Mbed Device Server M2M Interfaces v2.3

Table of Contents

Introduction
The Device Server data model
M2M server interfaces
Resource discovery (/.well-known/core)
Resource directory (/rd)
  Register
  Update registration
  De-registration
Notification (/notification)
  Make observation
  Terminate observation
  Receive notification
Authentication interface (/auth)
  Examples
Batch notifications
  TLV example
  Notifications with node's timestamps

Introduction

The Device Server provides Machine-to-Machine (M2M) interfaces through which M2M nodes interact with it. The interfaces are available over the CoAP protocol [draft-ietf-core-coap-18] and over http (M2M-HTTP), which is disabled by default. Observe and notification messages are not supported over HTTP.

For easy integration of a devices with the Device Server, use the appropriate mbed Device Client (Java and C).

The Device Server data model

The Device Server data model is divided into:

- Resources.
- Endpoints.
- Groups.
- Domains.

Resources can be perceived as individual URI paths, for example, `/path, /longer/path`.
Several resources can be assigned to a single endpoint. Resources are typically used to provide access to sensors, actuators and configuration parameters on an M2M device.
An endpoint is the web server software running on a device and it belongs to some domain.
In short, a domain contains endpoints that contain resources. Several domains can be configured in the Device Server.

Device Server endpoints can also be grouped into logical groups. Each endpoint can belong to several groups.

Device Server allows endpoints and resources to be associated with semantic naming. This means that during registration, naming metadata can be associated with them.
An endpoint includes a host name that uniquely identifies it in a domain. An endpoint can also be associated with a node type, for example 'MotionDetector'.

Resources can be associated with a resource type, for example, 'LightSensor', an interface description, and other metadata, such as content types and observability.

M2M server interfaces

The following components are available through the Device Server M2M interfaces. The M2M interfaces are accessed using CoAP.

Device Server Component	Uri path	Note
Resource	/.well-known/core	
Resource	/rd	
Authentication	/auth	Available only in secure connection (eDTLS).
Notification	/notification	Available only in M2M-HTTP interface.

Note that the default port for non-encrypted communication is 5683 and for encrypted 5684. You can configure the ports in the `device-server.properties` file.

Resource discovery (/.well-known/core)

The well-known CoRE resource includes the M2M interfaces that the Device Server offers under /.well-known/core with the REST interface defined in RFC 6690. An M2M node uses it to discover the Device Server's interface automatically. Note that this interface does not support query filtering.

To retrieve the Device Server resources, use this method:

```
GET /.well-known/core
```

The /.well-known/core resource contains the following link-format descriptions:

```
</rd>;rt="core-rd"
```

Resource directory (/rd)

The resource directory enables the registration of endpoints with the Device Server. An endpoint is registered by making a request to /rd. The request query string may contain any of the endpoint attributes defined above, and the request body is a list of the resources to register for that endpoint in the Link Format. Each link may contain any of the resource attributes defined above.

A registration with the resource directory has a soft state, and must be refreshed periodically. A non-default lifetime can be specified by including the lifetime parameter in the request. All the resources that are previously stored in the resource directory but are not delivered in the request body, are removed, and all the new resources are added to the resource directory.

Register

To register an endpoint, use this method:

```
POST /rd?ep={endpoint-name}&et={endpoint-type}&lt={lifetime}&con={context-address}&d={domain}&b={binding}
```

This request registers an endpoint with the Device Server with specified query parameters and a list of resources formatted in the Link Format (RFC 6690) as the payload of the message.

The following attributes are supported:

Attribute	Name	Restrictions
url	Resource path	
rt	Resource type	Only once for registration
if	Interface description	Only once
ct	CoAP Content type	Convert Ascii MIME type
obs	Observable	Boolean
aobs	Auto-observable	Boolean
id	Resource id (used in auto observation)	Integer

The following query parameters are supported:

Query parameter	Name	Restrictions
ep	Endpoint name. A unique name for the registering node in a domain.	(Optional) 63 Bytes. If the parameter is not present in the request, the name is generated. The name has to be RFC1123 compliant, with the exclusion that dots(.) are not allowed in a hostname.
et	Endpoint type	(Optional) 63 Bytes
lt	Lifetime. Number of seconds that this registration will be valid for. Must be updated within this time, or will be removed.	(Optional) 32-bit uint
d	Domain name	(Optional) 63 bytes. If this parameter is missing, a default domain is assumed.
con	Address at which this endpoint is available. In the absence of this parameter, the scheme of the protocol, a source address of the register request is assumed. This parameter is mandatory for M2M-HTTP binding.	(Optional for CoAP) scheme:// host:port
b	Indicates the current binding mode and queue mode of the registering device. U - udp, Q - queue mode, S - sms binding. On M2M-HTTP interface this parameter is not supported.	(Optional) SMS binding is not supported.

The following deprecated query parameters are still supported:

- h - endpoint name
- rt - endpoint type

Response codes:

CoAP code	Description
2.01 Created	Successful registration, returns the uri-location (that is used in registration updated) and max-age.
4.00 Bad request	Malformed message: - missing required query parameter - template not found - domain does not exist

Example:

```
POST /rd?h=node-001&rt=Light&lt=6000
Uri-Host: domain.com

</light>;rt="ucum:Lux";if="ns.wadl#s",</uv-light>;rt="ns:image";if="ns.wadl#s"
=>
2.01 Created
Location-Path: /rd/domain.com/node-001
```

Template

Registrations can be done by using endpoint templates described in the configuration guide. Templates allow predefined resource sets to be used for a registered endpoint. A template is mapped by an endpoint type.

Update registration

Endpoint information can be updated by sending a PUT request to /rd/{reg-location-path}. Resources can also be added by including a body in the request. The resources delivered in the body of the messages are handled as new resources for the endpoint.

To update endpoint information, use this method:

```
PUT /rd/{domain}/{name}?lt={lifetime}
```

Example:

```
PUT /rd/domain.com/node-001?lt=6000
=>
2.04 Changed
```

M2M-HTTP

On the Http interface the *con* parameter is mandatory for Update registration.

De-registration

Endpoint information can be removed by sending a DELETE request to /rd/{reg-location-path}. The message does not contain a body and all the resources associated with the deleted endpoint are removed from the resource directory.

To remove endpoint information, use this method:

```
DELETE /rd/{domain}/{name}
```

Example:

```
DELETE /rd/domain.com/node-001
=>
2.02 Deleted
```

Notification (/notification)

This interface used by endpoints that are communication with the Device Server through the proprietary HTTP protocol.

Make observation

Observation request is made by the Device Server as a GET operation, where the `X-Observe` header is set to `0` and the `X-Token` header holds the observation identifier generated by the server. In successful cases endpoints must respond with the `X-Observer` header as 0 and the actual value of the observed resource.

```
GET /dev/resource/path
X-Observe: 0
X-Token: 0x1472
=>
HTTP/1.1 200 OK
X-Observe: 0
X-Token: 0x1472
Content-Type: text/plain
Cache-Control: max-age=3600

21C
```

When resource does not exist then endpoint must respond with HTTP status `404`.

```
HTTP/1.1 404 ...
```

In some cases, where the resource exists but not observable then endpoint must respond with HTTP OK but `X-Observe` header must be omitted.

```
HTTP/1.1 200 OK
```

Terminate observation

In order to terminate subscription, server sends the same GET operation as in case of observation requests but the `X-Observe` header is set to `1`. Endpoint must cancel observing the resource.

```
GET /dev/resource/path
X-Observe: 1
X-Token: 0x1472
->
HTTP/1.1 200 OK
X-Observe: 1
X-Token: 0x1472
```

Receive notification

Notifications are sent by the endpoints using Device Server's M2M interface. Endpoint name is mandatory and together with the domain it is used to identify the endpoint. The `con` parameter is optional and if it is set then the context address of the endpoint, where server initiated requests are sent, will be changed.

The `X-Token` header must be the same as sent by the Device Server while making the observation requests. The `X-Observe` header must be an increasing number that helps identifying the order of the values for the same observation.

```
POST /notification?ep={endpoint-name}&d={domain}[&con=scheme://host:port]
X-Observe: ??+1
X-Token: 0x1472
Content-Type: text/plain
Cache-Control: max-age=3600

21C
->
HTTP/1.1 200 OK
```

In error cases, where the endpoint is unknown for the Device Server or the resource is not observed or observation has already been terminated, the server sends HTTP error code `404`.

```
-> HTTP/1.1 404 ...
```

Authentication interface (/auth)

With authentication functionality the PANA server can request EDTLS pre-shared secret of any endpoint. This is enabled only via eDTLS secured UDP connection. PANA server must be defined in whitelist.csv with Authentication enabled. Successful response contains secret of requested keyID in hex format.

```
GET /auth?pskid={pre-shared-key-id (HEX)}
accept: 42 | 0 (APPLICATION/OCTET-STREAM | TEXT/PLAIN)
```

Response codes:

CoAP code	Description
2.05 Content	Returns valid secret (as raw bytes or HEX)
4.01 Unauthorized	Key-id could not be found whitelist
4.03 Forbidden	Requester does not have permission to access this interface

Examples

Batch notifications

The Device Server binary TLV format is used to package a batch of resource representations in a single payload. This format can be used for the Device Server root resource (/) or for a function set resource (e.g. /sen). This format can be used as the payload in notifications, or in the response to a GET request.

Media Type	CoAP Code
application/nanoservice-tlv	200

In this format the ID field corresponds to the id= integer attribute assigned to the resource by the node. This same id= attribute is used for auto-observation. The binary format is simply the following structure repeated for each representation in the payload. The overall length of the payload is calculated from the IP packet length, thus there is no length field.

Field	Format and Length	Description
ID	16-bit unsigned integer	The Device Server id= link attribute corresponding to the resource
Length	16-bit unsigned integer	The Length of the following field in bytes
Value	Sequence of bytes, length indicated by the Length field	Either a plain text or opque value depending on the Resource's data format

Reserved ID Values:

ID	Name	Description
0	Current Timestamp	MUST be Unix timestamp in seconds since 1970. Value represented as a string
1	Time Offset	The time in seconds before the Current Timestamp that the following measurements were made. Value represented as a string.

TLV example

In this example a node has two resources and the following link description:

```
</sen>;aobs;id="20";ct="200",
</sen/temp>;aobs;id="25",
</sen/light>;aobs;id="26"
```

When the node sends a notification to the server for /sen, or responds to a GET on /sen, when temp=22.3 and light=100 the payload would be as follows:

Notifications with node's timestamps

In order to send resource notification with timestamp, TLV batch message must contain current timestamp and optionally offset. Every following resource representation has measurement time of current timestamp minus offset. It is possible to have multiple time series by including multiple timest offset. Note that current timestamp must appear always first and only one time.

Byte #	Value	HEX	Note
0	0	00	id=25
1	25	19	
2	0	00	length=4
3	4	04	
4	'2'	32	value=22.3
5	'2'	32	
6	'.'	2e	
7	'3'	33	
8	0	00	id=26
9	26	1a	
10	0	00	length=3
11	3	03	
12	'1'	31	value=100
13	'0'	30	
14	'0'	30	

Example with latest measurements:

ID	Value	Note	TLV raw data (hex)
0	1357056000	2013.01.01 10:00	`ID LEN VALUE` 0000 000A 3133 3537 30353 6303 030
25	20.1	temperature-value=20.1 timestamp='2013.01.01 10:00'	0019 0004 3230 2e31
26	98	light-value=20.1 timestamp='2013.01.01 10:00'	001A 0004 3938

Example with historical measurements:

ID	Value	Note	TLV raw data (hex)
0	1357056000	2013.01.01 10:00	`ID LEN VALUE` 0000 000A 3133 3537 3035 3630 3030
1	300	5 minutes offset	0001 0003 3330 30
25	20.1	temperature-value=20.1 timestamp='2013.01.01 9:55'	0019 0004 3230 2e31
26	98	light-value=20.1 timestamp='2013.01.01 9:55'	001A 0004 3938
1	600	10 minutes offset	0001 0003 3630 30
25	22.0	temperature-value=22.0 timestamp='2013.01.01 9:50'	0019 0004 3232 2e30