CAMP Resource Model Issues

Version: 2011.11.05

# Abstract

Section 2.1.1 of the CAMP 1.0 specification[[1]](#endnote-1) describes the resources that are used to manage an application and its dependencies on the services provided by the platform and, possibly, other applications. There are a number of high-level, architectural issues with this model. The purpose of this write-up is to explore these issues and outline some of the key decision points in their resolution.

# Issues and Questions

## Requirement Resolution: Modeling, Names and Values

The CAMP 1.0 specification defines the process of “requirement resolution” as the matching of requirements (encoded as Requirement resources) against capabilities (encoded as Capability resources). Although this process sounds reasonable in the abstract, in practice there are a number of practical issues. To examine these issues, consider the following scenario:

A POJO application models its requirement on the JRE as a CAMP Platform Requirement. This requirement states, effectively, that the application requires Java version 1.6.0\_30 or greater.

If the requirement resolution process is to be of help in deploying our example application, the target platforms for this application must agree to model their available JREs as Platform Component Capabilities. However, nothing in CAMP 1.0 defines this agreement. Furthermore, it isn’t clear that modeling JREs in this fashion is appropriate for all platforms; a platform might support a single version of the JRE across all applications and simply document this on some help page.

Simply agreeing to expose your JRE as a Platform Component Capability, however, is not enough. In order to automatically match requirements with capabilities there must be some agreement on the ‘name’ or ‘type’ of both the Platform Component Capability and the Platform Component Requirement (e.g. “JRE”). Without this agreement there is no way for the CAMP implementation to know which Capabilities are potential matches for which Requirements. Next there must be agreement on the names of the properties to be matched, their value space, and their comparison rules. For example, it must be agreed to that the JRE version is communicated via a property called “version” (and not “Version” or “jreVesion”) and that the value of this property is a string and that the string “1.6.0\_31-b05” represents a version that is greater than “1.6.0\_30”.

Secondly there is the issue of optionality. If CAMP were to require that all platforms that support the use of Java must expose a “JRE” capability with, at minimum, a “version” attribute in the correct format, this would increase the ability to port applications from one platform to the other, but decrease the ability to implement CAMP for those platforms that are unable or unwilling to model their platform in this way. Conversely, defining the requirement to expose a “JRE” capability as optional within CAMP increases the ability to implement CAMP on different platforms, but decreases the ability to port applications.

Keep in mind that the above scenario only supported matching against a simple service using a single attribute value. More complicated scenarios involving multiple services that needed to be matched across a number of attribute values would be considerably more difficult.

**Q1: Should CAMP 1.x include details on the capabilities and requirements for specific platform components?**

**Q2: If not, should the CAMP TC undertake additional “sub-specifications” that define the capabilities and requirements of platform components?**

1. The “requirement resolution” process is only relevant for scenarios in which the Application Administrator attempts to deploy a PDP on an “unfamiliar” platform. An “unfamiliar” platform is a platform that the Application Developer has not targeted or otherwise directly interacted with. If the Application Developer is working directly with a specific platform, the metadata and artifacts in the PDP can directly reference the URIs of any required Component Templates.

**Q3: Should CAMP omit the notion of “requirement resolution”, at least for the next version of the specification?**

## What is an Application Component?

It is not clear what Application Components and, by association, Application Component Templates are supposed to represent. While it is true that an application can be composed of many modules (code, images, libraries, datasets, etc.), it is not clear that these modules need to be represented as separate resources within CAMP. In general, CAMP exposes unique resources to represent artifacts and interfaces that either have a unique lifecycle or are managed by different actors. Neither of these criteria seem to apply to Application Components.

**Q4: Is the lifecycle of Application Components/Application Component Templates separate from the Assembly/Assembly Template that references them?**

1. Currently the specification implies that the Application Components that are referenced by an Assembly are instantiated when the Assembly is instantiated and deleted when the Assembly is deleted.
2. Similarly, the specification implies that the Application Component Templates that are referenced by an Assembly Template are created when the Assembly Template is registered and deleted when the Assembly Template is deleted.

## Configuration Injection

Although CAMP 1.0 describes how to “configure” an application by linking the Assembly Template to various Component Templates, it doesn’t say anything about the results of such linking effect the configuration of the application itself. Although CAMP is “language, framework, and platform neutral” it seems that, to support portability across various CAMP implementations, the specification may need to say something about the relationship between CAMP-level links and application-level configuration values. For example, a configuration file of an application may contain the URL of the database that the application should connect to. As provided in the PDP, this configuration value may be specified as:

dbURL=${CAMP.MYAPP.DBURL}

Although the name, location, format etc. of the configuration file are clearly outside the scope of CAMP, it may aid portability if the name of the annotation (i.e. the stuff between the curly braces above) were related to CAMP resource links in some, normative fashion.

**Q4: What, if anything, should CAMP say about the relationship between the links defined in the CAMP spec (e.g a link from an Assembly Template to a Platform Component Template) and the effects on the configuration of the underlying framework?**

1. <http://cloudspecs.org/CAMP/CAMP_v1-0.pdf> [↑](#endnote-ref-1)