M2M Interfaces v1.9
The NanoService Platform provides Machine-to-Machine (M2M) interfaces, through which M2M nodes interact with NSP. These interfaces are available over the CoAP protocol [draft-ietf-core-coap-12].

For easy integration of a device with the NanoService Platform, use the appropriate NanoService Device Library (Java and C).

The NanoService Platform Data Model

The NSP data model is divided into domains, groups, end-points and resources. Resources can be perceived as individual URI paths (e.g. /path, /longer/path) and as such there can be several resources assigned for a single end-point. Resources are typically used to provide access to sensors, actuators and configuration parameters on an M2M device. An end-point is the web server software running on a device, and it belongs to some domain. In short, domain contains end-points and end-points contain resources and there can be several domains configured in the NanoService Platform.

NanoServices allows end-points and resources to be associated with semantic naming, and during registration naming meta-data can be associated with them. An end-point includes a host name, which uniquely identifies the end-point in a domain. An end-point can also be associated with a node type, for example 'MotionDetector'. Finally resources can be associated with a resource type (e.g. 'LightSensor'), an interface description and other meta-data about the resource such as its content-types and observability.

M2M Server Interfaces

The following components are available through the NSP M2M interfaces. These M2M interfaces are accessed using CoAP.
<table>
<thead>
<tr>
<th>NSP Component</th>
<th>Uri path</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource discovery</td>
<td>/.well-known/core</td>
<td></td>
</tr>
<tr>
<td>Resource directory</td>
<td>/rd</td>
<td></td>
</tr>
<tr>
<td>Authentication</td>
<td>/auth</td>
<td>Available only in secure connection (eDTLS)</td>
</tr>
</tbody>
</table>

Note that default port for non-encrypted communication is 5683 and for encrypted is 5684. Those can be configured in `nsp.properties`.

**Resource Discovery (/.well-known/core)**

The well-known CoRE resource includes the NSP's offered M2M interfaces under /.well-known/core with the REST interface defined in RFC 6690. This is used by an M2M node to discover the NSP's interface automatically. Note that this interface does not support query filtering.

```
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 end-points to the NanoService Platform. End-point registration is done by making a request to /rd. The request query string may contain any of the end-point attributes defined above, and the request body is a list of the resources to register for that end-point in the Link Format. Each link may contain any of the resource attributes defined above. The Uri-Host option of the request is used to indicate the domain this end-point is related to. If no Uri-Host is included, then the end-point is associated with the configurable default NSP domain. A registration with the resource directory has 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**

```
POST /rd?h={endpoint-name}&rt={endpoint-type}&lt={lifetime}
content-format: 0 (text/plain)
Uri-Host: {domain-name}
```

Registers endpoint to NSP with specified query parameters and list of resources formatted in 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 
os | Observable | Boolean 
aobs | Auto-observable | Boolean 
id | Resource id (used in auto observation) | Integer 

Supported query parameters:

<table>
<thead>
<tr>
<th>Query parameter</th>
<th>Name</th>
<th>Restrictions</th>
</tr>
</thead>
</table>
h | End-point name. A unique name for the registering node in domain. | (Optional) 63 Bytes. If parameter is not present in the request, the name is generated. Name has to be RFC1123 compliant, with the exclusion that dots(.) are not allowed in hostname. 
rt | End-point 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 

Response codes:

<table>
<thead>
<tr>
<th>CoAP code</th>
<th>Description</th>
</tr>
</thead>
</table>
| 2.01 Created | Successful registration, returns 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 

**Template**

Registrations can be done by using endpoint templates described in the configuration guide. Templates allow predefined resource sets to be used for registered endpoint. Template is mapped by endpoint type.
**Update registration**

Updating end-point information can be performed 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 end-point.

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

**De-registration**

End-point 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 to the deleted end-point are removed from the resource directory.

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

**Example**

Register:

```plaintext
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
```

Update registration

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

De-register:

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

**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.

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

<table>
<thead>
<tr>
<th>CoAP code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.05</td>
<td>Content Returns valid secret (as raw bytes or HEX)</td>
</tr>
<tr>
<td>4.01</td>
<td>Unauthorized Key-id could not be found white-list</td>
</tr>
<tr>
<td>4.03</td>
<td>Forbidden Requester does not have permission to access this interface</td>
</tr>
</tbody>
</table>

Examples

**CoAP HTTP Proxy Interface**

The CoAP HTTP proxy interface can be used to access HTTP URI resources with CoAP messages. The NSP platform performs protocol translation between CoAP and HTTP so that HTTP resources can be accessed using CoAP requests. In addition to providing access support for full http URI:s the interface also provides the possibility to use CoAP side URI shortcuts to both decrease required message size on the CoAP side and to decouple the actual HTTP URI from the CoAP URI if necessary.

Accessing web resources directly can be done by providing the CoAP Proxy-Uri option in the request parameters. The URI needs to be the absolute URI path (scheme://domain:port/path?query_string) to the HTTP resource. At minimum the scheme and domain parts have to be defined for the request.

The URI shortcut can be used by providing URI-Host and URI-Path options in the request. Uri-Host is mapped to an NSP domain and Uri-Path is mapped into a domain specific shortcut within the platform. The domain can be configured with a default shortcut and if only the Uri-Host option is provided the domain default shortcut is used. Further information on how to configure domain shortcuts and the default shortcut can be found from the NSP installation guide.

Shortcut functionality supersedes Proxy-Uri functionality. If both are present, the shortcut option is used.

| Attribute | Description                                           | Restrictions                                                      |
|-----------|-------------------------------------------------------|                                                                   |
| Proxy-Uri | Absolute HTTP URI to be accessed via the proxy        | scheme://domain:port/path?query_string                             |
| Uri-Host  | NSP domain under which the URI shortcut is used       | Domain needs to be found in the proxy URI shortcut configuration.   |
| Uri-Path  | Domain shortcut that is used to resolve the eventual request URI | Shortcut needs to be found under the specified domain in the proxy configuration. |

**Batch notifications**

The NanoService binary TLV format is used to package a batch of resource representations in a single payload. This format can be used for the NanoService
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.

<table>
<thead>
<tr>
<th>Media Type</th>
<th>CoAP Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>application/nanoservice-tlv</td>
<td>200</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Format and Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>16-bit unsigned integer</td>
<td>The NanoService id= link attribute corresponding to the resource</td>
</tr>
<tr>
<td>Length</td>
<td>16-bit unsigned integer</td>
<td>The Length of the following field in bytes</td>
</tr>
<tr>
<td>Value</td>
<td>Sequence of bytes, length indicated by the Length field</td>
<td>Either a plain text or opaque value depending on the Resource’s data format</td>
</tr>
</tbody>
</table>

Reserved ID Values:

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Current Timestamp</td>
<td>MUST be Unix timestamp in seconds since 1970. Value represented as a string</td>
</tr>
<tr>
<td>1</td>
<td>Time Offset</td>
<td>The time in seconds before the Current Timestamp that the following measurements were made. Value represented as a string.</td>
</tr>
</tbody>
</table>

**TLV example**

Here is an example, where 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:**

<table>
<thead>
<tr>
<th>ID</th>
<th>Value</th>
<th>Note</th>
<th>TLV raw data (hex)</th>
</tr>
</thead>
</table>
| 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:**
<table>
<thead>
<tr>
<th>ID</th>
<th>Value</th>
<th>Note</th>
<th>TLV raw data (hex)</th>
</tr>
</thead>
</table>
| 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                                     |