviceBoundary> <uri>sip:munich-police@example.com</uri> <uri>xmpp:munich-police@example.com</uri> <serviceNumber>110</serviceNumber> </mapping> <locationValidation> <valid>country A1 A3 A6</valid> <invalid>PC</invalid> <unchecked>HNO</unchecked> </locationValidation> <path> <via source="resolver.example"/> <via source="authoritative.example"/> </path> <locationUsed id="627b8bf819d0bad4d"/> </findServiceResponse>	Passed

Reference format serviceBoundary	<?xml version="1.0" encoding="UTF-8"?> <findServiceResponse xmlns="urn:ietf:params:xml:ns:lost1" xmlns:p2="http://www.opengis.net/gml"> <mapping expires="2007-01-01T01:44:33Z" lastUpdated="2006-11-01T01:00:00Z" source="authoritative.example" sourceId="7e3f40b098c711dbb6060800200c9a66"> <displayName xml:lang="en"> New York City Police Department </displayName> <service>urn:service:sos.police</service> <serviceBoundaryReference source="authoritative.example" key="7214148E0433AFE2FA2D48003D31172E"/> <uri>sip:nypd@example.com</uri> <uri>xmpp:nypd@example.com</uri> <serviceNumber>911</serviceNumber> </mapping> <path> <via source="resolver.example"/> <via source="authoritative.example"/> </path> <locationUsed id="6020688f1ce1896d"/> </findServiceResponse>	source: ng911-lost1.irt.cs.columbia.edu key: 8c435895f2f531ddb753d937ed6cbf6c

C. Integration Tests

Here, we run tests that require both query generation and response parsing to work correctly.

Description	**Test parameters**	**Query**	**Response**	**Parsing**
Civic address with validate location on	civic: {country: "US", A1: "New York", A3: "New York", A6: "Broadway", HNO: 2920, PC: 10027}, validateLocation: (pageVars.validateLocation == "validateLocation"),	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" recursive="true" validateLocation="true" serviceBoundary="value"> <location id="627b8bf819d0bad4d" profile="civic">	<?xml version="1.0" encoding="UTF-8"?> <findServiceResponse xmlns="urn:ietf:params:xml:ns:lost1"> <mapping expires="NO-CACHE" lastUpdated="2009-01-01T00:00:00Z" source="lost.cs.columbia.edu" sourceId="4c594c32-1389-11de-bfdc-8da325ae7c2f"> <displayName xml:lang="en">	Passed: Notable displayed fields: xmlns: urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr

		<civicAddress xmlns="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"> <country>US</country> <A1>New York</A1> <A3>New York</A3> <A6>Broadway</A6> <HNO>2920</HNO> <PC>10027</PC> </civicAddress> </location> <service>urn:service:sos</service> </findService>	New York </displayName> <service> urn:service:sos </service> <serviceBoundary profile="civic"> <civicAddress xmlns="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"> <country>US</country> <A1>NY</A1> </civicAddress> </serviceBoundary> <uri>sip:psap_ny@irt.cs.columbia.edu</uri> <serviceNumber>911</serviceNumber> </mapping> <locationValidation> <valid>Country A1</valid> </locationValidation> <path> <via source="ng911-lost1.irt.cs.columbia.edu"/> </path> <locationUsed id="627b8bf819d0bad4d"/> </findServiceResponse>	country: US a1: NY valid: Country A1
Exception propogation	A civic address outside of the US	<?xml version="1.0" encoding="UTF-8"?> <findService xmlns="urn:ietf:params:xml:ns:lost1" recursive="true" validateLocation="false" serviceBoundary="value"> <location id="627b8bf819d0bad4d" profile="civic"> <civicAddress xmlns="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"> <country>DE</country> <A1>Bavaria</A1> <A3>Munich</A3> <A6>Otto-Hahn-Ring</A6> <HNO>6</HNO> <PC>81675</PC> </civicAddress>	LoST EXCEPTION: HTTP/1.1 200 OK Server: Apache-Coyote/1.1 Server: CU-LoST-RI/1.1.3.20091220 Content-Type: application/lost+xml Content-Length: 176 Date: Fri, 17 Dec 2010 08:12:44 GMT <?xml version="1.0" encoding="UTF-8"?><errors xmlns="urn:ietf:params:xml:ns:lost1" source="ng911-lost1.irt.cs.columbia.edu"><notFound message="No Data" xml:lang="en"/></errors>	Passed. Exception was propagated through the error_callback methods and eventually displayed in the exceptions text field on the demo page.

		</location> <service>urn:service:sos</service> </findService>		