|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| **ISSUE REPORT** | | | | |
| (All fields are required to be filled, except the Owner assigned by TG and Filename which is automatically updated) | | | | |
| Spec Name | **Core Framework** | | |  | |  |
| *Use the following names:*  *Core Framework*  *Security*  *Smart Home Device*  *Smart Home Resource* | | | *If the CR applies to multiple specifications, list all.* |
| **Filename*:*** ⌘ ***(incl. Automatically assigned CR Number and Version)*** | | *CR ATG xxxx - Dynlink.docx* | | |

|  |  |
| --- | --- |
| ***Title:*** ⌘ | Dynamic Links and embedded client |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Submitter:*** ⌘ | (Samsung) Michael Koster | | | | | |
| ***Owner:*** ⌘ | | (Samsung) Michael Koster | | | | |
| ***Certification:*** ⌘ | X | |  | ***Date:*** ⌘ | | 2018-04-10 |
| ***Category:*** ⌘ | F | |  |  | |  |
|  | *Use one of the following categories:* ***C*** *(correction)* ***F***  *(functional modification of feature)* ***E*** *(editorial modification)* | | | |  | |
| ***Summary and Reason for change:*** ⌘ | | | Add Dynamic Links (Dynlink) and embedded client specification for rules, alerts, scenes, and push pattern support | | | |
|  | | | | | | |
| ***Location of changes:*** ⌘ | | | (new section) | | | |
|  | | | | | | |
| ***Other comments:*** ⌘ | | |  | | | |

**How to propose a technical solution to the Issue submitted using this form (we are using the MS Word revision marks feature (also known as track changes) to designate the modified text in a Change Request. Please use revision marks as instructed below):**

Once the above Issue Report is filled out:

1. With “revision marks” disabled, copy the clause(s) from the Word version of the target specification subject to the Change Request into the following pages.
2. Then with “revision marks” enabled, make the changes to the clause(s).
3. If the CR adds or modifies references or includes new specification text, include all references in the CR and use bookmarks to create the references.
4. Do not denote new specification text, clauses, acronyms, references, etc. with revision marks. Instead, highlight (as appropriate) with a comment with an instruction; e.g., “Editor: New text to be inserted after Clause x.y.z”.
5. Do not try to force auto header numbering to work. When you copy in a heading, delete the auto-number and manually type in the correct number.

Note: if you are just submitting and issue and not submitting a technical solution, filling the second (and subsequent) page(s) is optional.

\*\*\*\*\*\* Paste the Change Request content here \*\*\*\*\*\*

\*\*\*\*\* First Change

**11.10 Dynamic Links (Dynlink)**

**11.10.1 Overview**

Dynamic Links are OCF Links which define a data transfer relationship between two Resource instances. One of the resource instances is located on the same device the Dynamic Link is stored on, and the other Resource instance may be on the same device or on some other device.

Dynamic Links are based on the IETF CoRE Dynlink draft [REF].

A Dynamic link may be used to observe a local or remote resource, and apply notifications as updates to a local resource (Observe Dynlink). A Dynamic Link may also be used to push updates which are applied to a local resource, to a local or remote resource (Push Dynlink).

Two resource types are defined, oic.r.dynlink and oic.r.dynlinkstatus.

The oic.r.dynlink resource type defines a collection of Dynamic Links and their associated metadata which are used to initiate and maintain automatic data transfers, using the functionality of an embedded client.

Each Dynamic Link in the collection has an associated instance of oic.r.dynlinkstatus, also in the collection, which exposes the configuration and state of the internal embedded client for the associated Dynamic Link.

The resource type oic.r.dynlink may be used to define a standalone resource as a container for one or more dynamic links, or it may be composed together with a resource instance of another type in order to add dynamic link functionality.

Dynamic Links are created using the oic.if.linkupdate interface and a link payload that contains the mandatory parameters needed to define the link, described in section 11.10.2. When the Dynamic Link is created, the server automatically creates the corresponding instance of oic.r.linkstatus, initializes the state of the link, and configures the embedded client to process the link.

A newly created Dynamic Link becomes active immediately upon creation. Creating an Observe Dynlink issues the Observe operation and updates the linkstatus resource accordingly upon receipt of the first response. Creating a Push Dynlink will prepare the embedded client to issue a push operation upon the next update, and all subsequent updates, of the local resource.

Conditional Notification parameters are respected when present at the source of the data transfer, for both Observe Dynlinks and Push Dynlinks.

**11.10.2 Dynamic Link Structure**

A Dynamic Link has parameters as shown in table 34. The target URI of the link, in the "href" parameter is always the source of the data transfer, and the context of the link, in the "anchor" parameter, is always the destination of the data transfer, for both Observe Dynlinks and Push Dynlinks.

A Dynamic Link is indicated by the link relation value rel=boundto.

Observe Dynlinks are indicated by a link attribute value bind=obs.

The Observe Dynamic Link example in Figure 38 specifies that the remote resource at the URI "**ocf://850faa5d-ccaf-4293-9452-f4fcab2e2c39/sensors/temperature**" is to be observed using **oic.if.s** and **oic.r.temperature**, and that notifications are to be applied to the resource at the relative referenced URI "**temperatureinput**".

When a notification is received from an Observe Dynlink, all of the resource properties in the received representation shall be applied to the destination resource.

**Figure 38. Example Observe Dynlink**

|  |
| --- |
| {  "href": "ocf://850faa5d-ccaf-4293-9452-f4fcab2e2c39/sensors/temperature",  "if": "oic.if.s",  "rt": "oic.r.temperature",  "rel": "boundto",  "anchor": "temperatureinput",  "bind": "obs",  "ins": "ca557af3"  } |

Push Dynlinks are indicated by a link attribute value bind=push.

The Push Dynamic Link example in Figure 39 specifies that, when updated, the value of the resource at the relative references URI "**heatcontrol**" is to be applied to the remote resource at the URI "**ocf://850faa5d-ccaf-4293-9452-f4fcab2e2c39/heater/onoff**" using an update request with **oic.if.a** and **oic.r.switch.binary**.

When an update from Push Dynlink is issued in response to a resource update, the transmitted representation shall contain all of the properties of the source resource, pointed to by "href", that are implemented by the interface specified in the Dynamic Link.

**Figure 39. Example Push Dynlink**

|  |
| --- |
| {  "href": "heatcontrol",  "if": "oic.if.a",  "rt": "oic.r.switch.binary",  "rel": "boundto",  "anchor": "ocf://850faa5d-ccaf-4293-9452-f4fcab2e2c39/heater/onoff",  "bind": "push",  "ins": "cdaf5503"  } |

**11.10.3 Dynamic Link Status Resource**

The resource type oic.r.dynlinkstatus contains properties that expose the status of a particular Dynamic Link.

One instance of a oic.r.dynlinkstatus resource is created by the server when a Dynamic Link created. This resource instance is updated when the data transfer state of its corresponding Dynamic Link changes.

The Link Status resource exposes the following properties:

The "**linkstate**" property exposes the state of the Dynamic Link and is defined by the enumeration ["init", "active", "error", "timeout"]. This property is read-only.

* The "**init**" state indicates that the Dynamic Link has been created but is still in its initial state, that is no requests have been issued.
* The "**active**" state indicates that the Dynamic Link has issued at least one request and has not received any error responses.
* The "**error**" state indicates that the Dynamic Link has issued at least one request and has received an error response. For CoAP, this would generally be a response other than 2.XX.
* The "**timeout**" state indicates that the Dynamic Link has been idle for a length of time exceeding the "maxidletime" setting.

The "**laststatus**" property contains information the last status code that was received from the remote server on a request generated from the corresponding Dynamic Link. Laststatus is a numeric code that is normalized to the HTTP status code set. The initial state of laststatus is set to 0 (zero), and is updated with each received status code until an error code is received. This property is read-only.

The "**maxidle**" property exposes the maximum time the corresponding Dynamic Link may remain idle without any activity before the timeout state is entered. Units are milliseconds. This property is read/write with a maximum value of (2^32)-1. This property is by default set to a value of 0 (zero) which disables the timeout state.

The "**idletime**" property exposes the amount of time that has elapsed since activity has occurred on the corresponding Dynamic Link. Units are in milliseconds. This property is read-only.

The "**linkinstance**" property exposes the "ins" value of the associated Dynamic Link

**11.10.4 Dynamic Link Life Cycle**

Figure 40 shows the life cycle of an Observe Dynlink.

1. The client creates a Dynamic Link using oic.if.linkupdate on a resource that exposes oic.r.dynlink
2. The server creates an instance of oic.r.linkstatus that corresponds to the Dynamic Link
3. The server returns a link representation with an "ins" attribute that has a unique value
4. The embedded client issues an Observe request to the target resource of the link.
5. If the client wishes to set an idle time limit, it uses the "ins" attribute to select and sets the maxidle property of the corresponding Link Status resource instance.
6. On response from the observed resource, the embedded client updates the Link Status Resource with any response status code, and updates the local resource at the context or anchor of the Dynamic Link with the returned value if the response indicates success.
7. On subsequent notify responses from the observed resource, the embedded client updates the link status resource with the received status code, and updates the resource at the context of the Dynamic Link with the received value if the notify response indicates success.
8. If the response from the observed resource indicates an error, the embedded client will hold the error status code in the laststatus property and clean up the cancelled observe request.
9. If the maxidle property is greater than zero and the idletime counter has exceeded this value, the server will set and hold the "timeout" value in the linkstate property and cancel the observe request.

Figure 41 shows the life cycle of a Push Dynlink.

1. The client creates a Dynamic Link using oic.if.linkupdate on a resource that exposes oic.r.dynlink
2. The server creates an instance of oic.r.linkstatus that corresponds to the Dynamic Link
3. The server returns a link representation with an "ins" attribute that has a unique value
4. If the client wishes to set an idle time limit, it sets the maxidle property of the corresponding Link Status resource instance.
5. Whenever the resource at the target of the link is updated, the embedded client issues an update request to the resource at the context or anchor of the Dynamic Link
6. On response from the updated resource, the embedded client updates the Link Status Resource with any response status code received
7. If the response from the observed resource indicates an error, the embedded client will hold the status code in the laststatus property and disable outgoing update requests.
8. If the maxidle property is greater than zero and the idletime counter has exceeded this value, the server will set and hold the "timeout" value in the linkstate property, and disable outgoing update requests.

**11.10.5 Link Parameters and Resource Properties**

Table 34 shows the link parameters of a Dynamic Link

Table 34. Link Parameters of Dynamic Link

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Link Parameter | Name | Value rule | Mandatory | Description |
| Link Target | href | URIreference | yes | Data Source |
| Link Context | anchor | URIreference | yes | Data Destination |
| Link Relation | rel | "boundto" | yes | Indicates a Dynamic Link Type |
| Binding Type | bind | "obs","push" | yes | Type of binding |
| Interface | if | string | no | Interface type to use in requests, may override the default interface |
| Resource Type | rt | string | no | Resource Type to use in requests, may override the default resource type |
| Instance | ins | number | yes | Server or client supplied |

Table 35 shows the properties of the oic.r.linkstatus resource type

Table 35. Link Status Resource Property definition

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Property title | Property name | Value type | Value rule | Unit | Access mode | Mandatory | Description |
| Link State | linkstate | string | enum |  | R | yes | Life cycle state of the Dynamic Link |
| Last Status | laststatus | string | enum |  | R | yes | Status code from the last response |
| Maximum Idle Time | maxidle | number |  | milliseconds | RW | yes | Timeout value, 0 (zero) to disable timeout |
| Idle Time | idletime | number |  | milliseconds | R | yes | Time since the last activity on the Dynamic Link |
| Link Instance | linkinstance | number |  |  | R | yes | Value of the "ins" link parameter of the Dynamic Link |

Table 36 shows the link parameters for the oic.r.linkstatus resource type

Table 36. Link Parameters of Link Status Resuorce

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Link Parameter | Name | Value rule | Mandatory | Description |
| Link Status Resource URI | href | URIreference | yes | Location of the Link Status Resource |
| Link Instance | linkinstance | "ins" of the Dynamic Link | yes | To identify the Link Status resource corresponding to a particular Dynamic Link |
| Resource Type | rt | "oic.r.linkstatus" | yes | Link Status Resource Type |
| Interface | if | "oic.if.rw" | yes | Link Status Resource Interface Type |