CAMP Lifecycle Issues

Version: 2011.10.30

1 Abstract

Section 3 of the CAMP 1.0 specification[i] describes the lifecycle of an application. There are a number of issues with this lifecycle and its description. The purpose of this write-up is to explore these issues and outline some of the key decision points in their resolution.

2 Issues and Questions

2.1 State Representation

Although section 3 provides the names of various lifecycle states, nowhere in the CAMP 1.0 model are these names surfaced (e.g. as the value of a resource property).

Q1: Should CAMP surface the names of the application states?

1. Although applications in the "Uploaded" state are not represented by a CAMP resource, applications in the "Deployed" state are described and managed by an Assembly Template resource and applications in the "Instantiated" state are described and managed by an Assembly resource (though the Assembly Template used to instantiate the application continues to exist).
2. The CAMP 1.0 specification implies, but is not definitive, that applications in the "Suspended" state are described and managed by Assembly resources.
3. If applications in the "Suspended" state are described and managed by Assembly resources, then the representation of the Assembly resource **must** provide some information that allows the Application Administrator to determine which state the application is in (i.e. the name of the state).
4. If applications in the "Suspended" state were described and managed by a hypothetical, new "SuspendedAssembly" resource, there would be no need to communicate the state name as it would be implicit in the existence of such a resource.
5. A strategy of representing application state via separate resource types has implications on extensibility. Rather than having to define a new string, providers that wish to extend the CAMP lifecycle with additional states would need to define a new resource type (and extend the Platform type to include an array of links to all the instances of that type, etc.)

2.2 Abstract States

In the description of the lifecycle in section 3 it is noted that:

> The states and transitions shown in this diagram are abstract and do not necessarily correspond in a one-to-one fashion with any specific information or activities maintained or performed by either the Consumer or the Provider.

3 The meaning of this note is unclear. Taken literally, it implies that all of section 3 is a non-normative example. A Provider could implement an arbitrary set of states or implement the CAMP 1.0 defined states but with arbitrary names and, because there is no "one-to-one correspondence with specific information" claim to conform to CAMP. If the state of an application is communicated and controlled by the state name values (see "Issues and Questions

State Representation" above), there are obvious interoperability issues with this concept.

Q2: Should CAMP define a normative set of applications states and their names?

1. The normative states do not necessarily have to be those in the CAMP 1.0 specification. The TC is free to remove states, add states, change names, etc.
2. Providers could be allowed to omit states not implemented by their platform.
3. As currently specified by the third bullet-point note (lines 469-470), providers should be able to extend the model with additional states.
4. Providers may not combine (2) and (3), above, to create new states with the same semantics as states defined by CAMP but with different names.

3.1 Automatic State Transitions

The state diagram in Figure 8 shows the pathway to instantiating an application as two, separate steps: the "register->Deployed" step (triggered by a POST to the Platform resource), and the "instantiate->Instantiated" step (triggered by a POST to the Assembly Template). However, most existing PaaS systems instantiate applications on deployment. One could model such a system in CAMPs lifecycle if we allowed providers to automatically invoke the "instantiate" transition on deployment. If we consider that this scenario may not be the only one in which a provider's state transitions may not match those of CAMP, we arrive at the following question:

Q3: Should CAMP allow provider's to automatically execute state transitions to match their underlying implementation?

1. If we did allow provider's to automatically execute state transitions, we need to define a way for the provider to communicate what happened to the consumer. In the scenario above, for example, the provider could indicate that the "instantiate" transition was automatically triggered by including link to the resulting Assembly (rather than an Assembly Template) in the `Location` header. However, unless the provider also includes a representation of the Assembly resource in the response message body, it will be difficult for the consumer to figure out what whether they are being given a reference to an Assembly (meaning that the "instantiate" transition was automatically triggered) or an Assembly Template (meaning that it was not). This also has implications with respect to the discussion on media types.

3.2 Extended State Transitions

The state diagram in Figure 8 has no transition from the "Instantiated" state to the "deleted" end state but many providers support this transition in their existing implementations. One way to address this would be to allow providers to extend CAMPs lifecycle with additional transitions.

Q4: Should CAMP allow provider's to extend the defined lifecycle with additional transitions between existing, CAMP-defined, states?

1. This question does not apply to the case where a provider has extended the CAMP lifecycle with additional states. In such a situation it is obvious that the provider will have to also have to define extension transitions into and out of the extended states.
2. Any extensions to the existing state transitions must define a triggering mechanism that does not conflict or overload the mechanisms for existing state transitions. For example, the "delete instance" transition is triggered by performing an HTTP DELETE on the Assembly resource. If a provider wants to add a transition from the "Instantiated" state to the final state, they will have to define some other mechanism for triggering this transition.

3.3 Multiple Assembly Instances

Section 6.10 of the CAMP specification defines how to create an Assembly from an Assembly Template. What the specification does not go into is "what happens when you attempt to create multiple Assembly instances from the same Assembly Template?"

Q5: Should CAMP allow the creation of multiple Assembly instances from the same Assembly Template?

If the answer to the above question is "yes", that brings up another question:

Q6: Do the Assembly instances created from a single template represent unique applications or replicas of "the same application"?

1. Replicas share state (whether in memory or via a database), unique applications have separate state.
2. From the point of view of an application user, replicas are indistinguishable but unique applications are distinguishable (whether via a different URL or different appearance, etc.).

3.4 Optional States

CAMP 1.0 defines the "Suspended" as optional but says nothing about how the provider can advertise that it does or doesn't support this state nor what the provider's response should be to a request to transition to a state that the provider does not support.

---

[i] http://cloudspecs.org/CAMP/CAMP_v1-0.pdf