Specification of inter-architecture communications

Deliverable 3.2 (version 1)

Lionel Touseau, Nicolas LE SOMMER

16 September 2017

Project ASAWoO

Adaptive Supervision of Avatar / Object Links for the Web of Objects

Grant Agreement: ANR-13-INFR-0012-04






Abstract

This document describes how independant infrastructures communicate. The mechanisms of discovery and invocation of remote functionalities hosted on independant ASAWoO platforms are also specified in this document.

Contents

1 Introduction

The concepts of Web of Things (WoT), ASAWoO platform, avatar, containers/managers, functionalities and WoTApps are considered prerequisites to the understanding of this document. As a reminder, ASAWoO platform architecture specification deliverable can be referred to.
On an ASAWoO platform different kind of services are exposed: the ones exposed by the platform managers, and the ones bound to avatar functionalities. By using platform services, particularly the service registry broadcast mechanism, ASAWoO platforms are able to discover each other and thus inter-architecture communications are enabled. Providing service registries have been advertised, avatars hosted on different platforms are able to communicate by using each other functionalities.
The remainder of the document is structured as follows. The next section specifies the RESTful API that exposes platform-level services. The third section describes how functionalities exposed by avatars hosted on each platform are discovered and invoked.

2 Platform services API

An ASAWoO platform publishes services redirecting to managers specific features. They are provided by each ASAWoO instance and are listed under http://<host>/registry/system. For instance, the avatar manager handles avatars life-cycle and provides its features over RESTful services, while WoTApp manager lists available applications and allows to start/stop WoTApps.
The following table specifies the URI paths for each manager.

Manager	Method	Path	Description
Neighborhood	GET	/neighborhood	Returns a list of connected ASAWoO platforms
Avatar	GET	/avatars	Lists all avatars hosted on this platform
	POST	/avatars	Creates a new avatar from an avatar configuration
	GET	/avatars/<avatarId>	Returns the description of avatar matching avatarId
	PUT	/avatars/<avatarId>	Updates the configuration of avatar matching avatarId
	DELETE	/avatars/<avatarId>	Deletes avatar matching avatarId
Device configuration	GET	/deviceconfigurations	Lists device configurations (JSONLD graph)
	GET	/deviceconfigurations/<id>	Gets a specific device configuration
	DELETE	/deviceconfigurations/<id>	Deletes a device configuration
	POST	/deviceconfigurations	Creates or updates one or many device configurations
WoT Application	GET	/wotapps	Lists available WoTApps
	GET	/wotapps/<wotappId>	Shows WoTApp configuration matching wotappId
	PATCH	/wotapps/<wotappId>	Starts/Stops a WoTApplication
	POST	/wotapps	Loads a WoTApp configuration
WoT functionality	GET	/registry	Returns platform's local and remote registries
	POST	/registry	Adds new entries into platform's remote registry
	GET	/registry/system	Lists platform "system" functionalities
	GET	/registry/local	Returns platform's local registry
	GET	/registry/remote	Returns platform's remote registry
	GET	/registry/dtn	Returns platform's registry substituting DTN URIs

3 Functionalities annoucements, discovery and invocation

The /registry services described above allow to query a functionality registry or to publish one's registry from/to a remote ASAWoO platform. Besides enabling these features, ASAWoO WoT functionality module should also implement an automatic publication/discovery mechanism. This module broadcasts its local registry to neighbor ASAWoO platforms thanks to the multicast communication layer implemented by the DTN communication layer. Meanwhile it also listens to multicasted functionality registries from neighbor platforms, and adds their registry to its remote functionalities registry.

This way, from a platform's perspective, avatars hosted on remote platforms are listed with their functionalities in the remote registry. Remote services (i.e., functionalities) can therefore be invoked using functionality URIs stated in the remote registry.

Moreover, remote functionalities can also be invoked through the DTN communication module.