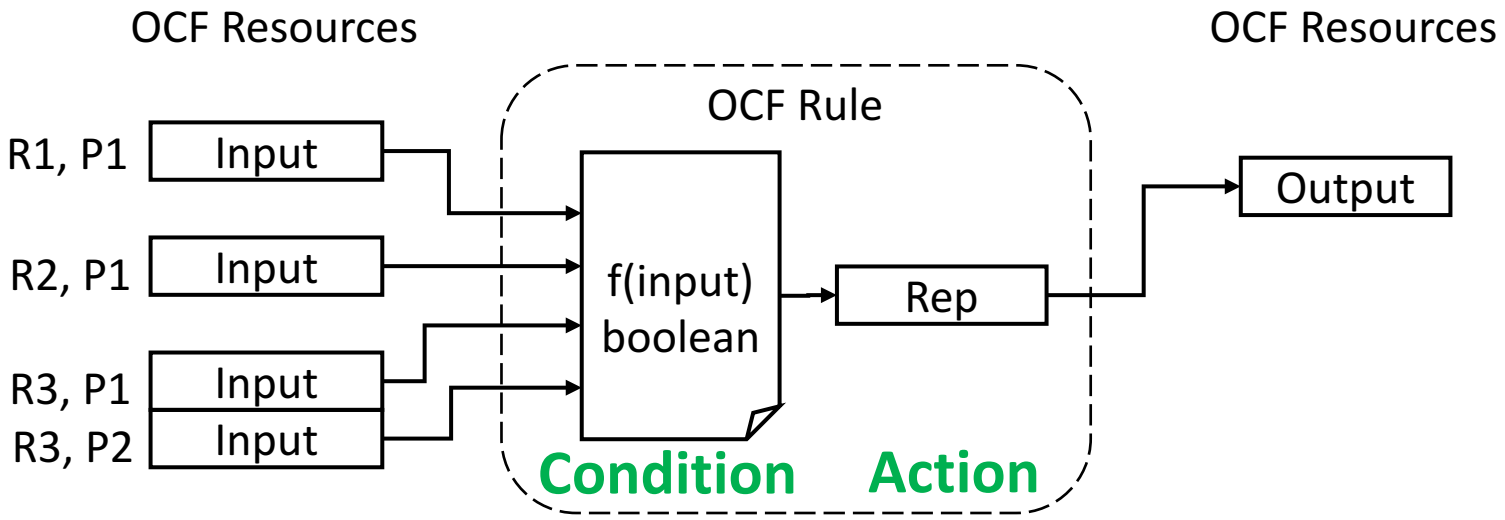
Rules, Scenes, Scripts, Modes, and Groups

Design Patterns for OCF Links and Collections

Michael J Koster

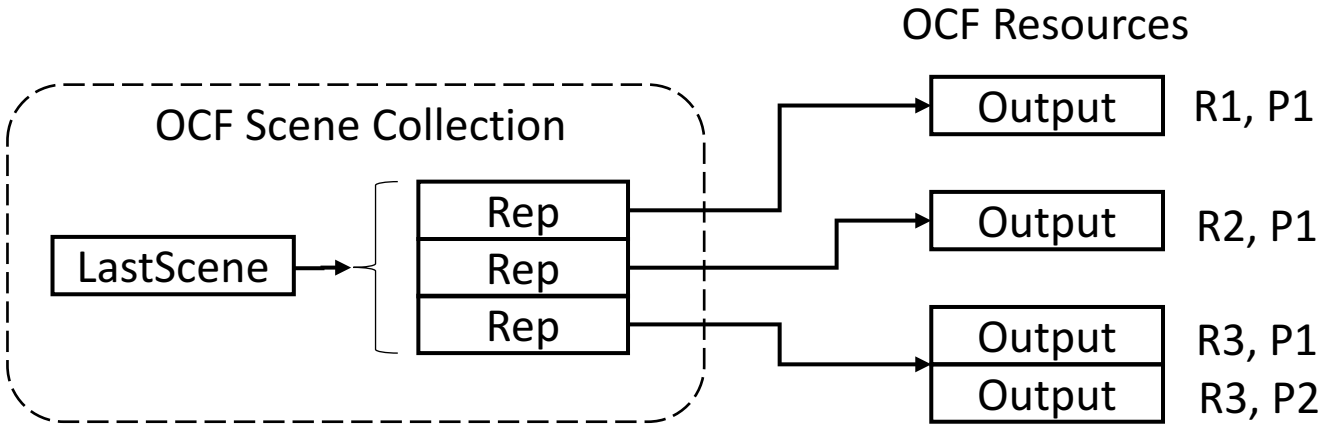
December 14, 2017

Use case for Rules



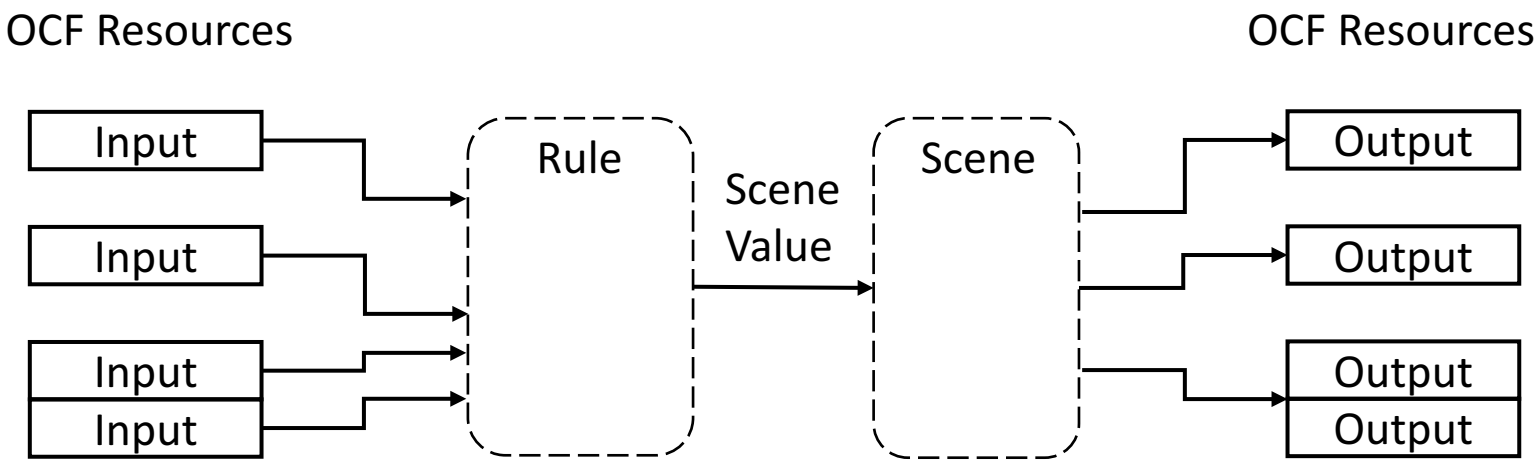
- When Condition f() evaluates to true, Action is to update some Output resource using a particular Representation

Use Case for Scenes



- Setting the LastScene value results in an update of the Output Resources using a particular set of representations selected by the LastScene value
- Different Scene Values may result in different output states

Use Case for Rules + Scenes

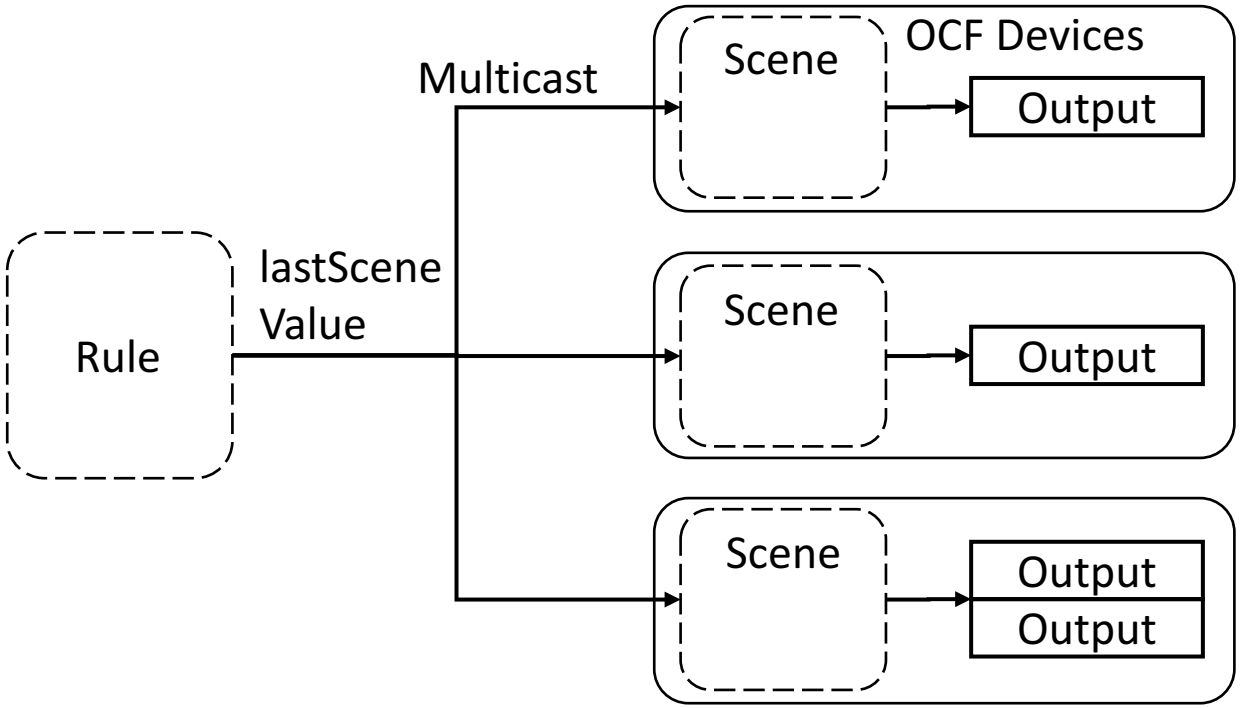


- Rules evaluating to true, trigger Scene changes
- SceneValue is output from the rule to update the LastScene value in the Scene Collection

Use Case for Groups

- Scenes execute in more than one device, triggered by multicast update of the lastScene property
- This is how large numbers of things may be orchestrated using a multicast update
  - Using a lastScene multicast solves the problem of actuating diverse resources using a single payload

Use Case for Groups



- lastScene update is multicast from some rule to a group of scene collections in separate devices

Script Use Case

- Programming language functionality triggered by rule evaluation
- Complex Behavior
- Sequences and timing

Use Case for Modes

- A mode is a group of rules
- A Mode can be modeled by a state in the Script Machine
- "home", "away" are modes

Additional requirements relative to the current designs

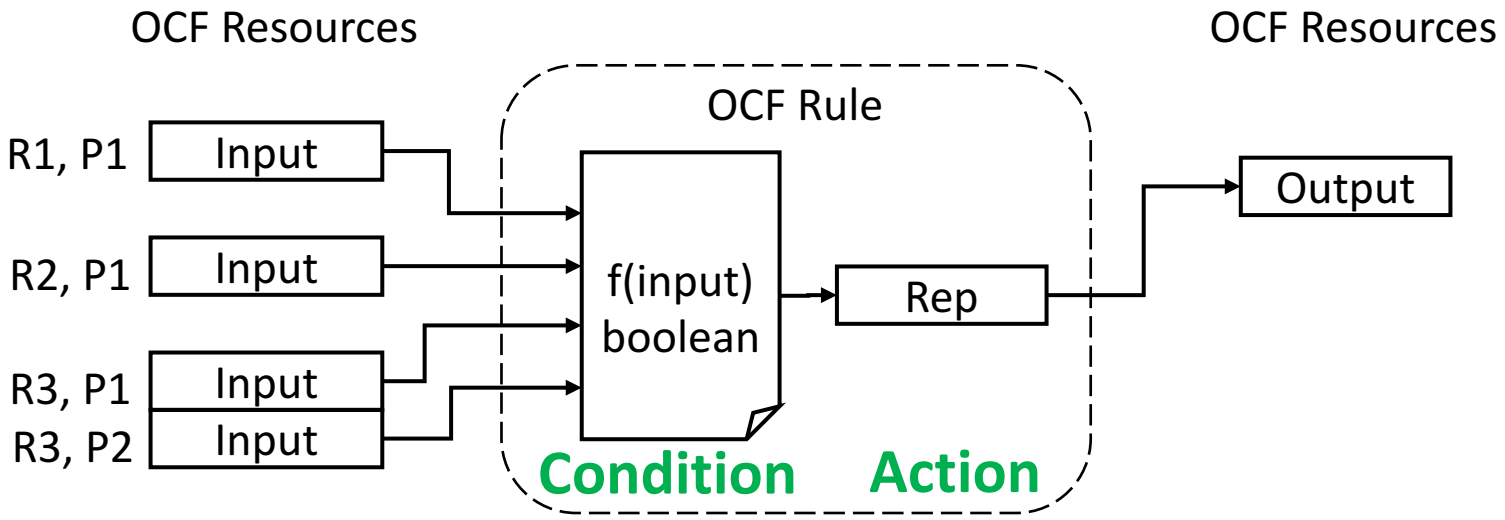
1. Rule inputs should be mapped to resources using links for reusability of rules e.g. marketplace
2. Scene and rule outputs need to perform batch updates and actuation
3. Scenes need to be able to be split up and stored in the end devices or intermediaries
4. Groups need to interact at the resource and property level on target devices

Designs

- Rules
- Scenes
- Groups
- Modes and Scripts

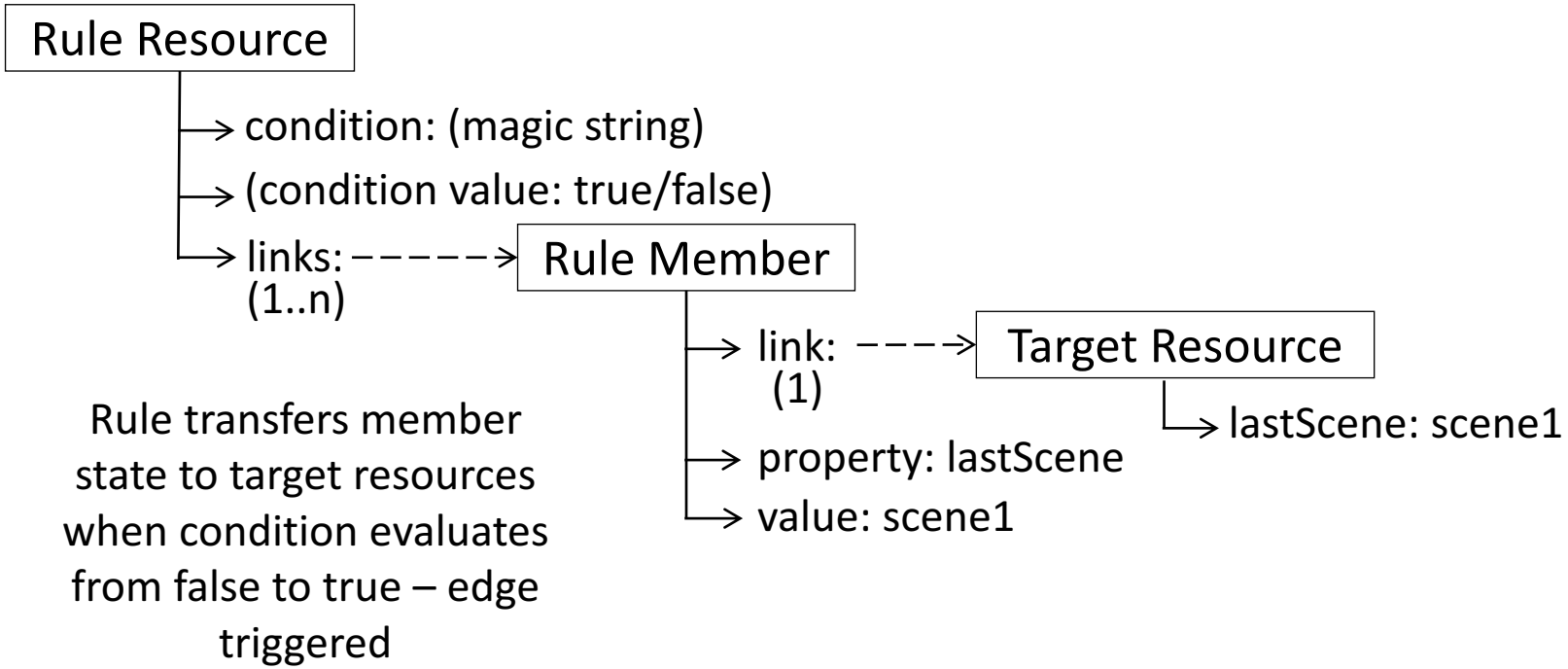
Rules Design Proposal

Use case for Rules

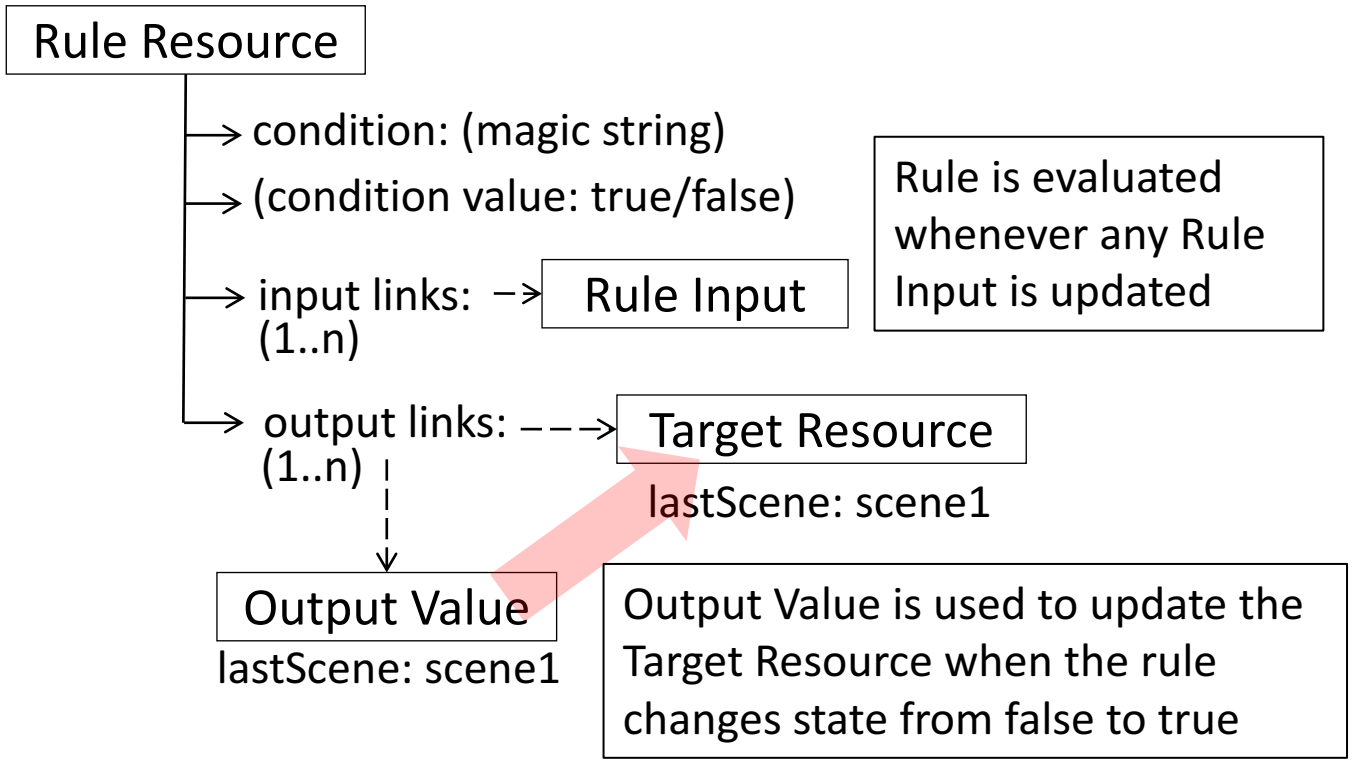


- When f() evaluates to true, update the Output resource using a particular Representation

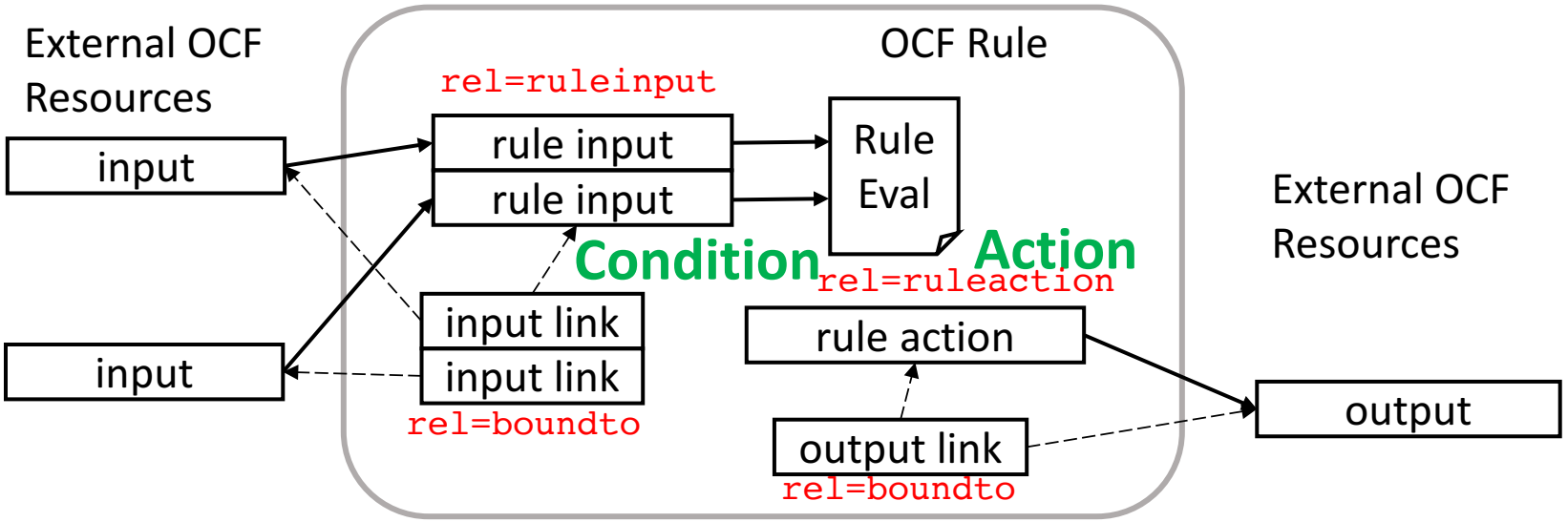
Rules as currently defined



Refactor Rules (Req. 1 & 3)



Rule Design


External OCF
Resources
input
input
OCF Rule
rel=ruleinput
rule input
rule input
Rule
Eval
Condition
Action
rel=ruleaction
input link
input link
rel=boundto
rule action
output link
rel=boundto
External OCF
Resources
output

Rule Resource Design

- Local resources for rule condition inputs and rule action output payload
- Input Links point to external input resources and target attributes e.g. resource type, interface
- EBNF resource that specifies input resource names, extracts properties by name, and applies conditions
- Output Links that point to action destinations and specify the resource from which to obtain a representation for the update payload

Rule Behavior

- Input registers are local resources that have the correct RT for the rule to evaluate as rule inputs
  - Rule inputs can observe external resources, using links, or may be updated from some client
  - The rule is evaluated each time an input is updated
- Rule actions are updated each time the rule transitions from evaluating false to evaluating true
  - Rule actions may be observed and notify on rule trigger
  - Output links are processed when rule actions are updated, resulting in data transfer to the external resource

Property Selection within a Rule

- EBNF definitions for text format
- Resources may have many (named) properties
- Resource properties to select for rule inputs are identified in the rule text using a delimiter
- For example, "switch:value identifies that the "value" property of the "switch" input resource is to be evaluated, e.g.:

```
(switch:value == true) &&
(temperature:temperature >=  30)
```

Examples

https://github.com/mjkoster/ocf-newmodel-examples/blob/master/rule-baseline.json

rule input (local link)

```
{
  "href": "switch",
  "rel": ["item", "ruleinput"],
  "rt": "oic.r.switch.binary",
  "if": ["oic.if.a"]
}
```

input link

```
{
  "anchor": "switch",
  "rel": ["boundto"],
  "bind": "obs",
  "href": "ocf://deviceID/switchhref",
  "if": ["oic.if.s"]
}
```

rule (rep)

```
"rule": "(switch:value == true) and
(temperature:temperature >= set-
temp:temperature)"
```

rule action (local link)

```
{
  "href": "scenevalue",
  "rel": ["item", "ruleaction"],
  "rt": "oic.wk.scenevalue",
  "if": ["oic.if.a"]
}
```

output link

```
{
  "href": "scenevalue",
  "rel": ["boundto"],
  "bind": "update",
  "anchor": "ocf://deviceID/scenehref",
  "if": ["oic.if.s"]
}
```

rule action (rep)

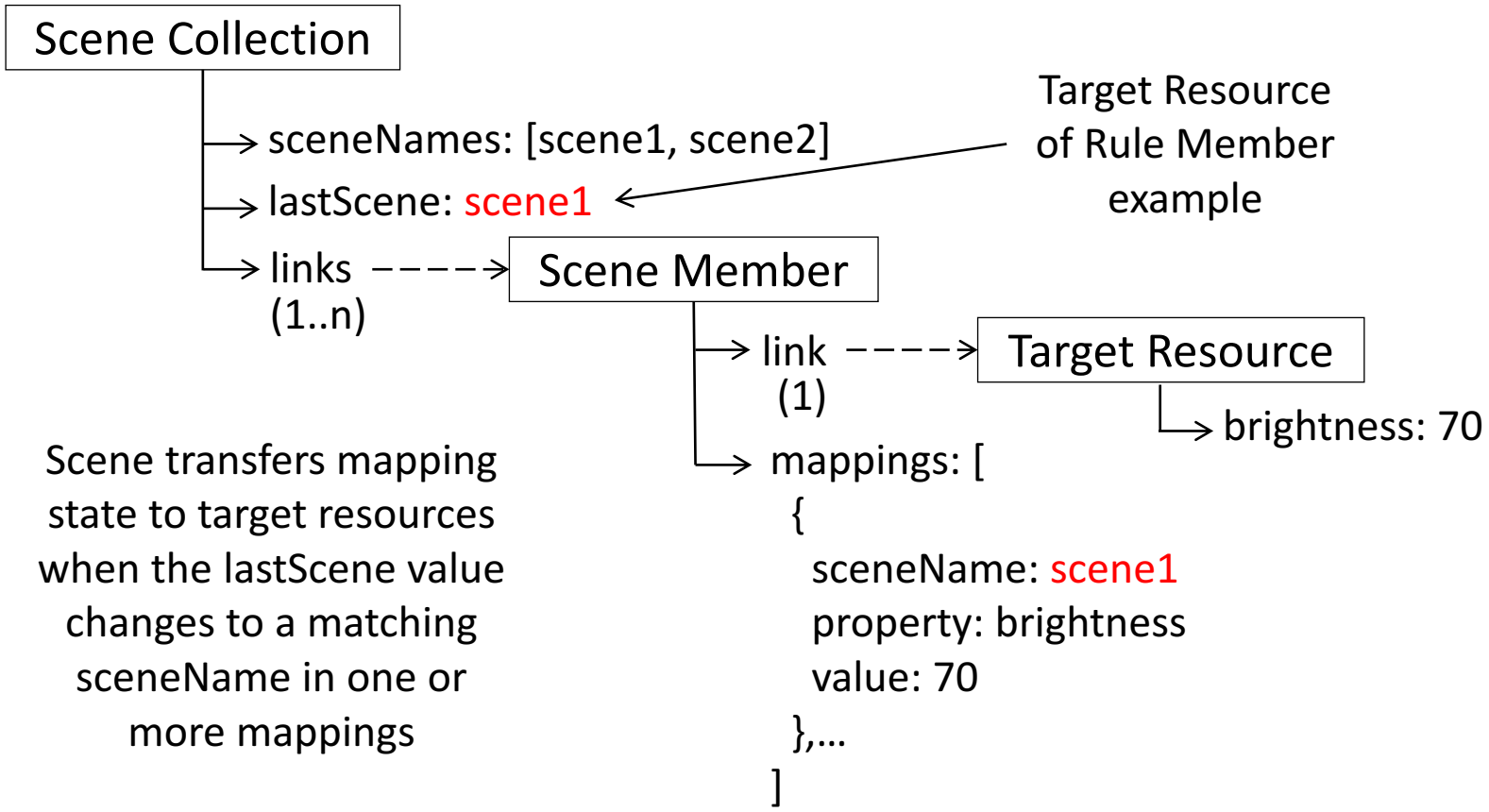
```
{
   "lastScene": "heat-off",
}
```

How Rules (and Scenes, etc.) are created and managed

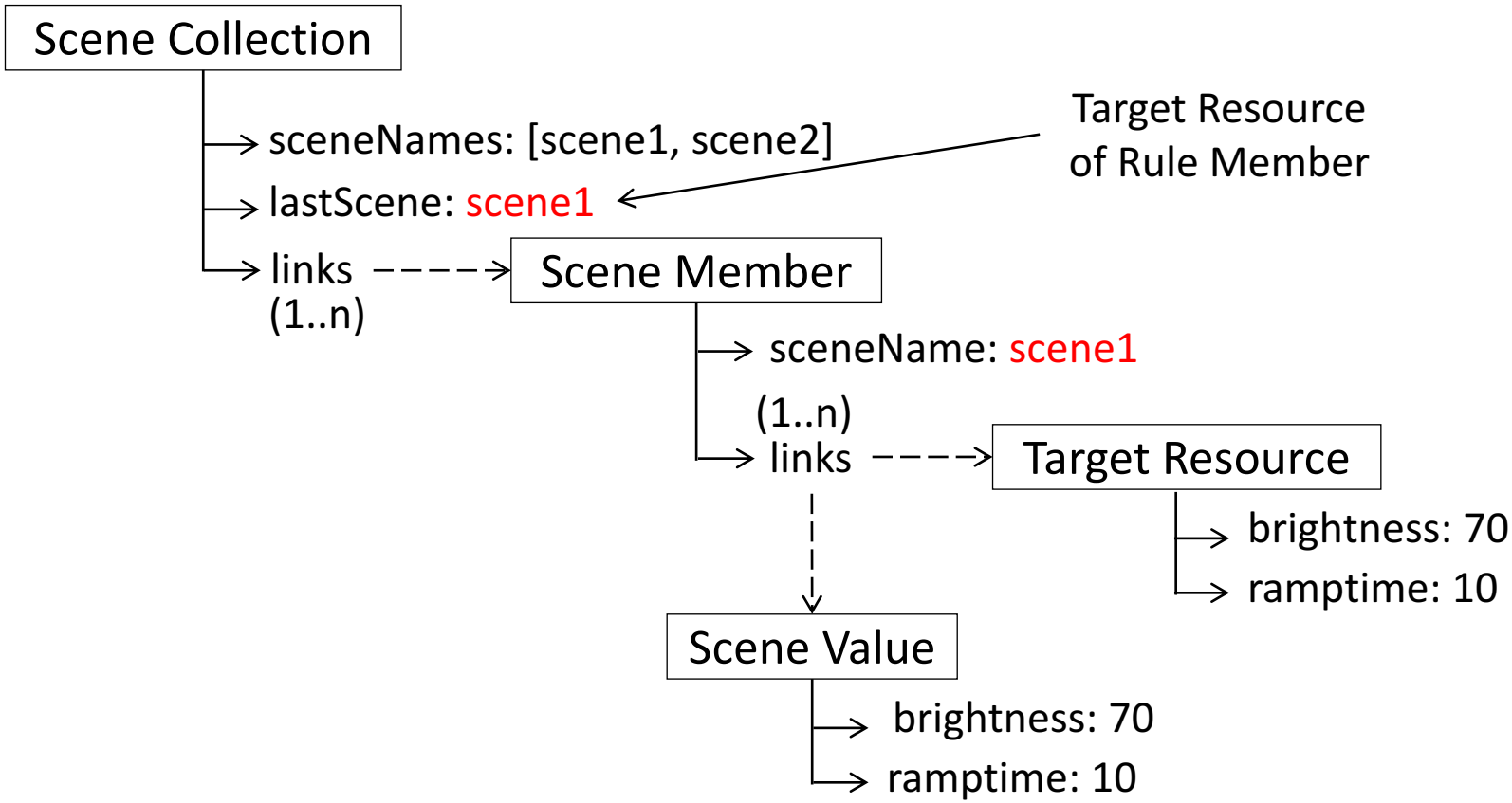
- We need a new ability to create items in a collection and links that point to them
- Ad-hoc method is to create a resource type with the desired behavior on POST
- Should we create an "Batch Create" interface type, e.g. if=oic.if.batchcreate, to standardize this?
- Payload would need to contain links and item representations – the oic.if.b schema extended with more link parameters like "if" and "rt"

Scenes Design Proposal

Scenes as currently defined



Refactor Scenes (Req. 1&2)



New pattern

- Use a link to define the transfer of an arbitrary payload (scenevalue) to a target resource:

```
{
  "href": "<rep-to-transfer>",
  "rel": "scenevalue",
  "anchor": "<targetresource>"
}
```

- Example Scene Value pointed to by "href" above

```
{
  "brightness": 70,
  "ramptime": 10
}
```

Scene Collection

- LastScene resource
- SupportedScenes array (allowed scene identifiers)
- Links to output resources
- Scene identifier stored in LastScene triggers update operations using a set of output links
- Output links work like Rule Outputs
  - transfer structured representations
  - contain target attributes, e.g. resource type, interace

Scene Value Collection

- Refactored to enable batch payloads, etc.
- Collection of output links associated with a scene value
- Links point to target resources, contain target attributes, and specify resources from which to obtain representations:

```
"anchor": "/example/controlpayloads/pwr-on",
"rel": "sceneoutput",
"href": "/example/device/pwrcontrol",
"rt": "x-com.example.rt.pwrcontrol",
"if": "oic.if.a"
```

Modes and Scripts Design Proposal

Script Machine Resource

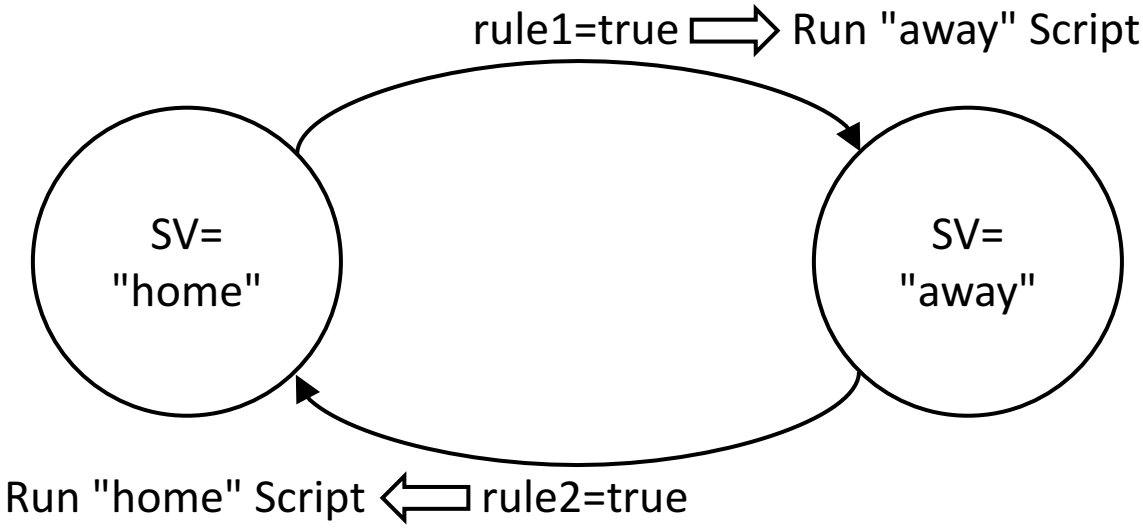
- Links to rules and scene collections
- Mapping of scene values to rule selectors
- Mapping of scene values to handler scripts
- State Machine description using rules and SceneValues:

```
when state == home:
  when rule1 == true: state <= away
when state == away:
  when rule2 == true: state <= home
```

Script State Machine

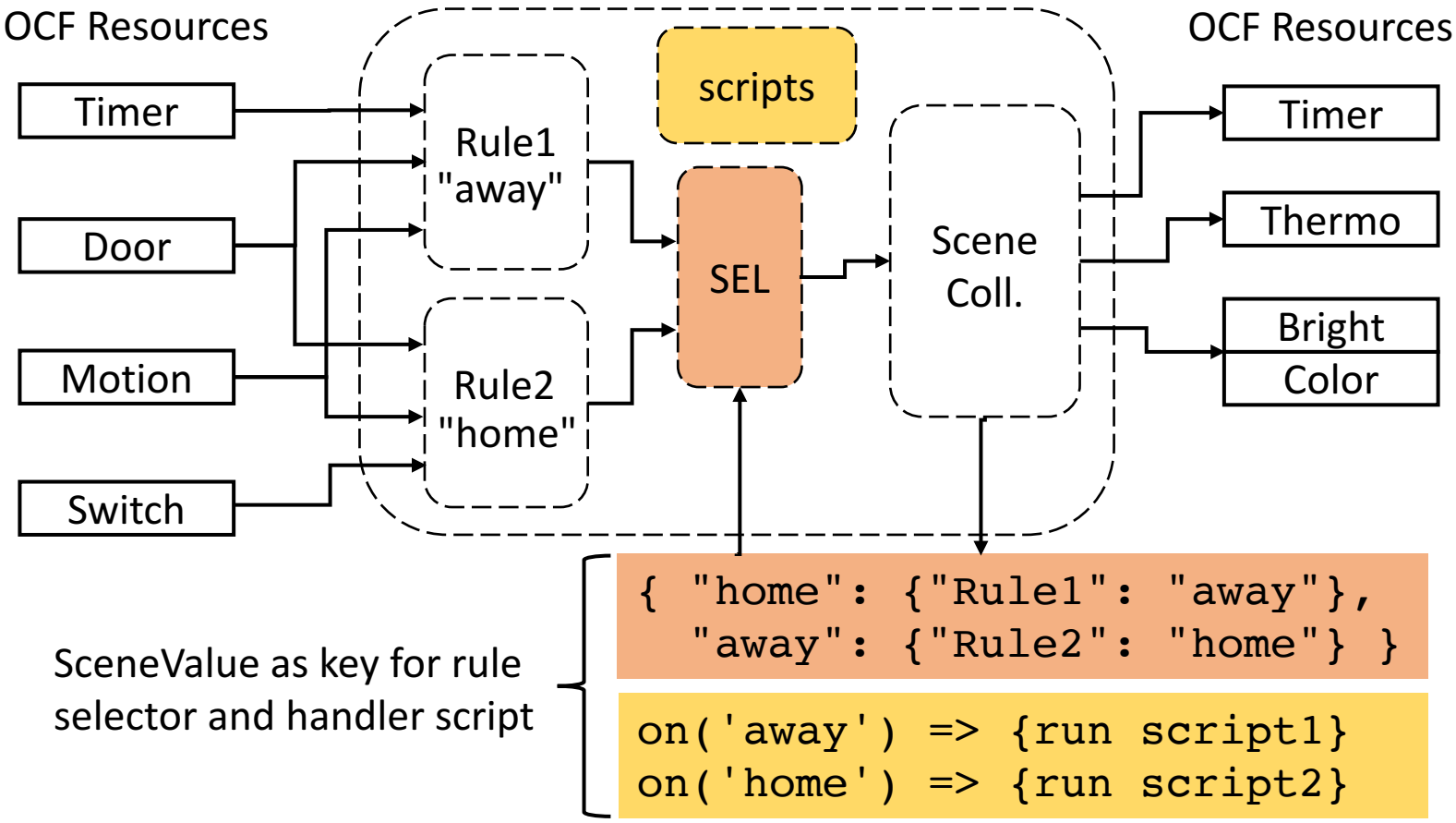
- Scene Value == State variable (bubble)
- State Transitions (arcs) are triggered by Rules
- Scene Value selects one or more rules that define outgoing state transitions (arcs)
- State Machine allows the definition of complex behaviors while avoiding conflicts
- State Machine allows controlled triggering of script execution; Script executes when a state is entered
- States may be used for Modes, selecting rulesets

Script Machine Example (2 Modes)



- Script runs when entering a state, triggered by a rule evaluation

Script Machine Example (2)



Groups Design Proposal

Groups

- Multicast groups need to be able to map multicast network addresses at the device level
- Multicast requests apply the same path and options to all devices
- Desired resources in a group may be at different paths in each device
- Linked rule inputs can point to resources at well known paths for multicast targets

Groups

- A Group resource should contain a multicast addresses with associated security material
- The multicast address should be a link containing a network address that a client can use directly, or a link to a proxy resource that exposes the multicast group