Reusable Interface Definitions for Constrained RESTful Environments
draft-ietf-core-interfaces-06

Abstract

   This document defines a set of Constrained RESTful Environments
   (CoRE) Link Format Interface Descriptions [RFC6690] applicable for
   use in constrained environments. These include the: Actuator,
   Paramter, Read-only parameter, Sensor, Batch, Linked Batch and Link
   List interfaces.

   The Batch, Linked Batch and Link List interfaces make use of resource
   collections. This document further describes how collections relate
   to interfaces.

   Many applications require a set of interface descriptions in order
   provide the required functionality. This document defines the
   concept of function sets to specify this set of interfaces and
   resources.

   _Editor's note: The git repository for the draft is found at
   https://github.com/core-wg/interfaces_

   This Internet-Draft will expire on May 1, 2017.

Copyright Notice

Table of Contents

1. Introduction

   IETF Standards for machine to machine communication in constrained
   environments describe a REST protocol and a set of related
   information standards that may be used to represent machine data and
   machine metadata in REST interfaces. CoRE Link-format is a standard
   for doing Web Linking [RFC5988] in constrained environments. SenML
   [I-D.ietf-core-senml] is a simple data model and representation
   format for composite and complex structured resources. CoRE Link-
   Format and SenML can be used by CoAP [RFC7252] or HTTP servers.

   The discovery of resources offered by a constrained server is very
   important in machine-to-machine applications where there are no
   humans in the loop. Machine application clients must be able to
   adapt to different resource organizations without advance knowledge
   of the specific data structures hosted by each connected thing. The
   use of Web Linking for the description and discovery of resources
   hosted by constrained web servers is specified by CoRE Link Format
   [RFC6690]. CoRE Link Format additionally defines a link attribute
   for interface description ("if") that can be used to describe the
   REST interface of a resource, and may include a link to a description
   document.

   This document defines a set of Link Format interface descriptions for
   some common design patterns that enable the server side composition
   and organization, and client side discovery and consumption, of
   machine resources using Web Linking. A client discovering the "if"

link attribute will be able to consume resources based on its
knowledge of the expected interface types. In this sense the
Interface Type acts in a similar way as a Content-Format, but as a
selector for a high level functional abstraction.

An interface description describes a resource in terms of it's
associated content formats, data types, URI templates, REST methods,
parameters, and responses. Basic interface descriptions are defined
for sensors, actuators, and properties.

A set of collection types is defined for organizing resources for
discovery, and for various forms of bulk interaction with resource
sets using typed embedding links.

Interface descriptions may be used in the composition of Function
Sets and Profiles. Function Sets and Profiles are described and an
example is given of a sensor and actuator device profile using
Function Sets composed from the interface descriptions described in
this document.

This document first defines the concept of collection interface
descriptions. It then defines a number of generic interface
descriptions that may be used in contrained environments. Several of
these interface descriptions utilise collections. The interface
descriptions are then used by the function sets.

Whilst this document assumes the use of CoAP [RFC7252], the REST
interfaces described can also be realized using HTTP [RFC7230].

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and
"OPTIONAL" in this document are to be interpreted as described in
[RFC2119].

This document requires readers to be familiar with all the terms and
concepts that are discussed in [RFC5988] and [RFC6690]. This
document makes use of the following additional terminology:

Device: An IP smart object running a web server that hosts a group
   of Function Set instances from a profile.

Function Set: A group of well-known REST resources that provides a
   particular service.

Gradual Reveal: A REST design where resources are discovered
   progressively using Web Linking.

Interface Description: The Interface Description describes the
   generic REST interface to interact with a resource or a set of
   resources. Its use is described via the Interface Description
   'if' attribute which is an opaque string used to provide a name or
   URI indicating a specific interface definition used to interact
   with the target resource. One can think of this as describing
   verbs usable on a resource.

Profile: A group of well-known Function Sets defined by a
   specification.

Resource Discovery: The process allowing a web client to identify
   resources being hosted on a web server.

Service Discovery: The process making it possible for a web client
   to automatically detect devices and Function Sets offered by these
   devices on a CoRE network.

3. Collections

3.1. Introduction to Collections

A Collection is a resource which represents one or more related
resources. [RFC6573] describes the "item" and "collection" Link
Relation. "item" link relation identifies a member of collection.
"collection" indicates the collection that an item is a member of.
For example: A collection might be a resource representing catalog of
products, an item is a resource related to an individual product.

Section 1.2.2/[RFC6690] also describes resource collections.

This document uses the concept of "collection" and applies it to
interface descriptions. A collection interface description consists
of a set of links and a set of items pointed to by the links which
may be sub-resources of the collection resource. The collection
interface descriptions described in this document are Link List,
Batch and Linked Batch.

The links in a collection are represented in CoRE Link-Format
Content-Formats including JSON and CBOR variants, and the items in
the collection may be represented by senml, including JSON and CBOR
variants. In general, a collection may support items of any
available Content-Format.

A particular resource item may be a member of more than one
collection at a time by being linked to, but may only be a
subresource of one collection.

Some collections may have pre-configured items and links, and some
collections may support dynamic creation and removal of items and
links. Likewise, modification of items in some collections may be
permitted, and not in others.

Collections may support link embedding, which is analogous to an
image tag (link) causing the image to display inline in a browser
window. Resources pointed to by embedded links in collections may be
interacted with using bulk operations on the collection resource.
For example, performing a GET on a collection resource may return a
single representation containing all of the linked resources.

Links in collections may be selected for processing by a particular
request by using Query Filtering as described in CoRE Link-Format
[RFC6690].

3.2. Use Cases for Collections

Collections may be used to provide gradual reveal of resources on an
endpoint. There may be a small set of links at the .well-known/core
location, which may in turn point to other collections of resources
that represent device information, device configuration, device
management, and various functional clusters of resources on the
device.

A collection may provide resource encapsulation, where link embedding
may be used to provide a single resource with which a client may
interact to obtain a set of related resource values. For example, a
collection for manufacturer parameters may consist of manufacturer
name, date of manufacture, location of manufacture, and serial number
resources which can be read as a single senml data object.

A collection may be used to group a set of like resources for bulk
state update or actuation. For example, the brightness control
resources of a number of luminaries may be grouped by linking to them
in a collection. The collection type may support receiving a single
update form a client and sending that update to each resource item in
the collection.

Items may be sub-resources of the collection resource. This enables
updates to multiple items in the collection to be processed together
within the context of the collection resource.

3.3. Content-Formats for Collections

The collection interfaces by default use CoRE Link-Format for the
link representations and SenML or text/plain for representations of
items. The examples given are for collections that expose resources

and links in these formats. In addition, a new "collection" Content-
Format is defined based on the SenML framework which represents both
links and items in the collection.

The choice of whether to return a representation of the links or of
the items or of the collection format is determined by the accepts
header option in the request. Likewise, the choice of updating link
metadata or item data or the collection resource itself is determined
by the Content-Format option in the header of the update request
operation.

The default Content-Formats for collection types described in this
document are:

Links: application/link-format, application/link-format+json

Items: application/senml+json, text/plain

3.4. Links and Items in Collections

Links use CoRE Link-Format representation by default and may point to
any resource reachable from the context of the collection. This
includes absolute links and links that point to other network
locations if the context of the collection allows. Links to sub-
resources in the collection MUST have a path-element starting with
the resource name, as per [RFC3986]. Links to resources in the
global context MUST start with a root path identifier [RFC5988].
Links to other collections are formed per [RFC3986].

Examples of links:

</sen/>;if="core.lb"': Link to the /sen/ collection describing it as
   a core.lb type collection (Linked Batch)

</sen/>;rel="grp"': Link to the /sen/ collection indicating that
   /sen/ is a member of a group in the collection in which the link
   appears.

<"/sen/temp">;rt="temperature"': An absolute link to the resource at
   the path /sen/temp

<temp>;rt="temperature": Link to the temp subresource of the
   collection in which this link appears."

<temp>;anchor="/sen/": A link to the temp subresource of the
   collection /sen/ which is assumed not to be a subresource of the
   collection in which the link appears ,but is expected to be
   identified in the collection by resource name.

Links in the collection MAY be Read, Updated, Added, or Removed using the CoRE Link-Format or JSON Merge-Patch Content-Formats on the collection resource. Reading links uses the GET method and returns an array or list containing the link-values of all selected links. Links may be added to the collection using POST or PATCH methods. Updates to links MUST use the PATCH method and MAY use query filtering to select links for updating. The PATCH method on links MUST use the JSON Merge-Patch Content-Format (application/merge-patch+json) specified in [RFC7396].

Items in the collection SHOULD be represented using the SenML (application/senml+json) or plain text (text/plain) Content-Formats, depending on whether the representation is of a single data point or multiple data points. Items MAY be represented using any supported Content-Format.

Link Embedding enables the bulk processing of items in the collection using a single operation targeting the collection resource. A subset of resources in the collection may be selected for operation using Query Filtering. Bulk Read operations using GET return a SenML representation of all selected resources. Bulk item Update operations using PUT or POST apply the payload document to all selected resource items in the collection, using either a Batch or Group update policy. A Batch update is performed by applying the resource values in the payload document to all resources in the collection that match any resource name in the payload document. Group updates are performed by applying the payload document to each item in the collection. Group updates are indicated by the link relation type rel="grp" in the link.

3.5. Queries on Collections

Collections MAY support query filtering as defined in CoRE Link-Format [RFC6690]. Operations targeting either the links or the items MAY select a subset of links and items in the collection by using query filtering. The Content-Format specified in the request header selects whether links or items are targeted by the operation.

3.6. Observing Collections

Resource Observation via [I-D.ietf-core-dynlink] using CoAP [RFC7252] MAY be supported on items in a collection. A subset of the conditional observe parameters MAY be specified to apply. In most cases pmin and pmax are useful. Resource observation on a collection's items resource MAY report any changes of resource state in any item in the collection. Observation Responses, or notifications, SHOULD provide representations of the resources that have changed in SenML Content-Format. Notifications MAY include

multiple observations of a particular resource, with SenML time
stamps indicating the observation times.

3.7. Collection Types

There are three collection types defined in this document:

```
+----------------+---------+--------------------+
| Collection Type | if=     | Content-Format    |
+----------------+---------+--------------------+
| Link List       | core.ll | link-format       |
|                 |         |                    |
| Batch           | core.b  | link-format, senml |
|                 |         |                    |
| Linked Batch    | core.lb | link-format, senml |
+----------------+---------+--------------------+
```

Table 1: Collection Type Summary

The interface description defined in this document describes the
methods and functions that may be applied to the collections.

4. Interface Descriptions

This section defines REST interfaces for Link List, Batch, Sensor,
Parameter and Actuator resources. Variants such as Linked Batch or
Read-Only Parameter are also presented. Each type is described along
with its Interface Description attribute value and valid methods.
These are defined for each interface in the table below. These
interfaces can support plain text and/or SenML Media types.

The if= column defines the Interface Description (if=) attribute
value to be used in the CoRE Link Format for a resource conforming to
that interface. When this value appears in the if= attribute of a
link, the resource MUST support the corresponding REST interface
described in this section. The resource MAY support additional
functionality, which is out of scope for this document. Although
these interface descriptions are intended to be used with the CoRE
Link Format, they are applicable for use in any REST interface
definition.

The Methods column defines the methods supported by that interface,
which are described in more detail below.

```
+--------------+---------+----------------+-------------------+
| Interface    | if=     | Methods        | Content-Formats   |
+--------------+---------+----------------+-------------------+
| Link List    | core.ll | GET            | link-format       |
|              |         |                |                   |
| Batch        | core.b  | GET, PUT, POST | link-format, senml|
|              |         |                |                   |
| Linked Batch | core.lb | GET, PUT, POST,| link-format, senml|
|              |         |                |                   |
|              |         | DELETE         |                   |
|              |         |                |                   |
| Sensor       | core.s  | GET            | link-format,      |
|              |         |                |                   |
|              |         |                | text/plain        |
|              |         |                |                   |
| Parameter    | core.p  | GET, PUT       | link-format,      |
|              |         |                |                   |
|              |         |                | text/plain        |
|              |         |                |                   |
| Read-only    | core.rp | GET            | link-format,      |
|              |         |                |                   |
| Parameter    |         |                | text/plain        |
|              |         |                |                   |
| Actuator     | core.a  | GET, PUT, POST | link-format,      |
|              |         |                |                   |
|              |         |                | text/plain        |
+--------------+---------+----------------+-------------------+
```

                 Table 2: Interface Description Summary

   The following is an example of links in the CoRE Link Format using
   these interface descriptions. The resource hierarchy is based on a
   simple profile defined in Appendix B. These links are used in the
   subsequent examples below.

   Req: GET /.well-known/core
   Res: 2.05 Content (application/link-format)
   </s/>;rt="simple.sen";if="core.b",
   </s/lt>;rt="simple.sen.lt";if="core.s",
   </s/tmp>;rt="simple.sen.tmp";if="core.s";obs,
   </s/hum>;rt="simple.sen.hum";if="core.s",
   </a/>;rt="simple.act";if="core.b",
   </a/1/led>;rt="simple.act.led";if="core.a",
   </a/2/led>;rt="simple.act.led";if="core.a",
   </d/>;rt="simple.dev";if="core.ll",
   </l/>;if="core.lb",

                    Figure 1: Binding Interface Example

[4.1](#). **Link List**

The Link List interface is used to retrieve (GET) a list of resources on a web server. The GET request SHOULD contain an Accept option with the application/link-format content format; however if the resource does not support any other form of GET methods the Accept option MAY be elided. The Accept option SHOULD only include the application/link-format content format.

Editor's note: This use of Accept is not very clear, should probably explain this is due to this interface description being extended by Batch and Linked Batch later.

The request returns a list of URI references with absolute paths to the resources as defined in CoRE Link Format. This interface is typically used with a parent resource to enumerate sub-resources but may be used to reference any resource on a web server.

Link List is the base interface to provide gradual reveal of resources on a CoRE web server, hence the root resource of a Function Set SHOULD implement this interface or an extension of this interface.

The following example interacts with a Link List /d containing Parameter sub-resources /d/name, /d/model.

```
Req: GET /d/ (Accept:application/link-format)
Res: 2.05 Content (application/link-format)
</d/name>;rt="simple.dev.n";if="core.p",
</d/model>;rt="simple.dev.mdl";if="core.rp"
```

[4.2](#). **Batch**

The Batch interface is used to manipulate a collection of sub-resources at the same time. The Batch interface description supports the same methods as its sub-resources, and can be used to read (GET), update (PUT) or apply (POST) the values of those sub-resource with a single resource representation. The sub-resources of a Batch MAY be heterogeneous, a method used on the Batch only applies to sub-resources that support it. For example Sensor interfaces do not support PUT, and thus a PUT request to a Sensor member of that Batch would be ignored. A batch requires the use of SenML Media types in order to support multiple sub-resources.

In addition, The Batch interface is an extension of the Link List interface and in consequence MUST support the same methods.

_Editor's note: hould probably explain that this means doing a GET
with an Accept:application/link-format will return the sub-resource
links_

The following example interacts with a Batch /s/ with Sensor sub-
resources /s/light, /s/temp and /s/humidity.

```
Req: GET /s/
Res: 2.05 Content (application/senml+json)
{"e":[
    { "n": "light", "v": 123, "u": "lx" },
    { "n": "temp", "v": 27.2, "u": "degC" },
    { "n": "humidity", "v": 80, "u": "%RH" }],
}
```

4.3. Linked Batch

The Linked Batch interface is an extension of the Batch interface.
Contrary to the basic Batch which is a collection statically defined
by the web server, a Linked Batch is dynamically controlled by a web
client. A Linked Batch resource has no sub-resources. Instead the
resources forming the batch are referenced using Web Linking
[RFC5988] and the CoRE Link Format [RFC6690]. A request with a POST
method and a content format of application/link-format simply appends
new resource links to the collection. The links in the payload MUST
reference a resource on the web server with an absolute path. A
DELETE request removes the entire collection. All other requests
available for a basic Batch are still valid for a Linked Batch.

The following example interacts with a Linked Batch /l/ and creates a
collection containing /s/light, /s/temp and /s/humidity in 2 steps.

```
Req: POST /l/ (Content-Format: application/link-format)
</s/light>,</s/temp>
Res: 2.04 Changed

Req: GET /l/
Res: 2.05 Content (application/senml+json)
{"e":[
   { "n": "/s/light", "v": 123, "u": "lx" },
   { "n": "/s/temp", "v": 27.2, "u": "degC" },
}

Req: POST /l/ (Content-Format: application/link-format)
</s/humidity>
Res: 2.04 Changed

Req: GET /l/ (Accept: application/link-format)
Res: 2.05 Content (application/link-format)
</s/light>,</s/temp>,</s/humidity>

Req: GET /l/
Res: 2.05 Content (application/senml+json)
{"e":[
   { "n": "/s/light", "v": 123, "u": "lx" },
   { "n": "/s/temp", "v": 27.2, "u": "degC" },
   { "n": "/s/humidity", "v": 80, "u": "%RH" }],
}

Req: DELETE /l/
Res: 2.02 Deleted
```

4.4. Sensor

The Sensor interface allows the value of a sensor resource to be read
(GET). The Media type of the resource can be either plain text or
SenML. Plain text MAY be used for a single measurement that does not
require meta-data. For a measurement with meta-data such as a unit
or time stamp, SenML SHOULD be used. A resource with this interface
MAY use SenML to return multiple measurements in the same
representation, for example a list of recent measurements.

The following are examples of Sensor interface requests in both text/
plain and application/senml+json.

```
Req: GET /s/humidity (Accept: text/plain)
Res: 2.05 Content (text/plain)
80

Req: GET /s/humidity (Accept: application/senml+json)
Res: 2.05 Content (application/senml+json)
{"e":[
    { "n": "humidity", "v": 80, "u": "%RH" }],
}
```

4.5. Parameter

The Parameter interface allows configurable parameters and other
information to be modeled as a resource. The value of the parameter
can be read (GET) or update (PUT). Plain text or SenML Media types
MAY be returned from this type of interface.

The following example shows request for reading and updating a
parameter.

```
Req: GET /d/name
Res: 2.05 Content (text/plain)
node5

Req: PUT /d/name (text/plain)
outdoor
Res: 2.04 Changed
```

4.6. Read-only Parameter

The Read-only Parameter interface allows configuration parameters to
be read (GET) but not updated. Plain text or SenML Media types MAY
be returned from this type of interface.

The following example shows request for reading such a parameter.

```
Req: GET /d/model
Res: 2.05 Content (text/plain)
SuperNode200
```

4.7. Actuator

The Actuator interface is used by resources that model different
kinds of actuators (changing its value has an effect on its
environment). Examples of actuators include for example LEDs,
relays, motor controllers and light dimmers. The current value of
the actuator can be read (GET) or the actuator value can be updated
(PUT). In addition, this interface allows the use of POST to change

the state of an actuator, for example to toggle between its possible
values. Plain text or SenML Media types MAY be returned from this
type of interface. A resource with this interface MAY use SenML to
include multiple measurements in the same representation, for example
a list of recent actuator values or a list of values to updated.

The following example shows requests for reading, setting and
toggling an actuator (turning on a LED).

```
Req: GET /a/1/led
Res: 2.05 Content (text/plain)
0

Req: PUT /a/1/led (text/plain)
1
Res: 2.04 Changed

Req: POST /a/1/led (text/plain)
Res: 2.04 Changed

Req: GET /a/1/led
Res: 2.05 Content (text/plain)
0
```

5. Function Sets and Profiles

This section defines how a set of REST resources can be created
called a function set. A Function Set is similar to a function block
in the sense that it consists of input, output and parameter
resources and contains internal logic. A Function Set can have a
subset of mandatory inputs, outputs and parameters to provide minimum
interoperability. It can also be extended with manufacturer/user-
specific resources. A device is composed of one or more Function Set
instances.

An example of function sets can be found from the CoRE Resource
Directory specification that defines REST interfaces for
registration, group and lookup [I-D.ietf-core-resource-directory].

The OMA Lightweight M2M standard [OMA-TS-LWM2M] also defines a
function set structure called an Objects that use integer path,
instance and resource URI segments. OMA Objects can be defined and
then registered with an OMA maintained registry [OMA-TS-LWM2M]. This
section is simply meant as a guideline for the definition of other
such REST interfaces, either custom or part of other specifications.

5.1. Defining a Function Set

   In a Function Set, types of resources are defined. Each type
   includes a human readable name, a path template, a Resource Type for
   discovery, the Interface Definition and the data type and allowed
   values. A Function Set definition may also include a field
   indicating if a sub-resource is mandatory or optional.

5.1.1. Path template

   A Function Set is a container resource under which its sub-resources
   are organized. The profile defines the path to each resource of a
   Function Set in a path template. The template can contain either
   relative paths or absolute paths depending on the profile needs. An
   absolute Function Set should be located at its recommended root path
   on a web server, however it can be located under an alternative path
   if necessary (for example multi-purpose devices, gateways etc.). A
   relative Function Set can be instantiated as many times as needed on
   a web server with an arbitrary root path. However some Function Sets
   (e.g. device description) only make sense as singletons.

   The path template includes a possible index {#} parameter, and
   possible fixed path segments. The index {#} allows for multiple
   instances of this type of resource, and can be any string. The root
   path and the indexes are the only variable elements in a path
   template. All other path segments should be fixed.

5.1.2. Resource Type

   Each root resource of a Function Set is assigned a Resource Type
   parameter, therefore making it possible to discover it. Each sub-
   resource of a Function Set is also assigned a Resource Type
   parameter. This Resource Type is used for resource discovery and is
   usually necessary to discover optional resources supported on a
   specific device. The Resource Type of a Function Set may also be
   used for service discovery and can be exported to DNS-SD [RFC6763]
   for example.

   The Resource Type parameter defines the value that should be included
   in the rt= field of the CoRE Link Format when describing a link to
   this resource. The value SHOULD be in the form "namespace.type" for
   root resources and "namespace.type.subtype" for sub-resources. This
   naming convention facilitates resource type filtering with the
   /.well-known/core resource. However a profile could allow mixing in
   foreign namespace references within a Function Set to import external
   references from other object models (e.g. SenML and UCUM).

5.1.3. Interface Description

The Interface Description parameter defines the REST interface for
that type of resource. Several base interfaces are defined in
Section 4 of this document. For a given profile, the Interface
Description may be inferred from the Resource Type. In that case the
Interface Description MAY be elided from link descriptions of
resource types defined in the profile, but should be included for
custom extensions to the profile.

The root resource of a Function Set should provide a list of links to
its sub-resources in order to offer gradual reveal of resources. The
CoRE Link List interface defined in Section 7.1 offers this
functionality so a root resource should support this interface or a
derived interface like CoRE Batch (See Section 7.2).

5.1.4. Data type

The Data Type field defines the type of value (and possible range)
that is returned in response to a GET for that resource or accepted
with a PUT. The interfaces defined in Section 4 make use of plain
text and SenML Media types for the actual format of this data. A
profile may restrict the list of supported content formats for the
CoRE interfaces or define new interfaces with new content types.

5.2. Discovery

A device conforming to a profile SHOULD make its resources
discoverable by providing links to the resources on the path /.well-
known/core as defined in [RFC6690]. All resources hosted on a device
SHOULD be discoverable either with a direct link in /.well-known/core
or by following successive links starting from /.well-known/core.

The root path of a Function Set instance SHOULD be directly
referenced in /.well-known/core in order to offer discovery at the
first discovery stage. A device with more than 10 individual
resources SHOULD only expose Function Set instances in /.well-known/
core to limit the size of this resource.

In addition, a device MAY register its resources to a Resource
Directory using the registration interface defined in
[I-D.ietf-core-resource-directory] if such a directory is available.

5.3. Versioning

A profile should track Function Set changes to avoid incompatibility
issues. Evolutions in a Function Set SHOULD be backward compatible.

6. Security Considerations

   An implementation of a client needs to be prepared to deal with
   responses to a request that differ from what is specified in this
   document. A server implementing what the client thinks is a resource
   with one of these interface descriptions could return malformed
   representations and response codes either by accident or maliciously.
   A server sending maliciously malformed responses could attempt to
   take advantage of a poorly implemented client for example to crash
   the node or perform denial of service.

7. IANA Considerations

   This document registers the following CoRE Interface Description
   (if=) Link Target Attribute Values.

7.1. Link List

   Attribute Value: core.ll

   Description: The Link List interface is used to retrieve a list of
      resources on a web server.

   Reference: This document. Note to RFC Editor - please insert the
      appropriate RFC reference.

   Notes: None

7.2. Batch

   Attribute Value: core.b

   Description: The Batch interface is used to manipulate a collection
   of sub-resources at the same time.

   Reference: This document. Note to RFC Editor - please insert the
      appropriate RFC reference.

   Notes: None

7.3. Linked Batch

   Attribute Value: core.lb

   Description: The Linked Batch interface is an extension of the Batch
      interface. Contrary to the basic Batch which is a collection
      statically defined by the web server, a Linked Batch is
      dynamically controlled by a web client.

Reference: This document. Note to RFC Editor - please insert the
   appropriate RFC reference.

Notes: None

7.4. Sensor

Attribute Value: core.s

Description: The Sensor interface allows the value of a sensor
   resource to be read.

Reference: This document. Note to RFC Editor - please insert the
   appropriate RFC reference.

Notes: None

7.5. Parameter

Attribute Value: core.p

Description: The Parameter interface allows configurable parameters
   and other information to be modeled as a resource. The value of
   the parameter can be read or update.

Reference: This document. Note to RFC Editor - please insert the
   appropriate RFC reference.

Notes: None

7.6. Read-only parameter

Attribute Value: core.rp

Description: The Read-only Parameter interface allows configuration
   parameters to be read but not updated.

Reference: This document. Note to RFC Editor - please insert the
   appropriate RFC reference.

Notes: None

7.7. Actuator

Attribute Value: core.a

Description: The Actuator interface is used by resources that model
   different kinds of actuators (changing its value has an effect on

its environment). Examples of actuators include for example LEDs, relays, motor controllers and light dimmers. The current value of the actuator can be read or the actuator value can be updated. In addition, this interface allows the use of POST to change the state of an actuator, for example to toggle between its possible values.

Reference: This document. Note to RFC Editor - please insert the appropriate RFC reference.

Notes: None

8. Acknowledgements

Acknowledgement is given to colleagues from the SENSEI project who were critical in the initial development of the well-known REST interface concept, to members of the IPSO Alliance where further requirements for interface descriptions have been discussed, and to Szymon Sasin, Cedric Chauvenet, Daniel Gavelle and Carsten Bormann who have provided useful discussion and input to the concepts in this document.

9. Changelog

Changes from -05 to -06:

o Updated the abstract.

o Section 1: Updated introduction.

o Section 2: Alphabetised the order

o Section 2: Removed the collections definition in favour of the complete definition in the collections section.

o Removed section 3 on interfaces in favour of an updated definition in section 1.3.

o General: Changed interface type to interface description as that is the term defined in RFC6690.

o Removed section on future interfaces.

o Section 8: Updated IANA considerations.

o Added Appendix A to discuss current state of the art wrt to collections, function sets etc.

Changes from -04 to -05:

o Removed Link Bindings and Observe attributes. This functionality
   is now contained in I-D.ietf-core-dynlink.

o Hypermedia collections have been removed. This is covered in a
   new T2TRG draft.

o The WADL description has been removed.

o Fixed minor typos.

o Updated references.

Changes from -03 to -04:

o Fixed tickets #385 and #386.

o Changed abstract and into to better describe content.

o Focus on Interface and not function set/profiles in intro.

o Changed references from draft-core-observe to RFC7641.

o Moved Function sets and Profiles to section after Interfaces.

o Moved Observe Attributes to the Link Binding section.

o Add a Collection section to describe the collection types.

o Add the Hypermedia Collection Interface Description.

Changes from -02 to -03:

o Added lt and gt to binding format section.

o Added pmin and pmax observe parameters to Observation Attributes.

o Changed the definition of lt and gt to limit crossing.

o Added definitions for getattr and setattr to WADL.

o Added getattr and setattr to observable interfaces.

o Removed query parameters from Observe definition.

o Added observe-cancel definition to WADL and to observable
   interfaces.

Changes from -01 to -02:

o Updated the date and version, fixed references.

o "Removed pmin and pmax observe parameters "[Ticket #336]"."

Changes from -00 to WG Document -01

o Improvements to the Function Set section.

Changes from -05 to WG Document -00

o Updated the date and version.

Changes from -04 to -05

o Made the Observation control parameters to be treated as resources
   rather than Observe query parameters. Added Less Than and Greater
   Than parameters.

Changes from -03 to -04

o Draft refresh

Changes from -02 to -03

o Added Bindings

o Updated all rt= and if= for the new Link Format IANA rules

Changes from -01 to -02

o Defined a Function Set and its guidelines.

o Added the Link List interface.

o Added the Linked Batch interface.

o Improved the WADL interface definition.

o Added a simple profile example.

10. References

10.1. Normative References

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119,
              DOI 10.17487/RFC2119, March 1997,
              <http://www.rfc-editor.org/info/rfc2119>.

   [RFC5988] Nottingham, M., "Web Linking", RFC 5988,
              DOI 10.17487/RFC5988, October 2010,
              <http://www.rfc-editor.org/info/rfc5988>.

   [RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link
              Format", RFC 6690, DOI 10.17487/RFC6690, August 2012,
              <http://www.rfc-editor.org/info/rfc6690>.

10.2. Informative References

   [I-D.ietf-core-dynlink]
              Shelby, Z., Vial, M., Koster, M., and C. Groves, "Dynamic
              Resource Linking for Constrained RESTful Environments",
              draft-ietf-core-dynlink-00 (work in progress), October
              2016.

   [I-D.ietf-core-resource-directory]
              Shelby, Z., Koster, M., Bormann, C., and P. Stok, "CoRE
              Resource Directory", draft-ietf-core-resource-directory-08
              (work in progress), July 2016.

   [I-D.ietf-core-senml]
              Jennings, C., Shelby, Z., Arkko, J., Keranen, A., and C.
              Bormann, "Media Types for Sensor Markup Language (SenML)",
              draft-ietf-core-senml-03 (work in progress), October 2016.

   [OIC-Core]
              "OIC Resource Type Specification v1.1.0", 2016,
              <https://openconnectivity.org/resources/specifications>.

   [OIC-SmartHome]
              "OIC Smart Home Device Specification v1.1.0", 2016,
              <https://openconnectivity.org/resources/specifications>.

   [OMA-TS-LWM2M]
              "Lightweight Machine to Machine Technical Specification",
              2016, <http://technical.openmobilealliance.org/Technical/
              technical-information/release-program/current-releases/
              oma-lightweightm2m-v1-0>.

[oneM2MTS0008]
            "TS 0008 v1.3.2 CoAP Protocol Binding", 2016,
            <http://www.onem2m.org/technical/published-documents>.

[oneM2MTS0023]
            "TS 0023 v2.0.0 Home Appliances Information Model and
            Mapping", 2016,
            <http://www.onem2m.org/technical/published-documents>.

[RFC3986]   Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
            Resource Identifier (URI): Generic Syntax", STD 66,
            RFC 3986, DOI 10.17487/RFC3986, January 2005,
            <http://www.rfc-editor.org/info/rfc3986>.

[RFC6573]   Amundsen, M., "The Item and Collection Link Relations",
            RFC 6573, DOI 10.17487/RFC6573, April 2012,
            <http://www.rfc-editor.org/info/rfc6573>.

[RFC6763]   Cheshire, S. and M. Krochmal, "DNS-Based Service
            Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013,
            <http://www.rfc-editor.org/info/rfc6763>.

[RFC7230]   Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
            Protocol (HTTP/1.1): Message Syntax and Routing",
            RFC 7230, DOI 10.17487/RFC7230, June 2014,
            <http://www.rfc-editor.org/info/rfc7230>.

[RFC7252]   Shelby, Z., Hartke, K., and C. Bormann, "The Constrained
            Application Protocol (CoAP)", RFC 7252,
            DOI 10.17487/RFC7252, June 2014,
            <http://www.rfc-editor.org/info/rfc7252>.

[RFC7396]   Hoffman, P. and J. Snell, "JSON Merge Patch", RFC 7396,
            DOI 10.17487/RFC7396, October 2014,
            <http://www.rfc-editor.org/info/rfc7396>.

Appendix A. Current Usage of Interfaces and Function Sets

This appendix analyses the current landscape with regards the
definition and use of collections, interfaces and function sets/
profiles. This should be considered when considering the scope of
this document.

In summary it can be seen that there is a lack of consistancy of the
definition and usage of interface description and function sets.

A.1. Constrained RESTful Environments (CoRE) Link Format (IETF)

   [RFC6690] assumes that different deployments or application domains
   will define the appropriate REST Interface Descriptions along with
   Resource Types to make discovery meaningful. It highlights that
   collections are often used for these interfaces.

   Whilst 3.2/[RFC6690] defines a new Interface Description 'if'
   attribute the procedures around it are about the naming of the
   interface not what information should be included in the
   documentation about the interface.

   Function sets are not discussed.

A.2. CoRE Resource Directory (IETF)

   [I-D.ietf-core-resource-directory] uses the concepts of collections,
   interfaces and function sets.

   If defines a number of interfaces: discovery, registration,
   registration update, registration removal, read endpoint links,
   update endpoint links, registration request interface, removal
   request interface and lookup interface. However it does not assign
   an inteface description identifier (if=) to these interfaces.

   It does define a resource directory function set which specifies
   relevant content formats and interfaces to be used between a resource
   directory and endpoints. However it does not follow the format
   proposed by this document.

A.3. Open Connectivity Foundation (OCF)

   The OIC Core Specification [OIC-Core] most closely aligns with the
   work in this specification. It makes use of interface descriptions
   as per [RFC6690] and has registered several interface identifiers
   (https://www.iana.org/assignments/core-parameters/core-
   parameters.xhtml#if-link-target-att-value). These interface
   descriptors are similar to those defined in this specification. From
   a high level perspective:

   links list: OCF (oic.if.ll) -> IETF (core.ll)
                 Note: it's called "link list" in the IETF.
   linked batch: OCF (oic.if.b) -> IETF (core.lb)
   read-only: OCF (oic.if.r) -> IETF (core.rp)
   read-write: OCF (oic.if.rw) -> IETF (core.p)
   actuator: OCF (oic.if.a) -> IETF (core.a)
   sensor: OCF (oic.if.s) -> IETF (core.s)
   batch: No OCF equivalent -> IETF (core.b)

Some of the OCF interfaces make use of collections.

The OIC Core specification does not use the concept of function sets.
It does however discuss the concept of profiles. The OCF defines two
sets of documents. The core specification documents such as
[OIC-Core] and vertical profile specification documents which provide
specific information for specific applications. The OIC Smart Home
Device Specification [OIC-SmartHome] is one such specification. It
provides information on the resource model, discovery and data types.

A.4. oneM2M

OneM2M describes a technology independent functional architecture
[oneM2MTS0023]. In this archictecture the reference points between
functional entities are called "interfaces". This usage does not
match the [RFC6690] concept of interfaces. A more direct comparison
is that of 10.2/[oneM2MTS0023] that defines basic procedures and
resource type-specific procedures utilising REST type create,
retrieve, update, delete, notify actions.

[oneM2MTS0023] does not refer to resource collections however does
define "Group Management Procedures" in 10.2.7/[oneM2MTS0023]. It
does allow bulk management of member resources.

[oneM2MTS0023] does not use the term "function set". [oneM2MTS0008]
describes the binding with the CoAP protocol. In some respects this
document provides a profile of the CoAP protocol in terms of the
protocol elements that need to be supported. However it does not
define any interface descriptions nor collections.

A.5. OMA LWM2M

[OMA-TS-LWM2M] utilises the concept of interfaces. It defines the
following interfaces: Bootstrap, Client Registration, Device
Management and Service Enablement and Information Reporting. It
defines that these have a particular direction (Uplink/Downlink) and
indicates the operations that may be applied to the interface (i.e.
Request Bootstrap, Write, Delete, Register, Update, De-Register,
Create, Read, Write, Delete, Execute, Write Attributes, Discover,
Observe, Cancel Observation, Notify). It then further defines which
objects may occur over the interface. In 6/[OMA-TS-LWM2M] resource
model, identifier and data formats are described.

Whilst it does not formally describe the use of "collections" the use
of a multiple resource TLV allows a hierarchy of resource/sub-
resource.

It does not identify the interfaces through an Interface Description (if=) attribute.

It does not use the term function set. Informally the specification could be considered as a function set.

Note: It refers to draft-ietf-core-interfaces-00. It also makes use of the binding/observation attributes from draft-ietf-dynlink-00 but does not refer to that document.

Appendix B. Profile example

The following is a short definition of simple profile. This simplistic profile is for use in the examples of this document.

Function Set	Root Path	RT	IF
Device Description	/d	simple.dev	core.ll
Sensors	/s	simple.sen	core.b
Actuators	/a	simple.act	core.b

Table 3: List of Function Sets

Type	Path	RT	IF	Data Type
Name	/d/name	simple.dev.n	core.p	xsd:string
Model	/d/model	simple.dev.mdl	core.rp	xsd:string

Table 4: Device Description Function Set

```
+-------------+------------+---------------+--------+------------+
| Type        | Path       | RT            | IF     | Data Type  |
+-------------+------------+---------------+--------+------------+
| Light       | /s/light   | simple.sen.lt | core.s | xsd:decimal |
|             |            |               |        |            |
|             |            |               |        | (lux)      |
|             |            |               |        |            |
| Humidity    | /s/humidity| simple.sen.hum| core.s | xsd:decimal |
|             |            |               |        |            |
|             |            |               |        | (%RH)      |
|             |            |               |        |            |
| Temperature | /s/temp    | simple.sen.tmp| core.s | xsd:decimal |
|             |            |               |        |            |
|             |            |               |        | (degC)     |
+-------------+------------+---------------+--------+------------+
```

Table 5: Sensors Function Set

```
+------+-----------+---------------+--------+-------------+
| Type | Path      | RT            | IF     | Data Type   |
+------+-----------+---------------+--------+-------------+
| LED  | /a/{#}/led| simple.act.led| core.a | xsd:boolean |
+------+-----------+---------------+--------+-------------+
```

Table 6: Actuators Function Set

Authors' Addresses

   Zach Shelby
   ARM
   150 Rose Orchard
   San Jose 95134
   FINLAND

   Phone: +1-408-203-9434
   Email: zach.shelby@arm.com

   Matthieu Vial
   Schneider-Electric
   Grenoble
   FRANCE

   Phone: +33 (0)47657 6522
   Email: matthieu.vial@schneider-electric.com

Michael Koster
SmartThings
665 Clyde Avenue
Mountain View 94043
USA

Email: michael.koster@smartthings.com

Christian Groves
Huawei
Australia

Email: Christian.Groves@nteczone.com