Abstract

The scale and scope of the worldwide web has been in part driven by the availability of HTML as a common serialization, data model, and interaction model for structured resources on the web. By contrast, the general use of appropriate hypermedia techniques for machine interfaces has been limited by the lack of a common format for serialization and exchange of structured machine resources and sensor/actuator data which includes or embeds standardized hypermedia controls. The IRTF Thing to Thing Research Group [T2TRG] has a charter to investigate the use of REST design style [REST] for machine interactions. The W3C Web of Things Interest Group [W3C-WoT] are investigating abstract hypermedia controls and interaction models for machines. Machine optimized content formats exist for web links [RFC5988] [RFC6690] and for data items [I-D.ietf-core-senml].

Structured data which contains both links and items is known as the collection pattern. This draft describes media types for representation of machine resources structured as collections. A simple, reusable data model is described with a representation format, using a well known set of keywords to expose hypermedia controls, which inform clients how to perform state transfer operations on resources. The underlying assumptions regarding transfer layer processing are specified in this document. The HSML media type described in this document is compatible with SenML and CoRE Link-format by reusing the keyword identifiers and element structures from these content formats. Representations of HSML document content may be obtained in CoRE Link-Format and SenML content formats.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.
Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 14, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Scope of this document ........................................  4
2. Overview and Use Case Requirements ...........................  4
3. Data Model and Interaction Model .............................  4
   3.1. Informative Representation Examples ........................  5
   3.2. Links ...................................................  5
   3.3. Collections ............................................  5
   3.4. Link Embedding .........................................  5
       3.4.1. Batch operations on multiple items in a collection  6
       3.4.2. Collective operation on groups of linked resources  6
4. Abstract Transfer Model .......................................  6
5. Collections ..................................................  7
   5.1. Base element ............................................  7
   5.2. Link element ............................................  7
   5.3. Item element ............................................  8
       5.3.1. Items embedded in the collection ....................  8
       5.3.2. Items stored as collections .......................  9
6. Representation Formats ........................................ 11
   6.1. Example Serialization Formats ............................ 11
       6.1.1. Collection Formats ................................. 12
       6.1.2. Link Formats ....................................... 13
       6.1.3. Item Formats ....................................... 13
7. URI and Parameter Processing ................................. 15
   7.1. URI Path Processing .................................... 15
   7.2. URI Parameter processing ............................... 15
7.2.1. Resource selection ........................................ 16
8. Transfer Model Mapping to Collections .......................... 16
  8.1. Target Resource is Collection, Format is Collection ...... 16
    8.1.1. RETRIEVE ............................................. 16
    8.1.2. UPDATE ............................................... 17
    8.1.3. CREATE ............................................... 18
    8.1.4. DELETE ............................................... 20
  8.2. Target Resource is Collection, Format is Link ............ 21
    8.2.1. RETRIEVE ............................................. 21
    8.2.2. UPDATE ............................................... 21
    8.2.3. CREATE ............................................... 22
    8.2.4. DELETE ............................................... 23
  8.3. Target Resource is Collection, Format is Item ............ 24
    8.3.1. Link Embedding Items ................................ 24
    8.3.2. RETRIEVE ............................................. 24
    8.3.3. UPDATE ............................................... 25
    8.3.4. CREATE ............................................... 26
    8.3.5. DELETE ............................................... 27
  8.4. Target Resource is Item .................................... 28
    8.4.1. RETRIEVE ............................................. 28
    8.4.2. UPDATE ............................................... 28
    8.4.3. CREATE ............................................... 29
    8.4.4. DELETE ............................................... 29
  8.5. Groups .................................................... 29
    8.5.1. RETRIEVE ............................................. 30
    8.5.2. UPDATE ............................................... 31
    8.5.3. CREATE ............................................... 32
    8.5.4. DELETE ............................................... 33
9. Link extensions ................................................ 34
  9.1. Actions ................................................... 34
  9.2. Link Bindings and Monitors ................................ 35
10. Reserved Identifiers ........................................... 36
  10.1. Default namespace ........................................ 37
  10.2. URI Processing Parameters ................................ 37
  10.3. Link Keywords ........................................... 37
    10.3.1. Link Relation Types ................................ 38
    10.3.2. Link Attribute Types ................................ 38
  10.4. Item Keywords ............................................ 38
  10.5. Link Parameters used in Actions .......................... 39
  10.6. Link Parameters used in Link Bindings and Monitors ... 40
  10.7. Conditional Observe Parameters used in Monitors ........ 41
  10.8. Link Attribute Values ................................... 41
11. IANA Considerations ........................................... 42
  11.1. Media Types ............................................. 42
  11.2. CoRE Parameters Content Formats .......................... 43
  11.3. Link Parameters ......................................... 43
  11.4. Link Relation Types ..................................... 44
  11.5. New CoRE Interface Types ................................ 44
1. Scope of this document

This is a broadly scoped document which specifies representation formats, data models, interaction models, transfer mapping, URI processing, and design pattern extensions including actions and monitors.

The features listed above and new features may be specified and extended as needed in other documents which refer to this document.

2. Overview and Use Case Requirements

Use case requirements include the following.

A standardized way to expose self-describing resource representations using embedded hyperlinks and link annotations.

A standardized way of organizing and interacting with resource instances using hypermedia controls such as links and forms.

A standardized encapsulation of resources for modeling things, capabilities, groups, indices, and other common structures.

3. Data Model and Interaction Model

The HSML data model consists of collections containing links which point to items. An instance of a collection is a resource and is identified by a URI.

Links are standard web links as in [RFC5988] or [RFC6690]. Items are identified by links in collections.

Links in a collection may point to items within the context of the collection or they may point to items external to the collection, on the same server or on other servers.
Items are data elements, either within the context the collection, or outside the context of the collection. An instance of an item is a resource and is identified by a URI.

An Item may only be in the context of one collection, but may be identified by any number of links in any number of collections.

Items in the collection that use an HSML compatible data model, for example SenML, see [I-D.ietf-core-senml], may be embedded in the collection and transferred either along with the links or separately from links.

3.1. Informative Representation Examples

JSON formatted examples are used in this document to illustrate normative and informative concepts. Representations in other formats may be derived from the JSON representations. For example, compact binary mappings may be defined using available models.

3.2. Links

Links follow the specifications in [RFC5988] and [RFC6690] with extensions to implement actions and monitors as described in this document and any referencing extension documents.

HSML Links may be stored in Resource Directories for discovery using CoRE Resource Directory [I-D.ietf-core-resource-directory].

3.3. Collections

Collections contain one or more links and extended links, and may contain data items referred to by the links. A representation of a collection may contain both links and data items, plus any extended links such as action forms.

3.4. Link Embedding

Link embedding enables the transfer of one or more items in a collection using a single transfer operation. This document describes two types of link embedding for items in the collection. Batch link embedding allows one or more resource instances (item) to each contribute part of an aggregate (collection) representation. Group link embedding allows a particular operation to be repeated for each member (item) of a group (collection).
3.4.1. Batch operations on multiple items in a collection

A collection of items enables operations on more than one item at a time by exposing a structured representation of multiple resources in the collection.

Applications may select resources by using URI parameters, and transfer representations of multiple named resources using the HSML or SenML multi-item formats.

3.4.2. Collective operation on groups of linked resources

Resource links in the collection may specify group transfer semantics, where transfer operations are routed to each resource in the collection specified by a group link. Group responses are aggregated using a multi-item format which identifies each item by URI.

4. Abstract Transfer Model

The HSML media type assumes a transfer model capable of interacting with representations using a simple CRUD model, allowing for basic life cycle operations on resources and collections.

CREATE
Create an instance of a resource as specified using the payload as a constructor. Optionally return a reference to the created resource. Typically uses POST in CoAP [RFC7252] or HTTP, may use PUBLISH in pubsub protocols.

RETRIEVE
Obtain a representation of the selected resource. Typically uses GET in CoAP or HTTP, could use SUBSCRIBE with message retention in pubsub.

UPDATE
Replace or partially replace the representation of the selected resource. Typically uses PUT or PATCH in CoAP and HTTP, could use PUBLISH in pubsub in the frequent case that CREATE and UPDATE are not needed on the same resource.

DELETE
Remove the representation of the selected resource. Typically uses DELETE in CoAP or HTTP. There is no natural mapping to pubsub if a remove operation is not provided.

OBSERVE
Obtain a sequence of representations of the selected resource, indicating state updates which occur on the resource. Typically uses CoAP Observe, HTTP EventSource, MQTT SUBSCRIBE. OBSERVE is the transfer equivalent of performing a RETRIEVE on the resource immediately following each state change of the resource.

5. Collections

Collection representations in HSML include Base Elements, Link Elements, and Item Elements.

5.1. Base element

A base element describes the context under which to interpret values embedded in subsequent items within the representation of a collection.

The base identifier element (bi) may contain an absolute URI or an absolute path reference from which to base relative references found in the links. It functions as a base URI embedded in content as per [RFC3986] Section 5.1.1

URI reference follows the definition in [RFC3986] Section 5.

The format of base elements are specified in [I-D.ietf-core-senml]. Figure 1 is an example of a base element.

```json
{
  "bi": "/sensors/"
}
```

Figure 1: Example Base URI

Other base items from SenML are permissible, including base time (bt) and base value (bv). If additional senml base values are present, the client MUST interpret the items in the collection in the context of the applicable base elements. For example, if there is a "bv" or base value element, all of the returned values from items in the collection MUST be added to the base value as per [I-D.ietf-core-senml].

5.2. Link element

A link element is a hyperlink based on the structure and syntax of [RFC6690] and [I-D.ietf-core-links-json]. An example link element is shown in Figure 2.
5.3. Item element

An item element in a collection is a data element that is referenced by a link in the collection.

Items in the collection are indicated by hyperlink references ("href") that serve as selection variables for matching to URI parameters and resource names ("n") in multi-resource representations. Reference resolution should use the rules defined in [RFC3986].

Items may be embedded in the collection, they may be subresources of the collection, or they may be items in other collections referenced by links in the collection. An example item element is shown in Figure 3

```
{
   "n": "temp",
   "v": 27
}
```

5.3.1. Items embedded in the collection

Items may be stored as simple sets of key-value pairs in the context of the collection. Links about these items may be obtained from the collection that contains them.
Alternatively, items themselves may be stored as single-item collections, pointed to by links in another collection. Items stored as collections may contain an item with a zero length href and name, and a self link for the item as shown in the collection representation of the item in Figure 5. Items stored in this way may be augmented by adding additional resources and link content to the collection. Items stored as collections may offer link format and collection format representations.
base collection:
[
    { "bi": "/sensors/" },
    { "href": "temp/" },
    { "href": "humid/" }
]

"temp" item:
[
    { "bi": "/sensors/temp/" },
    { "href": "", //may be elided
      "rel": ["self","item"] },
    { "n": "", //may be elided
      "v": 27 }
]

"humid" item:
[
    { "bi": "/sensors/humid/" },
    { "rel": ["self","item"] },
    { "v": 50 }
]

Figure 5: Items as Separate Collections

Items embedded in collections, and items linked and stored as separate collections, will all be returned using the item representation format as shown in Figure 6. A client interacting with the items representation of the example collection at /sensors/
would not need to understand the difference between embedded items and linked items that exposed similar content.

```
[
  {
    "bi": "/sensors/"
  },
  {
    "n": "temp",
    "v": 27
  },
  {
    "n": "humid",
    "v": 50
  }
]
```

Figure 6: Example Items Representation

6. Representation Formats

The HSML media type includes multiple content types and interface types [I-D.ietf-core-interfaces] to enable the client to select representations that optimize communication for the workflow. Representation formats include links and items together (collection formats), links alone (link formats), or items alone (item formats).

Link formats are useful for discovery workflow, item formats are useful for interaction with resource state machines, and link+item formats are useful for constructing and modifying resource instances.

In addition to HSML native formats, standard CoRE Link-Format [RFC6690] and SenML formats [I-D.ietf-core-senml] may be exposed.

6.1. Example Serialization Formats

Figure 7 shows an example document in hsml+json format. This example contains a base element, three link elements, and two item elements.
The HSML media type defines content formats and corresponding CoRE Interface Types that may select partial representations of the resource for interaction.

6.1.1. Collection Formats

Collection formats as shown in Figure 7 expose all of the elements of a resource, including items, links, and link extensions.
6.1.2. Link Formats

Link content formats, when used in an "accept" option or "content-type" option in a transfer header, or when selected by the "if=hsml.link" URI parameter, will select the link elements in the collection for interaction as in Figure 8.

```
RETRIEVE /sensors/ accept=application/hsml.link+json
or
RETRIEVE /sensors/ accept=application/hsml+json?if=hsml.link
```

Response Payload:

```
[
    {
        "anchor": "/sensors/",
        "rel": ["self", "index"]
    },
    {
        "href": "temp",
        "rt": "some.sensor.temp"
    },
    {
        "href": "humid",
        "rt": "some.sensor.humid"
    }
]
```

Figure 8: Example Lnk Format

CoRE link-format content formats, for example application/link-format+json, select RFC6690 compliant links, and may not include representations of extended links (rel=action, rel=monitor).

6.1.3. Item Formats

Item content formats, when used in an "accept" option or "content-type" option in a transfer header, or when selected by the "if=hsml.item" URI parameter, will select the item elements in the collection for interaction as in Figure 9.
RETRIEVE /sensors/ accept=application/hsml.item+json
or
RETRIEVE /sensors/ accept=application/hsml+json?if=hsml.item
Response Payload:
[
  {
    "bi": "/sensors/"
  },
  {
    "n": "temp",
    "v": 27
  },
  {
    "n": "humid",
    "v": 50
  }
]

Figure 9: Example Item Format

URI Parameters for matching link attributes and relations may be used to select items when item representations are being specified using either content-format (accept) or interface parameters (if=). For example:

RETRIEVE /sensors/?if=hsml.item&rt=some.sensor.temp
Response Payload:
[
  {
    "bi": "/sensors/"
  },
  {
    "n": "temp",
    "v": 27
  }
]

Figure 10: Item Selection Using Link Parameter

SenML content formats select data records and return SenML compliant resource names. "bn" may optionally be returned when compliant resource names "n" may be resolved through simple string concatenation as per [I-D.ietf-core-senml].
7. URI and Parameter Processing

The HSML media type defines URI reference processing and URI Query processing but does not in general define fragment (#) references in URIs.

If fragment references are provided in a particular transfer implementation, they should be used to select single items in collections in accordance with current practice.

7.1. URI Path Processing

The path part of the URI reference used to indicate HSML resources may be used as a reference to a collection or to an item in a collection. Collection references should contain the trailing slash character "/" in accordance with [RFC3986]. Server implementations should return links to collections with the trailing "/", and should attempt to accept references to collections without the trailing "/" if such references can be used to construct unambiguous references.

References to items in a collection should not contain the trailing "/" character. Servers should return items in response to references that do not contain the trailing "/" character, and should attempt to accept references to items in collections with the trailing "/" if such references can be used to construct unambiguous references.

URI references may be routed to collections in the order in which path segments appear in the reference, from left to right reading the path string, separated by "/" characters.

URI references may alternatively be routed as opaque strings to resources. In this case, the resolution of relative references to items in a collection should be possible by concatenating the relative reference to the context URI of the collection. Note that this may enforce certain naming conventions such as the trailing slash in practice.

7.2. URI Parameter processing

URI Parameters, typically mapped as query parameters in HTTP and CoAP, are used for selecting resources, selecting partial representations, and otherwise modifying aspects of the expected or included representation. In this way, they may be considered part of the URI, since they help identify a unique representation to be transferred.
7.2.1. Resource selection

URI Parameters may be used to select resources in a collection for transfer. This is done using the common parameter matching rules specified in [RFC6690].

Resource selection is performed based on matching URI Parameters with Link Parameters of all links in the collection which are exposed by the indicated media type and interface type. URI Parameters listed in Section 10.2 are excluded from the matching process.

The target resource selection depends on the content-format specified in the request or the interface type specified in the URI parameters.

The collection content-formats or interface types select all links and items in the collection, including link extensions. URI parameters included in the request should be matched against link parameters for selecting links and associated items.

The link content formats or interface types select all links in the collection. URI parameters included in the request should be matched against link parameters for link selection.

The item content formats or interface types select all items in the collection. URI parameters included in the request should be matched against link parameters associated with items in the collection for item selection as shown in Figure 10.

8. Transfer Model Mapping to Collections

8.1. Target Resource is Collection, Format is Collection

When the reference of a request targets a collection resource, using a collection format, the representation may contain both links and items. It is implied that operations using this format will interact with both links and items. The collection format is indicated by using a collection content type in the accept or content-type header, or by specifying a collection interface type e.g. if=hsml.collection.

8.1.1. RETRIEVE

Retrieve returns a representation of selected elements, consisting of a list of elements in the collection, including base element, links, and optionally representations of items, as shown in Figure 11. Elements may include link extensions, for example action links and monitor links.
RETRIEVE /sensors/ accept=application/hsm1l.collection+json
Response Payload:

```
[
    {
        "bi": "/sensors/"
    },
    {
        "anchor": "/sensors/",
        "rel": ["self", "index"]
    },
    {
        "href": "temp",
        "rt": "some.sensor.temp"
    },
    {
        "href": "humid",
        "rt": "some.sensor.humid"
    },
    {
        "n": "temp",
        "v": 27
    },
    {
        "n": "humid",
        "v": 50
    }
]
```

Figure 11: Retrieve Collection

8.1.2. UPDATE

Update replaces all selected elements in the collection with elements included in the payload. Update operations may include replace (PUT) and partial update (PATCH) operations where supported in the transfer protocol. The server response should indicate that the resource was Changed.
UPDATE /sensors/?href=temp
   content-type=application/hsml.collection+json
Payload:
[
   {
      "rt": ["some.sensor.temp", "some.other.type"]
   }
]
Response: Changed

RETRIEVE /sensors/ accept=application/hsml.collection+json
Response Payload:
[
   {
      "bi": "/sensors/"
   },
   {
      "anchor": "/sensors/",
      "rel": ["self", "index"]
   },
   {
      "href": "temp",
      "rt": ["some.sensor.temp", "some.other.type"]
   },
   {
      "href": "humid",
      "rt": "some.sensor.humid"
   },
   {
      "n": "temp",
      "v": 27
   },
   {
      "n": "humid",
      "v": 50
   }
]

Figure 12: Update Item in Collection

8.1.3. CREATE

Create adds elements to a collection, including links and items, where the elements are specified by representations included in the payload. Hints and directives about the created resource may be included in the payload as link parameters, for example a value for href, specifying the location of the created resource. The server is expected to return the location of created resource instances to the
client in a header or other response parameter. For example, the
"Location" option in CoAP or "Location" header in HTTP should be used
to identify the created resource. The server response should
indicate that a resource was Created.

CREATE /sensors/ content-type=application/hsml.collection+json
Payload:
[
  {
    "href": "barometer",
    "rt": "some.sensor.mbar"
  },
  {
    "n": "barometer",
    "v": 993
  }
]
Response: Created
Location: "barometer"

RETRIEVE /sensors/ accept=application/hsml.collection+json
Response Payload:
[
  {
    "bi": "/sensors/"
  },
  {
    "anchor": "/sensors/",
    "rel": ["self", "index"]
  },
  {
    "href": "barometer",
    "rt": "some.sensor.mbar"
  },
  {
    "href": "temp",
    "rt": ["some.sensor.temp", "some.other.type"]
  },
  {
    "href": "humid",
    "rt": "some.sensor.humid"
  },
  {
    "n": "barometer",
    "v": 993
  },
  {
    "n": "temp",
    "v": 993
  }
]
"v": 27
},
{
"n": "humid",
"v": 50
}
]

Figure 13: Create Item in Collection

8.1.4. DELETE

Delete removes selected elements from the collection. If no elements are selected, delete removes the entire collection. The server response should indicate that the resource was Deleted.

DELETE /sensors/?href=barometer
Response: Deleted

RETRIEVE /sensors/ accept=application/hsml.collection+json
Response Payload:
[
{
"bi": "/sensors/"
},
{
"anchor": "/sensors/",
"rel": ["self", "index"]
},
{
"href": "temp",
"rt": ["some.sensor.temp", "some.other.type"]
},
{
"href": "humid",
"rt": "some.sensor.humid"
},
{
"n": "temp",
"v": 27
},
{
"n": "humid",
"v": 50
}
]

Figure 14: Delete Item in Collection
8.2. Target Resource is Collection, Format is Link

When a collection is referenced and the link format is indicated, using a link content format in the header or specifying a link interface type, e.g. if=hsml.link, it is expected that the request will interact with the links in the collection.

8.2.1. RETRIEVE

Retrieve returns a list containing selected links, as shown in Figure 15.

```
RETRIEVE /sensors/ accept=application/hsml.link+json
Response Payload:
[
  {
    "anchor": "/sensors/",
    "rel": ["self", "index"]
  },
  {
    "href": "temp",
    "rt": ["some.sensor.temp", "some.other.type"]
  },
  {
    "href": "humid",
    "rt": "some.sensor.humid"
  }
]
```

```
RETRIEVE /sensors/?rt=some.sensor.temp accept=application/hsml.link+json
Response Payload:
[
  {
    "href": "temp",
    "rt": ["some.sensor.temp", "some.other.type"]
  }
]
```

Figure 15: Retrieve Links

8.2.2. UPDATE

Update modifies selected links, replacing link elements with elements included in the payload. Update operations may include replace (PUT) and partial update (PATCH) operations where supported in the transfer
protocol. The server response should indicate that the resource was changed.

```
UPDATE /sensors/?href=temp
content-type=application/hsml.link+json
Payload:
[
  {
    "rt": "some.sensor.temp"
  }
]
```

```
RETRIEVE /sensors/ accept=application/hsml.link+json
Response Payload:
[
  {
    "anchor": "/sensors/",
    "rel": ["self", "index"]
  },
  {
    "href": "temp",
    "rt": "some.sensor.temp",
  },
  {
    "href": "humid",
    "rt": "some.sensor.humid"
  }
]
```

Figure 16: Update Links

### 8.2.3. CREATE

Create adds links to the collection, where the links are included in the payload. The server response should indicate that the resource was changed.
CREATE /sensors/ content-type=application/hsml.link+json
Payload:
[
    {
        "href": "/sensor-group/
    }
]
Response: Changed

RETRIEVE /sensors/ accept=application/hsml.link+json
Response Payload:
[
    {
        "href": "/sensor-group/"
    },
    {
        "anchor": "/sensors/",
        "rel": ["self", "index"]
    },
    {
        "href": "temp",
        "rt": "some.sensor.temp",
    },
    {
        "href": "humid",
        "rt": "some.sensor.humid"
    }
]

Figure 17: Create Links

8.2.4.  DELETE

Delete removes selected links from the collection. The server response should indicate that the resource was Changed. If links point to items in the context of the collection, either remove the items as well as the links, or leave the collection as is and return a method error (Method Not Allowed).
DELETE /sensors/?href=sensor-group
Response: Changed

RETRIEVE /sensors/ accept=application/hsml.link+json
Response Payload:
[
  {
    "anchor": "/sensors/",
    "rel": ["self", "index"]
  },
  {
    "href": "temp",
    "rt": "some.sensor.temp"
  },
  {
    "href": "humid",
    "rt": "some.sensor.humid"
  }
]

Figure 18: Delete Links

8.3. Target Resource is Collection, Format is Item

When a collection is referenced and the item format is indicated, either by including an item content type in the request header or using an item interface type, e.g. if=hsml.item, it is expected that the request will interact with the items in a collection.

Specifying item interaction with a collection invokes the link embedding operations.

8.3.1. Link Embedding Items

Collective operations on items in collections are invoked by using the URI of the collections, along with URI parameters, to select one or more items in the collection.

Items which are compatible with the HSML item format may be returned with multiple items embedded in a single representation.

8.3.2. RETRIEVE

Retrieve returns a list containing a base element and a composite representation of the selected items as shown in Figure 19.
RETRIEVE /sensors/ accept=application/hsml.item+json
Response Payload:
[
  {
    "bi": "/sensors/"
  },
  {
    "n": "temp",
    "v": 27
  },
  {
    "n": "humid",
    "v": 50
  }
]

RETRIEVE /sensors/?href=temp
accept=application/hsml.item+json
Response Payload:
[
  {
    "bi": "/sensors/"
  },
  {
    "n": "temp",
    "v": 27
  }
]

Figure 19: Retrieve Items

8.3.3. UPDATE

Update modifies selected items, replacing items in the collection
with items included in the payload which match by name "n" value.
Update operations may include replace (PUT) and partial update
(PATCH) operations where supported in the transfer protocol. The
server response should indicate that the resource was Changed.
UPDATE /sensors/ content-type=application/hsml.item+json
Payload:
[
    {
        "n": "temp",
        "v": 30
    }
]
Response: Changed

RETRIEVE /sensors/ accept=application/hsml.item+json
Response Payload:
[
    {
        "bi": "/sensors/"
    },
    {
        "n": "temp",
        "v": 30
    },
    {
        "n": "humid",
        "v": 50
    }
]

Figure 20: Update Items

8.3.4. CREATE

Create adds new items to the collection along with system-constructed links. Link content is determined by the resource type or traits defined by application semantics. Server is expected to return the location of created resource instances to the client in a header or other response parameter. For example, the "Location" option in CoAP or "Location" header in HTTP should be used to identify the created resource. The server response should indicate that a resource was Created.
CREATE /sensors/ content-type=application/hsml.item+json
Payload:
[
  {
    "n": "barometer",
    "v": 1002
  }
]
Response: Created

RETRIEVE /sensors/ accept=application/hsml.item+json
Response Payload:
[
  {
    "bi": "/sensors/"
  },
  {
    "n": "temp",
    "v": 30
  },
  {
    "n": "barometer",
    "v": 1002
  },
  {
    "n": "humid",
    "v": 50
  }
]

Figure 21: Create Items

8.3.5. DELETE

Delete removes selected items and corresponding links from the collection. The server response should indicate that the resource was Deleted. If no items are selected, return a not found error.
DELETE /sensors/?href=barometer
Response: Deleted

RETRIEVE /sensors/ accept=application/hsml.item+json
Response Payload:
[
    {
        "bi": "/sensors/"
    },
    {
        "n": "temp",
        "v": 30
    },
    {
        "n": "humid",
        "v": 50
    }
]

Figure 22: Delete Items

8.4. Target Resource is Item

When the URI of a reference points to an item in a collection, it is expected that the request will interact with a single item.

8.4.1. RETRIEVE

Retrieve returns a representation of the item in the content type according to the accept option of the RETRIEVE request, or using a system defined content-format if there is no accept option provided.

RETRIEVE /sensors/temp accept=text/plain
Response Payload:
30

Figure 23: Retrieve One Item

8.4.2. UPDATE

Update replaces the resource state with the state defined in the supplied representation according to the content-type or ct option. Update operations may include replace (PUT) and partial update (PATCH) operations where supported in the transfer protocol. The server response should indicate that the resource was Changed.
UPDATE /sensors/temp content-type=text/plain
Payload:
33

RETRIEVE /sensors/temp accept=text/plain
Response Payload:
33

Figure 24: Update One Item

8.4.3. CREATE

Not Defined, application dependent.

8.4.4. DELETE

Delete removes any links to the item from the collection, and removes the item. If the item is stored as a collection, delete removes the collection. The server response should indicate that the resource was Deleted.

DELETE /sensors/temp

RETRIEVE /sensors/temp accept=text/plain
Response: Not Found

Figure 25: Delete One Item

8.5. Groups

Group transfer operations are provided by collections that contain links with the "grp" relation value.

Transfer operations which specify the collection URI as target and use the item content format are routed to the resolved URI of each link in the collection that contains the "grp" relation.

URI Parameters used for resource selection and matching are sent to the target URIs of all links that contain the "grp" relation.

Responses from the selected group resources are aggregated and by default returned as a single response. The group response SHOULD be returned as an outer array where such representation is available, for example a JSON array which contains elements consisting of SenML responses.
Optionally, a chunked response may be specified, if provided by the transfer implementation, in which the response from each group member is returned individually within a sequence of responses.

The return code should be based on successful responses from link targets. An implementation of a group collection may choose to allow some rejected responses from link targets, depending on the composition of the link targets. A group may not be required to be composed of link targets that always accept all requests; this is at the discretion of the resource designer.

No mechanism is provided in this document to enable a client to inspect the separate return codes from each group link target resource. Multiple transfer headers may be supplied in some representations, or mapped to metadata in others.

The following examples assume the prior example from Figure 5 indexed by a group collection as in Figure 26.

```plaintext
RETRIEVE /sensor-group/ accept=application/hsm1.collection+json
Response Payload:
[
  {
    "bi": "/sensor-group/
  },
  {
    "anchor": "/sensor-group/",
    "rel": ["self", "index"]
  },
  {
    "href": "/sensors/temp/
  },
  {
    "href": "/sensors/humid/",
    "rel": "grp"
  }
]
```

Figure 26: Example Group Collection

8.5.1. RETRIEVE

Retrieve requests are routed to each link in the collection that contains the "grp" relation. The response from each link target is returned as an element in an array representation.
RETRIEVE /sensor-group/ accept=application/hsml.item+json
Response Payload:
[
  {
    "bi": "/sensors/temp/"
  },
  {
    "v": 33
  }
],
[
  {
    "bi": "/sensors/humid/"
  },
  {
    "v": 41
  }
]

Figure 27: Group Retrieve

8.5.2. UPDATE

Update requests are routed to each link in the collection that contains the "grp" relation. The target resource of each group link processes the request, including URI parameters and content format. The result code returned should indicate that the resource is Changed if any resource state may have been updated.
UPDATE /sensor-group/ content-type=application/hsml.item+json
Payload:
[
   {
      "v": 0
   }
]
Response: Changed

RETRIEVE /sensor-group/ accept=application/hsml.item+json
Response Payload:
[
   [
      {
         "bi": "/sensors/temp/"
      },
      {
         "v": 0
      }
   ],
   [
      {
         "bi": "/sensors/humid/"
      },
      {
         "v": 0
      }
   ]
]

Figure 28: Group Update

8.5.3. CREATE

Create requests are routed to each link in the collection that contains the "grp" relation. In the example shown in Figure 29, an additional named resource is being created within each (collection type) item to hold a location value for that item. The result code should indicate that a resource was Created if any resource was created as a result of the create operation.
CREATE /sensor-group/ content-type=application/hsml.item+json
Payload:
[
  {
    "n": "location",
    "vs": "living room"
  }
]
Response: Created

RETRIEVE /sensor-group/ accept=application/hsml.item+json
Response Payload:
[
  [
    {
      "bi": "/sensors/temp/"
    },
    {
      "v": 0
    },
    {
      "n": "location",
      "vs": "living room"
    }
  ],
  [
    {
      "bi": "/sensors/humid/"
    },
    {
      "v": 0
    },
    {
      "n": "location",
      "vs": "living room"
    }
  ]
]

Figure 29: Group Create

8.5.4. DELETE
Delete requests are routed to each link in the collection that contains the "grp" relation. In the example shown in Figure 30, the URI parameter ?href=location selects the resource at the relative URI reference "location" at each group link target for delete. The
result code should indicate that a resource was Deleted if any resource was deleted as a result of the delete operation.

DELETE /sensor-group/?href=location
Response: Deleted

RETRIEVE /sensor-group/ accept=application/hsml.item+json
Response Payload:
[
    [
        "bi": "/sensors/temp/"
    ],
    [
        "v": 0
    ]
],
[
    [
        "bi": "/sensors/humid/"
    ],
    [
        "v": 0
    ]
]

Figure 30: Group Delete

9. Link extensions

9.1. Actions

Actions are hypermedia controls, indicated by a rel=action value in a link, used to construct transfer operations that change the state of resources. The use roughly follows the use of forms in HTML [RFC1866], with semantics more consistent with links. See Section 10.5 for more information.

An example Action element is shown in Figure 31.
Figure 31: Example Action Element

These Action elements inform the client that to perform a type "st.on" or "st.off" action on the context resource, perform a CREATE method on the "switchcommand" URI relative to the context URI, using the text/plain content type, with a payload as defined by the "schema" parameter. This example uses a free-form fragment of JSON-Schema language to differentiate, by action payloads, the "st.on" and "st.off" actions, which are mapped to the same URI and method.

9.2. Link Bindings and Monitors

Link Bindings and Monitors are hypermedia controls, indicated by a rel=boundto or rel=monitor value in a link, used to construct transfer operations that consume or expose state changes of resources. A monitor invokes a state transfer operation from the link context to a target resource. A Link Binding follows the semantics defined in [I-D.groves-core-dynlink], and invokes a state transfer in the opposite direction, that is from the link target to the link context.

Monitors use the IANA registered link relation "monitor", defined in [RFC5989]. Link Bindings use the link relation type "boundto", defined in [I-D.groves-core-dynlink].

Monitors have a set of accept parameters that indicate how the context resource is being observed, a set of filter parameters that indicate the conditions for generating a state change in the monitor, and a set of target parameters that indicate how state changes are to
be applied to the monitor resource. See Section 10.6 for more information.

An example Monitor element is shown in Figure 32.

```
{
  "rel": "monitor",
  "href": "tank-level-events",
  "content-type": "application/senml+json",
  "transfer-method": "create",
  "pmin": 600,
  "pmax": 3600,
  "nbul": 20,
  "nbll": 80
}
```

Figure 32: Example Monitor Element

This Monitor element defines a monitor resource at the "tank-level-events" URI relative to the context URI, which OBSERVEs the context URI, and updates the "tank-level-events" resource using the CREATE method to add JSON items to the collection, according to the given conditional parameters no more frequently than once every 600 seconds, at least once every 3600 seconds, when the reading is in the notification band, which has a lower limit of 80 and wraps around zero to an upper limit of 20. This has the effect of defining a low level alert notification and high level alert notification.

10. Reserved Identifiers

This section defines the common reserved identifiers that are expected to be processed by implementations of HSML clients and servers. There are many more relation types and link parameters defined and registered with IANA. Implementations should not restrict processing to the keywords identified in this document; they should accept all IANA registered keywords as valid identifiers.

Many of the keywords listed are defined in other RFCs and IETF documents. This document does not redefine any existing keywords. Where a definition exists, the existing definition will be used. Where multiple conflicting definitions exist, this document will indicate the required definition.

New definitions are summarized in Section 11.
10.1. Default namespace

Identifiers in representations using the HSML media types are assumed to use the default namespace defined in Section 10 of this document. An identifier that does not contain an explicit namespace identifier is assumed to be in the default namespace.

For example, if the identifier "method" is encountered and it doesn’t resolve to an IANA registered parameter (reg-parameter in [RFC5988]) resolution should be attempted using the definition of "method" in this document.

10.2. URI Processing Parameters

The following URI Parameters are used to filter representations according to specific processing rules and should not be used to attempt to match link parameters.

"if"  Interface type, used to select a partial representation of a collection

"count"  Indicates the number of items to be returned from the collection

"start"  Indicates the array index of the item in the collection to select as the first item to be returned

"page"  Page number, in units of count

10.3. Link Keywords

The following keywords are reserved for use in an HSML serialization to indicate elements of a web link

"anchor"  Overrides the default resource context of the link

"rel"  Link relation type as defined in [RFC5988] and [RFC6690]

"href"  Target of a link reference. This may be a relative path reference in the collection, e.g. "currentValue" or an absolute path reference on the server, "/sensors/temp/currentValue", or an absolute URI, for example "https://example.com/sensors/temp/currentValue"
10.3.1. Link Relation Types

The following keywords are reserved for use in a HSML serialization to indicate types of link relations, and are used for values of "rel".

"self"
  Refers to the collection that contains the link

"item"
  The link points to an item in the collection, indicating eligibility for collective interaction using link embedding as described in Section 3.4 and Section 8.3.1.

"grp"
  The item the link points to is available for collective interaction through the collection URI according to group semantics described in Section 8.5.

10.3.2. Link Attribute Types

The following keywords are reserved for use in a HSML serialization to indicate types of link attributes

"rt"
  The resource type(s) of the item

"u"
  Units of measure

"ct"
  The CoAP content-format number(s) associated with the item

"content-type"
  The media type string(s) associated with the item

"obs"
  Presence of this attribute indicates that the associated resource is observable

10.4. Item Keywords

The following keywords are reserved for use in a HSML serialization to indicate elements within the serialization. Some of these are defined in [I-D.ietf-core-senml].

"bi"
The base URI of the collection, relative to the service location
e.g. "/sensors/temp/" This is a new definition for HSML

"bt"
The base time that corresponds to the encapsulated state of the
collection

"t"
The time stamp that corresponds to the encapsulated state of the
item in the collection, relative to the base time "bt"

"n"
The name or URI of the resource, relative to the base name or base
URI

"u"
Units of measure

"v"
Number value

"vb"
Boolean value

"vs"
String value

10.5. Link Parameters used in Actions

"anchor"
May override the default context of an action

"rel"
Indicates that this control is an action when rel contains the
value "action"

"href"
URI for mapping or invoking the action specified in the action
control.

"type"
Additional indicator of the action being exposed, can be used with
"rel"

"method"
Transfer method to use on a particular action

"accept"
The Content-Types or CoAP content-formats that are accepted on create and update methods

"content-type"
The media type string(s) that are exposed by retrieve and observe methods

"ct"
The CoAP content-format number(s) exposed

"schema"
Indicates the schema to use for constructing or interpreting transfer payloads, may be a literal value or a URI pointing to an instance of a schema

10.6. Link Parameters used in Link Bindings and Monitors

"anchor"
May override the context URI of a link binding or monitor with any observable resource

"rel"
Indicates that this control is a monitor when rel contains the value "monitor" or a link binding when rel contains the value "boundto"

"href"
The URI of the resource used to monitor context URI, where transfer operations will be sent.

"accept"
The media type string or CoAP content-format to request from the observed resource

"content-type"
The media type string to use in the transfer operation

"ct"
the CoAP content-format number to use in the transfer operation

"accept-method"
(HSML extension) Transfer method to use in request from the observed resource, default is OBSERVE

"transfer-method"
(HSML extension) Transfer method to use for notifications, default is UPDATE
"accept-schema"
   (HSML extension) Schema to use in interpreting the observed
   resource payload, required if transfer-schema is used.

"transfer-schema"
   (HSML extension) Schema to use in constructing the notification
   transfer payload, default is to transfer the accepted payload
   unmodified to the target resource.

10.7. Conditional Observe Parameters used in Monitors

"pmin"
   Minimum time between notifications from a monitor

"pmax"
   Maximum time between notifications from a monitor

"gth"
   Value to match or exceed to determine notification condition

"lth"
   Value to match or be less than to determine notification condition

"st"
   Value change +/- from last report to determine notification
   condition

"eq"
   Value to match, or change from, to determine notification
   condition

"bmn"
   Defines a lower limit, at or above which notification is enabled

"bmx"
   Defines an upper limit, at or below which notification is enabled

"iv"
   Starts the notification state machine with an initial value

10.8. Link Attribute Values

The following keywords are reserved for use in a HSML serialization
   to indicate values of link attributes

"create"
   Transfer layer CREATE operation, value of "method" or "target-
   method"
"retrieve"
Transfer layer RETRIEVE operation, value of "method" or "accept-method"

"update"
Transfer layer UPDATE operation, value of "method" or "target-method"

"delete"
Transfer layer DELETE operation, value of "method" or "target-method"

"observe"
Transfer layer OBSERVE operation, value of "method" or "accept-method"

11. IANA Considerations

11.1. Media Types

Type

  o application

Subtypes

  o hsml
  o hsml.collection
  o hsml.link
  o hsml.item

Media type strings

  o application/hsml
  o application/hsml.collection
  o application/hsml.link
  o application/hsml.item
  o application/hsml+json
  o application/hsml.collection+json
11.2. CoRE Parameters Content Formats

(subject to Structured Syntax encoding rules TBD)

- 22000 - application/hsml+json
- 22001 - application/hsml.link+json
- 22002 - application/hsml.item+json

11.3. Link Parameters

- method
- schema
- content-type
- ct
- accept-method
- transfer-method
- accept-schema
- transfer-schema

The following should be registered in the CoRE dynamic linking draft [I-D.groves-core-dynlink].

- pmin
- pmax
- bmn
- bmx
- iv
- lth
- gth
11.4. Link Relation Types

- st
- eq

11.5. New CoRE Interface Types

- hsml.collection
- hsml.item
- hsml.link

11.6. Transfer Layer Methods

These definitions may use the default namespace and do not need to be registered with IANA

- create
- retrieve
- update
- delete
- observe

12. Security Considerations

12.1. Object Signing

Collection representations are resource state encapsulations and may be transmitted and stored as signed objects in order to protect the integrity of data and metadata, including time and embedded access control information.

12.2. Signed Embedded Time Stamps

The collection may include time stamps (bt and t) that are signed with the object data and metadata.
12.3. Signed Embedded Access Control

The collection representation may include embedded access control information, also signed with the metadata, that can instruct the server to enforce a particular access policy for transfer requests.

12.4. Secure State Updates

Representations submitted to a server to update the state of a resource (UPDATE, CREATE, DELETE) may also contain embedded signed assertions which may be used by the server to decide whether to apply or reject the update.

12.5. Object Signing and Encryption

Object signing and encryption SHOULD use the mechanisms specified in IETF documents for secure JSON Objects [RFC7516] and CBOR Objects [I-D.ietf-cose-msg] [I-D.selander-ace-object-security].

13. Terminology

Client
Having a client role in a REST operation, transmitting a request and receiving one or more responses.

Server
Having a server role in a REST operation, the origin of data items or proxy for the origin. A server is also an authority for a URI namespace [RFC3986].

Resource
Server endpoint for a REST operation, identified by a URI [RFC3986]

Representation
An encoded form of the state of a resource. The encoding rules may be specified in a media type or content type. Clients and servers exchange representations of resources in order to effect application state changes [REST].

URI
Uniform Resource Identifier, used to identify a resource in a link or as a reference [RFC3986]

Reference
An identifier used to select or identify a particular resource. References are constructed by clients to identify resources when
interacting with servers. Servers match references in client requests against URIs of hosted resources.

Media Type, also Content-Format, Content-Type
A set of rules for encoding, transfer, and processing resource representations

Hypermedia
Design style which uses metadata in the form of hyperlinks to structure resources in relation to each other

Collection
A composite resource that contains links and optionally data items

Link, also Hyperlink
A metadata element as described in [RFC5988] and [RFC6690] that contains a pointer to and description of some data element.

Item
A data item pointed to by one or more links in one or more collections.

Context
The context of a link is the subject of the link or the enclosing scope. In this document the collection is the default context for links in the collection.

Target
The target of a link is the resource being pointed to or described. Links in a collection point to and describe items as link targets.

Transfer Layer
A set of predefined message types used to implement state transfer semantics, for example REST.

Request
A message sent from a client to a server identifying the resource, representation, and method to use for the interaction with the server.

Response
A message sent from a server to a client in response to a request, which communicates the state of the identified resource.

Operation or Method
The state transition type requested by the client for the server to perform on the identified resource. Indicated by the transfer layer method, for example, RETRIEVE, UPDATE, CREATE, DELETE.

Pubsub
A transfer layer semantic interface based on the publish-subscribe paradigm, allowing for asynchronous messages to be routed on demand.

14. Informative References

[I-D.groves-core-dynlink]

[I-D.ietf-core-interfaces]

[I-D.ietf-core-links-json]

[I-D.ietf-core-resource-directory]

[I-D.ietf-core-senml]

[I-D.ietf-cose-msg]

[I-D.selander-ace-object-security]


Author’s Address
Michael Koster
SmartThings
1281 Lawrence Station Road
Sunnyvale 94089
USA

Phone: +1-707-502-5136
Email: michael.koster@smartthings.com