Access to Mobility Information in M2M Communications

Anastas Nikolov[1], Evelina Pencheva[1], Ivaylo Atanasov[1]

Abstract –**Service capabilities in Machine-to-Machine (M2M) communications represent generic functionality accessible for different applications through application programming interfaces. For M2M communications Representational State Transfer (REST) style is adopted. This paper presents an approach to design of RESTful web services for M2M mobility. An abstraction of mobility information is synthesized and represented as resources accessible through four standard operations: create, retrieve, update and delete. Implementation aspects are considered.**

Keywords – **Machine-to-Machine Communications, Service capabilities, Resource structure, Web Services**

I. Introduction

Machine-to-Machine (M2M) communications allow intelligent objects which are uniquely identified and gather data from their environment to exchange information with network applications. In order to deploy the M2M solutions in a wide scale it is necessary to design capabilities that can be reused across several applications [1], [2]. Best software practices in information technology and communications (ICT) adopt separation of applications from service capabilities and network capabilities.

Service capabilities are software modules that are exposed to M2M applications through the use of application programming interfaces (APIs). Several API sets have been designed in the ICT area. The Web services model has been used to define communication functions such as call control, messaging, charging and access to mobility information.

The Web services model is based on the assumption that communications are like invocation of remote service whose nature can be ignored. But this model is not suitable for M2M applications, as M2M devices are constrained resources with tangible states that can be manipulated. REST (REpresentational State Transfer) is adopted as a method for M2M modeling. In REST, each physical or logical entity is represented as a resource which has particular state. The resources can be addressed through HTTP Uniform Resource Identifier (URI) and its states can be retrieved and updated. Resources can be created and destroyed respectively.

In this paper, we present an approach to design Mobility service capability that may be used to build horizontal service platform. Mobility as an M2M service capability provides information about device location. The access to the location of a device is through a request for the device location, a notification of a change in the device location and notifications of device location on a periodic basis. The related research concerning access to location information addresses specific solutions, but does not study the necessary generic functionality [3], [4], [5], [6], [7]. The related implementations deal with positioning methods and the usage of location data, but do not concern programmability issues. We apply the REST architectural style to identify the generic context functions providing access to location information and to design valuable resource structure with operations that can manipulate it.

Our approach, presented in the next sections, consists of annotation of semantic information for location services and synthesis of resource structure as specified in ETSI TS 102 690. The procedures used to manipulate the location information are described as create, retrieve, update and delete operations. Some implementation aspects of the designed RESTful services are considered.

II. Annotation of Location Service Information

The location information depends on the application. For example it is required for healthcare and fleet tracking applications but not for smart metering. Additionally, location information may be reported periodically or on demand, and notifications of distance changes between monitored devices may also be available to applications.

The location information is presented by latitude, longitude, altitude, accuracy and time stamp. Latitude, longitude and altitude values are expressed as floating point numbers. The accuracy values express the desire of the application for the location information to be provided. The choice of values may influence the price that the service provider charges. In triggered notifications, a tracking accuracy is defined. Two accuracy values (requested and accepted) may be used. For example, a taxi tracking service that locates the nearest taxi to the client requires fine grained accuracy while coarse grained accuracy may be appropriate for a truck nearing the vicinity of a warehouse. The accuracy of location provided in meters is expressed as an integer number. In some applications, the maximum age of location information may be useful, e.g. the location information may be cached rather than directly accessed. The maximum acceptance age, in seconds, is expressed in integers.

Fig.1 shows the resource structure for device location. Following the ETSI resource structure defined in TS 102690, the deviceLocation resource contains standard attributes common for all resources such as expirationTime, accessRightID, creationTime, and lastModifiedTime (not shown in Fig.1) and specific attributes such as deviceLatitude, deviceLongitute, deviceAltitude, accuracy and timestamp. The deviceLatitude, deviceLongitude and deviceAltitude attributes represent the measured device location and the accuracy attribute represents the measurement accuracy. The timestamp attribute represents the date and time that location

[1]The authors are with the Faculty of Telecommunications at Technical University of Sofia, 8 Kl. Ohridski Blvd, Sofia 1000, Bulgaria, E-mails: nikolov.anastas@gmail.com; enp@tu-sofia.bg; iia@tu-sofia.bg

was collected. The <subscriptions> sub-resource of the deviceLocation resource contains a collection of 0..n <subscription> resources which represent active subscriptions to location information. Each <subscription> resource in addition to specified in ETSI TS 102 690 mandatory attributes has also requestedAccuracy and acceptedAccuracy attributes. The requestedAccuracy express the range in which the subscribed application wants to receive location information. The acceptedAccuracy expresses the range that the subscribed application considers to be feasable. If the location cannot be provided within this range, the application prefers not to receive the information. The <deviceDistances>, <devicePeriodicReporting> and <deviceChangeReporting> are sub-resources of the <deviceLocation> resource.

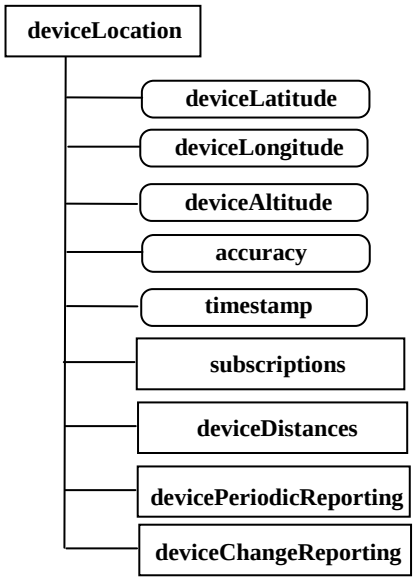

Fig.1 Structure of deviceLocation resource

Some applications may be interested in device distance from a location. The <deviceDistances> collection resource represents the collection of <deviceDistanceFrom> resources. The <deviceDistanceFrom> resource structure is shown in Fig.2, where the remoteLatitude and remoteLongitude attributes represent the latitude and longitude of the location to measure from, respectively. The distance attribute represents the distance from device to the location specified in meters. The subscriptions sub-resource of the <deviceDistances> resource contains a collection of 0..n <subscription> resources which represent active subscriptions to device distance from the specified location.

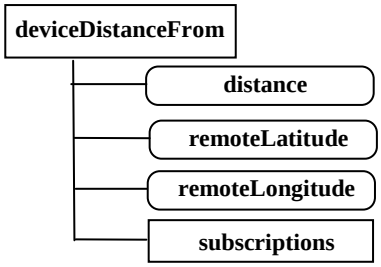

Fig.2 Structure of deviceDistanceFrom resource

Notifications of device location may be provided on a periodic basis. The periodic notifications provide location information at an application defined interval. The <devicePeriodicReproting> collection resource represents the collection of <devicePeriodicLocation> resources. Fig.3 shows the <devicePeriodicLocation> resource structure.

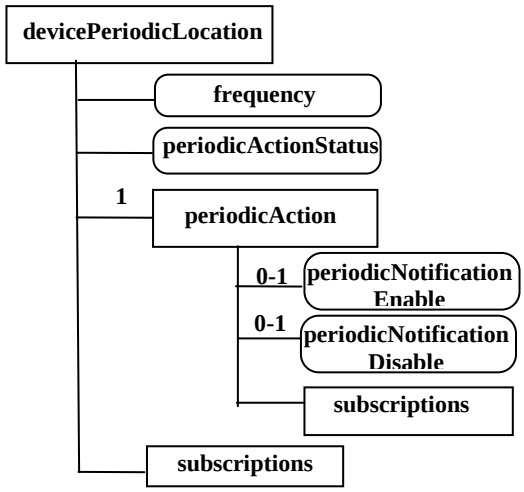

Fig.3 Structure of devicePeriodicLocation resource

The frequency attribute represents the minimum time between notifications (maximum frequency of notifications can also be considered). The periodicActionStatus attribute indicates the status of the action. The periodicAction resource represents the action. The periodicNotificationEnable attribute represents the action that enables the periodic notification. The periodicNotificationDisable attribute represents the action that disables the periodic notification.

An application can be notified of a device entering or leaving a geographical area. When a matching event occurs, a notification message will be sent to the application. An application may define a target area and notification criteria e.g. entering the target area or leaving the target area. The <deviceChangeReporting> collection resource represents the collection of <deviceLocationChange> resource. The <deviceLocationChange> resource structure for triggered location change notifications is shown in Fig.4. The locationChangeCriteria attribute is of enumerated type (entering or leaving an area). The areaLatitude, areatLongitude attributes represent the latitude and longitude of the center point, and radius attribute represents the radius of the circle around the center point in meters. The triggeredActionStatus attribute indicates the status of the action. The <triggeredAction> resource represents the action.

III. Procedures for Location Services

Each procedure related to manipulation of device location information consists of Create, Retrieve, Update and Delete.
For collection resource such as subscriptions and deviceDistances collection resources only Retrieve and Update procedures are defined for managing the retrieval and update information associated with the specified collection

resources. Such type of resource (like any other collection) cannot be created or deleted by means of request.

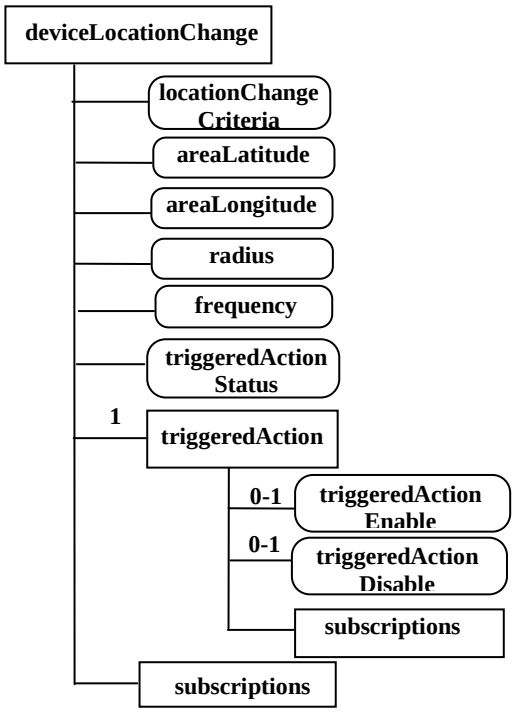

Fig.4 Structure of deviceLocationChange resource

Fig.5 illustrates the procedure used to retrieve the representation of deviceDistances collection resource. The representation includes the values of all the attributes and the references to the deviceDistanceFrom child resources for which the issuer is authorized to discover.

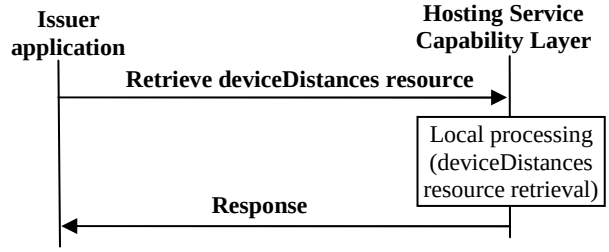

Fig.5 Retrieval of deviceDistances resource

Create procedure is to create a resource as a child of a collection resource. Fig.6 illustrates the procedure for creation of <deviceDistanceFrom> resource as a sub-resource of the <deviceDistances> collection resource.

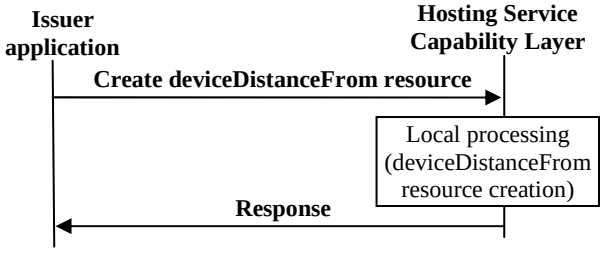

Fig.6 Creation of deviceDistanceFrom resource

Delete procedure are used to remove resources. For example, Delete <subscription> resource procedure may be used to delete an active subscription which means that the subscriber unsubscribes from notifications.

Update procedures are used to update/modify the content of existing resources. Fig.7 illustrates the procedure used to modify the attributes of the <devicePeriodicLocation> resource (e.g. frequency of notifications).

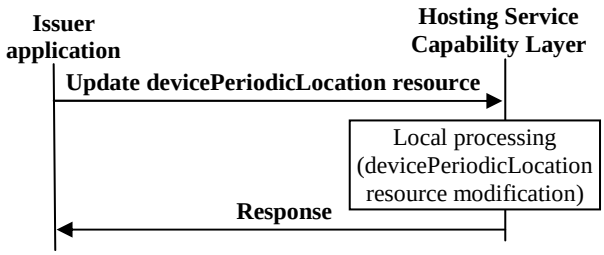

Fig.7 Update of devicePeriodicLocation resource

Table 1 lists the specific operations related to location service and the corresponding REST procedures used for these operations.

Table 1 Supported location service operations

Specific location service operations	*REST procedures*
getDeviceLocation	Retrieve <deviceLocation> resource
getDeviceDistance	Retrieve <deviceDistance> resource
startPeriodicNotification	Update <priodicAction> resource to enable
stopPeriodicNotification	Update <priodicAction> resource to disable
startTriggeredNotification	Update <triggeredAction> resource to enable
stopTriggeredNotification	Update <triggeredAction> resource to disable
deviceLocationNotification	Update <deviceLocation> resource
deviceLocationChanged	Update <deviceLocation> resource

The getDeviceLocation operation retrieves the location of a device. The getDeviceDistance operation determines the distance of a device from a location. The startPeriodic-Notification operation makes available notifications of a device location periodically, while the stopPeriodic-Notification operation disables periodic notifications about a device location. Similarly, the startTriggeredNotification operation makes available notifications of a device location change, while the stopTriggeredNotification operation disables notifications about a device location change. The deviceLocationNotification operation notifies periodicaly the location of a monitored device. The deviceLocationChanged operation notifies about location changes of a monitored device. These operations are implemented by retrieving or updating the content of the resources.

For example, in order to start device location change reporting, the <triggeredAction> resource has to be modified to triggeredActionEnable. Notify procedure is used to indicate the operation for reporting a notification about a change of a resource as a consequence of a subscription. Notify procedure is mapped to an Update procedure as the asynchronous mechanisms are used. For example, Fig.8 illustrates the notification procedures about device location as a result of periodic location reporting.

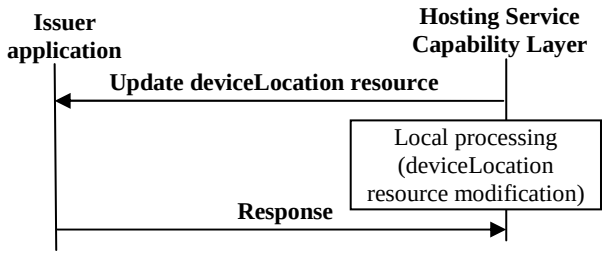

Fig.8 Periodic reporting of device location

Typical exceptions that may arise include the following: accuracy is out of limit, invalid remote location, service error, and invalid input value.

IV. Implementation Aspects

Location information may be used by different M2M applications for healthcare and transportation. Asset tracking functionality is useful in tracking of high-value assets such as intravenous pumps, wheel chairs, and stretchers within hospitals, and marking medications with RFIDs to eliminate errors in hospitals when administering medications, as well as in for tracking objects within trucks or other forms of transportation. Location information is valuable in fleet management to ascertain location of vehicles and sending dispatch notifications, in insurance for remotely monitor location and theft prevention, for emergency support, navigation, toll ticketing, remote patient monitoring, etc.

The support of the designed resource structure is up to device capabilities. Depending on its computing capabilities, the device may provide the full range of the designed operations or just some of them.

Each resource, in REST terms, can be addressed using HTTP URI. The resource is located inside a web server using the host, port, and path parts of the URI. The action to be taken is determined by the HTTP request. The HTTP request GET is used to retrieve the resource state, the HTTP request PUT is used to update the resource state. The HTTP request POST and DELETE can be used to create (to add) and destroy a resource.

Let us assume that the device runs a mobility application named deviceMobility that will generate location data. The device runs a device service capability layer (DSCL) that is configured to have the <sclBase>: HTTP://device5555.example.com. The network application that uses location services is already registered with the network service capability layer (NSCL) agreed to have <sclBase>: HTTP://mobility.com. The network application subscribes for registering of devices. When a device registers to DSCL, the DSCL requests registration through the creation of a resource under HTTP://example.com/scl/ with the resource identifier device5555, which will be the unique device identifier. The hosting NSCL responds positively to the request. After the registration of the device application deviceMobility to the DSCL and announcing to the NSCL, the device application requests the creation of a deviceLocation resource under the HTTP://example.com/scls/device5555/applications/deviceMobility/ collection with the identifier deviceLocation. The device location will be addressable through the link HTTP://example.com/scls/device5555/applications/deviceMobility/deviceLocation using HTTP GET request.

On request for device location, an HTTP response is returned where the location information is provided in the response body e.g. in JSON format: {"latitude":42.6795551, "longitude":23.2916345,"altitude":602.2, "accuracy": 20, "timestamp": 1388678400} where "timestamp" is the number of milliseconds elapsed since 1 January 1970 00:00:00 UTC (UNIX TimeStamp), the value is equal to 'Wed, 01 Jan 2014 23:00:00 GMT', and the location is Sofia, Bulgaria.

Following the same procedures the device application may request creation of child resource of deviceLocation resource.

Another protocol that may be used in REST-based architectures is CoAP (Constrained Application Protocol), which defines some primitives that allow REST on the top of TCP and UDP.

V. Conclusion

Location services may be used in different M2M application area. Design of M2M service capabilities for mobility provides access to location information that can be shared by different applications through reusable software modules. The paper presents an approach to design REST-based service capabilities for mobility. The semantic information related to mobility is synthesized by identification of basic use cases. The mobility semantic information is presented in a resource tree structure, where resources may be manipulated through their states. REST-based location services are designed by definition of operation performed on the resources. As each of the resources is uniquely addressable, it can be accessed using standard HTTP methods or CoAP primitives.

References

[1] C. Pereira, A. Aguiar, “Towards Efficient Mobile M2M Communications: Survey and Open Challenges”, *Sensors* vol.14, pp.19582-19608; DOI: 10.3390/S141019582, 2014.

[2] C. Im, C. Jeong, “ISOMP: Instant Service Orchestration Mobile M2M Platform”, *International Journal of Distributed Sensor Networks*, Hindawi, Article ID 298251, 2015.

[3] S. K. Datta, C. Bonnet, “Smart M2M Gateway Based Architecture for M2M Device and Endpoint Management”, Available at: http://www.Eurecom.Fr/Fr/Publication/4318/Download/Cm-Publi-4318.Pdf, 2014.

[4] J. Kim, J. Lee, J. Kim, J. Yun, “M2M Service Platforms: Survey, Issues, and Enabling Technologies”, *IEEE Communications Surveys & Tutorials*, vol.16, no.1, pp. 61-76, pp.2014.

[5] N. A. Surobhi, A. Jamalipour, “M2M-Based Service Coverage For Mobile Users In Post-Emergency Environments,” *IEEE Transactions On Vehicular Technology*, vol. 63, no. 7, pp. 3294-3303, 2014.

[6] S. Wahle, T. Magedanz, F. Schulze, “Demonstration of OpenMTC – M2M Solutions for Smart Cities and the Internet of Things”, Available at: http://Www.Ieeelcn.Org/Prior/LCN37/Lcn37demos/Lcndemos12_Wahle.Pdf, 2013.

[7] F. Bai, K. S. Munasinghe, A. Jamalipour, “A Novel Information Acquisition Technique For Mobile-Assisted Wireless Sensor Networks,” *IEEE Transactions on Vehicular Technology*, vol. 61, no. 4, pp.1752-1761, 2012.