REMOTE MANAGEMENT AND MONITORING OF DISTRIBUTED OSGI APPLICATIONS

TIM VERBELEN
Senior Researcher
imec – Ghent University

JAN S. RELLERMeyer
Assistant Professor
TU Delft
REMOTE MANAGEMENT AND MONITORING
FROM IOT TO CLOUD

- On-site management too expensive
- The device has no screen or input devices
- The device is unreachable
- There are too many devices
- …
OSGI – MODULARITY FOR THE JVM

• Modular Software Management
  • Can be managed individually
  • Understand versioning
  • The implementation of modularity for the JVM
    ▪ is the Bundle

• Dynamism
  • Well-defined life-cycle
  • Lose coupling
  • Complex systems are inherently dynamic
OSGI – THE PLATFORM FOR LONG-RUNNING DEVICES

- Dynamic Module System for the Java Runtime
  - Java on the Embedded Device
  - Pros:
    - bridges heterogeneity
    - can run the same code in the cloud or on the device
  - Cons:
    - updates and maintenance

- Original domain: Home Gateway
  - Multiple software packages need to co-exist on the same machine
  - Long-running software
  - Periodic maintenance and updates requires
  - Ideally with minimal downtime and no interruption of unrelated software packages
REMOTE MANAGEMENT IS NOT NEW (IN OSGI)
TR-069 CONNECTOR SERVICE SPECIFICATION

- TR-069 is a remote management protocol specified by the Broadband Forum
- Specifies a bi-directional protocol based on SOAP 1.1 over HTTP
- For managing “Customer-Premise Equipment (CPE)”
- Designed for routers, gateways, set-op boxes,…
REMOTE MANAGEMENT IS NOT NEW (IN OSGI)

DEVICE MANAGEMENT TREE (DMT)

- A logical view of manageable entities implemented by plugins and structured in a tree with named nodes
- One generic API to hide a multitude of remote management protocols
- Used in IoT space (i.e. Bosch SI)
REMOTE MANAGEMENT OVER REST

- RESTful interface
- Exposing the framework and its internal state as resources
  - Bundles
  - Services
- Easy management of OSGi deployments through REST
  - Command line tools
  - Web interfaces
  - Application-level
- REST service implementation ships with Concierge 5.1
REST SERVICE ARCHITECTURE

- framework
- framework/state
- framework/startlevel
- framework/bundles
- framework/bundles/representations
- framework/bundle/{bundleid}
- framework/bundle/{bundleid}/state
- framework/bundle/{bundleid}/startlevel
- framework/bundle/{bundleid}/header
- framework/services
- framework/services/representations
- framework/service/{serviceid}
REST SERVICE ARCHITECTURE

137.3.1 Framework Startlevel Resource

/framework/startlevel

The startlevel resource represents the active start level of the framework. It supports the GET and PUT requests.

137.3.1.1 GET

The GET request retrieves a Framework Startlevel Representation from the REST management service. The request can return the following status codes:

- 200 (OK): the request has been served successfully and the body of the response is a startlevel representation.
- 406 (NOT ACCEPTABLE): the REST management service does not support any of the requested representations.

137.3.1.2 PUT

The PUT request sets the target framework startlevel. The body of the request needs to be a Framework Startlevel Representation. The request can return the following status codes:

- 204 (NO CONTENT): the request was received and valid. The framework will asynchronously start to adjust the framework startlevel until the target startlevel has been reached.
- 415 (UNACCEPTABLE MEDIA TYPE): the request had a media type that is not supported by the REST management service.
- 400 (BAD REQUEST): the REST management service received an IllegalArgumentException when trying to adjust the framework startlevel, e.g., because the requested startlevel was zero or negative.
137.4.5 Framework Startlevel Representation

137.4.5.1 JSON

```
{
    "startLevel": 6,
    "initialBundleStartLevel": 4
}
```

137.4.5.2 XML

```
<frameworkStartLevel>
  <startLevel>6</startLevel>
  <initialBundleStartLevel>4</initialBundleStartLevel>
</frameworkStartLevel>
```
REST CLIENT

- **RestClientFactory**
  
  ```java
  public interface RestClientFactory {
    RestClient createRestClient(URI uri);
  }
  ```

- **RestClient**
  
  ```java
  public interface RestClient {
    ...
    FrameworkStartLevelDTO getFrameworkStartLevel() throws Exception;
    void setFrameworkStartLevel(FrameworkStartLevelDTO startLevel) throws Exception;
    ...
  }
  ```
var client = new OSGiRestClient('http://localhost:8080/restendpoint');
client.installBundle(bundle, {
    success : function(res) {
        // Start the bundle once the install has finished
        client.startBundle(res.id);
    },
    failure : function(httpCode, res) {
        // handle failure
    }
});
var client = new OSGiRestClient('http://localhost:8080/restendpoint');

client.installBundle(bundle, {
    success : function(res) {
        // Start the bundle once the install has finished
        client.startBundle(res.id);
    }, failure : function(httpCode, res) {
        // handle failure
    }
});
REMOTE MANAGEMENT OVER REST

<!-- RESTED -->

![Image of REST client interface](image)

**Request**

- Action: GET
- URL: localhost:8888/framework/bundle/0
- Headers:
  - Basic auth

**Response (0.014s)**

- Status: 200 OK
- Headers:
  - "symbolicName": "org.eclipse.conclierge", "location": "System Bundle", "lastModified": 0,
  - "id": 0, "state": 32, "version": "1.5.8"
REST SPEC LIMITATIONS

- Requires a REST server on the device to manage
  - Why not use the OSGi service layer? And OSGi Remote Services?
- Limited to a single OSGi framework
  - Why not manage a cluster of OSGi frameworks?
- Limited to OSGi-specific information (i.e. bundles, services, etc)
  - Why no metrics on the underlying (hardware?) platform?
CLUSTER INFORMATION SPECIFICATION
THE NEW KID IN TOWN
FRAMEWORK NODE STATUS
CLUSTER INFORMATION SPECIFICATION

- Presence services indicating a framework / node is there

- Service properties exposing node metadata
  - Unique identifier (and parent)
  - Location information (country, region, zone, …)
  - Endpoints (private and public)
  - Tags (application-specific)
FRAMEWORK NODE STATUS (2)
CLUSTER INFORMATION SPECIFICATION

- `HashMap<String, String> metrics(String... keys)`
  - Provides implementation-specific metrics about the node
  - Typically things that change over time (i.e. resource usage)

- Integrated with Remote Service Admin specification
  - `*NodeStatus` exported as remote service
  - Remote services automatically shared within cluster
FRAMEWORK MANAGER
CLUSTER INFORMATION SPECIFICATION

- Managing of an OSGi framework
- Similar API as the REST specification, but now as remote service
- List / inspect bundles and services
- Install / start / update bundles
ECLIPSE CONCIERGE

- Full OSGi core R5 compatibility
  - R6 almost done
  - R7 to come
- Keep a small footprint to work well on embedded devices
  - 250kiB without debug symbols
  - 330kiB with debug symbols
- Remain “readable”
  - Currently 9 classes
- Remain backwards-compatible
  - Java 5
CONCLUSIONS

- Concierge is an OSGi framework optimized for embedded devices and the Internet of Things.

- It hosts the reference implementations for the REST service and the Cluster Information specifications.

- Check them out to ease the remote management of your OSGi deployments.

http://eclipse.org/concierge

https://github.com/eclipse/concierge