Extending SDF and CoAP for AI Agent Interaction with IoT Devices

draft-corneo-t2trg-sdf-coap-agents-00

Lorenzo Corneo, Jaime Jiménez, Ari Keränen

Overview

- CoAP and SDF are part of the IoT stack and can be used by AI agents, see Jaime’s T2TRG presentation @ IETF 123 Madrid
- In this draft we aim at further improve these protocols for Agentic AI consumption
- Discussion on CoAP extensions
- Discussion on SDF extensions

Problem Statement

- We tested CoAP and SDF in projects using AI agents and we identified the following gaps:
  - CoAP has no way to distinguish between an AI agent and a human (or a machine)
  - SDF models leads to hallucinations when certain fields are missing or are poorly written.

AI agent identification in CoAP

- Proposed procedure in three steps
  - `Agent-ID` CoAP option
  - Key discovery via Resource Directory
  - Agent-Aware Server Behavior

Agent-ID CoAP Option [CoAP 1/3]

- CoAP elective option that carries:
  - An agent identifier string
  - A cryptographic signature allowing the server to verify the agent's claimed identity.
  - A reference to where the agent's public key can be retrieved.

```
GET /sensors/temperature
Agent-ID: h'a4012667...'
(COSE_Sign1, keyid "sig1")
Accept: 60
(application/cbor)

2.05 Content
Content-Format: 60
(application/cbor)

22.5
```

Key Discovery via Resource Directory [CoAP 2/3]

- Use Resource Directory RFC 9176
- Extend it to host agent key material (or links to externally hosted keys)
- A server receiving an Agent-ID with unknown key identifier can query the RD

Agent-Aware Server Behavior [CoAP 3/3]

Once the server verified the key, the server can:

- Return a richer descriptions alongside sensor data.
- Indicate content usage restrictions (e.g., data not to be used for model training).
- Apply rate limiting or access restrictions specific to automated clients.

SDF Extensions

- SDF content negotiation for AI agents
- AI agent-friendly SDF descriptions
  - Developer-defined templates
  - Structured examples for language models

Signaling Agent Intent [SDF 1/3]

- Define a new `content-type`, e.g., `text/sdf` and use it for agents' requests

```
GET /sdf/thermometer
Accept: TBD (text/sdf)

2.05 Content
Content-Format: TBD (text/sdf)

Temperature sensor. Property /temperature is the reading in
Celsius.
GET /temperature to read it.
```

AI agent-friendly SDF descriptions: Developer-Defined Templates [SDF 2/3]

- A new SDF quality, e.g., `sdfBot` adds natural-language instructions directly in SDF document
- A simple reference mechanism to construct descriptions by reference
- `text/sdf` requests trigger the delivery of AI agents content

```
{
  "sdfBot": "Temperature sensor. @T is the reading in
Celsius.",
  "sdfObject": {
    "thermometer": {
      "sdfProperty": {
        "temperature": {
          "description": "Measured temperature",
          "type": "number",
          "unit": "Cel",
          "botRef": "T",
          "sdfBot": "Read-only. GET /temperature"
        }
      }
    }
  }
}
```

Output:

Temperature sensor. Property /temperature is the reading in Celsius. GET /temperature to read it.

AI agent-friendly SDF descriptions: Structured Examples for Language Models [3/3]

- A new SDF quality `sdfStructuredExamples` includes examples and instructions for AI agents.
- `sdfStructuredExamples` includes:
  - `task`: describing the example task:
  - `steps` which includes:
    - `action`: the action the agent would take to complete the task
    - `parameters`: giving associated parameter values needed for the action

```
{
  "sdfStructuredExamples": [
    {
      "task": "Update firmware of the device to version 2",
      "steps": [
        {
          "action": "Set sdfProperty/firmware to v2 image",
          "parameters": "<new firmware file>"
        },
        {
          "action": "Execute sdfAction/startFirmwareUpdate"
        },
        {
          "action": "Observe sdfProperty/firmwareVersion",
          "readyWhen": "sdfProperty/firmwareVersion has v2"
        }
      ]
    }
  ]
}
```