Secure OTA updates for small devices with Uptane and RIOT

Anton Gerasimov

HERE Technologies

RIOT Summit, 2018

Outline

Background

- **Uptane** is a specification for secure over-the-air updates. Alternative to e.g. **SUIT**.
- Uptane is based in **TUF** (The Update Framework), adopted among the others by Docker, Digital Ocean and pip.
- **HERE OTA Connect** is one of the implementations of Uptane. Another is **OTAmatic** from Airbiquity.
- We had an Uptane client implementation for Linux-based devices, MCU support was missing.

Goals

- Develop a static library that the users/customers will be able to integrate into their bootloaders.
- Make a demo project using this library.
- Hardware platform:
  - Raspberry Pi with CAN dongle as an Uptane primary (gateway)
  - KEA129LEDLIGHTRD as an Uptane secondary (target device)

Why RIOT?

- BSP code is preserved in an open source project instead of slowly decaying in our own repo.
- RIOT already has ISO/TP implementation.
  Developed by Vincent Dupont:
  http://riot-os.org/files/RIOT-Summit-2017-slides/6-1-Network-Session-OTAkeys-CAN.pdf
- RIOT already has a bootloader implementation.

Why RIOT?

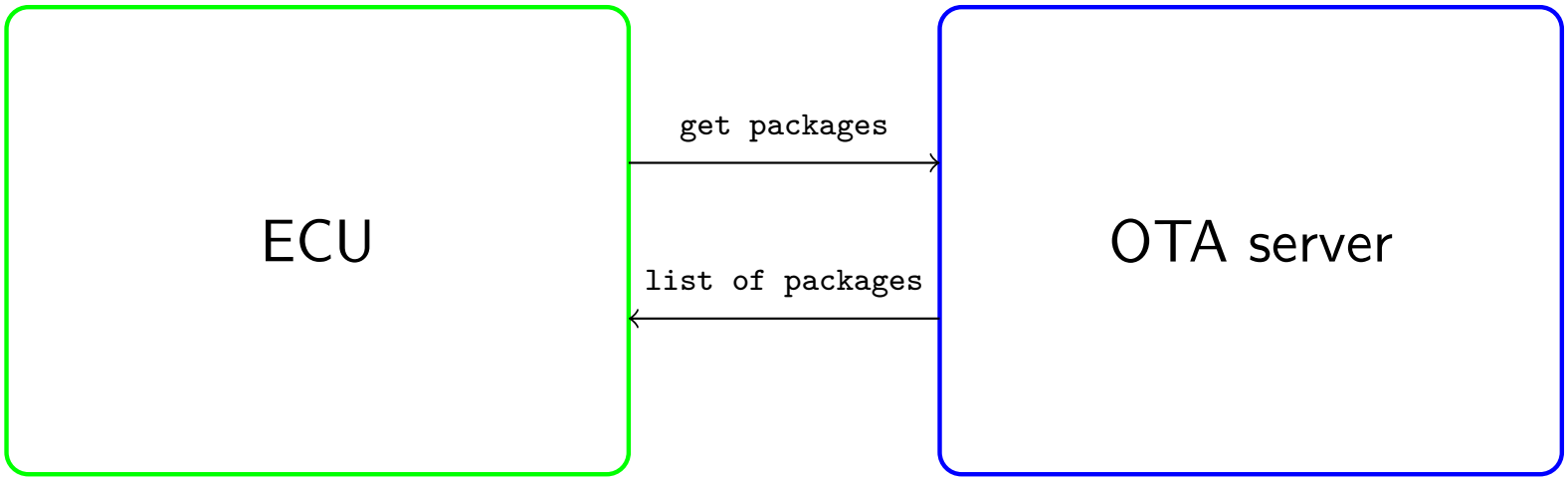
- BSP code is preserved in an open source project instead of slowly decaying in our own repo.
- RIOT already has ISO/TP implementation.
  Developed by Vincent Dupont:
  http://riot-os.org/files/RIOT-Summit-2017-slides/6-1-Network-Session-OTAkeys-CAN.pdf
- ~~RIOT already has a bootloader implementation.~~
- A great opportunity to contribute to RIOT OTA.

Results

- Static library (**libuptiny**)
- BSP code for KEA128LEDLIGHTRD for RIOT (on a PR now)
- Basic demo on KEA128LEDLIGHTRD (on a PR)
- Good intentions

Outline

Basic layout



- Installation is initiated by the device.
- Server gives out the list of available packages, device chooses what to install.
- This scenario is typical for user-controlled devices (i.e. infotainment panel).

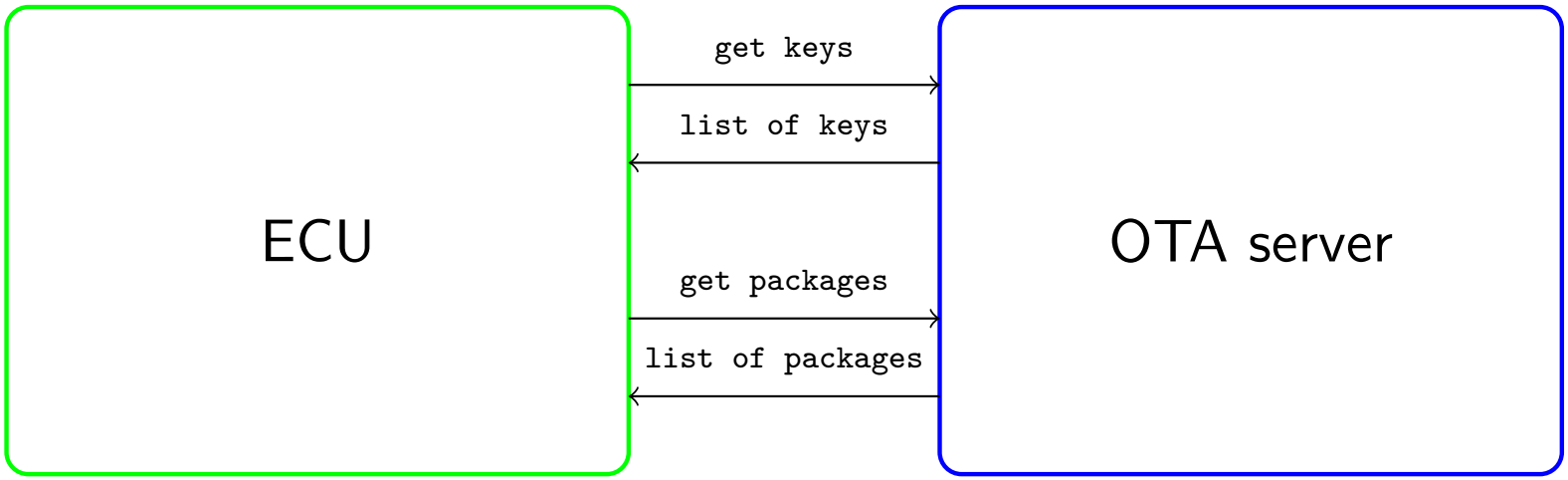
Basic layout (meatadata)

targets.json

```
{
  "signatures": [
    {
      "keyid": "ce69f17a69ca6...",
      "method": "ed25519",
      "sig": "gd945ORLpvCGo..."
    }
  ],
  "signed": {
    "_type": "Targets",
    "expires": "3021-07-13T01:02:03Z",
    "targets": {
      "first_firmware.txt": {
        "hashes": {
          "sha256": "1bbb15aa921..."
        },
        "length": 2092
      },
      "second_firmware.txt": {
        "hashes": {
          "sha256": "f309846c846..."
        },
        "length": 3120
      }

    },
    "version": 2
  }
}
```

Key rotation



- Additional piece of metadata (**root**) holding public keys for **targets** metadata and for itself.
- Makes key rotation possible: new keys are signed with the old ones.
- Initial set of keys needs to be provisioned to the device.

Key rotation (metadata)

targets.json

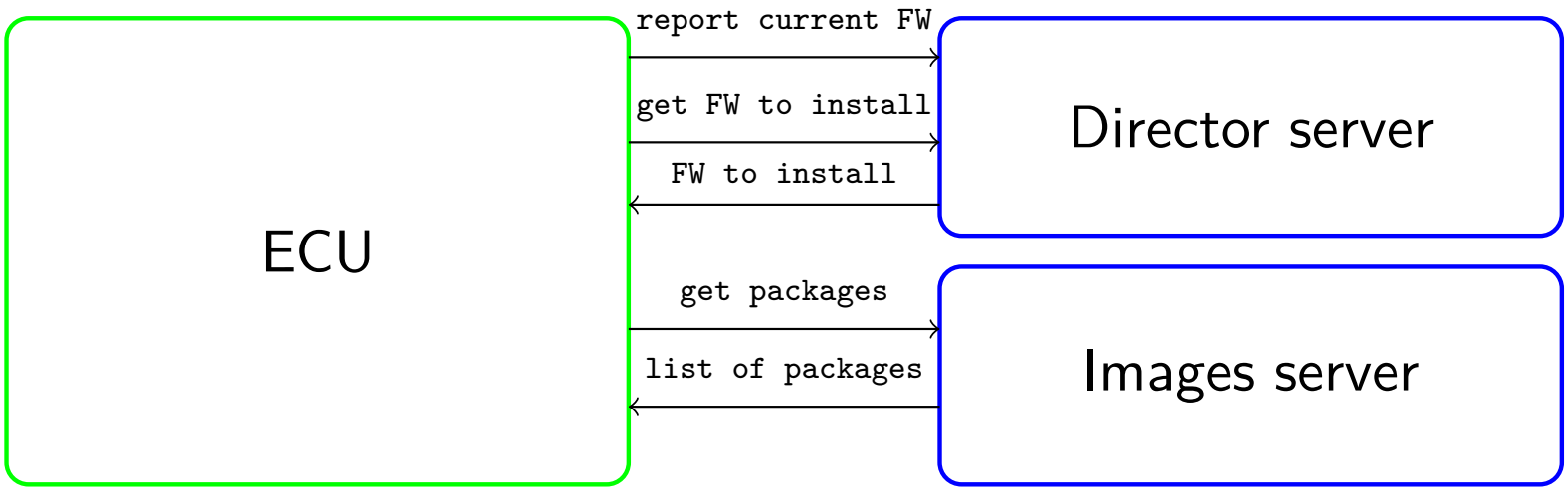
```
{
  "signatures": [
    {
      "keyid": "ce69f17a69ca6...",
      "method": "ed25519",
      "sig": "gd945ORLpvCGo..."
    }
  ],
  "signed": {
    "_type": "Targets",
    "expires": "3021-07-13T01:02:03Z",
    "targets": {
      "first_firmware.txt": {
        "hashes": {
          "sha256": "1bbb15aa921..."
        },
        "length": 2092
      },
      "second_firmware.txt": {
        "hashes": {
          "sha256": "f309846c846..."
        },
        "length": 3120
      }

    },
    "version": 2
  }
}
```

root.json

```
{
  "signatures": [...],
  "signed": {
    "_type": "Root",
    "expires": "3021-07-13T01:02:03Z",
    "keys": {
      "ce69f17a6...": {
        "keytype": "ED25519",
        "keyval": {
          "public": "1FAE772EF364C0..."
        }
      }
    },
    "roles": {
      "root": {
        "keyids": [
          "ce69f17a69ca6..."
        ],
        "threshold": 1
      },
      "targets": {
        "keyids": [
          "ce69f17a69ca6..."
        ],
        "threshold": 1
      }
    },
    "version": 1
  }
}
```

Director



- Now control is inverted: **director** tells the device what to install.
- **Images server** still tells what images exist and are valid.
- Both have **root** and **targets** metadata, format of the latter is slightly different for the two.
- **Director** metadata will be normally generated on the fly and is signed with **online keys**.
- **Images** metadata can be signed with **offline keys** (i.e. by the ECU vendor or OEM) for more security.

Director (metadata)

Director targets.json

```
{
  "_type": "Targets",
  "expires": "3021-07-13T01:02:03Z",
  "targets": {
    "second_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "IVI-Unit": {
            "hardwareId": "RPi3b"
          }
        }
      },
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    }
  },
  "version": 2
}
```

Images targets.json

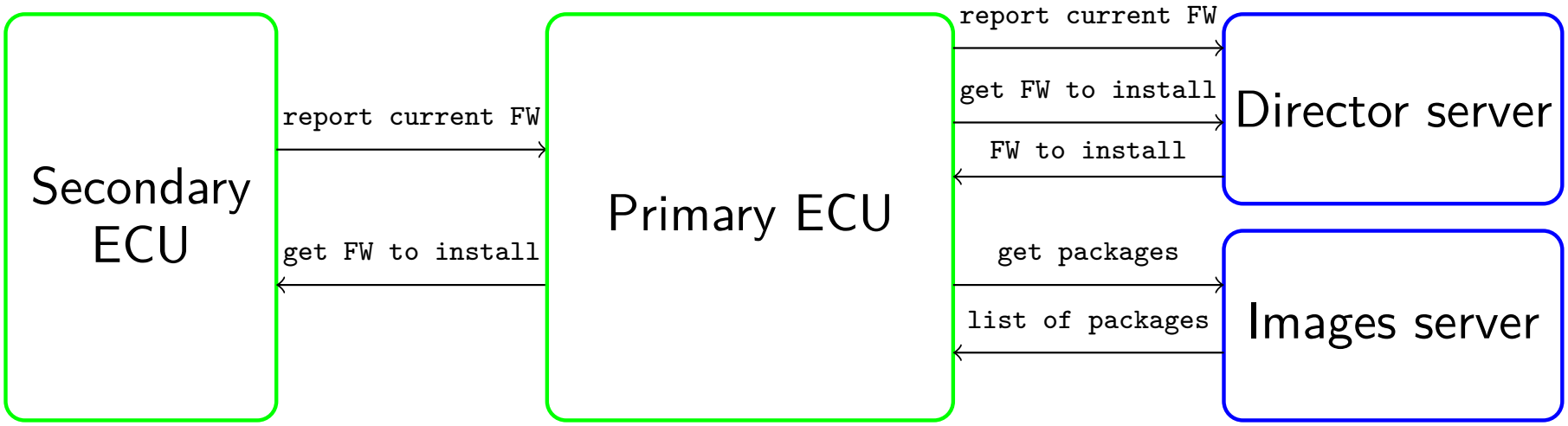
```
{
  "_type": "Targets",
  "expires": "3021-07-13T01:02:03Z",
  "targets": {
    "first_firmware.txt": {
      "hashes": {
        "sha256": "1bbb15aa921..."
      },
      "length": 2092
    },
    "second_firmware.txt": {
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    }
  },
  "version": 2
}
```

Director (continued metadata)

manifest.json

```
{
  "attacks_detected": "",
  "ecu_serial": "IVI-Unit",
  "installed_image": {
    "fileinfo": {
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    },
    "filepath": "second_firmware.txt"
  }
}
```

Multiple ECUs



- Normally only one device can communicate to the remote server directly, it is called **primary** in Uptane specification.
- Other devices (**secondaries**) are connected to the primary.
- Secondaries also verify metadata and firmware, so that compromised primary doesn't immediately compromise secondaries.
- **Libuptiny** is targeted at secondary devices.

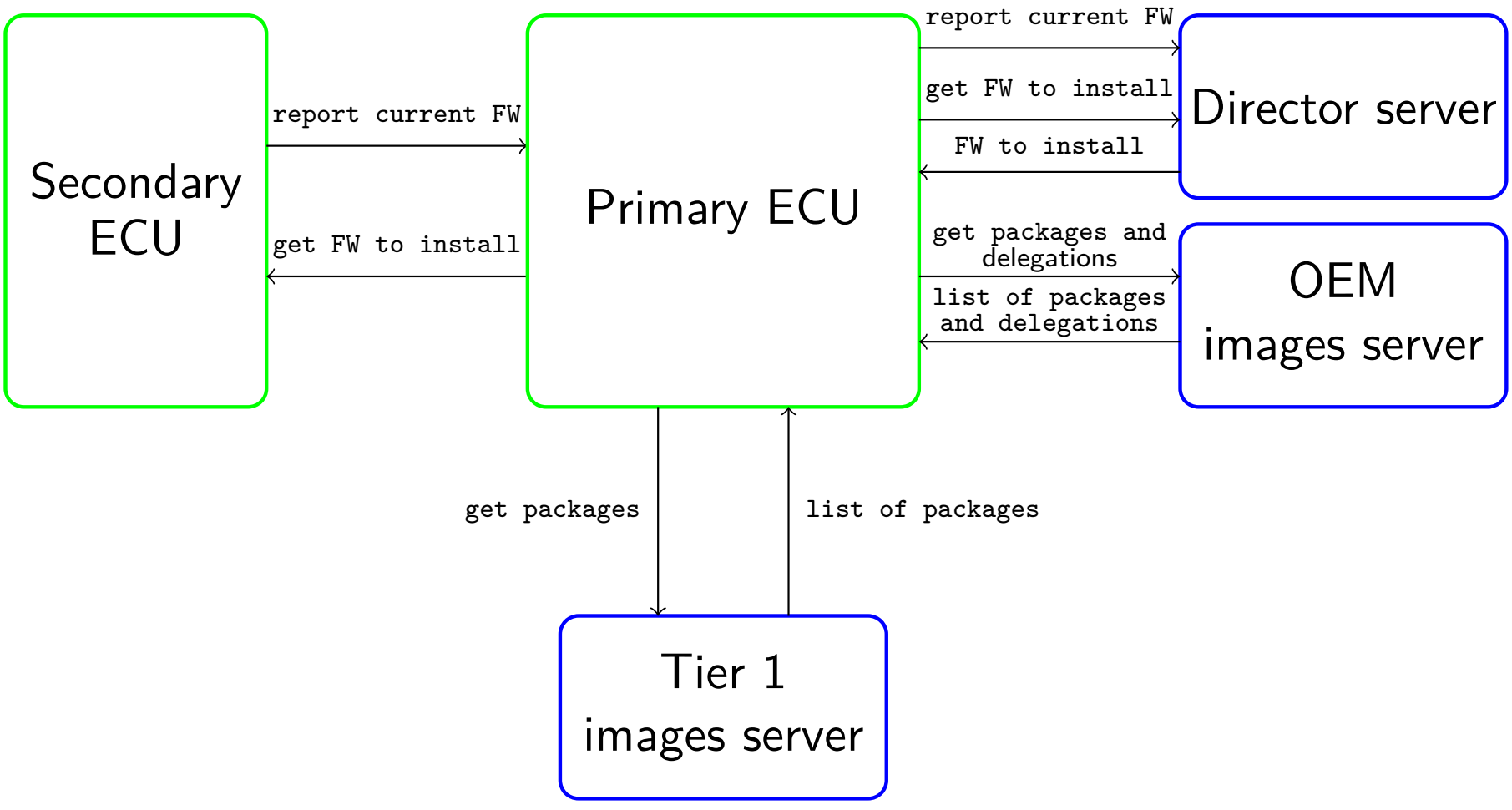
Multiple ECUs (director targets meatadata)

```
{
  "_type": "Targets",
  "expires": "3021-07-13T01:02:03Z",
  "targets": {
    "second_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "IVI-Unit": {
            "hardwareId": "RPi3b"
          }
        }
      },
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    },
    "first_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "Lights-controller": {
            "hardwareId": "LEDLIGHTRD"
          }
        }
      },
      "hashes": {
        "sha256": "1bbb15aa921..."
      },
      "length": 2092
    }
  },
  "version": 2
}
```

Multiple ECUs (manifest)

```
{
  "ecu_version_manifests": {
    "Lights-controller": {
      "signatures": [
                ...
      ],
      "signed": {
        "attacks_detected": "",
        "ecu_serial": "Lights-controller",
        "installed_image": {
          "fileinfo": {
            "hashes": {
              "sha256": "f309846c846..."
            },
            "length": 2092
          },
          "filepath": "first_firmware.txt"
        }
      }
    },
    "IVI-Unit": {
      "signatures": [
                ...
      ],
      "signed": {
                ...
      }
    }
  },
  "primary_ecu_serial": "IVI-Unit"
}
```

Delegations



- Main images repository may **delegate** trust to other images repositories with their own keys.
- E.g. car vendor (OEM) delegates to component manufacturer.

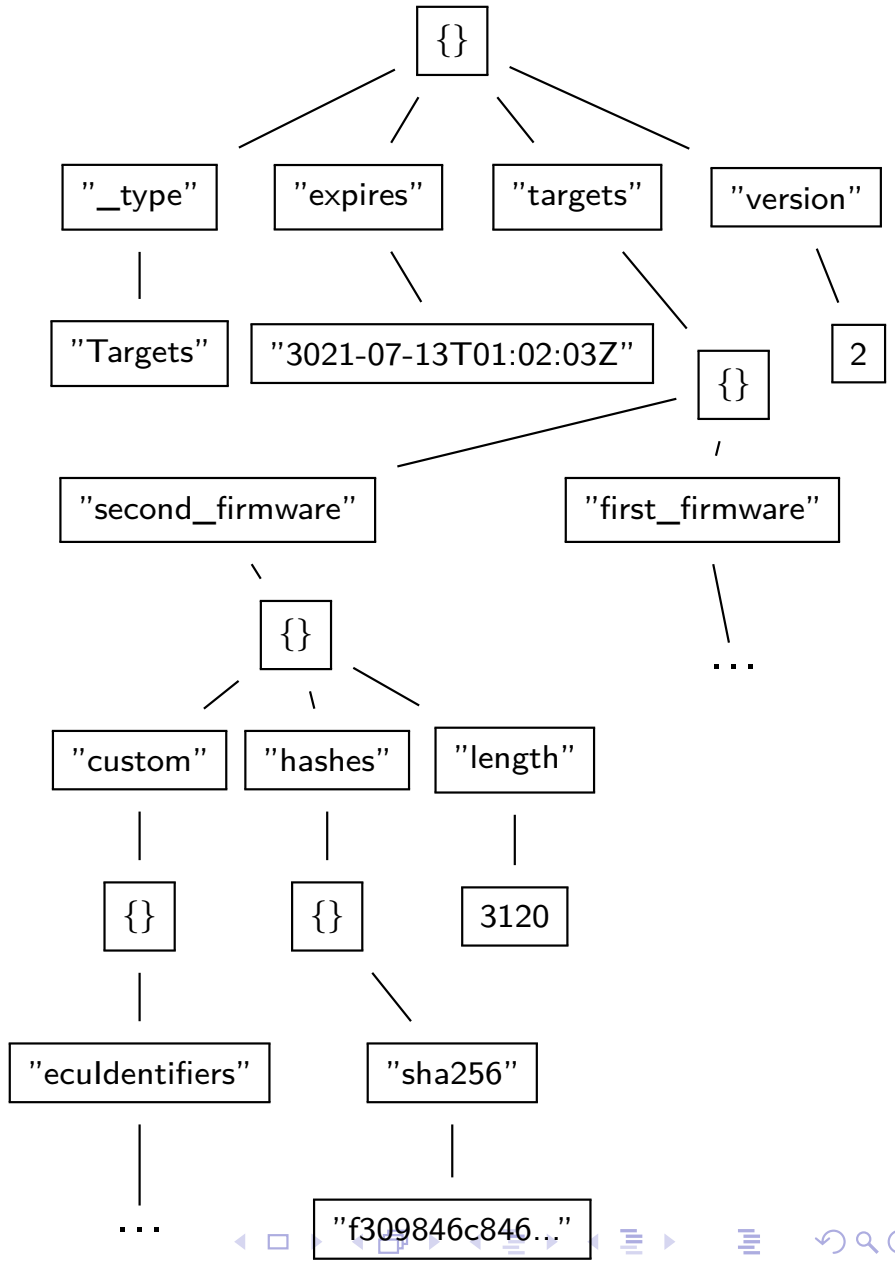
Outline

How do you parse JSON on MCU

- Backend developers love JSON.
- After all it could have been XML.
- So how do we implement it?
  - Parse the whole JSON object into traversable structure ("DOM" approach)
  - Parse JSON token by token ("SAX" approach)
  - Something hacky

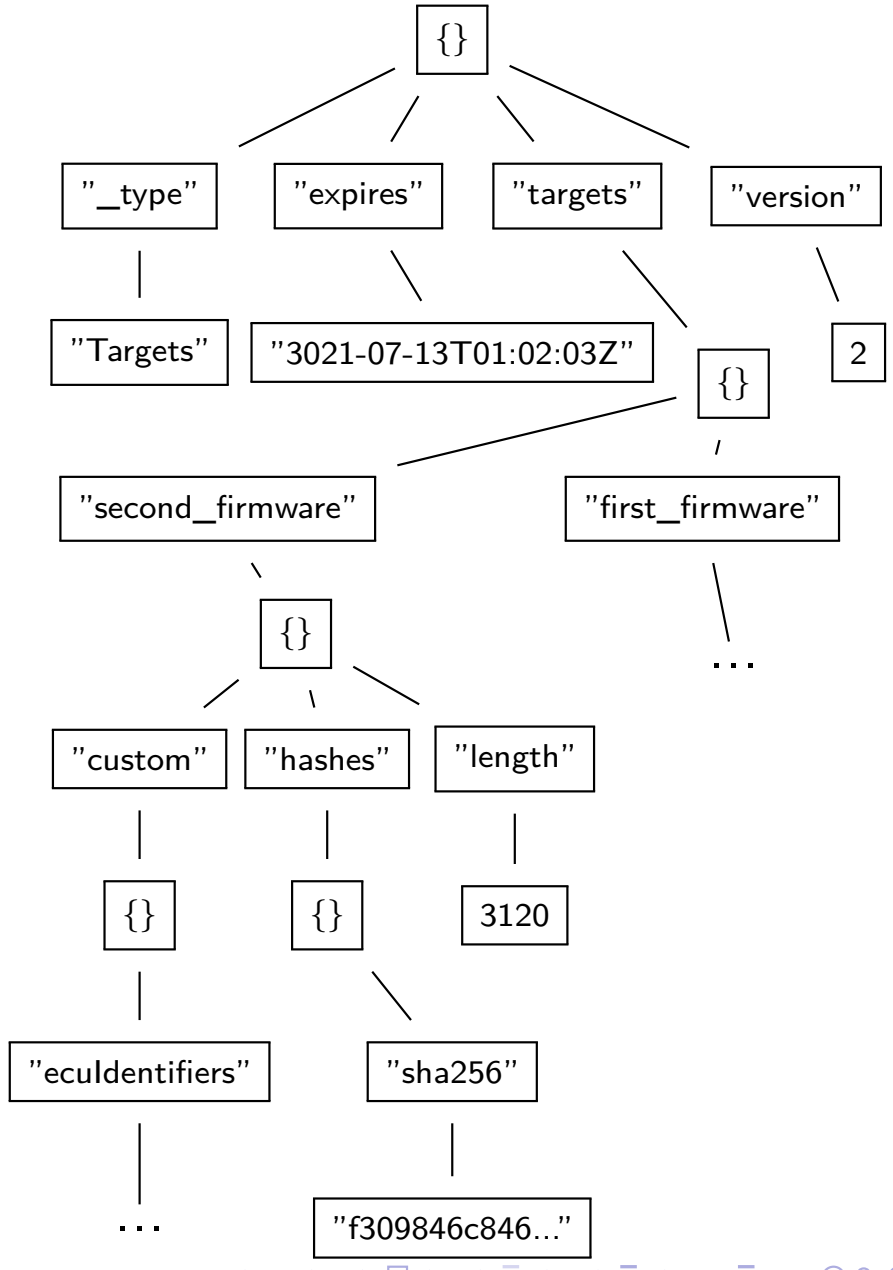
Why not the whole tree?

```
1  {
2    "_type": "Targets",
3    "expires": "3021-07-13T01:02:03Z",
4    "targets": {
5      "second_firmware.txt": {
6        "custom": {
7          "ecuIdentifiers": {
8            "IVI-Unit": {
9              "hardwareId": "RPi3b"
10           }
11         }
12       },
13       "hashes": {
14         "sha256": "f309846c846..."
15       },
16       "length": 3120
17     },
18     "first_firmware.txt": {
19       "custom": {
20         "ecuIdentifiers": {
21           "Lights-controller": {
22             "hardwareId": "LEDLIGHTRD"
23           }
24         }
25       },
26       "hashes": {
27         "sha256": "1bbb15aa921..."
28       },
29       "length": 2092
30     }
31   },
32   "version": 2
33 }
```



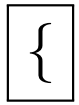
Why not the whole tree?

- Around 200-300 (and growing!) bytes per target in metadata.
- 10s to 100s of devices in a vehicle.
- Holding that array of data in RAM is not really an option.



Why not token by token?

```
{
  "_type": "Targets",
  "expires": "3021-07-13T01:02:03Z",
  "targets": {
    "second_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "IVI-Unit": {
            "hardwareId": "RPi3b"
          }
        }
      },
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    },
    "first_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "Lights-controller": {
            "hardwareId": "LEDLIGHTRD"
          }
        }
      },
      "hashes": {
        "sha256": "1bbb15aa921..."
      },
      "length": 2092
    }
  },
  "version": 2
}
```



Why not token by token?

```
1  {
2    "_type": "Targets",
3    "expires": "3021-07-13T01:02:03Z",
4    "targets": {
5      "second_firmware.txt": {
6        "custom": {
7          "ecuIdentifiers": {
8            "IVI-Unit": {
9              "hardwareId": "RPi3b"
10           }
11         }
12       },
13       "hashes": {
14         "sha256": "f309846c846..."
15       },
16       "length": 3120
17     },
18     "first_firmware.txt": {
19       "custom": {
20         "ecuIdentifiers": {
21           "Lights-controller": {
22             "hardwareId": "LEDLIGHTRD"
23           }
24         }
25       },
26       "hashes": {
27         "sha256": "1bbb15aa921..."
28       },
29       "length": 2092
30     }
31   },
32   "version": 2
33 }
```

"_type"

Why not token by token?

```
{
  "_type": "Targets",
  "expires": "3021-07-13T01:02:03Z",
  "targets": {
    "second_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "IVI-Unit": {
            "hardwareId": "RPi3b"
          }
        }
      },
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    },
    "first_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "Lights-controller": {
            "hardwareId": "LEDLIGHTRD"
          }
        }
      },
      "hashes": {
        "sha256": "1bbb15aa921..."
      },
      "length": 2092
    }
  },
  "version": 2
}
```

"Targets"

Why not token by token?

```
1  {
2    "_type": "Targets",
3    "expires": "3021-07-13T01:02:03Z",
4    "targets": {
5      "second_firmware.txt": {
6        "custom": {
7          "ecuIdentifiers": {
8            "IVI-Unit": {
9              "hardwareId": "RPi3b"
10           }
11         }
12       },
13       "hashes": {
14         "sha256": "f309846c846..."
15       },
16       "length": 3120
17     },
18     "first_firmware.txt": {
19       "custom": {
20         "ecuIdentifiers": {
21           "Lights-controller": {
22             "hardwareId": "LEDLIGHTRD"
23           }
24         }
25       },
26       "hashes": {
27         "sha256": "1bbb15aa921..."
28       },
29       "length": 2092
30     }
31   },
32   "version": 2
33 }
```

"expires"

Why not token by token?

```
1  {
2    "_type": "Targets",
3    "expires": "3021-07-13T01:02:03Z",
4    "targets": {
5      "second_firmware.txt": {
6        "custom": {
7          "ecuIdentifiers": {
8            "IVI-Unit": {
9              "hardwareId": "RPi3b"
10           }
11         }
12       },
13       "hashes": {
14         "sha256": "f309846c846..."
15       },
16       "length": 3120
17     },
18     "first_firmware.txt": {
19       "custom": {
20         "ecuIdentifiers": {
21           "Lights-controller": {
22             "hardwareId": "LEDLIGHTRD"
23           }
24         }
25       },
26       "hashes": {
27         "sha256": "1bbb15aa921..."
28       },
29       "length": 2092
30     }
31   },
32   "version": 2
33 }
```

"3021-07-13T01:02:03Z"

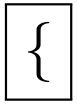
Why not token by token?

```
{
  "_type": "Targets",
  "expires": "3021-07-13T01:02:03Z",
  "targets": {
    "second_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "IVI-Unit": {
            "hardwareId": "RPi3b"
          }
        }
      },
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    },
    "first_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "Lights-controller": {
            "hardwareId": "LEDLIGHTRD"
          }
        }
      },
      "hashes": {
        "sha256": "1bbb15aa921..."
      },
      "length": 2092
    }
  },
  "version": 2
}
```

"targets"

Why not token by token?

```
{
  "_type": "Targets",
  "expires": "3021-07-13T01:02:03Z",
  "targets": {
    "second_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "IVI-Unit": {
            "hardwareId": "RPi3b"
          }
        }
      },
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    },
    "first_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "Lights-controller": {
            "hardwareId": "LEDLIGHTRD"
          }
        }
      },
      "hashes": {
        "sha256": "1bbb15aa921..."
      },
      "length": 2092
    }
  },
  "version": 2
}
```



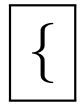
Why not token by token?

```
{
  "_type": "Targets",
  "expires": "3021-07-13T01:02:03Z",
  "targets": {
    "second_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "IVI-Unit": {
            "hardwareId": "RPi3b"
          }
        }
      },
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    },
    "first_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "Lights-controller": {
            "hardwareId": "LEDLIGHTRD"
          }
        }
      },
      "hashes": {
        "sha256": "1bbb15aa921..."
      },
      "length": 2092
    }
  },
  "version": 2
}
```

"second_firmware.txt"

Why not token by token?

```
{
  "_type": "Targets",
  "expires": "3021-07-13T01:02:03Z",
  "targets": {
    "second_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "IVI-Unit": {
            "hardwareId": "RPi3b"
          }
        }
      },
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    },
    "first_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "Lights-controller": {
            "hardwareId": "LEDLIGHTRD"
          }
        }
      },
      "hashes": {
        "sha256": "1bbb15aa921..."
      },
      "length": 2092
    }
  },
  "version": 2
}
```



Why not token by token?

```
{
  "_type": "Targets",
  "expires": "3021-07-13T01:02:03Z",
  "targets": {
    "second_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "IVI-Unit": {
            "hardwareId": "RPi3b"
          }
        }
      },
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    },
    "first_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "Lights-controller": {
            "hardwareId": "LEDLIGHTRD"
          }
        }
      },
      "hashes": {
        "sha256": "1bbb15aa921..."
      },
      "length": 2092
    }
  },
  "version": 2
}
```

"custom"

Why not token by token?

```
1  {
2    "_type": "Targets",
3    "expires": "3021-07-13T01:02:03Z",
4    "targets": {
5      "second_firmware.txt": {
6        "custom": {
7          "ecuIdentifiers": {
8            "IVI-Unit": {
9              "hardwareId": "RPi3b"
10           }
11         }
12       },
13       "hashes": {
14         "sha256": "f309846c846..."
15       },
16       "length": 3120
17     },
18     "first_firmware.txt": {
19       "custom": {
20         "ecuIdentifiers": {
21           "Lights-controller": {
22             "hardwareId": "LEDLIGHTRD"
23           }
24         }
25       },
26       "hashes": {
27         "sha256": "1bbb15aa921..."
28       },
29       "length": 2092
30     }
31   },
32   "version": 2
33 }
```

{

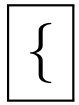
Why not token by token?

```
1  {
2    "_type": "Targets",
3    "expires": "3021-07-13T01:02:03Z",
4    "targets": {
5      "second_firmware.txt": {
6        "custom": {
7          "ecuIdentifiers": {
8            "IVI-Unit": {
9              "hardwareId": "RPi3b"
10           }
11         }
12       },
13       "hashes": {
14         "sha256": "f309846c846..."
15       },
16       "length": 3120
17     },
18     "first_firmware.txt": {
19       "custom": {
20         "ecuIdentifiers": {
21           "Lights-controller": {
22             "hardwareId": "LEDLIGHTRD"
23           }
24         }
25       },
26       "hashes": {
27         "sha256": "1bbb15aa921..."
28       },
29       "length": 2092
30     }
31   },
32   "version": 2
33 }
```

"ecuIdentifiers"

Why not token by token?

```
{
  "_type": "Targets",
  "expires": "3021-07-13T01:02:03Z",
  "targets": {
    "second_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "IVI-Unit": {
            "hardwareId": "RPi3b"
          }
        }
      },
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    },
    "first_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "Lights-controller": {
            "hardwareId": "LEDLIGHTRD"
          }
        }
      },
      "hashes": {
        "sha256": "1bbb15aa921..."
      },
      "length": 2092
    }
  },
  "version": 2
}
```



Why not token by token?

```
{
  "_type": "Targets",
  "expires": "3021-07-13T01:02:03Z",
  "targets": {
    "second_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "IVI-Unit": {
            "hardwareId": "RPi3b"
          }
        }
      },
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    },
    "first_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "Lights-controller": {
            "hardwareId": "LEDLIGHTRD"
          }
        }
      },
      "hashes": {
        "sha256": "1bbb15aa921..."
      },
      "length": 2092
    }
  },
  "version": 2
}
```

"IVI-Unit"

Why not token by token?

- State explodes really fast.
  - Where are we?
  - What was already read and what is still expected?
- The code quickly becomes unmanageable.
- Code generators might have helped

"IVI-Unit"

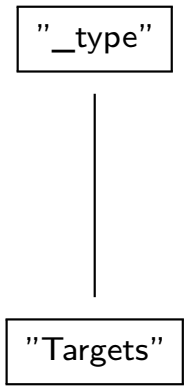
Something hacky

```
{
  "_type": "Targets",
  "expires": "3021-07-13T01:02:03Z",
  "targets": {
    "second_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "IVI-Unit": {
            "hardwareId": "RPi3b"
          }
        }
      },
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    },
    "first_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "Lights-controller": {
            "hardwareId": "LEDLIGHTRD"
          }
        }
      },
      "hashes": {
        "sha256": "1bbb15aa921..."
      },
      "length": 2092
    }
  },
  "version": 2
}
```

{

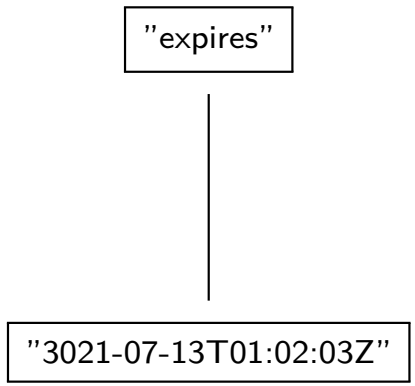
Something hacky

```
{
  "_type": "Targets",
  "expires": "3021-07-13T01:02:03Z",
  "targets": {
    "second_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "IVI-Unit": {
            "hardwareId": "RPi3b"
          }
        }
      },
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    },
    "first_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "Lights-controller": {
            "hardwareId": "LEDLIGHTRD"
          }
        }
      },
      "hashes": {
        "sha256": "1bbb15aa921..."
      },
      "length": 2092
    }
  },
  "version": 2
}
```



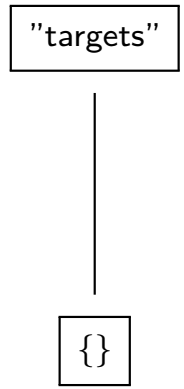
Something hacky

```
{
  "_type": "Targets",
  "expires": "3021-07-13T01:02:03Z",
  "targets": {
    "second_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "IVI-Unit": {
            "hardwareId": "RPi3b"
          }
        }
      },
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    },
    "first_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "Lights-controller": {
            "hardwareId": "LEDLIGHTRD"
          }
        }
      },
      "hashes": {
        "sha256": "1bbb15aa921..."
      },
      "length": 2092
    }
  },
  "version": 2
}
```



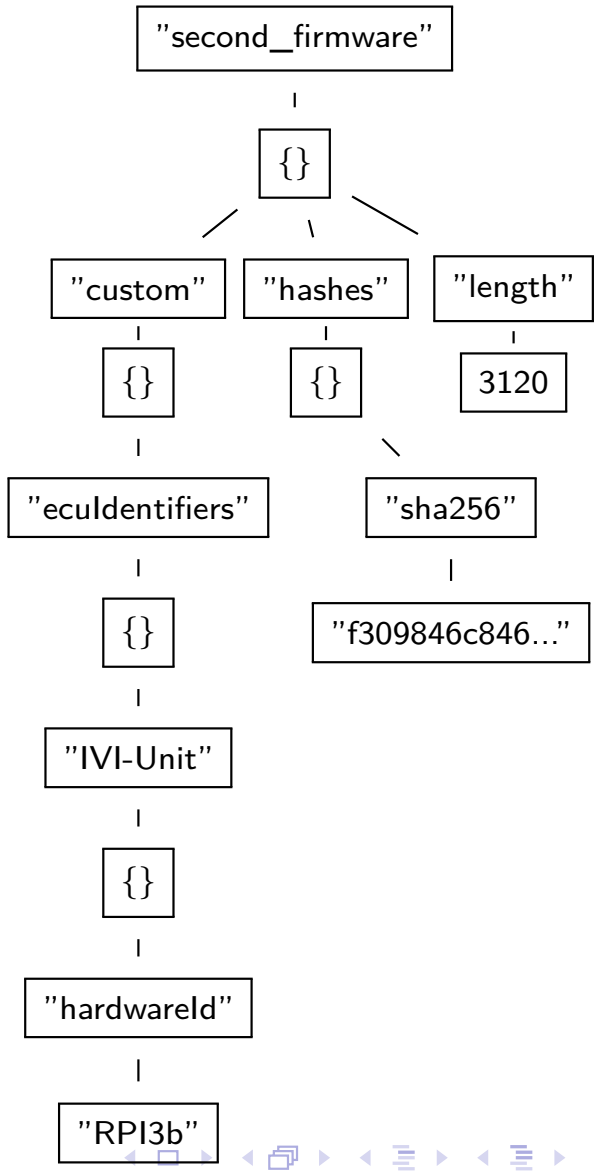
Something hacky

```
{
  "_type": "Targets",
  "expires": "3021-07-13T01:02:03Z",
  "targets": {
    "second_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "IVI-Unit": {
            "hardwareId": "RPi3b"
          }
        }
      },
      "hashes": {
        "sha256": "f309846c846..."
      },
      "length": 3120
    },
    "first_firmware.txt": {
      "custom": {
        "ecuIdentifiers": {
          "Lights-controller": {
            "hardwareId": "LEDLIGHTRD"
          }
        }
      },
      "hashes": {
        "sha256": "1bbb15aa921..."
      },
      "length": 2092
    }
  },
  "version": 2
}
```



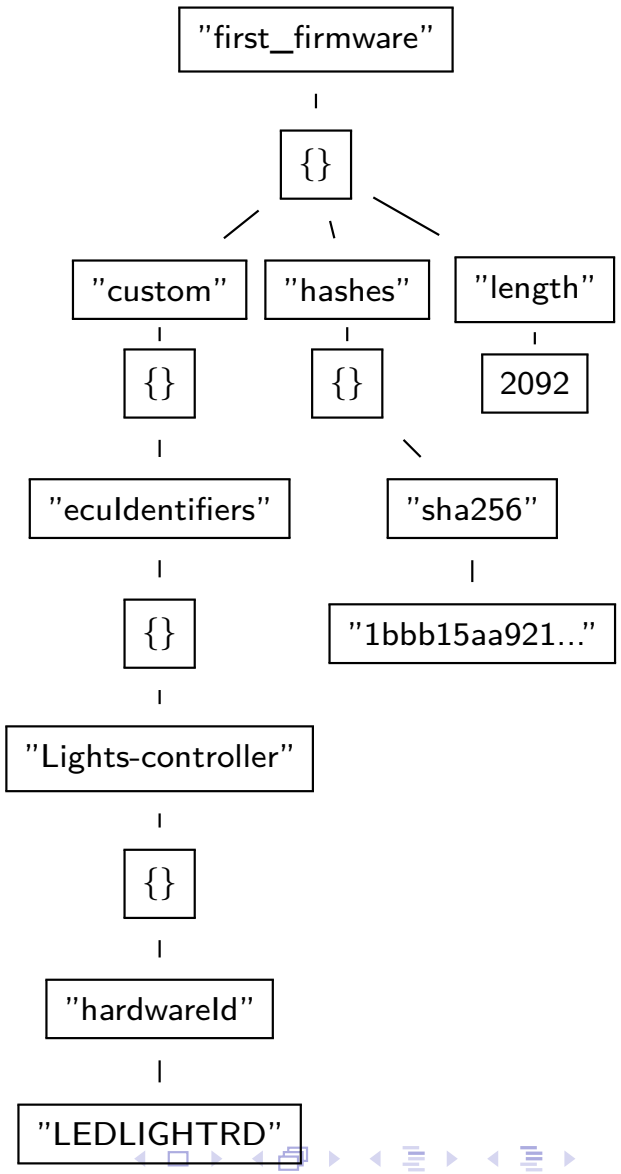
Something hacky

```
1  {
2    "_type": "Targets",
3    "expires": "3021-07-13T01:02:03Z",
4    "targets": {
5      "second_firmware.txt": {
6        "custom": {
7          "ecuIdentifiers": {
8            "IVI-Unit": {
9              "hardwareId": "RPi3b"
10           }
11         }
12       },
13       "hashes": {
14         "sha256": "f309846c846..."
15       },
16       "length": 3120
17     },
18     "first_firmware.txt": {
19       "custom": {
20         "ecuIdentifiers": {
21           "Lights-controller": {
22             "hardwareId": "LEDLIGHTRD"
23           }
24         }
25       },
26       "hashes": {
27         "sha256": "1bbb15aa921..."
28       },
29       "length": 2092
30     }
31   },
32   "version": 2
33 }
```



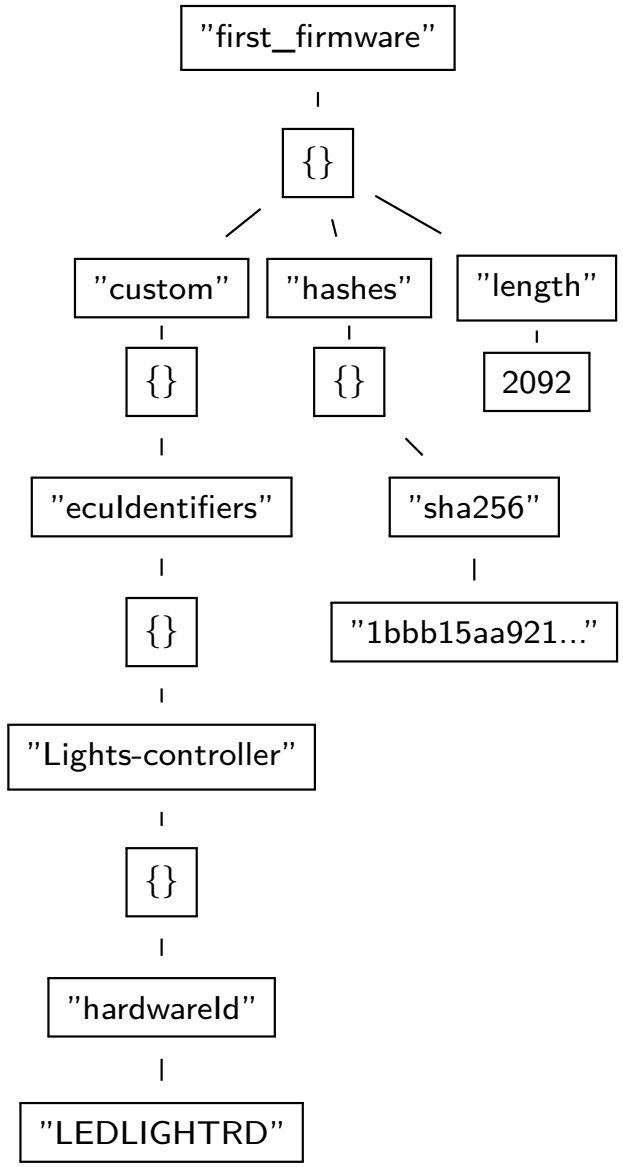
Something hacky

```
1  {
2    "_type": "Targets",
3    "expires": "3021-07-13T01:02:03Z",
4    "targets": {
5      "second_firmware.txt": {
6        "custom": {
7          "ecuIdentifiers": {
8            "IVI-Unit": {
9              "hardwareId": "RPi3b"
10           }
11         }
12       },
13       "hashes": {
14         "sha256": "f309846c846..."
15       },
16       "length": 3120
17     },
18     "first_firmware.txt": {
19       "custom": {
20         "ecuIdentifiers": {
21           "Lights-controller": {
22             "hardwareId": "LEDLIGHTRD"
23           }
24         }
25       },
26       "hashes": {
27         "sha256": "1bbb15aa921..."
28       },
29       "length": 2092
30     }
31   },
32   "version": 2
33 }
```



Something hacky

- **libuptiny** uses **jsmn** as a tokenizer and structure parser.
- JSON is fed to libuptiny in a streamed fashion. Data that was not consumed is returned to the caller to be fed again.
- jsmn is "rewound" when necessary.
- The result is still not that tiny (8Kb flash for metadata parsing and Uptane logic only), but it was the best we could achieve with JSON.



Outline

Isn't it RIOT Summit?

- libuptiny and crypto library are interfaced as RIOT modules.

```
MODULE = libuptiny
include $(RIOTBASE)/Makefile.base
```

- Libuptiny demo is a RIOT application.
- Contributions and suggestions for improvements are most welcome.
- BSP code for the demo board is on the review in RIOT repo.
- Improvements for CAN and ISO/TP are following.
- Project be integrated with RIOT's OTA as soon as it's there.

Links

- Uptane: https://uptane.github.io/
- Aktualizr: inmplementation of Uptane primary, secondaries and minimal backend: https://github.com/advancedtelematic/aktualizr
- libuptiny is a part of aktualizr, available under https://github.com/advancedtelematic/aktualizr/tree/master/partial/libuptiny
- jsmn, a minimalistic JSON tokenizer and primary parser: https://github.com/zserge/jsmn

Thanks!