Cloud Application Management for Platforms Version 1.1

Committee Specification Draft 03 / Public Review Draft 01

31 July 2013

Specification URIs

This version:

http://docs.oasis-open.org/camp/camp-spec/v1.1/csprd01/camp-spec-v1.1-csprd01.pdf (Authoritative)
http://docs.oasis-open.org/camp/camp-spec/v1.1/csprd01/camp-spec-v1.1-csprd01.html
http://docs.oasis-open.org/camp/camp-spec/v1.1/csprd01/camp-spec-v1.1-csprd01.doc

Previous version:

http://docs.oasis-open.org/camp/camp-spec/v1.1/csd02/camp-spec-v1.1-csd02.pdf Authoritative)
http://docs.oasis-open.org/camp/camp-spec/v1.1/csd02/camp-spec-v1.1-csd02.html
http://docs.oasis-open.org/camp/camp-spec/v1.1/csd02/camp-spec-v1.1-csd02.doc

Latest version:

http://docs.oasis-open.org/camp/camp-spec/v1.1/camp-spec-v1.1.pdf (Authoritative)
http://docs.oasis-open.org/camp/camp-spec/v1.1/camp-spec-v1.1.html
http://docs.oasis-open.org/camp/camp-spec/v1.1/camp-spec-v1.1.doc

Technical Committee:

OASIS Cloud Application Management for Platforms (CAMP) TC

Chair:

Martin Chapman (martin.chapman@oracle.com), Oracle

Editors:

Jacques Durand (jdurand@us.fujitsu.com), Fujitsu Limited
Adrian Otto (adrian.otto@rackspace.com), Rackspace Hosting, Inc.
Gilbert Pilz (gilbert.pilz@oracle.com), Oracle
Tom Rutt (trutt@us.fujitsu.com), Fujitsu Limited

Abstract:

This document defines the artifacts and APIs that need to be offered by a Platform as a Service (PaaS) cloud to manage the building, running, administration, monitoring and patching of applications in the cloud. Its purpose is to enable interoperability among self-service interfaces to PaaS clouds by defining artifacts and formats that can be used with any conforming cloud and enable independent vendors to create tools and services that interact with any conforming cloud using the defined interfaces. Cloud vendors can use these interfaces to develop new PaaS offerings that will interact with independently developed tools and components.

Status:

This document was last revised or approved by the OASIS Cloud Application Management for Platforms (CAMP) TC on the above date. The level of approval is also listed above. Check the “Latest version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/camp/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/camp/ipr.php).

Citation format:

When referencing this specification the following citation format should be used:

[CAMP-v1.1]

Cloud Application Management for Platforms Version 1.1. 31 July 2013. OASIS Committee Specification Draft 03 / Public Review Draft 01. http://docs.oasis-open.org/camp/camp-spec/v1.1/csprd01/camp-spec-v1.1-csprd01.html.

Notices

Table of Contents

1 Introduction

1.1 Overview

Platform as a Service (PaaS) is a term that refers to a type of cloud computing in which the service provider offers customers/consumers access to one or more instances of a running application computing platform or application service stack. NIST defines PaaS [SP800-145] as a "service model" with the following characteristics:

> The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations.

There are multiple commercial PaaS offerings in existence using languages such as Java, Python and Ruby and frameworks such as Spring and Rails. Although these offerings differ in such aspects as programming languages, application frameworks, etc., there are inherent similarities in the way they manage the lifecycle of the applications that are targeted for, and deployed upon them. *The core proposition of this specification is that these similarities can be leveraged to produce a generic application and platform management API that is language, framework, and platform neutral.*

For PaaS consumers this management API would have the following benefits:

- "Portability between clouds" is emerging as one of the primary concerns of cloud computing. By standardizing the management API for the use cases around deploying, stopping, starting, and updating applications, this specification increases consumers' ability to port their applications between PaaS offerings.
- It is likely that implementations of this specification will appear as plugins for application development environments (ADEs) and application management systems. Past experience has shown that, over time, such generic implementations are likely to receive more attention and be of higher quality than the implementations written for solitary, proprietary application management interfaces.

For PaaS providers this management API would have the following benefits:

- Because the strength and features of a PaaS offering's application management API are unlikely to be perceived as key differentiators from other PaaS offerings, the existence of a consensus management API allows providers to leverage the experience and insight of the specification's contributors and invest their design resources in other, more valuable areas.
- By increasing the portability of applications between PaaS offerings, this management API helps "grow the pie" of the PaaS marketplace by addressing one of the key pain points for PaaS consumers.

1.2 Purpose

This document defines the artifacts and APIs that need to be offered by a Platform as a Service (PaaS) cloud to manage the building, running, administration, monitoring and patching of applications in the cloud. Its purpose is to enable interoperability among self-service interfaces to PaaS clouds by defining artifacts and formats that can be used with any conforming cloud and enable independent vendors to create tools and services that interact with any conforming cloud using the defined interfaces. Cloud vendors can use these interfaces to develop new PaaS offerings that will interact with independently developed tools and components.

The following is a non-exhaustive list of the use cases which are supported by this specification.

- Building and packaging an application in a local Application Development Environment (ADE)
- Building an application in an ADE running in the cloud

- Importing a Platform Deployment Package into the cloud
- Uploading application artifacts into the cloud
- Run, stop, suspend, snapshot, and patch an application

1.3 Example (non-normative)

This example illustrates a scenario in which the application administrator wants to run and monitor an application. It assumes that the application package was previously made available to the platform, either because it was uploaded to the platform or developed directly on the platform.

The administrator starts by deploying the application package to the platform. This is done by sending an HTTP POST request to the platform entry point URL as shown below, where "/myPaaS" is the entry point and "/myPaas/pkgs/1" is the location of the application package.

```
POST /myPaaS HTTP/1.1
Host: example.org
Content-Type: application/json
Content-Length: ...

{
  "pdpUri": "/myPaaS/pkgs/1"
}
```

On receiving such a request the platform deploys the application package and creates a new resource "/myPaas/templates/1" that represents the deployed application. The response from the platform is show below.

```
HTTP/1.1 201 Created
Location: http://example.org/myPaaS/templates/1
Content-Type: ...
Content-Length: ...

...
```

Once the application is deployed, the administrator starts the application by sending an HTTP POST request to the resource that represents the deployed application, which was obtained in the previous step ("/myPaaS/templates/1").

```
POST /myPaaS/templates/1 HTTP/1.1
Host: example.org
```

On successful start the platform creates a new resource representing the running application and provides the URL of that resource "/myPaaS/apps/1" in the response as shown below.

```
HTTP/1.1 201 Created
Location: http://example.org/myPaaS/apps/1
Content-Type: ...
Content-Length: ...

...
```

The administrator can now monitor the running application by sending an HTTP GET request to the resource that represents the running application, which was obtained in the previous step ("/myPaas/apps/1").

```
GET /myPaaS/apps/1 HTTP/1.1
Host: example.org
```

The response contains the JSON representation of the running application as shown below.


```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: ...

{
  "uri": "http://example.org/myPaaS/apps/1",
  "name": "Hello Cloud App",
  "type": "assembly",
  "description": "Hello Cloud Application Running in a PaaS Env",
  "applicationComponents": [
    {
      "href": "/myPaaS/apps/1/acs/1", "targetName": "appComp1"
    },
    {
      "href": "myPaaS/apps/1/acs/2", "targetName:: "appComp2"
    }
  ],
  "platformComponents": [
    {
      "href": "/myPaaS/pcs/1", "targetName": "dbPlatComp"
    },
    {
      "href": "myPaaS/pcs/2", "targetName": "msgBusPlatComp"
    }
  ],
  "assemblyTemplate": "/myPaaS/templates/1",
}
```

1.4 Non-Goals

The specification of functional interfaces specific to services provided by individual components (see Application Components and Platform Components, below) is out of scope for this document. This is because such interfaces may be quite diverse and differ significantly from platform to platform.

1.5 PaaS Roles

There are many roles that can be defined for a PaaS environment. For the purposes of this specification we identify the following roles:

Application Developer: The person that builds and tests an application and presents the developed artifacts for deployment.

Application Administrator: The person that deploys applications and manages the application throughout its life-cycle.

Together these two roles make up the consumers of the management API described in this specification. This specification is intended mainly for Application Administrators, though it does constraint the artifacts that an Application Developer presents for deployment.

Platform Administrator: The person that manages the platform. This specification describes some of the functions of a Platform Administrator, though most of the functions of this role are outside its scope.

Application End-User: A user of an application deployed on the platform. The interactions of the Application end-user and the application are outside the scope of this specification.

Extension Developer: The person who creates new Extensions for Platforms.

1.6 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 [RFC2119].

Upper case versions of the RFC 2119 keywords are used in this document for emphasis. All text except examples, unless otherwise labeled, is normative. All examples are non-normative.

Three conformance targets are defined in this specification:

1. A CAMP Provider, or "Provider" for short, is an implementation of the server aspects of this specification.
2. A CAMP Consumer, or "Consumer" for short, is an implementation of the client aspects of this specification.
3. Platform Deployment Package (PDP) defines the format of a deployment plan of an application and associated artifacts and meta-data.

See Section 8, "Conformance", for details on Conformance to this specification.

1.7 Specification Version

Each version of a CAMP specification is identified by a unique string termed the "Specification Version String". The Specification Version String for this specification is "CAMP 1.1".

1.7.1 Backwards Compatibility

This version of the CAMP specification is not backwards compatible with any previous version of the CAMP specification.

1.8 Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.

[RFC2616] R. Fielding et al, "Hypertext Transfer Protcol – HTTP/1.1", IETF RFC 2616, June 1999. http://www.w3.org/Protocols/rfc2616/rfc2616.txt

[RFC2617] J. Franks et al, "HTTP Authentication: Basic and Digest Access Authentication", IETF RFC 2617, June 1999. http://www.ietf.org/rfc/rfc2617.txt

[RFC 3986] T.Berners-Lee et al, "Uniform Resource Identifiers (URI): Generic Syntax", IETF RFC 3986, August 1998. http://www.ietf.org/rfc/rfc3986.txt

[RFC4346] T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", IETF RFC 4346, April 2006. http://www.ietf.org/rfc/rfc4346.txt

[RFC4627] D. Crockford, "The application/json Media Type for JavaScript Object Notation (JSON)", IETF RFC 4627, July 2006. http://www.ietf.org/rfc/rfc4627.txt

[RFC5246] T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", IETF RFC 5246, August 2008. http://www.ietf.org/rfc/rfc5246.txt

[RFC5789] L. Dusseault, Linden Lab, J. Snell, "PATCH Method for HTTP", IETF RFC 5789, March 2010. http://www.ietf.org/rfc/rfc5789.txt

[JSON Patch] P. Bryan, M. Nottingham, "JSON Patch", IETF Draft, January 2013. http://tools.ietf.org/html/draft-ietf-appsawg-json-patch-10

[ISO 8601:2004] International Organization for Standardization, Geneva, Switzerland, "Data elements and interchange formats -- Information interchange - - Representation of dates and times", March 2008. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=40874

[YAML 1.1] Oren Ben-Kiki, Clark Evans, Brian Ingerson, "YAML Ain't Markup Language (YAML) Version 1.1", January 2005. http://yaml.org/spec/1.1/

[OVF] Distributed Management Task Force, "Open Virtualization Format Specification", DSP0243, 13 December 2012. http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_2.0.0.pdf

[TAR] "IEEE Std 1003.1-2001, IEEE Standard for Information Technology - Portable Operating System Interface (POSIX)"

[GZIP] RFC-1952 "GZIP file format specification version 4.3," http://tools.ietf.org/html/rfc1952

[ZIP] ZIP File Format Specification, http://www.pkware.com/documents/APPNOTE/APPNOTE-6.3.0.TXT

[SHA256] FIPS PUB 180-4, Federal Information Processing Standards Publication, “Secure Hash Standard (SHS),” http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

1.9 Non-Normative References

[SP800-145] Peter Mell, Timothy Grance, “The NIST Definition of Cloud Computing”, Special Publication 800-145, September 2011. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[Git] The Software Freedom Conservancy, “Git, the fast version control system”, March 2012. http://git-scm.com/

2 Concepts and Types

This document specifies the self-service management API that a Platform as a Service offering presents to the consumer of the platform. The API is typically the interface into a platform implementation layer that controls the deployment of applications and their use of the platform.

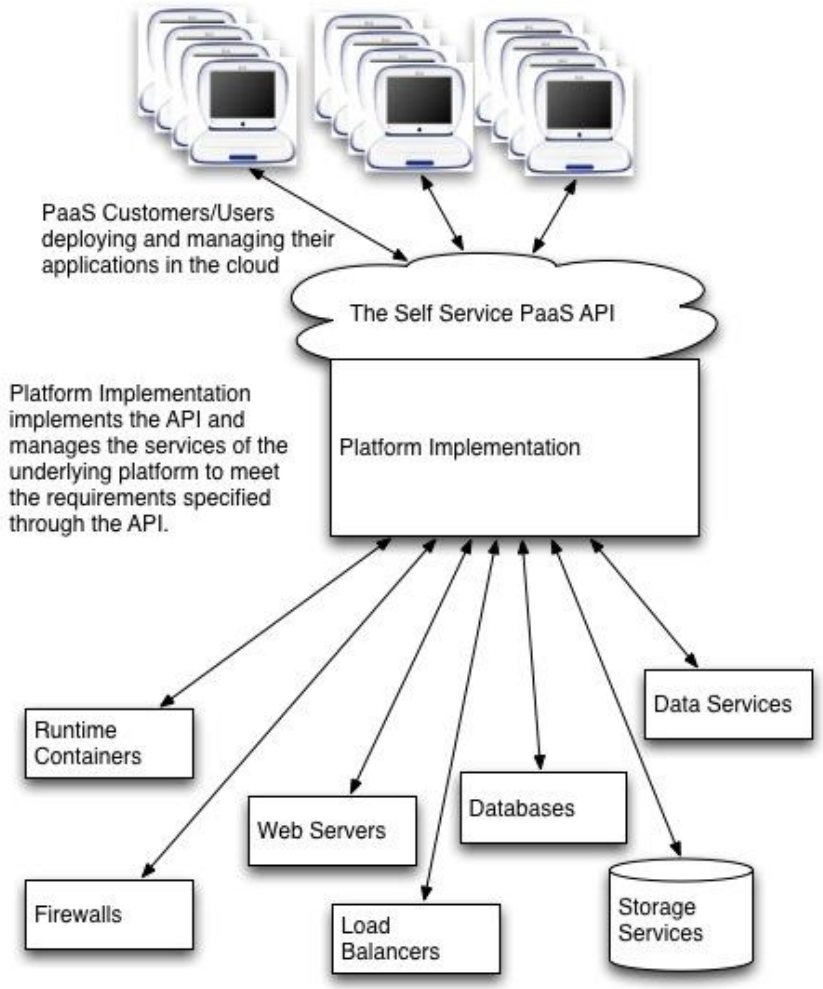

Figure 2-1: Typical PaaS Architecture

The figure above shows a typical architecture of a Platform as a Service cloud. The platform implementation is a management client of the underlying resources that transforms (through policies) the application requirements expressed by the Application Administrator into provisioning and other operations on those resources. The Platform Administrator manages the underlying hardware, storage, networks, and software services that make up the platform through existing administrative interfaces. Thus the Application Administrator is able to concentrate on their application and its deployment environment rather than having to be a systems administrator, database administrator and middleware administrator as well (as is the case with IaaS).

The goal of the management interface is to provide the PaaS consumer with a model that is as simple as possible, and yet still provides the essential elements that give them control over the deployment, execution, administration and metering of their application and its deployment environment.

2.1 Application Assemblies

The model of resources manipulated through the interface, in combination with the protocol used to remotely accomplish this, constitutes the self-service PaaS management API. The model contains resource types corresponding to the artifacts discussed earlier.

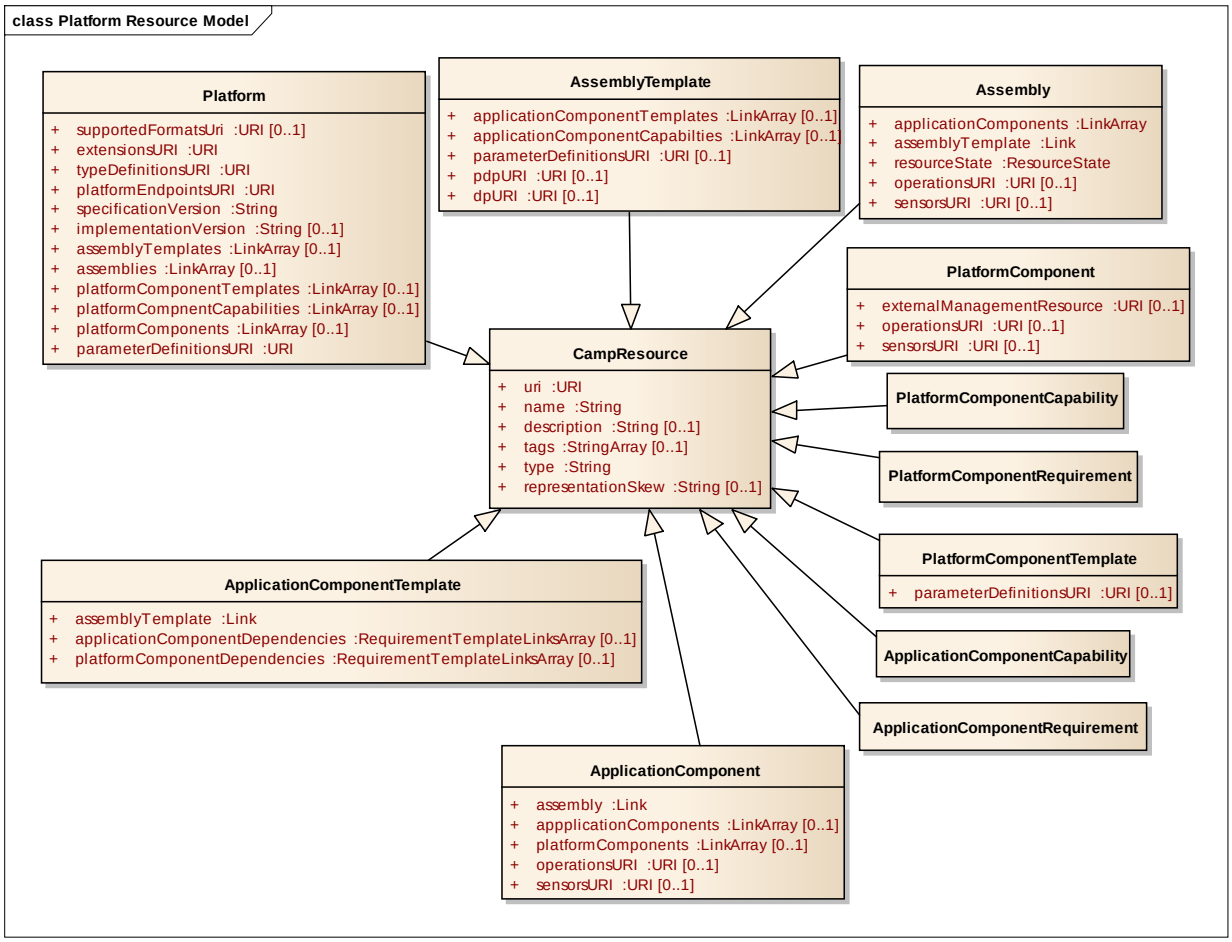

Figure 2-2: CAMP Resources as UML Classes

Figure 2-2 is a UML Class Diagram showing the CAMP resources as UML classes. All CAMP resources share a set of common attributes shown here as a CampResource parent class.

Each attribute shown in these UML class diagrams has a CAMP common type. The '+' symbol before each attribute name in the boxes indicates that the attribute access is public (i.e. available thru the API). Non-mandatory resource attributes are indicated using the [0..1] UML multiplicity tag.

2.1.1 Platform

The Platform resource is the primary view of the platform and what is running on it. The Platform references templates for creating applications (as Assembly Templates) as well as running applications (as Assemblies) and enables discovery of the PaaS offering in terms of Platform Components and their capabilities. The Platform also determines the scope of access for sharing amongst multiple applications.

2.1.2 Assemblies

Assembly resources represent running applications. Operations on an Assembly resource affect the components and elements of that application. Assembly Template resources are templates for the creation of Assemblies.

2.1.3 Components

There are two kinds of components, application components and platform components, each of which can exist in either template or instantiated forms.

The Application Component Template resource represents a discrete configuration of a part of an Assembly Template supplied by the Application Administrator. The attributes of this resource represent the configuration values that will be applied to the component upon instantiation.

The Application Component resource represents an instantiated instance of an Application Component Template, such as a deployed WAR file.

The Platform Component Template resource represents a discrete configuration of a part of an Assembly Template supplied by the Platform. The attributes of this resource represent the configuration values that will be applied to the component upon instantiation.

The Platform Component resource represents an instantiated instance of a Platform Component Template in use by an application.

2.1.4 Capabilities and Requirements

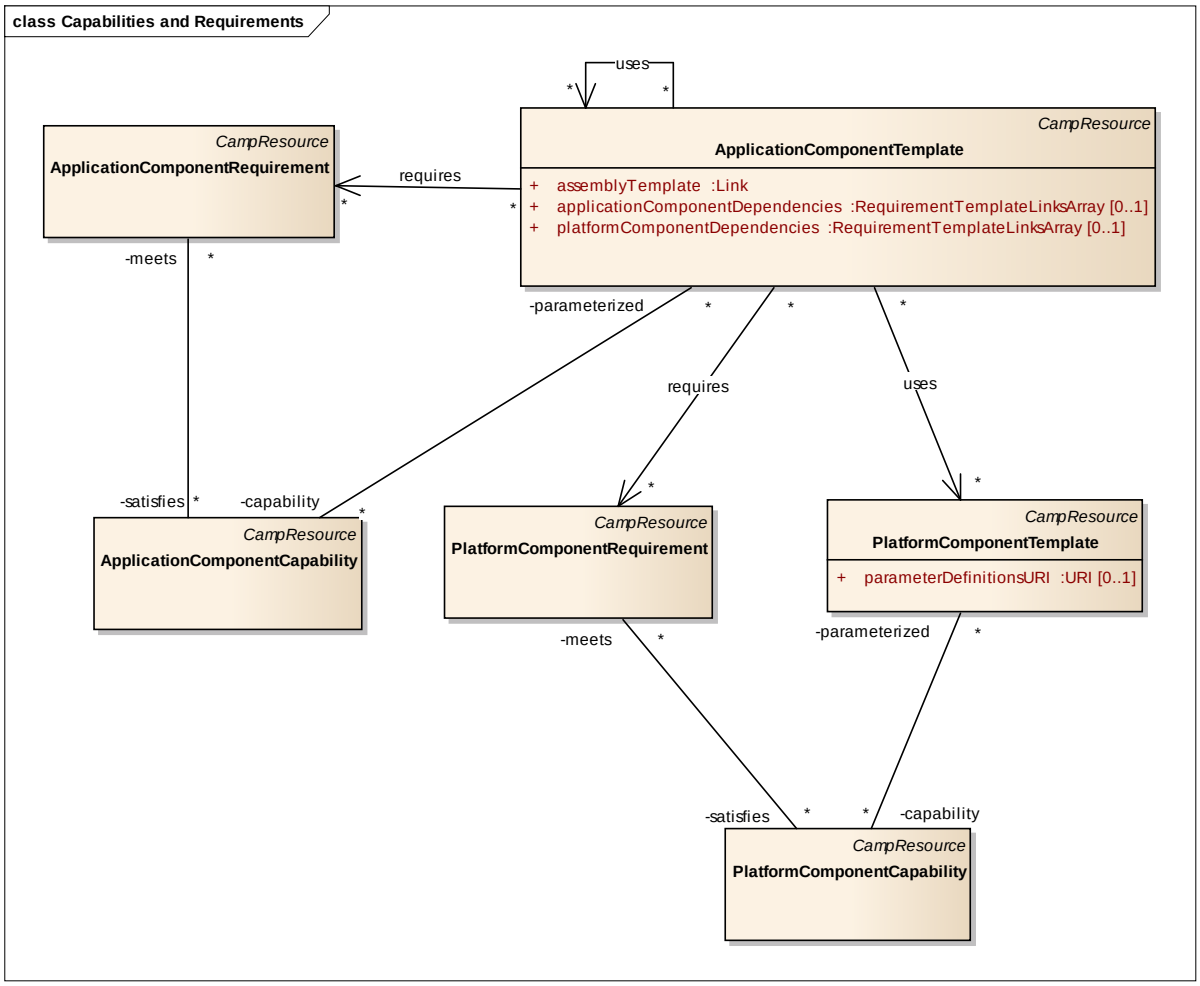

Figure 2-3: Capabilities, Requirements, and Template Relationships

Figure 2-3 shows the relationships between capabilities, requirements, and template resources. Associations which are visible thru pointer attributes in resources (i.e. URI, Link, or LinksArray attribute types) are show using UML named associations with navigation arrows. Associations which model implementation specific relationships, not visible thru the API, are represented using the UML association end notation, without navigation arrows. The association end names were chosen to represent the role that resource plays in the relationship. The '–' symbol before the association end names expresses that they have private access (i.e. navigation across resource links is not available thru the API).

Like Templates, Capability resources represent the configuration of instantiable Components (Application Components or Platform Components). Unlike Templates, which delineate discrete configurations, Capabilities specify ranges of configuration values.

Requirement resources are created by the Application Developer or Application Administrator to express an application's dependency on a component that is capable of satisfying a certain set of requirements. For example, an application component might depend upon a messaging service that supports a certain version of an AMQP API, can accept messages of up to 2MB in size, and which provides a persistent message store.

The process of matching Requirements with Capabilities is referred to as "requirement resolution".

2.1.5 Operations and Sensors

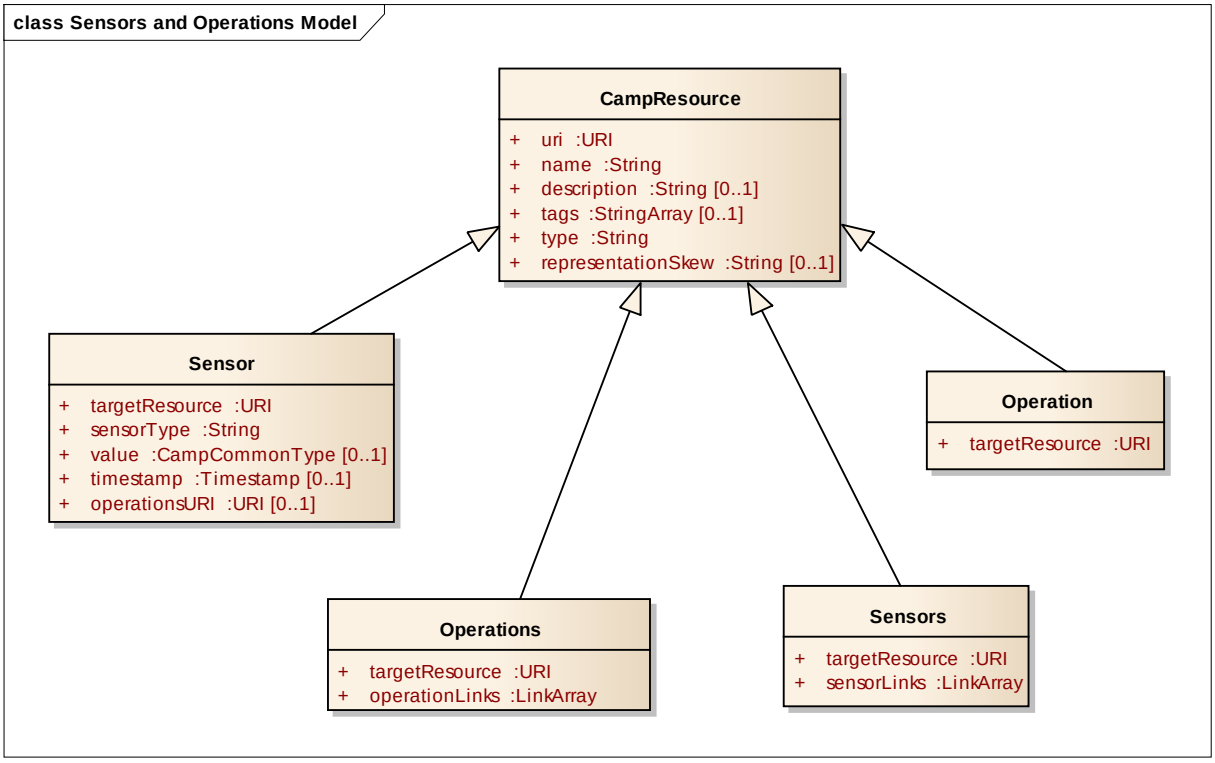

Figure 2-4: Operations and Sensors

Figure 2-4 is a UML class diagram showing the attributes of the Operations and Sensors resources.

Operations and Sensors provide a way of interacting with an application through the CAMP API. Operation resources represent actions that can be taken on a resource, and Sensor resources represent dynamic data about resources, such as metrics or state. Sensor resources are useful for exposing data that changes rapidly, or that might need to be fetched from a secondary system. Sensor Resources can also offer Operations to allow resetting metrics, or adjusting frequency collection settings.

Operation and Sensor resources are exposed on Assembly resources, Application Component resources and Platform Component resources. Operations are also known as effectors. The combination of sensors and operations enables ongoing management. This can include automation techniques such as using policies, event-condition-action paradigms, or autonomic control. A Consumer can use the REST API to

perform such management. A Provider can also use them. For example, a Platform Component could be offered that allows for “autoscaling” capacity based on the volume of work an application performs.

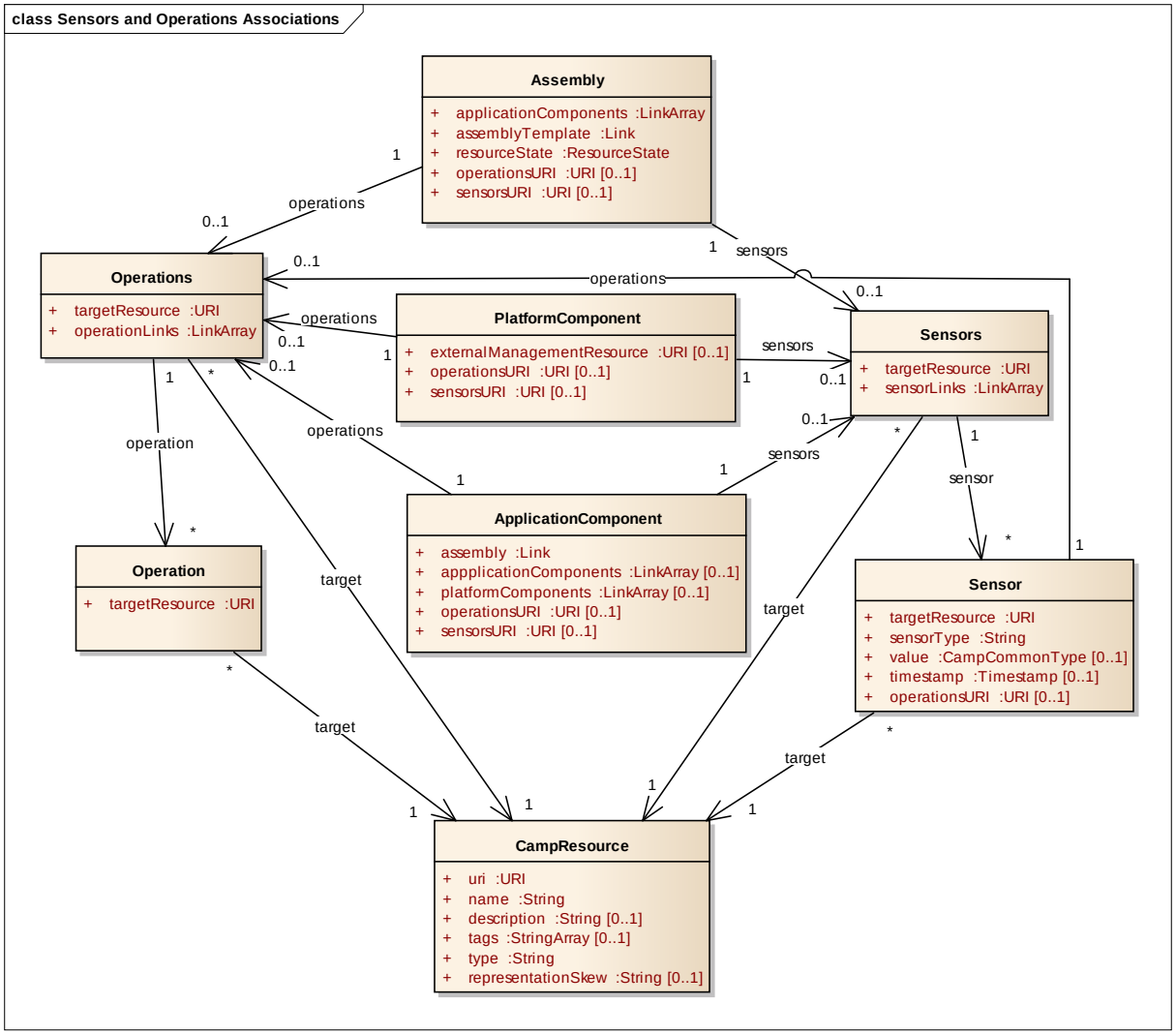

Figure 2-5: Operations and Sensors Associations

Figure 2-5 is a UML class diagram showing Operations and Sensors resources, and the CAMP resources that they are associated with.

2.1.6 Resource Relationships

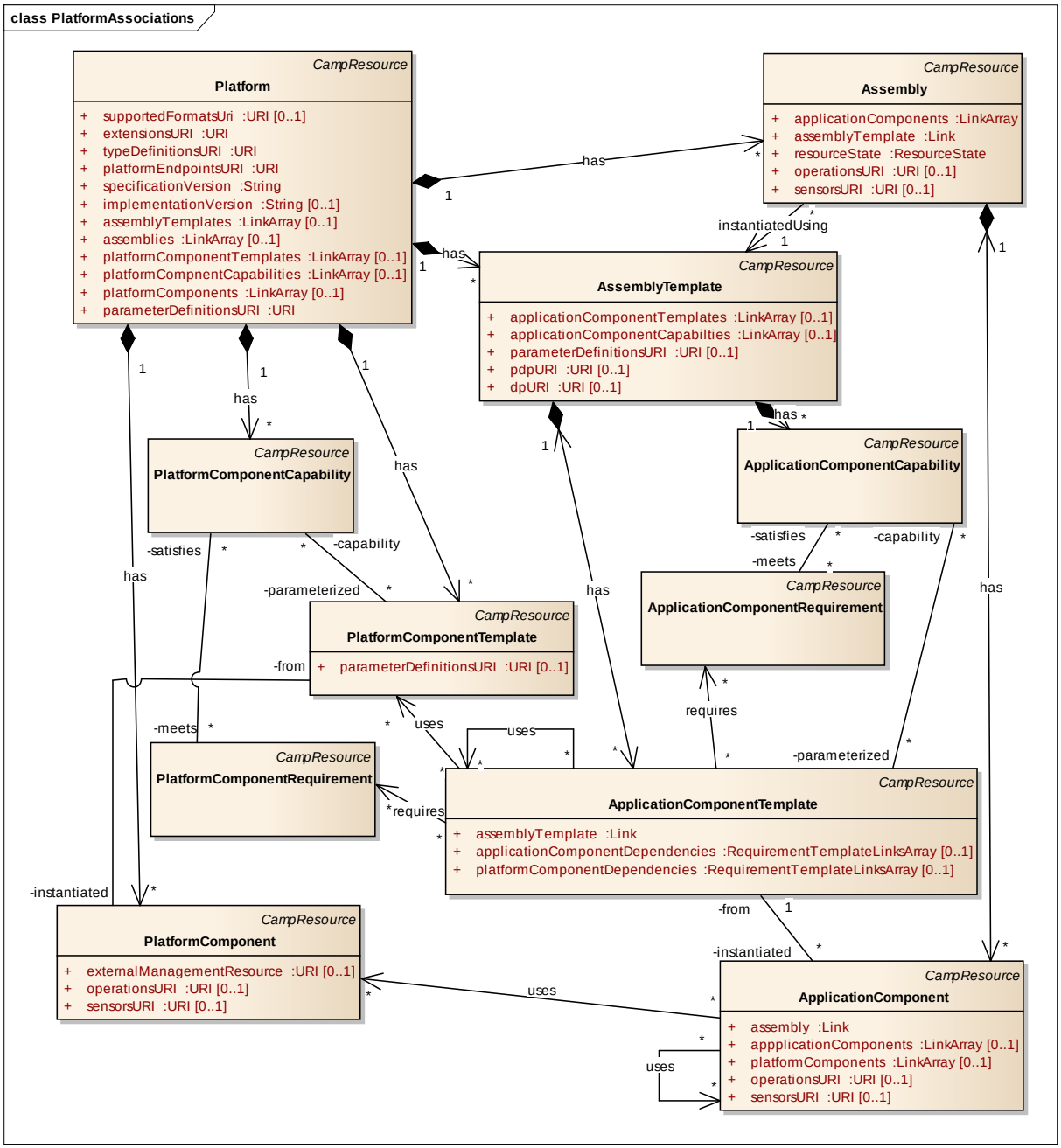

Figure 2-6: Platform Resource Relationships

Figure 2-6 shows the relationships between Platform Resources using a UML class diagram.

Associations which are visible thru pointer attributes in resources (i.e. URI, Link, or LinkArray attribute types) are shown using UML named associations with navigation arrows.

Associations which model implementation specific relationships, not visible thru the API, are represented using the UML association end notation, without navigation arrows. The '–' symbol on these association ends expresses that access is private (i.e. navigation using resource links is not available thru the API).

Strict aggregation (i.e. "has" relationship) is indicated using a solid diamond on the association end attached to the owning resource. This implies that the owned resource cannot exist independent of its owner.

A Platform provides a set of Platform Components that can be used by the invoking applications. Examples of Platform Components include servlet containers, web servers, LDAP stores, and database instances. Platforms can also provide higher-level business components such as a business rules manager to gain competitive advantage and developer loyalty. The Platform Administrator manages and operates the implementation of Platform Components.

An application is composed of a set of Application Components that depend on one or more Platform Components. Examples of Application Components include Ruby gems, Java libraries, and PHP modules. Application Components can also include non-code artifacts such as datasets and collections of identity information.

Application Components can also interact with other Application Components. Thus, an Application Component has two different sets of dependencies. It depends on the Components provided by the platform, and depends on services provided by other Application Components. Such Application Components can be on the same platform or can reside at some other location. The Assembly resource is used to aggregate the management of these components as shown in Figure 2-6.

Platform Components have a set of Platform Component Capabilities that an application can choose from to meet its requirements. Applications can tailor Platform Component Requirements (a refinement or narrowing of the configuration ranges in the Capabilities) to meet their needs based on the range of parameters expressed in the Platform Component Capability.

The relationships of an Assembly Template to Application Component Templates and Platform Component Templates, or their Requirements are shown in Figure 2-6.

An Application Component can express the exact configuration of its dependency on other Components using one of the Component Template resources (either an Application Component Template or Platform Component Template). Alternatively, it can express a range of configuration values that are acceptable for that dependency by using one of the Component Requirement resources (either an Application Component Requirement or Platform Component Requirement). This might be done, for example, in an ADE when the existing Component Templates are not known. During the deployment, these Requirements are matched with Capabilities that have attribute values that fall within the ranges specified by the Requirements.

An Application Component Template cannot be instantiated unless all of its dependencies are satisfied. An Application Component Template SHALL be referenced by a single Assembly Template. **[CO-01]** An Assembly Template SHALL NOT be instantiated until all of its Application Component Templates are successfully instantiated. **[CO-02]**

2.2 Deployment

A Deployment Plan (DP) is packaging management meta-data that provides a description of the artifacts that make up an application, the services that are required to execute or utilize those artifacts, and the relationship of the artifacts to those services.

A Platform Deployment Package (PDP) is an archive containing the DP together with application content files such as web archives, database schemas, scripts, source code, localization bundles, and icons; and metadata files such as manifests, checksums, signatures, and certificates. It can be used to move an Application and its Components from Platform to Platform, or between an Application Development Environment and a Platform.

A CAMP Consumer can create a new Assembly Template by uploading either a PDP or a DP to the Platform resource URI using an HTTP POST request. Assembly resources are created from the Assembly Templates. An Assembly resource represents a running instance of an application. An Assembly resource can be created using an HTTP POST request to the URI of an Assembly Template resource.

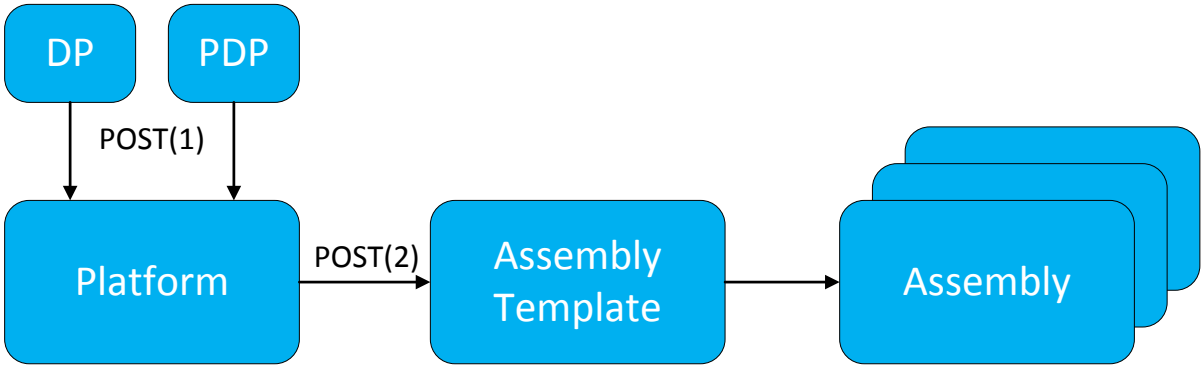

Figure 2-7: Creation of an Assembly resource

In Figure 2-7 the POST(1) request creates an Assembly Template resource by uploading either a PDP or a DP to the Platform resource's URI. The POST(2) request on the Assembly Template's URI creates an Assembly resource. Multiple Assembly resources can be created from an Assembly Template by submitting multiple HTTP POST requests.

2.3 Versions and Extensions

This specification supports multiple endpoints and versions, and extensions. All of these are represented in the resource model so they can be discovered by CAMP Consumers. The resources enabling discovery are shown in Figure 2-8, and their relationships are shown in Figure 2-9.

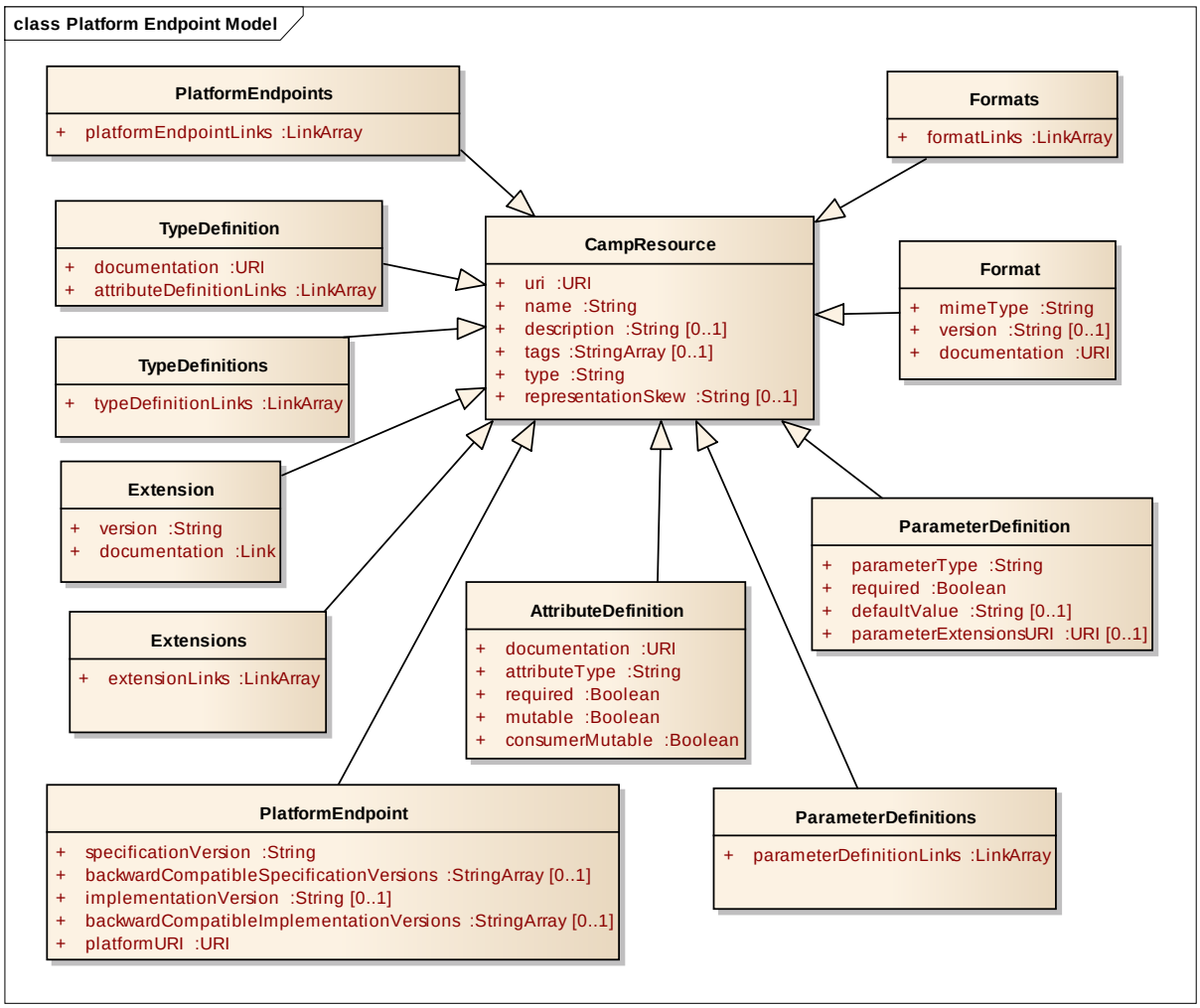

Figure 2-8: Platform Endpoint and Meta Data Resources

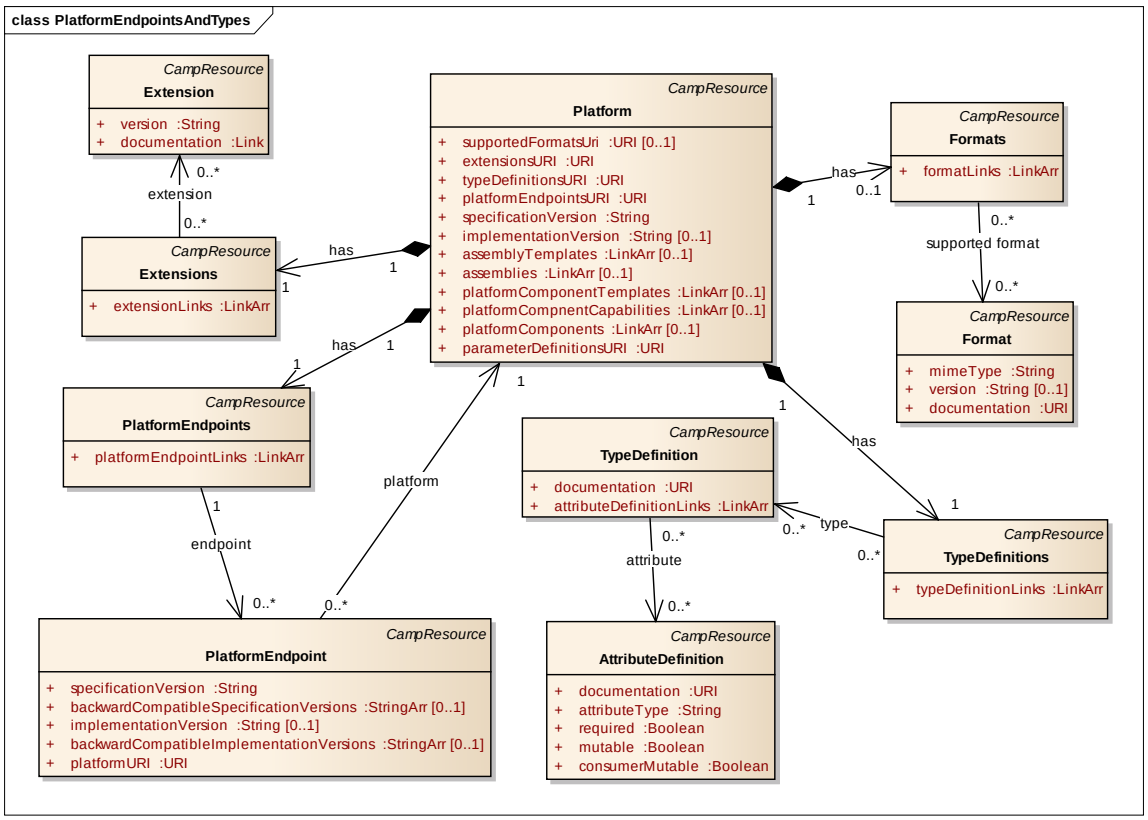

Figure 2-9: Platform Endpoint and Extension Resource Relationships

2.4 Parameters

Parameters can be defined on the Platform, Assembly Template, and Platform Component Template resources. Parameters affect the resources that are instantiated from these resources. Figure 2-10 illustrates the relationships between these resources and the resources used to represent Parameters.

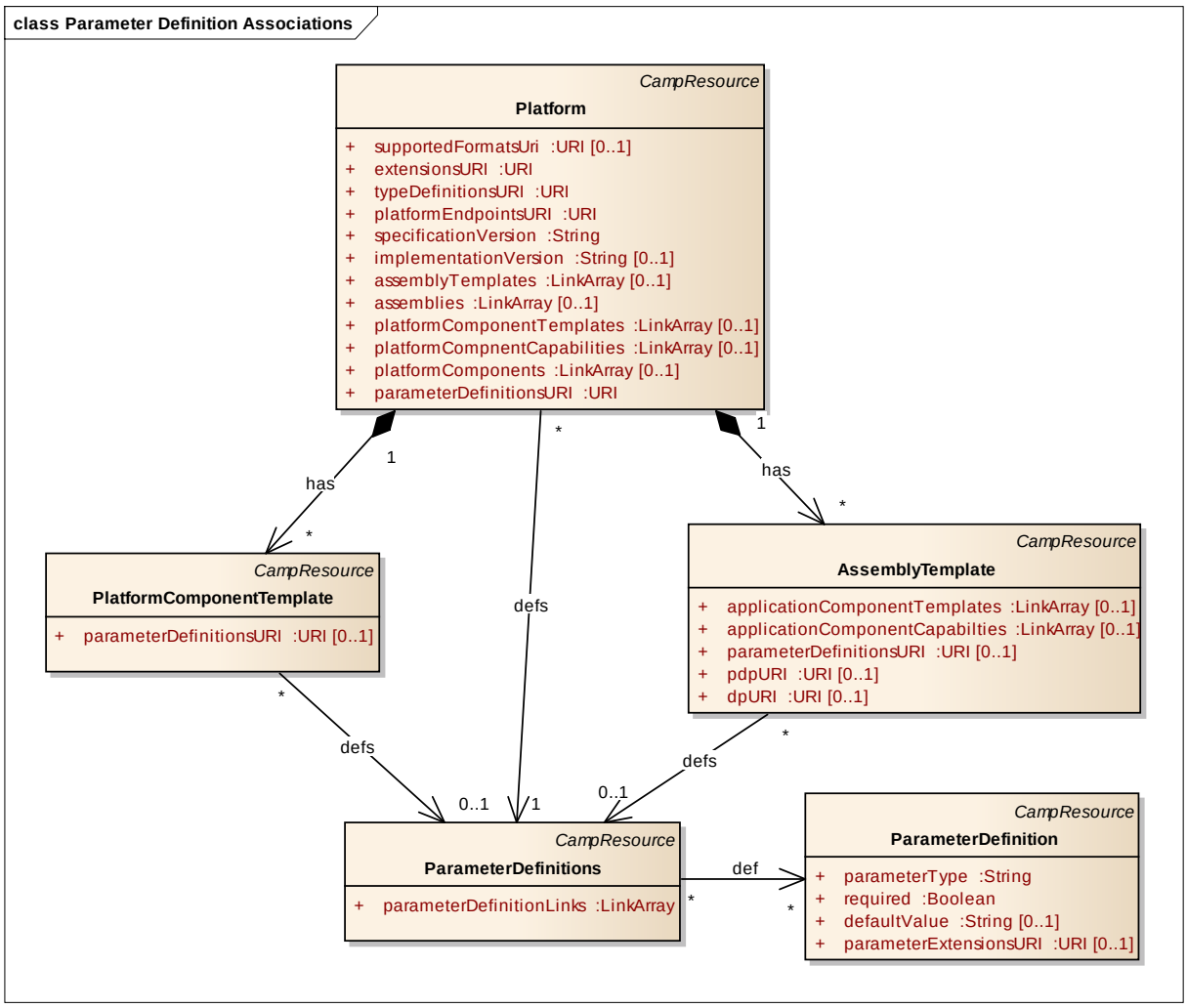

Figure 2-10: Parameter Definition Relationships

2.5 CAMP Common Types

Many of the attributes in the UML class diagrams have one of the CAMP common types, specified in Section 5.1, "Common Types". Figure 2-11 is a UML diagram showing the common data types as UML Data Types, which are used for the Types of these Resource Attribute definitions.

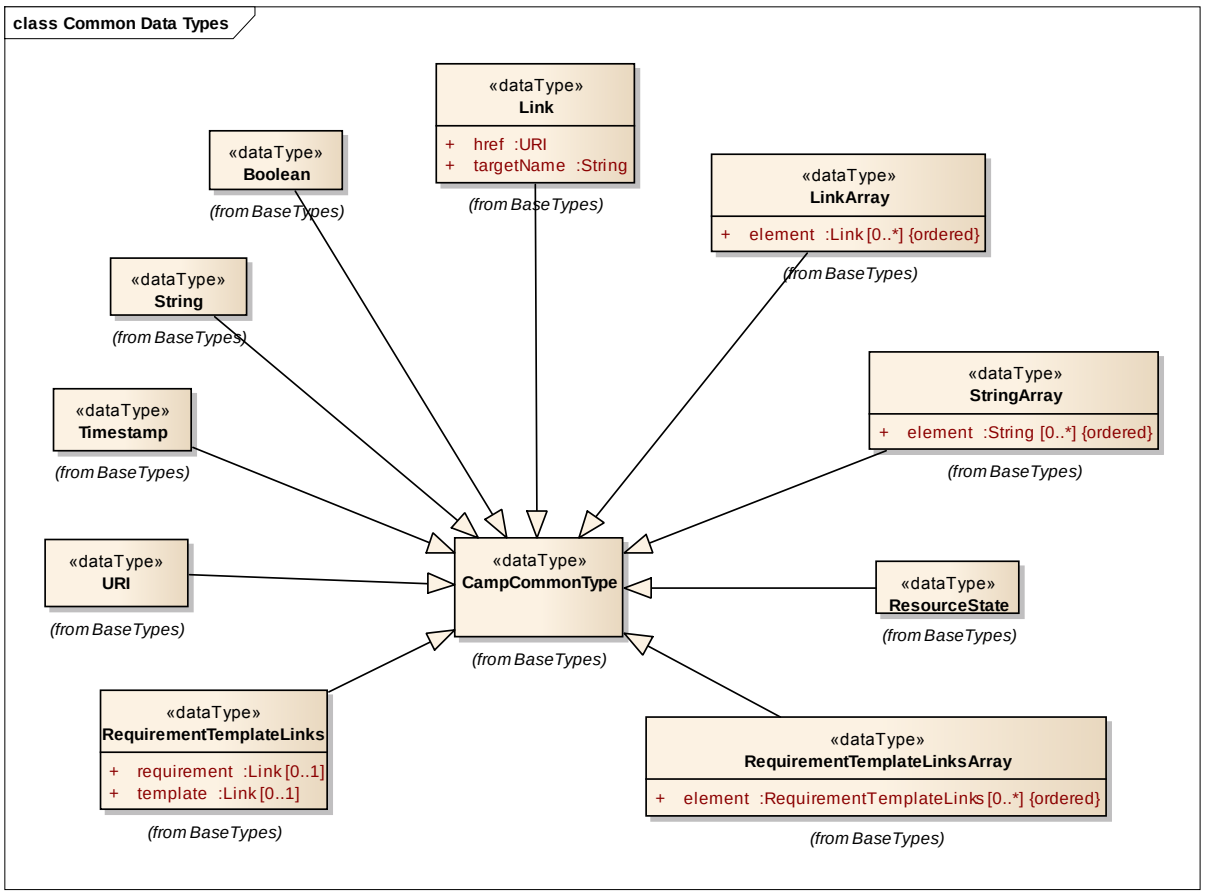

Figure 2-11: CAMP Common Base Types for Resource Attribute Definitions

Multi-valued member attributes are used to model the elements of the LinkArray, StringArray, and RequirementTemplateLinksArray types. This is for modeling purposes; the attribute name "element" does not appear in the JSON serialization for these common types.

The three array types have their elements tagged as ordered.

2.6 Representation Skew

There can be situations in which the information in the resources provided by the CAMP API is not a complete or accurate representation of the state of the underlying platform implementation. For example, while instantiating a new instance of an application, a CAMP server might be asked to provide a representation of an Application Component that corresponds to a dataset that is in the process of being loaded onto a database service. While the CAMP server might not be able to provide all of the information about this Application Component, it would be inaccurate to say that Application Component does not exist; it exists but it an intermediate state. It is expected that these sorts of situations will be the exception and that, during the majority of its existence, a CAMP resource will be in synch with the state of its underlying platform implementation.

The significance of this skew is the manner in which it affects the client's interactions with, and expectations about, the resource. In the above example, while the client cannot reasonably expect to make any changes to the Application Component until it has reached a steady state, the client can expect that the resource will reach this state in the near future. There are other situations in which, through some sort of error or breakdown, the CAMP API cannot tell when or if the information in the resource will ever be synchronized with the underlying implementation.

Details on how this skew is exposed in the CAMP API are provided in Section 5.3.6, "representationSkew".

3 Application Management Lifecycle

The figures in this section are UML object instance diagrams, which represent related Resource instances at various stages of Platform Resource instantiation. For simplification, attributes for these resources are not shown. For a comprehensive list of attributes for resources see Section 5, "Resources".

Instances in these diagrams are indicated by boxes, with an underlined "*object-name: Class"* label. Relationships visible thru the API are shown using associations with navigation arrows. Implementation specific relationships are indicated using the association end notation, without navigation arrows.

3.1 Initial Platform Resources

The CAMP model includes the resources below when no Assemblies have been created.

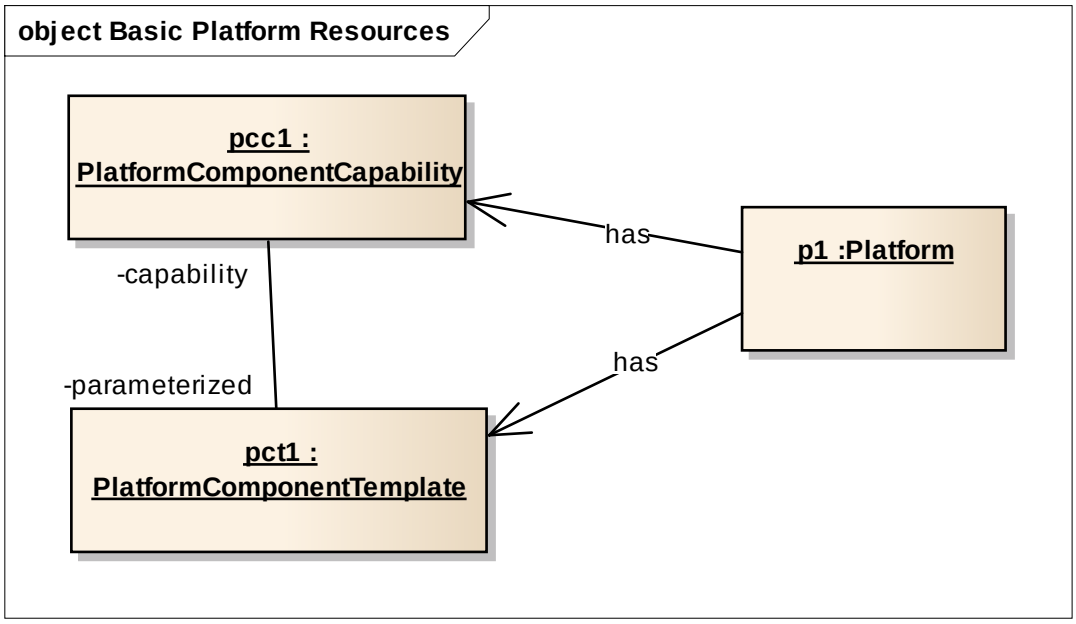

Figure 3-1: Initial Platform Resources

When the Application Administrator first accesses a new account a Platform will have a number of resources visible through the API. The Platform resource is used to find the other resources in this diagram. The various Platform Component Capabilities allow for discovery of all the platform services that are available along with value ranges for each service's attributes. The various Platform Component Templates may represent pools of previously configured platform resources and represent this configuration as specific values for each of the attributes. These values are chosen from the range of values in the corresponding Platform Component Capability.

3.2 Creating an Assembly Template from a PDP

A CAMP Consumer can create a new Assembly Template by uploading either a PDP or a DP to the Platform resource URI using an HTTP POST request (see Section 2.2, "Deployment"). When the Assembly Template is created other resources such as Application Component Templates or Platform Component Requirements might also be created by the CAMP Provider to facilitate the future creation of an Assembly resource. The model might then appear as follows:

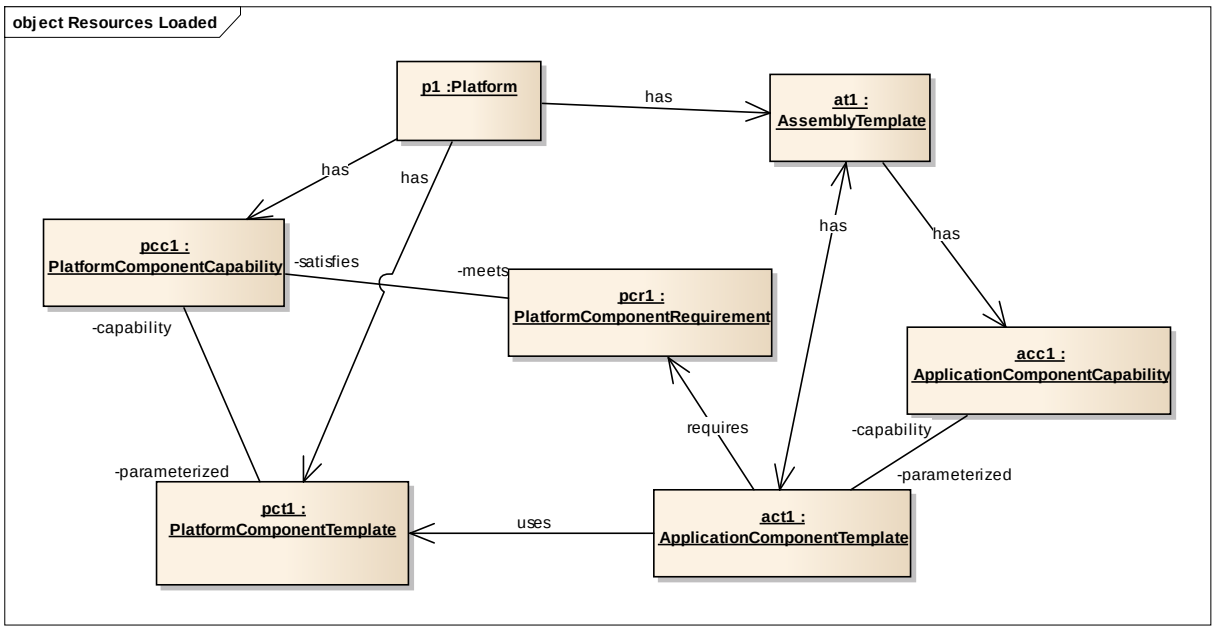

Figure 3-2: Loaded Resources

An Assembly Template is now present with one or more Application Component Templates, Application Component Requirements and Platform Component Requirements. The Application Component Template resources represent code and/or associated resources that were loaded with the PDP. The Application Component Requirement resources were used to find Application Component Templates (pre-existing software libraries for example) that were not part of the PDP but are required for the application. The Application Component Capability resources show the range of configurable attributes that the loaded Application Components can take on, when reused by future Application Components.

The Platform Component Requirement resources were used to find Platform Component Templates (pre-configured platform resources) that were part of the platform and required by the loaded Application Components, but perhaps unknown at the time the PDP was created (in an ADE for example).

If any of its requirements are not resolved, an Assembly Template could require modification before it can be used to create Assembly resources.

3.3 Creating and Managing an Application Assembly

To start an application an Assembly is created from the Assembly Template using a POST request (see Section 2.2, “Deployment”). This also creates Application Components and Platform Components corresponding to their respective templates, and the model would look as follows:

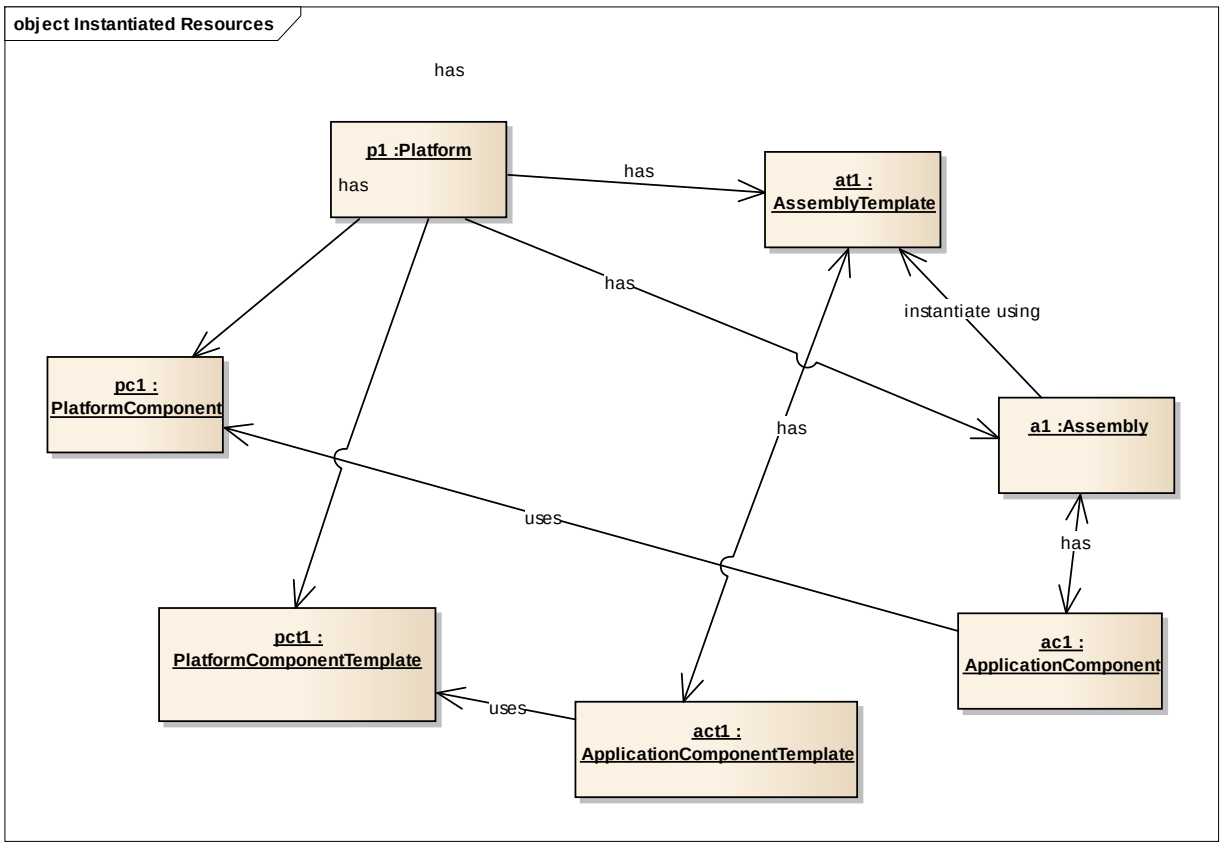

Figure 3-3: Instantiated Resources

To manage the operation of the application, the Application Administrator interacts with the Assembly resource and the related Application Component and Platform Component resources.

The traversal of the resources in the model can be accomplished by following the navigation arrows on the associations in these object instance diagrams, from each resource to the other resources it depends on.

The Application Administrator can observe real-time operational metrics through Sensor resources on Assembly and Component resources. In response to these metrics, the Application Administrator — or an automated process such as a management system — can affect changes to those resources through the Operation resources linked from those same resources.

3.4 Removing Assemblies and Assembly Templates

When finished working with an application, an Application Administrator can delete an Assembly using a DELETE request. The CAMP platform will typically soon thereafter remove the Assembly resource and all associated resources which are dedicated to that Assembly, such as Application Components. Where such a resource is not removed immediately, for example, when it is in the process of shutting down, it ought to present a representation skew of DESTROYING in the interim.

When the original Assembly Template is no longer needed, an Application Administrator can, again, delete it using a DELETE request. Again, the CAMP platform will typically delete the Assembly Template and all associated resources which are dedicated to that Assembly Template. Where this deletion is accepted but not immediate, such as because an Assembly is in use that references the Assembly Template, again the CAMP platform ought to present a representation skew of DESTROYING for the resources being deleted.

Following a lifecycle where an Assembly Template is created, then an Assembly is created, then the Assembly is deleted, then the Assembly Template is deleted, the model of Resources in the CAMP Provider will typically look similar to the initial model shown in Figure 3-1.

4 Platform Deployment Package

The Platform Deployment Package (PDP) ensures portability across platforms. It can be created by a platform to export to another platform, which then imports it. It can also be created by an Application Development Environment running locally or deployed as Software as a Service in the cloud. The PDP (and the Deployment Plan (DP), see Section 4.1.1, "Deployment Plan Overview") defines the formats for onboarding new applications into a CAMP-enabled Provider. A Consumer can create a new Assembly Template resource in the Platform by submitting a valid PDP or DP to a Platform URI using a POST request (see Section 6.11, "Registering a PDP").

4.1 PDP Package Structure

A PDP is an archive which contains a Deployment Plan (DP) file named `camp.yaml` at the root of the archive (see Section 4.1.1, "Deployment Plan Overview"). A PDP archive MAY include other files related to the application including, but not limited to, language-specific bundles, resource files, application content files such as web archives, database schemas, scripts, source code, localization bundles, and icons; and metadata files such as manifests, checksums, signatures, and certificates. **[PDP-01]**

4.1.1 Supported Archive Formats

A Provider SHALL support the following archive formats for a PDP:

- A PDP as a ZIP archive [ZIP] **[PDP-02]**
- A PDP as a TAR archive [TAR] **[PDP-03]**
- A PDP as a GZIP [GZIP] compressed TAR archive **[PDP-04]**

Providers MAY support additional archive formats for the PDP. **[PDP-05]**

4.1.2 Validating Integrity

A PDP MAY contain a manifest file, named camp.mf, at the root of the archive. **[PDP-06]** This file contains SHA256 [SHA256] digests of some or all files in the package. A Provider SHOULD reject a PDP if any digest listed in the manifest does not match the computed digest for that file in the package. **[PDP-07]**

A PDP MAY contain a certificate, named camp.cert, at the root of the archive. **[PDP-08]** This file contains a signed SHA256 digest for the manifest file and the corresponding X.509 certificate. A Provider SHOULD reject any PDP for which the signature verification fails. **[PDP-09]**

The format of the manifest file and the certificate file SHALL be as defined by the OVF specification [OVF]. **[PDP-10]**

4.2 Deployment Plan Overview

The Deployment Plan (DP) provides a description of the artifacts that make up an application, the services that are required to execute or utilize those artifacts, and the relationship of the artifacts to those services.

Example 1: Minimal DP describing an application consisting of a single RPM file.

```
00 campVersion: CAMP 1.1
01 artifacts:
02   -
03     artifactType: org.rpm:RPM
04     content: { href: my-app.rpm }
```


The above example describes an application that consists of a single RPM package, named “my-app.rpm”, which exists at the root of the PDP archive. The services required to install/run this package and the relationship of this package to those services are not specified.

4.2.1 Types

Deployment Plans can contain descriptions of artifacts, services and their relationships. However, it is outside the scope of this specification to provide detailed definitions of these entities. Instead Deployment Plans use ‘type’ nodes to identify these things. ‘Type’ nodes are Strings that describe entities that are managed by CAMP, but whose value and semantics are defined outside the CAMP specification. For example, a group of PaaS providers could agree to use the artifact type “org.rpm:RPM” to identify RPM packages. Line 03 in Example 1, above, is an example of the use of such a type.

To promote portability, both providers and consumers of the CAMP API are encouraged to namespace-qualify the types that they use. For example, if a PaaS provider supports a requirement type that expresses the relationship “deploy on a Spring container”, the value “com.paas-r-us.spring.DeployOn” is preferable to the value “DeployOn”, as the latter is likely to collide with similar types.

In addition to defining the labels for artifacts, services, and their relationships it is expected that those individuals and organizations that define such labels will also define additional attributes that qualify and constrain the entity that is referenced.

Example 2: Expanded DP describing details of how to install the RPM

```
00 campVersion: CAMP 1.1
01 artifacts:
02   -
03     artifactType: org.rpm:RPM
04     content: { my-app.rpm }
05     requirements:
06       -
07         requirementType: org.rpm:Install
08         org.rpm.installopts.excludedocs: true
```

The above example is an expansion of Example 1 that uses a Requirement Specification (lines 07-08) to describe the relationship of the RPM artifact to some (unspecified) service. This relationship is labeled on line 07 with a value of “org.rpm:Install”. It is assumed that this semantics associated with this label are documented, and that this documentation also describes the value space and semantics of the “org.rpm.installopts.excludedocs” node used on line 08.

4.2.2 Service Specifications

In addition to artifacts and their relationships to the platform, Deployment Plans can also contain descriptions of the services that can be used by those artifacts. These descriptions take the form of Service Specifications.

Example 3: Expanded DP that includes a Service Specification

```
00 campVersion: CAMP 1.1
01 artifacts:
02   -
03     artifactType: org.rpm:RPM
04     content: { my-app.rpm }
05     requirements:
06       -
07         requirementType: org.rpm:Install
08         org.rpm.installopts.excludedocs: true
09         fulfillment:
10           characteristics:
11             -
12               characteristicType: com.example:Linux
13               com.example.linux.kernelVersion: 3.9.6
14               org.iaas.bitsize: 64
```


The above example is an expansion of Example 2 that uses a Service Specification to outline the parameters of the service on which the RPM is to be installed. Lines 10-14 in the above example make up the Service Specification which, in this case, consists of a single Characteristic Specification (lines 12-14). Line 12 indicates that the target service for the RPM is a Linux instance. Line 13 indicates that this target has to run kernel version 3.9.6 and line 14 indicates that it has to be a 64-bit system.

4.2.2.1 Shared Services

There are situations in which an application can have two or more artifacts that need to share the same runtime instance of a service.

Example 4: DP with shared Service Specification

```
00 campVersion: CAMP 1.1
01 artifacts:
02   -
03     artifactType: com.java:WAR
04     content: { href: vitaminder.war }
05     requirements:
06       -
07         requirementType: com.java:HostOn
08         com.java.servlet.contextName: "/vitaM"
09         fulfillment:
10           …
11       -
12         requirementType: com.java.jdbc:ConnectTo
13         fulfillment:
14           id: db
15           characteristics:
16             -
17               characteristicType: org.storage.db:RDBM
18               …
19             -
20               characteristicType: org.storage.db:Replication
21               …
22             -
23               characteristicType: org.iso.sql:SQL
24               …
25   -
26     artifactType: org.sql:SqlScript
27     content: { href: vitaminder.sql }
28     requirements:
29       -
30         requirementType: org.sql:ExecuteAt
31         fulfillment: id:db
```

The above example describes an application with two components, a WAR file and an SQL script. In the case of this particular application, the SQL script is used to initialize the database that will be used by the WAR file. The two artifacts need to share a common database instance or the application will not work. Lines 14-24 describe the target database service. Line 14 is an ‘id’ node with the value ‘db’. This node is used as the target for the ‘fulfillment’ node on line 31. The use of the “id:db” reference on line 31 indicates that the database instance that will be used to fulfill the “org.sql:ExecuteAt” relationship on line 30 is the same instance that will be used to fulfill the “com.java.jdbc:ConnectTo” relationship on line 12.

The Deployment Plan in Example 4 can also be expressed in the following manner:

Example 5: DP with ‘services’ section

```
00 campVersion: CAMP 1.1
01 artifacts:
02   -
03     artifactType: com.java:WAR
04     content: { href: vitaminder.war }
05     requirements:
06       -
07         requirementType: com.java:HostOn
08         com.java.servlet.contextName: "/vitaM"
09         fulfillment:
10           …
11       -
12         requirementType: com.java.jdbc:ConnectTo
13         fulfillment: id:db
14   -
15     artifactType: org.sql:SqlScript
16     content: { href: vitaminder.sql }
17     requirements:
18       -
19         requirementType: org.sql:ExecuteAt
20         fulfillment: id:db
21 services:
22   -
23     id: db
24     characteristics:
25       -
26         characteristicType: org.storage.db:RDBM
27         …
28       -
29         characteristicType: org.storage.db:Replication
30         …
31       -
32         characteristicType: org.iso.sql:SQL
33         …
```

This example is equivalent to Example 4, but places the specification of the shared database service in the ‘services’ section that begins on line 21. In this case, both the WAR file and SQL script artifacts use references (lines 13 and 20, respectively) to indicate the service that will be used to fulfill their particular relationships.

4.2.2.2 Service Frameworks

There are situations in which the artifacts of an application are dynamically added (e.g. via a git [Git] push operation) after the creation of a “service framework” on which these artifacts can be deployed. Such a framework can be specified via a Deployment Plan that contains Service Specifications but no Artifact Specifications.

Example 6: DP with only Services Specifications

```
campVersion: CAMP 1.1
services:
  -
    name: Rails Runtime
    characteristics:
      -
        characteristicType: org.ruby-lang:Ruby
        …
      -
        characteristicType: org.rubyonrails:Rails
        …
  -
    name: Database
    characteristics:
      -
        characteristicType: org.storage.db:RDBM
        …
  -
    name: Git Repo
    characteristics:
      -
        characteristicType: com.git-scm:GIT
        …
```

The above example specifies a set of services onto which the user can deploy Rails components by pushing them to the git repository that will be created as a result of registering this DP and instantiating the resulting Assembly Template.

4.2.3 Names, Description, and Tags

Deployment Plans, artifacts and services can be decorated with names, descriptions, and tags. CAMP platform implementations can use this information when creating the resources that correspond to these entities. For example, the following Deployment Plan:

Example 7: DP with names, descriptions, and tags

```
name: Mike’s Drupal Instance
description: Drupal 6.28
tags: [ PHP, Drupal6, mikez ]
campVersion: CAMP 1.1
artifacts:
  -
    artifactType: net.php:Module
    content:
      href: ftp://ftp.drupal.org/files/projects/drupal-6.28.tar.gz
…
```

when successfully registered, could result in the creation of, among other resources, the following Assembly Template resource:

```
{
  "type": "assemblyTemplate",
  "uri: "http://uswest.paas-r-us.com/camp/AssemblyTemplate/101",
  "name": "Mike’s Drupal Instance",
  "description": "Drupal 6.28",
  "tags": [ "PHP", "Drupal6", "mikez" ],
  …
}
```


4.3 Deployment Plan Schema

A Platform Deployment Package (PDP) SHALL contain a single Deployment Plan. **[PDP-11]** The Deployment Plan SHALL be located at the root of the PDP archive. **[PDP-12]** The Deployment Plan file SHALL be named "camp.yaml" and SHALL consist of a well-formed YAML 1.1 [YAML 1.1] file that conforms to the description provided in this section. **[PDP-13]** Note the description of the structures and information in this section utilizes YAML's nomenclature.

4.3.1 General Attributes

The following nodes may appear as elements of the Deployment Plan, Artifact Specifications, or Service Specifications.

4.3.1.1 name

Type: String

Required: false

This node expresses the human-readable name of the Plan or Specification. Providers MAY reflect the value of this attribute in the names of any resources that are created in the processing the Deployment Plan. **[PDP-14]**

4.3.1.2 description

Type: String

Required: false

This node expresses the human-readable description of the Plan or Specification. Providers MAY reflect the value of this attribute in the descriptions of the resources that are in the processing the Deployment Plan. **[PDP-15]**

4.3.1.3 tags

Type: String[]

Required: false

This node expresses an array of human-readable tags for the Plan or Specification. Providers MAY reflect the values of this attribute in the tags of the resources that are created in the processing of the Deployment Plan. **[PDP-16]**

4.3.2 DeploymentPlan

This type defines the structure of the Deployment Plan. A Deployment Plan SHALL contain a single instance of a DeploymentPlan node. **[PDP-17]** This node has the following, general representation:

```
name: String ?
description: String ?
tags: ?
  - String +
campVersion: String
origin: String ?
artifacts: ?
  -
    ArtifactSpecification +
services: ?
  -
    ServiceSpecification +
```

In addition to the general attributes, the DeploymentPlan node contains the following attributes:

4.3.2.1 campVersion

Type: String

Required: true

The value of this node expresses the version of the CAMP specification to which the Deployment Plan conforms. This value SHALL be the Specification Version String of the CAMP specification to which this Deployment Plan conforms. **[PDP-18]**

For Deployment Plans that conform to this document, the value of this node SHALL be “CAMP 1.1” as defined in Section **Error! Reference source not found.** “Specification Version”. **[PDP-19]**

4.3.2.2 origin

Type: String

Required: false

The value of this node specifies the origin of the Deployment Plan. For example, when exporting an Assembly Template into a PDP, a platform instance might use the URL of its Platform resource for this value. Alternatively, an Application Development Environment could use its name and version.

4.3.2.3 artifacts

Type: ArtifactSpecification[]

Required: false

This node describes the artifacts that, together with the Deployment Plan itself, make up the PDP. Providers SHALL NOT regard the order of the ArtifactSpecifications within this array as semantically significant. **[PDP-20]**

4.3.2.4 services

Type: ServiceSpecification[]

Required: false

This node describes the services that the application in the PDP requires in order to function. Providers SHALL NOT treat the order of ServiceSpecifications within this array as semantically significant. **[PDP-21]**

4.3.3 ArtifactSpecification

This type describes an artifact of the application. The artifact MAY be contained within the PDP or MAY exist in some other location. **[PDP-22]** This type has the following, general representation:

```
name: String ?
description: String ?
tags: ?
  - String +
artifactType: String
content: ContentSpecification
requirements: ?
  -
    RequirementSpecification +
```

In addition to the general attributes, the ArtifactSpecification type contains the following attributes:

4.3.3.1 artifactType

Type: String

Required: true

This node defines the type of the artifact described by this Artifact Specification. See Section 4.2.1, “Types”, for a general description of the definition and treatment of these values.

4.3.3.2 content

Type: ContentSpecification

Required: true

This node defines the content of the artifact described by this Artifact Specification. See Section 4.3.5, "ContentSpecification", for a description of this node's type.

4.3.3.3 requirements

Type: RequirementSpecification[]

Required: false

This array specifies the ways in which the artifact described by this Artifact Specification engages with the services provided by the platform. See Section 4.3.6, "RequirementSpecification", for a description of the RequirementSpecification type. Providers SHALL NOT treat the order of RequirementSpecifications within this array as semantically significant. **[PDP-23]**

4.3.4 ServiceSpecification

A ServiceSpecification outlines a service that the application contained within the PDP requires in order to function. This outline is not intended to be a complete description of the service but, instead, delineate those characteristics that are significant to the application. This type has the following, general representation:

```
name: String ?
description: String ?
tags: ?
  - String +
id: String ?
characteristics: ?
  -
    CharacteristicSpecification +
```

In addition to the general attributes, the ServiceSpecification type contains the following attributes:

4.3.4.1 id

Type: String

Required: false

The value of this node serves as an anchor for intra-Deployment Plan references. See Section 4.3.6.2, "fulfillment", for information on how this anchor is used.

4.3.4.2 characteristics

Type: CharacteristicSpecification[]

Required: true

This array provides the characteristics of the service described by this Service Specification. See Section 4.3.7, "CharacteristicSpecification", for a description of the CharacteristicSpecification type. Providers SHALL NOT treat the order of CharacteristicSpecifications within this array as semantically significant. **[PDP-24]**

4.3.5 ContentSpecification

This type defines the content of a component. Content Specifications SHALL declare either a String attribute "href" that references the content or a String attribute "data" whose value is the data or, but not both. **[PDP-25]** It has the following, general representation:

```
href: URI
```


or

```
data: String
```

When “href” is used in a Content Specification its value is interpreted as follows:

- For IANA-assigned URI schemes (e.g. “http”, “https”, “ftp”, etc.) the Provider SHALL engage the protocol as per the relevant spec. **[PDP-26]** Providers SHALL support the “http” and “https” URI schemes. **[PDP-27]** A Provider MAY support additional URI schemes. **[PDP-28]**
- URL’s with the special scheme “pdp:” are interpreted as files contained in the PDP.
    - If the path segment (after the “pdp:”) begins with a “/” it is an absolute path.
    - If the path segment *is* “!” (i.e. the URL is “pdp:!”), the reference is to the PDP archive itself. This is useful in making an existing deployment package (such as a WAR) function as a PDP.
    - For any other path segment, the path is relative to the location of the file which contains the Content Specification, subject to the guidelines below.
    - Where the path segment contains the special character “!”, it is treated as a delimiter to look for the path to the right of “!” inside the archive at the path to the left of the “!". Providers SHALL understand this delimiter and SHALL NOT resolve any content if the archive format is unsupported. **[PDP-29]** For example “pdp:/certs.zip!/id_rsa.pub” refers to a file “id_rsa.pub” contained at the root of a “certs.zip” file located at the root of the PDP, and is valid only on platforms which support the ZIP format in conjunction with “!”. On other platforms the link will not be resolved.
- Where the value is not a URI, it is interpreted as a “pdp:” protocol link, as though it were preceded by “pdp:/”.

Example 7: A DP describing an application consisting of the contents of the PDP

```
00 artifacts:
01   -
02     type: org.oasis-open.tosca:CSAR
03     content: { href: pdp:! }
04     requirements:
05       -
06         type: com.oasis-open.tosca:DeployOn
07         …
```

The above example illustrates the use of the “pdp:!” construct wherein the content being referenced (on line 03) is the PDP itself. In this case the PDP is also an OASIS TOSCA v1 Cloud Service Archive.

4.3.6 RequirementSpecification

A RequirementSpecification describes the relationship between an artifact and a service. It has the following, general representation:

```
requirementType: String
fulfillment: [String | ServiceSpecification] ?
… ?
```

4.3.6.1 requirementType

The value of this node defines the relationship of the artifact that contains this RequirementSpecification to a service. For example, “com.java:HostOn”. See Section 4.2.1, “Types”, for a general description of the definition and treatment of these values.

It is expected that RequirementSpecifications will contain extension nodes that modify or provide additional information about the relationship that they describe. The value space and semantics of these extensions ought to be part of the definition of the value used in the “type” node. For example, the definition of the “com.java:HostOn” relationship might define a “com.java:contextPath” node whose value specifies the desired context path for the artifact when it is deployed on its selected service.

4.3.6.2 fulfillment

The value of this node either describes, or references a description of, the other party in the relationship (i.e. the service) defined by this RequirementSpecification. In the case where this node references a description, the value is a String that corresponds to the “id” node of a ServiceSpecification (e.g. “id:db”). In the case where this node contains the description, the value is a ServiceSpecification. See Section 4.3.4, “ServiceSpecification”, for a description of this type.

4.3.7 CharacteristicSpecification

A CharacteristicSpecification describes a desired characteristic or capability of a service. It has the following, general representation.

```
characteristicType: String
… ?
```

The inclusion of a CharacteristicSpecification in a ServiceSpecification indicates that the characteristic being described is significant to the application, but the degree of this significance (e.g. “absolutely necessary” versus “would be nice to have”) is not indicated.

4.3.7.1 characteristicType

The value of this node defines the characteristic being described by this CharacteristicSpecification. For example, “com.java:ServletContainer”. See Section 4.2.1, “Types”, for a general description of the definition and treatment of these values.

It is expected that CharacteristicSpecifications will contain extension nodes that modify or provide additional information about the characteristic that they describe. The value space and semantics of these extensions ought to be part of the definition of the value used in the “characteristicType” node. For example, the definition of the “org.rubyonrails:Rails” characteristic might define a “org.rubyonrails:version” node whose value specifies the version of Rails provided by the service.

5 Resources

The following sub-sections describe the resources defined by this specification.

5.1 Common Types

Resource attributes are defined using the following types:

5.1.1 Boolean

As defined by JSON [RFC4627], a token having a literal value of either `true` or `false`.

5.1.2 String

A UNICODE string as defined by JSON [RFC4627].

5.1.3 URI

A String (see above) that conforms to the syntax defined in RFC 3986 [RFC3986].

5.1.4 Timestamp

A String (see above) that conforms to the syntax defined in ISO 8601 [ISO 8601:2004]. Consumers and Providers SHALL express Timestamps in UTC (Coordinated Universal Time), with the special UTC designator ("Z"). **[RE-65]**

5.1.5 Link

The management model defined in this specification involves resource entity attribute values that link to other resource entities. For example, one of the Platform resource entity attribute values points to Assembly Templates. The “Link” type defined here is used for such attribute values.

```
{
  "href": URI
  "targetName": String
  …
}
```

The following attributes SHALL be present in a Link. **[RE-01]** Other attributes, not defined in this specification, MAY also be present. **[RE-02]**

5.1.5.1 href

This attribute is the URI [RFC 3986] of the resource referenced by this Link. The value of this attribute MAY be changed by the Provider. **[RE-03]** Consumers SHALL NOT change the value of this attribute. **[RE-04]**

5.1.5.2 targetName

This attribute echoes the value of the “name” attribute of the resource referenced by this Link. The value of this attribute may be changed by the Platform. Consumers SHALL NOT change the value of this attribute. **[RE-31]**

5.1.6 RequirementTemplateLinks

Requirement resources and Template resources are inherently related. The unsatisfied dependencies represented by Requirements are resolved by links to Templates that define services and capabilities that

satisfy those dependencies. This relationship is surfaced in the CAMP resource model through the "RequirementTemplateLinks" type. By grouping together a link to a Requirement and a link to a Template this type supports the expression the concept that "this Template satisfies this Requirement".

```
{
  "requirement": Link ?,
  "template": Link ?
  …
}
```

Though both attributes of this type are optional, a valid RequirementTemplateLinks type SHALL have at least one of the following attributes: **[RE-05]**

5.1.6.1 requirement

This optional attribute is a Link type that references either a Platform Component Requirement or an Application Component Requirement resource. The absence of this attribute indicates that the enclosing attribute (i.e. the attribute that is of type 'RequirementTemplateLinks') is expressing a link to a Template with no corresponding Requirement.

5.1.6.2 template

This optional attribute is a Link type that references either a Platform Component Template or an Application Component Template resource. The absence of this attribute indicates that the enclosing attribute (i.e. the attribute that is of type 'RequirementTemplateLinks') is expressing an unsatisfied Requirement.

5.2 Attribute Constraints

Resource attributes are constrained along a number of axes. These are:

5.2.1 Required

If the Required boolean constraint for an attribute of a resource type has a value of "true", then an instance of that resource type SHALL have the attribute present. **[RE-06]** If the value is "false" then the instance is valid with or without the attribute present.

5.2.2 Mutable

This boolean indicates the mutability of the attribute's value(s). "false" indicates that the value of the attribute, once set, SHALL NOT change for the lifetime of the resource. **[RE-07]** "true" indicates that the value of the attribute MAY change due to the actions or activity of either the provider or the consumer. **[RE-08]**

5.2.3 Consumer-mutable

This boolean indicates the ability of a consumer to set the value of the attribute. It is only relevant for mutable attributes. "false" indicates that the value(s) of the attribute SHALL NOT be changed by the Consumers. **[RE-09]** A value of "true" indicates that consumers MAY change the value of the attribute. **[RE-10]** Note that a value of 'true' does not preclude the Provider from changing the value of the attribute.

5.3 Common Resource Attributes

All the resources in this specification contain the following common attributes:

5.3.1 uri

Type: URI

Required: true

Mutable: false

This attribute expresses the URI of the resource.

5.3.2 name

Type: String

Required: true

Mutable: true

Consumer-mutable: true

This attribute expresses the human-readable name of the resource.

5.3.3 description

Type: String

Required: false

Mutable: true

Consumer-mutable: true

This attribute expresses the human-readable description of the resource.

5.3.4 tags

Type: String[]

Required: false

Mutable: true

Consumer-mutable: true

This attribute is an array of String values that may be assigned by the provider or the user. These values can be used for keywording and terms-of-interest.

5.3.5 type

Type: String

Required: true

Mutable: false

Consumer-mutable: false

This attribute expresses the CAMP resource type. Every CAMP resource type defined in this specification specifies the required value for this attribute.

5.3.6 representationSkew

Type: String

Required: false

Mutable: true

Consumer-mutable: false

The `representationSkew` attribute expresses the relationship between the information presented in the resource and the status of the platform implementation artifacts that are represented by that resource (see Section 2.6, "Representation Skew"). It is an optional, enumerated String. If present, representationSkew SHALL have one of the following values: **[RE-11]**

- "CREATING" – describes a resource that is in the process of being created. The client can expect that the resource will have a skew of "NONE" once this process has completed.

- “NONE” – is an assertion by the CAMP server that the information in the resource is an accurate representation of the underlying platform implementation. Absent some action by the client or some other event (e.g. platform shutdown), a resource with a skew of NONE can be expected to remain in synch with the platform implementation.
- “UNKNOWN” – indicates that the CAMP server cannot accurately depict the aspect of the platform implementation represented by this resource. Users can attempt to address the underlying issues(s) by manipulating this and/or other resources as specified by the API.
- “DESTROYING” – describes a resource that is in the process of being destroyed. The client can expect that the resource will cease to exist once this process has completed.

The absence of the `reprsentationSkew` attribute is semantically equivalent to the “NONE” value.

The value of the `representationSkew` attribute affects the availability of the HTTP methods for that resource. For example, resources with a `representationSkew` value of CREATING might support the GET, HEAD and DELETE methods, but no other HTTP methods. The following table lists the methods that SHALL be supported for each representationSkew value. **[RE-12]**

representationSkew value	Methods Available
CREATING	GET, DELETE
NONE	All supported methods for that resource.
UNKNOWN	All supported methods for that resource.
DESTROYING	GET

Table 5-1: representationSkew Available Methods

For each representationSkew value, CAMP Providers MAY support HTTP methods in addition to those listed in the corresponding row of Table 5-1. **[RE-13]**

5.4 Error Response Message Resource

Successful requests will generally return an HTTP status code of 200 (OK), 201 (Created), 202 (Accepted), or 204 (No Content), to indicate that the requested action has been successfully performed or submitted. In addition, they might include a response message body containing a representation of the requested information. However, it is possible for a number of things to go wrong. The various underlying causes are described (as discussed in the previous section) by various HTTP status codes in the range 400-499 (for client side errors) or 500-599 (for server side problems).

If a response is returned with an error status code (400-499 or 500-599), where appropriate, the server is encouraged to include a response message body containing an *ErrorMessage* object (as defined below). The text values of such messages might be used, for example, to communicate with a human user of the client side application.

The list of messages included in a single error response is encapsulated in the ErrorMessage data model.

Field	Type	Occurs	Description
message	Message	0..n	Zero or more message data for each individual message.

Table 5-2: ErrorMessage Data Model

An individual message contains the following fields:

Field	Type	Occurs	Description
code	String	0..1	Symbolic error code identifying the type of error reported by this message

field	String	0..1	Name of the field from the request data model that this message is associated with
hint	String	0..1	Localized text further describing the nature of the problem, possibly including potential workarounds that the client could try
text	String	1	Localized text describing the nature of the problem reported by this message
severity	String	0..1	Label indicating the severity of the error condition represented by this message Vendor shall publish the enumerators that are associated with this field and their semantics
stackTrace	String	0..1	Vendor specific stack trace associated with this message
source	String	0..1	Symbolic identifier of the implementation component that triggered this message
message-id	String	0..1	A unique string that identifies this particular message
profile	URI	0..1	A reference to the standard URI to indicate the meaning of this message

Table 5-3: Message Data Model

The *profile* attribute indicates the semantic meaning of the message which clients may handle automatically. Messages with the same profile shall adhere to the semantic requirements of that profile, but the payload (*hint*, *text*, *severity*, *stackTrace*) may be different. In other words, given a profile, clients processing the message should be able to subsequently interact with the providers in a consistent manner across.

Each provider may extend the profile to include specific scenarios and use cases.

The information captured in the *message* data element should be complementary to the HTTP status code, and could provide more detailed information. However, it SHALL NOT contradict the HTTP status code that is returned with the request. **[RE-14]**

The following table outlines the common profiles that would accompany this specification.

Profile	Description
/msg/unknown	Unknown error and information given is descriptive in nature
/msg/security	Security issues
/msg/security/authentication	An authentication error
/msg/access	Access violation error
/msg/allocation	Allocation related issues
/msg/allocation/insufficient	Insufficient resource to satisfy the request
/msg/infrastructure	Infrastructure related issues
/msg/infrastructure/maintenance	The request cannot be immediately responded due to the infrastructure being in maintenance status

Table 5-4: Common Message Profiles

5.5 HTTP Method Support

As described in Section 6.1, "Transfer Protocol", Consumers use HTTP [RFC2616] to interact with CAMP-defined resources. To foster interoperability it is necessary to define the HTTP methods supported by each resource. Note that a requirement on the Provider to support a particular HTTP method on a resource does not ensure that all requests to that resource using that method will succeed; it simply guarantees that the Provider will not fail such requests with a 405 (Method Not Allowed) error.

Providers SHALL support the HTTP GET, PUT, and PATCH methods on all of the resources defined in this section. **[RE-53]** Requirements for the support of additional HTTP methods are outlined in the descriptions of each resource below. Providers MAY elect to support additional HTTP methods in addition to those described here. **[RE-54]**

5.6 PlatformEndpoints Resource

A Provider MAY concurrently offer multiple instances of the CAMP API. **[RE-15]** The primary example of why a provider might do this is to simultaneously support two or more incompatible versions/implementations of the CAMP API, but there are many reasons for a provider to offer multiple instances of the CAMP API.

Concurrent instances are supported through the use of multiple instances of the Platform resource. The PlatformEndpoints resource allows Consumers to discover all the instances of the CAMP API that are currently available. It contains an array of Links to PlatformEndpoint resources (that each reference Platform resources), and has the following general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "platformEndpoints",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?,
  "platformEndpointLinks": Link[]
}
```

Because of the unique function of this resource, future versions of the CAMP specification SHALL NOT make non-backwards compatible changes to this resource. **[RE-16]**

NOTE: A Provider MAY expose the PlatformEndpoints and corresponding PlatformEndpoint resources in a way that allows for version discovery before the client has authenticated. **[RE-17]**

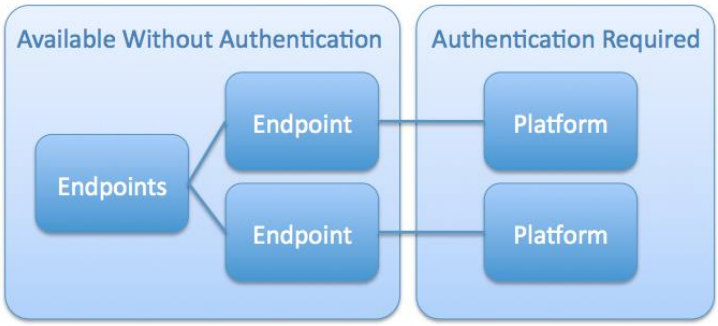

Figure 5-1: Example Implementation

The PlatformEndpoints resource contains the following attributes:

5.6.1 platformEndpointLinks

Type: Link []

Required: true

Mutable: false

This attribute is an array of Links to PlatformEndpoint Resources. This array SHALL contain at least one PlatformEndpoint Resource. **[RE-18]** References between the resources (PlatformEndpoints, PlatformEndpoint, and Platform) SHALL be self-consistent. **[RE-19]**

5.7 PlatformEndpoint Resource

Each PlatformEndpoint Resource SHALL refer to exactly one Platform Resource, and indicate the versions supported by the Platform. **[RE-20]** This specification is deliberately silent about any relationship between resources within different Platform trees. Each PlatformEndpoint Resource could represent a different CAMP API "view" of the same applications and services. On the other hand, each Endpoint could represent a completely independent system.

A PlatformEndpoint Resource has the following general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "platformEndpoint",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[], ?
  "platformUri": URI,
  "specificationVersion": String,
  "backwardCompatibleSpecificationVersions": String[] ?,
  "implementationVersion": String,
  "backwardCompatibleImplementationVersions": String[] ?
}
```

Because of the unique function of this resource, future versions of the CAMP specification SHALL NOT make non-backwards compatible changes to this resource. **[RE-21]**

Instances of the PlatformEndpoint resource contain the following attributes:

5.7.1 specificationVersion

Type: String

Required: true

Mutable: false

Each instance of a Platform resource is the root of a tree of resources, the syntax and semantics of which conform to one or more versions of the CAMP specification. The value of this attribute is the Specification Version String of the CAMP specification that is supported by the resources rooted in the Platform referenced by the platformUri attribute of this resource.

For Platforms that implement this version of the CAMP specification, the value of this attribute SHALL be "CAMP 1.1" as defined in section 1.7. **[RE-22]**

5.7.2 backwardCompatibleSpecificationVersions

Type: String[]

Required: false

Mutable: false

The values in this array identify each version of the CAMP specification that is backwards compatible with the current specificationVersion of the Platform (referenced in the platformUri attribute of this resource). The values in this array SHALL be the Specification Version Strings of previous CAMP specification versions. **[RE-23]**

If this attribute is not present, the version of the CAMP specification implemented by the Platform (referenced in the platformUri attribute of this resource) is not backwards compatible with any previous version of the CAMP specification.

PlatformEndpoint resources that reference Specification Version “CAMP 1.1” Platform Resources SHALL NOT include this attribute because no previous versions are compatible. **[RE-24]**

5.7.3 implementationVersion

Type: String

Required: false

Mutable: false

Multiple implementations of the same CAMP specification MAY be offered concurrently. **[RE-25]** For example, a Provider could offer an initial beta version of “CAMP 1.1” and, later, a production version; allowing a period of overlap for their customers to migrate from the beta to the production version. The value of this attribute is an arbitrary String that expresses the Provider-specific implementation version supported by the resources rooted in the Platform (referenced in the platformUri attribute of this resource).

5.7.4 backwardCompatibleImplementationVersions

Type: String[]

Required: false

Mutable: false

The values in this array list the provider-specific implementation versions that are backwards compatible with the implementation version of the Platform (referenced in the platformUri attribute of this resource). The values in this array are arbitrary Strings that correspond to previous implementationVersion Strings.

If this attribute is not present, the implementation version offered by the Platform (referenced in the PlatformURI attribute of this resource) is not backwards compatible with any previous implementation versions.

5.7.5 platformUri

Type: URI

Required: true

Mutable: false

This attribute is the URI of the Platform Resource that this PlatformEndpoint Resource represents.

5.8 Platform Resource

A Platform represents the Consumer’s initial view of the accessible resources and deployed entities. This resource has the following, general representation:


```
{
  "uri": URI,
  "name": String,
  "type": "platform",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?,
  "supportedFormatsUri": URI,
  "extensionsUri": URI,
  "typeDefinitionsUri": URI,
  "platformEndpointsURI": URI,
  "specificationVersion": String,
  "implementationVersion": String ?,
  "assemblyTemplates": Link[] ?,
  "assemblies": Link[] ?,
  "platformComponentTemplates": Link[] ?,
  "platformComponentCapabilities": Link[] ?,
  "platformComponents": Link[] ?,
  "parameterDefinitionsUri": URI
}
```

In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP POST method on the Platform resource as described in Section 6.11, "Registering an Application". **[RE-55]**

The Platform resource contains the following attributes:

5.8.1 supportedFormatsUri

Type: URI

Required: false

Mutable: false

This attribute is a URI reference to the Formats resource for the purpose of identifying all Supported Formats for this Platform. See Section 5.19, "Formats Resource", for details.

5.8.2 extensionsURI

Type: URI

Required: true

Mutable: false

This attribute provides a link to the Extensions this Platform supports. See Section 7.2, "Extensions Resource", for details.

5.8.3 typeDefinitionsURI

Type: URI

Required: true

Mutable: false

This attribute provides a link to the TypeDefinitions resource that provides information on the resource types that the Platform supports. See Section 5.21, "TypeDefinitions Resource", for details.

5.8.4 platformEndpointsURI

Type: URI

Requried: true

Mutable: false

This attribute provides a link to a PlatformEndpoints resource. The PlatformEndpoints resource enumerates the currently available CAMP implementations. See Section 5.6, "PlatformEndpoints Resource", for details.

5.8.5 specificationVersion

Type: String

Required: true

Mutable: false

Each instance of a Platform resource is the root of a tree of resources, the syntax and semantics of which conform to one or more versions of the CAMP specification. The value of this attribute is the Specification Version String of the CAMP specification that is supported by the resources rooted in this Platform.

For Platforms that implement this version of the CAMP specification, the value of this attribute SHALL be "CAMP 1.1" as defined in Section 1.7, "Specification Version". **[RE-26]**

The value of this attribute SHALL exactly match the value of the specificationVersion attribute of any PlatformEndpoint resource that references this Platform Resource. **[RE-27]**

5.8.6 implementationVersion

Type: String

Required: false

Mutable: false

A Provider MAY choose to offer multiple implementations of the same CAMP specification. **[RE-28]** For example, a Provider could offer an initial beta version of "CAMP 1.1" and, later, a production version; allowing a period of overlap for their customers to migrate from the beta to the production version. The value of this attribute is an arbitrary String that expresses the Provider-specific implementation version supported by the resources rooted in this Platform.

The value of this attribute SHALL exactly match the value of the implementationVersion attribute of any PlatformEndpoint resource that references this Platform resource. **[RE-29]**

5.8.7 assemblyTemplates

Type: Link[]

Required: false

Mutable: true

Consumer-mutable: false

This attribute is an array of Links to the AssemblyTemplates that are accessible to the Consumer.

5.8.8 assemblies

Type: Link[]

Required: false

Mutable: true

Consumer-mutable: false

This attribute is an array of Links to the Assembly resources that are accessible to the Consumer.

5.8.9 platformComponentTemplates

Type: Link[]

Required: false

Mutable: true

Consumer-mutable: false

This attribute is an array of Links to the PlatformComponentRequirement resources that are accessible to the Consumer.

5.8.10 platformComponentCapabilities

Type: Link[]

Required: false

Mutable: true

Consumer-mutable: false

This attribute is an array of Links to the PlatformComponentCapability resources that are accessible to the Consumer.

5.8.11 platformComponents

Type: Link[]

Required: false

Mutable: true

Consumer-mutable: false

This attribute is an array of Links to the PlatformComponent resources that are accessible to the Consumer.

5.8.12 parameterDefinitionsUri

Type: URI

Required: true

Mutable: true

Consumer-mutable: false

parameterDefinitionsURI points to a resource that contains links to parameterDefinitions that describe the parameters accepted by this resource on an HTTP POST method. Each of the parameterDefinitions provide metadata for a parameter as described in Section **Error! Reference source not found.**, "ParemeterDefinitions Resource". The Platform resource accepts the pdpUri parameter to create new AssemblyTemplate resources upon a POST. The Platform resource SHALL indirectly reference a parameterDefinition resource that describes the pdpUri parameter. **[RE-30]**

5.9 AssemblyTemplate Resource

An Assembly Template is the template for the creation of Assemblies. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "assemblyTemplate",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?,
  "applicationComponentTemplates": Link[] ?,
  "parameterDefinitionsUri": URI ?,
  "pdpUri": URI ?,
  "dpUri": URI ?
}
```

In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP POST and DELETE methods on instances of the AssemblyTemplate resource. **[RE-56]** As

described in Section 6.12, "Instantiating an Application", a successful POST request results in the instantiation of a new Assembly resource. As described in Section 6.14, "Deleting an Application Instance and a Deployed Application", a successful DELETE request removes the AssemblyTemplate from the system along with any ApplicationComponentTemplates referenced by the AssemblyTemplate and any PlatformComponentRequirements or ApplicationComponentRequirements referenced by those ApplicationComponentTemplates (i.e. the tree of resources that was created when the application was registered). A successful DELETE also removes the reference to the AssemblyTemplate from the Platform resource's `assemblyTemplates` array.

Instances of the AssemblyTemplate resource contain the following attributes:

5.9.1 applicationComponentTemplates

Type: Link[]

Required: false

Mutable: true

Consumer-mutable: false

This attribute is an array of Links to the ApplicationComponentTemplate resources that are part of this AssemblyTemplate. An AssemblyTemplate MAY have zero references to ApplicationComponentTemplate resources, **[RE-32]** but Providers SHALL NOT instantiate an Assembly using an AssemblyTemplate that does not reference at least one ApplicationComponentTemplate. **[RE-33]**

5.9.2 parameterDefinitionsUri

Type: URI

Required: false

Mutable: false

Consumer-mutable: false

This attribute references the URI of the ParameterDefinitions resource that defines parameters that may be passed to this resource. The ParameterDefinitions resource referenced by this attribute SHALL define parameters to allow setting the 'name', 'description', and 'tags' attributes of any new resource created in the course of interacting with this resource. **[RE-34]**

5.9.3 pdpUri

Type: URI

Required: false

Mutable: true

Consumer-mutable: false

This attribute specifies the URI of the PDP from which the Assembly Template resource was created. If present, it allows one to register another Assembly Template from the same PDP using the method defined in Section 6.11.1, "Registering a PDP by reference".

5.9.4 dpUri

Type: URI

Required: false

Mutable: true

Consumer-mutable: false

This attribute specifies the URI of the Deployment Plan from which the Assembly Template resource was created. If present, it allows one to register another Assembly Template from the same DP using the method defined in Section 6.11.1, "Registering a PDP by reference".

5.10 ApplicationComponentTemplate Resource

An Application Component Template represents the definition of the deployable Component. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "applicationComponentTemplate",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?,
  "assemblyTemplate": Link,
  "applicationComponentDependencies": RequirementTemplateLinks[] ?,
  "platformComponentDependencies": RequirementTemplateLinks[] ?
}
```

In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP DELETE method on instances of the ApplicationComponentTemplate resource. **[RE-57]** A successful DELETE request removes the ApplicationComponentTemplate from the system along with any PlatformComponentRequirements or ApplicationComponentRequirements referenced by that ApplicationComponentTemplates (i.e. the tree of resources rooted in the ApplicationComponentTemplate). A successful DELETE also removes the reference to the ApplicationComponentTemplate from the `applicationComponentTemplates` array of its containing AssemblyTemplate.

Instances of the ApplicationComponentTemplate resource contain the following attributes:

5.10.1 assemblyTemplate

Type: Link

Required: true

Mutable: true

Consumer-mutable: true

The attribute contains the link to the AssemblyTemplate that contains this resource. This attribute SHALL be present. **[RE-35]**

5.10.2 applicationComponentDependencies

Type: RequirementTemplateLinks[]

Required: false

Mutable: true

Consumer-mutable: true

This attribute is an array of RequirementTemplateLinks that reference related pairs of ApplicationComponentRequirement and ApplicationComponentTemplate resources.

If one of the elements in this array has a 'requirement' attribute with no matching 'template' attribute that indicates that this ApplicationComponentTemplate has an unsatisfied requirement and that the ApplicationComponent described by this template cannot be instantiated.

5.10.3 platformComponentDependencies

Type: RequirementTemplateLinks[]

Required: false

Mutable: true

Consumer-mutable: true

This attribute is an array of RequirementTemplateLinks that reference related pairs of PlatfromComponentRequirement and PlatformComponentTemplate resources.

If one of the elements in this array has a 'requirement' attribute with no match 'template' attribute that indicates that this ApplicationComponentTemplate has an unsatisfied requirement and that the ApplicationComponent described by this template cannot be instantiated.

5.11 ApplicationComponentRequirement Resource

An Application Component Requirement represents an Application Component's requirement on another Application Component and its required range of component capabilities. Each Application Component Requirement implements this class and adds its' own Attribute Ranges to the list below. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "applicationComponentRequirement",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?
}
```

In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP DELETE method on instances of the ApplicationComponentRequirement resource. **[RE-58]** A successful DELETE request removes the ApplicationComponentRequirement from the system as well as removing its reference from the `applicationComponentDependencies` array of any ApplicationComponentTemplates that refer to it.

5.12 ApplicationComponentCapability Resource

An Application Component Capability represents the definition of an Applicaton Component's range of component capabilities. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "applicationComponentCapability",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?
}
```

In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP DELETE method on instances of the ApplicationComponentCapability resource. **[RE-59]**

5.13 PlatformComponentTemplate Resource

A Platform Component Template represents the desired configuration of a Platform Component with specific values for the component capabilities. The specified value for each component attribute SHALL be in the range defined in the corresponding Platform Component Capability. **[RE-36]** This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "platformComponentTemplate",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?,
  "parameterDefinitionsUri": URI ?
}
```


Instances of the PlatformComponentTemplate resource contain the following attributes:

5.13.1 parameterDefinitionsUri

Type: URI

Required: false

Mutable: false

Consumer-mutable: false

This attribute references the URI of the ParameterDefinitions resource that defines parameters that may be passed to this resource. The ParameterDefinitions resource referenced by this attribute SHALL define parameters to allow setting the 'name', 'description', and 'tags' attributes of any new resource created in the course of interacting with this resource. **[RE-37]**

If this attribute is present in an instance of this resource, Providers SHALL support the POST method on that instance in addition to the methods defined in Section 0, "HTTP Method Support". **[RE-38]**

5.14 PlatformComponentRequirement Resource

A Platform Component Requirement represents an Application Component's requirement on a Platform Component and its required range of component capabilities. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "platformComponentRequirement",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?
}
```

In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP DELETE method on instances of the PlatformComponentRequirement resource. **[RE-60]** A successful DELETE request removes the PlatformComponentRequirement from the system as well as removing its reference from the `platformComponentDependencies` array of any ApplicationComponentTemplates that refer to it.

5.15 PlatformComponentCapability Resource

A Platform Component Capability represents the definition of Platform Component and its range of component capabilities. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "platformComponentCapability",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?
}
```

5.16 Assembly Resource

An Assembly represents an instantiated Application at runtime. This resource has the following, general representation:


```
{
  "uri": URI,
  "name": String,
  "type": "assembly",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?,
  "applicationComponents": Link[] ?,
  "assemblyTemplate": Link
  "operationsUri": URI ?,
  "sensorsUri": URI ?
}
```

In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP DELETE methods on instances of the Assembly resource as described in Section 6.14, "Deleting an Application Instance and a Deployed Application". **[RE-61]** A successful DELETE request removes the Assembly resource from the system along with any ApplicationComponents and PlatformComponents referenced by that Assembly. (i.e. the tree of resources that was created when the application was instantiated). A successful DELETE also removes the reference to the Assembly resource from the Platform resource's `assemblies` array.

Instances of the Assembly resource contain the following attributes:

5.16.1 applicationComponents

Type: Link[]

Required: true..*

Mutable: false

This attribute is an array of Links to the ApplicationComponent resources that are part of this Assembly. An Assembly resource SHALL have at least one reference to an ApplicationComponent resource. **[RE-39]**

5.16.2 assemblyTemplate

Type: Link

Required: true

Mutable: false

This attribute is a Link to the AssemblyTemplate resource from which this Assembly was created.

5.16.3 operationsUri

Type: URI

Required: false

Mutable: false

Consumer-mutable: false

This attribute contains a URI of the Operations resource listing the Operation resources available on this resource.

5.16.4 sensorsUri

Type: URI

Required: false

Mutable: false

Consumer-mutable: false

This attribute contains a URI of the Sensors resource listing the Sensor resources available on this resource.

5.17 ApplicationComponent Resource

An Application Component represents an instantiated Component at runtime. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "applicationComponent",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?,
  "assembly": Link,
  "applicationComponents": Link[] ?,
  "platformComponents": Link[] ?,
  "operationsUri": URI ?,
  "sensorsUri": URI ?
}
```

In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP DELETE method on instances of the ApplicationComponent resource. **[RE-62]** A successful DELETE request stops the underlying component, removes the ApplicationComponent resource from the system, and removes its reference from the `applicationComponents` array of its containing Assembly.

Instances of the ApplicationComponent resource contain the following attributes:

5.17.1 assembly

Type: Link

Required: true

Mutable: false

This attribute is a Link to the Assembly resource of which this ApplicationComponent is a member.

5.17.2 applicationComponents

Type: Link[]

Required: false

Mutable: false

This attribute is an array of Links to the ApplicationComponent resources that this ApplicationComponent depends on.

5.17.3 platformComponents

Type: Link[]

Required: false

Mutable: false

This attribute is an array of Links to the PlatformComponent resources that this ApplicationComponent depends on.

5.17.4 operationsUri

Type: URI

Required: false

Mutable: false

Consumer-mutable: false

This attribute contains a URI of the Operations resource listing the Operation resources available on this resource.

5.17.5 sensorsUri

Type: URI

Required: false

Mutable: false

Consumer-mutable: false

This attribute contains a URI of the Sensors resource listing the Sensor resources available on this resource.

5.18 PlatformComponent Resource

A Platform Component represents the runtime instance of a platform component and its configuration of component attributes as well as metrics associated with those attributes. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "platformComponent",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?,
  "externalManagementResource": URI ?
  "operationsUri": URI ?,
  "sensorsUri": URI ?
}
```

In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP DELETE method on instances of the PlatformComponent resource. **[RE-63]** A successful DELETE request stops the underlying component, removes the PlatformComponent resource from the system, and removes its reference from the Platform resource's `platformComponents` array.

Instances of the Platform Component resource contain the following attributes:

5.18.1 externalManagementResource

Type: URI

Required: false

Mutable: false

A URI to an external management interface to manage this platform component (such as an IaaS API to manage the virtual machines that support this component). This is platform dependent and requires external documentation to understand its meaning.

5.18.2 operationsUri

Type: URI

Required: false

Mutable: false

Consumer-mutable: false

This attribute contains a URI of the Operations resource listing the Operation resources available on this resource.

5.18.3 sensorsUri

Type: URI

Required: false

Mutable: false

Consumer-mutable: false

This attribute contains a URI of the Sensors resource listing the Sensor resources available on this resource.

5.19 Formats Resource

The Formats resource contains an array of Links to Format resources. It allows the identification of Supported Formats. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "formats",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?,
  "formatLinks": Link[]
}
```

The Formats resource contains the following attribute:

5.19.1 formatLinks

Type: Link[]

Required: true

Mutable: false

This attribute contains Links to Format resources that contain information about data serialization formats supported by the Platform. For every format that the Platform supports, there SHALL be a Format resource Link that represents such a format. **[RE-40]** The Required JSON Format Resource SHALL be listed first in the formatLinks array. **[RE-41]**

5.20 Format Resource

A Format resource represents exactly one supported data serialization format. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "format",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?,
  "mimeType": String,
  "version": String,
  "documentation": URI
}
```

Instances of the Format resource contain the following attributes:

5.20.1 mimeType

Type: String

Required: true

Mutable: false

This attribute contains the mime-type to be used by the Platform in HTTP [RFC2616] compliant content negotiation for this Format. For example: “application/json”.

5.20.2 version

Type: String

Required: false

Mutable: false

This attribute contains the version identifier of the data serialization format used.

5.20.3 documentation

Type: URI

Required: true

Mutable: false

5.20.4 Required JSON Format Resource

The Required JSON Format Resource is defined as:

```
{
  "uri": URI,
  "name": "JSON",
  "type": "format",
  "description": "JavaScript Object Notation",
  "tags": [ String, + ], ?
  "mimeType": "application/json",
  "version": "RFC4627",
  "documentation": "http://www.ietf.org/rfc/rfc4627.txt",
  "representationSkew": String ?
}
```

The *name*, *mimeType*, *version*, and *documentation* attribute values for the JSON Format Resource SHALL reflect the above values. **[RE-42]**

5.21 TypeDefinitions Resource

This resource contains an array of Links to all TypeDefinition resources. The Platform resource SHALL provide a Link to the TypeDefinitions resource in the required attribute named typeDefinitionsUri. **[RE-43]** This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "typeDefinitions",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?,
  "typeDefinitionLinks": Link[]
}
```

The TypeDefinitionss resource contains the following attribute:

5.21.1 typeDefinitionLinks

Type: Link[]

Required: true

Mutable: false

This attribute contains Links to TypeDefinition resources that contain information about resource types supported by the Platform. If the Platform does not extend this specification to add new resource types then the array can be empty. If the array is non-empty, for every resource type that the Platform supports, there SHALL be a TypeDefinition resource Link that represents such a resource type. **[RE-44]**

5.22 TypeDefinition Resource

A TypeDefinition resource represents exactly one resource type supported by the Platform. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "typeDefinition",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?,
  "documentation": URI,
  "attributeDefinitionLinks": Link[]
}
```

Instances of the TypeDefinition resource contain the following attributes:

5.22.1 documentation

Type: URI

Required: true

Mutable: false

This attribute contains a URI that points to the documentation for the resource type. For resource types that are defined in this Specification, the URI can point to this Specification.

5.22.2 attributeDefinitionLinks

Type: Link[]

Required: true

Mutable: false

This attribute contains an array of Links. Each Link in this array points to an AttributeDefinition resource. Each of these AttributeDefinition resources represents an attribute of the type represented by the Type resource. For every attribute of the type, there SHALL be an AttributeDefinition resource Link that represents the attribute. **[RE-45]** For more information on the AttributeDefinition resource see the next Section.

5.23 AttributeDefinition Resource

An AttributeDefinition resource represents exactly one supported attribute of one or more resource types. This resource has the following, general representation:


```
{
  "uri": URI,
  "name": String,
  "type": "attributeDefinition",
  "description": String ?,
  "representationSkew": String ?,
  "tags": String[] ?,
  "documentation": URI,
  "attributeType": String,
  "required": Boolean,
  "mutable": Boolean,
  "consumerMutable": Boolean
}
```

Instances of the AttributeDefinition resource contain the following attributes:

5.23.1 documentation

Type: URI

Required: true

Mutable: false

This attribute contains a URI that points to the documentation for the attribute that this resource represents. For attributes that are defined in this Specification, the URI can point to this Specification.

5.23.2 attributeType

Type: String

Required: true

Mutable: false

This attribute specifies the type of the attribute that this resource represents. For example, "String", "Timestamp".

5.23.3 required

Type: Boolean

Required: true

Mutable: false

This attribute specifies if the attribute that this resource represents is required.

5.23.4 mutable

Type: Boolean

Required: true

Mutable: false

This attribute specifies the mutability of the attribute that this resource represents.

5.23.5 consumerMutable

Type: Boolean

Required: true

Mutable: false

This attribute specifies if the attribute this resource represents is writable by a CAMP client.

5.24 ParameterDefinitions Resource

A ParameterDefinitions resource represents a collection of supported parameters for a particular resource. Multiple resources MAY reference the same ParameterDefinitions Resource. **[RE-46]** This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "parameterDefinitions",
  "description": String ?,
  "tags": String[] ?,
  "representationSkew": String ?,
  "parameterDefinitionLinks": Link[]
}
```

Instances of the ParameterDefinitions resource contain the following attributes:

5.24.1 parameterDefinitionLinks

Type: Link[]

Required: true

Mutable: true

This attribute is an array of Links to ParameterDefinition resources. Each Link in this array refers to one ParameterDefinition resource.

5.25 ParameterDefinition Resource

A ParameterDefinition resource represents exactly one supported parameter of one or more resource type. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "parameterDefinition",
  "description": String ?,
  "tags": String[] ?,
  "representationSkew": String ?,
  "parameterType": String,
  "required": Boolean,
  "defaultValue": String,
  "parameterExtensionUri": String ?
}
```

Instances of the ParameterDefinition resource contain the following attributes:

5.25.1 parameterType

Type: String

Required: true

Mutable: false

This attribute specifies the type of the attribute that this resource represents. For example, "String", "Timestamp".

5.25.2 required

Type: Boolean

Required: true

Mutable: false

This attribute specifies if the parameter that this resource represents is required.

5.25.3 defaultValue

Type: String

Required: false

Mutable: false

This attribute specifies the default value for this parameter, when present.

5.25.4 parameterExtensionUri

Type: URI

Required: false

Mutable: false

If this parameter is handled by an extension, this attribute refers to the Extension Resource that represents that extension and documents how the parameter is handled.

5.26 Operations Resource

An Operations resource represents a collection of Operation resources available on a target resource. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "operations",
  "description": String ?,
  "tags": String[] ?,
  "representationSkew": String ?,
  "targetResource": URI,
  "operationLinks": Link[]
}
```

Instances of the Operations resource contain the following attributes:

5.26.1 targetResource

Type: URI

Required: true

Mutable: false

This attribute indicates the CAMP resource on which the linked operations are invoked. Linked operations are those referred to by the operationLinks attribute. We use the term “target resource” to identify the resource referred to by this attribute.

5.26.2 operationLinks

Type: Link[]

Required: true

Mutable: false

This attribute contains Links to the Operation resources available on the target resource.

5.27 Operation Resource

An Operation resource represents exactly one operation or action available on a target resource. This resource has the following, general representation:


```
{
  "uri": URI,
  "name": String,
  "type": "operation",
  "description": String ?,
  "tags": String[] ?,
  "representationSkew": String ?,
  "documentation": URI,
  "targetResource": URI
}
```

In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP POST method on instances of the Operation resource. **[RE-64]**

A POST request on the Operation resource invokes the operation on the target resource. The Operation MAY require content in the body of the POST, such as parameters. **[RE-47]** The response to a POST request on an Operation resource SHOULD indicate what changes were made on the target resource. **[RE-48]** For asynchronous operations, the response SHOULD indicate how to track the progress of the request operation. **[RE-49]**

NOTE: For asynchronous operations, a Provider can accept a webhook URL from the Consumer as a parameter to the Operation POST request, and notify the client at that URL upon completion of the operation. It can also allow for polling of the resource to indicate completion.

Instances of the Operation resource contain the following attributes:

5.27.1 name

Type: String

Required: true

Mutable: false

This attribute contains the name of the operation that this resource represents. For example, "deploy" or "resize".

5.27.2 documentation

Type: URI

Required: true

Mutable: false

This attribute contains a URI of documentation for the operation this resource represents. The documentation SHOULD describe the behavior of the operation, the form of the body expected in POST requests, and the semantics and form of the response to such requests. **[RE-50]**

5.27.3 targetResource

Type: URI

Required: true

Mutable: false

This attribute indicates the CAMP resource on which the linked operation is invoked.

5.28 Sensors Resource

A Sensors resource represents a collection of Sensor resources available on a target resource. This resource has the following, general representation:


```
{
  "uri": URI,
  "name": String,
  "type": "sensors",
  "description": String ?,
  "tags": String[] ?,
  "representationSkew": String ?,
  "targetResource": URI,
  "sensorLinks": Link[]
}
```

Instances of the Sensors resource contain the following attributes:

5.28.1 targetResource

Type: URI

Required: true

Mutable: false

This attribute indicates the CAMP resource for which the linked sensors supply runtime data. Linked sensors are those referred to by the sensorLinks attribute. We use the term “target resource” to identify the resource referred to by this attribute.

5.28.2 sensorLinks

Type: Link[]

Required: true

Mutable: false

This attribute contains Links to the Sensor resources available on the target resource.

5.29 Sensor Resource

A Sensor resource represents exactly one supported sensor on one or more resources. Sensor resources represent dynamic data about resources, such as metrics or state. Sensor resources are useful for exposing data that changes rapidly, or that may need to be fetched from a secondary system. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "sensor",
  "description": String ?,
  "tags": String[] ?,
  "representationSkew": String ?,
  "documentation": URI,
  "targetResource": URI,
  "sensorType": String,
  "value": <sensorType> ?,
  "timestamp": Timestamp ?,
  "operationsURI": URI ?
}
```

Instances of the Sensor resource contain the following attributes:

5.29.1 documentation

Type: URI

Required: true

Mutable: false

This attribute contains a URI that points to the documentation for the sensor this resource represents.

5.29.2 targetResource

Type: URI

Required: true

Mutable: false

This attribute indicates the CAMP resource for which this Sensor resource supplies runtime data.

5.29.3 sensorType

Type: String

Required: true

Mutable: false

This attribute specifies the type of the data that this Sensor resource collects. For example, "String", "Timestamp". Common Types are defined in Section 5.1, “Common Types”. TypeDefinitions may also be used to specify types. See Section 5.21, “TypeDefinitions Resource”.

5.29.4 value

Type: As defined in *sensorType*

Required: false

Mutable: false

This attribute contains the current or most recent available value for this sensor. It can be omitted, for example, to indicate that no current value is available; either because no data has been collected or the collected data is stale.

5.29.5 timestamp

Type: Timestamp

Required: false

Mutable: false

This attribute contains the timestamp of the last collection or relevant activity of the sensor. When a “value” attribute is supplied, any timestamp provided in this attribute SHOULD correspond to when that value was observed. **[RE-51]**

5.29.6 operationsUri

Type: URI

Required: false

Mutable: false

This attribute contains URI of the Operations resource listing the Operation resources available for this resource.

Extensions MAY be defined to govern common sensor management operations, such as enabling, disabling, adjusting collection frequency, specifying the history of values which should be remembered, or collecting immediately. **[RE-52]**

6 Protocol

6.1 Transfer Protocol

The CAMP API is based on the Hypertext Transfer Protocol, version 1.1 [RFC2616]. Each request will be authenticated using HTTP Basic Authentication [RFC2617] unless otherwise noted. Therefore, requests sent from Consumers across unsecured networks SHOULD use the HTTPS protocol. **[PR-40]** TLS 1.1 [RFC4346] SHALL be implemented by the Provider. **[PR-41]** TLS 1.2 [RFC5246] is RECOMMENDED. **[PR-42]** When TLS is implemented, the following cipher suites are RECOMMENDED to ensure a minimum level of security and interoperability between implementations:

- TLS_RSA_WITH_AES_128_CBC_SHA (mandatory for TLS 1.1/1.2) **[PR-43]**
- TLS_RSA_WITH_AES_256_CBC_SHA256 (addresses 112-bit security strength requirements) **[PR-44]**

6.2 URI Space

The resources in the system are identified by URIs. Dereferencing the URI will yield a representation of the resource containing resource attributes and links to associated resources.

Consumers SHALL NOT make assumptions about the layout of the URIs or the structure of the URIs of the resources. **[PR-45]**

6.3 Media Types

6.3.1 Required Formats

In this specification, resource representations, request bodies, and error response messages are encoded in JSON, as specified in [RFC4627]. The media-type associated with CAMP JSON resource and error response message representations is "application/json".

Providers SHALL provide representations of all available resources in JSON. **[PR-01]**

6.3.1.1 Duplicate Keys in JSON Objects

Duplicate keys in JSON objects are allowed by **[RFC4627]**. This specification prohibits duplicate keys for interoperability reasons. Both Consumers and Providers SHALL NOT transmit JSON objects that contain duplicate keys. **[PR-02]** If a Consumer sends a Provider a request containing duplicate keys in a JSON object, the Provider SHOULD reject the request by sending back a ‘400 Bad Request’ status code. **[PR-03]** If a Provider sends a Consumer a response containing duplicate keys in a JSON object, the Consumer SHOULD raise an error to the user indicating the response from the server was malformed. **[PR-04]**

6.3.2 Supported Formats

If Supported Formats besides JSON are referenced in the supportedFormatsUri attribute of the Platform resource then resource representations, request bodies, and error response messages are allowed in the Supported Formats.

Supported Formats SHALL be applied uniformly for all resources defined by this specification. **[PR-05]**

A client can request any Supported Format using HTTP content negotiation.

6.4 Request Headers

This API does not impose any requirements on clients’ use of HTTP headers. All PUT requests that update a resource SHOULD contain the If-Match header field with a single entity tag value. **[PR-06]** If the

If-Match header field value in the request does not match the one on the server-side, the Provider SHALL send back a '412 Precondition Failed' status code. **[PR-07]**

6.5 Request Parameters

To retrieve a subset of the attributes in a resource, the Consumer MAY use the 'SelectAttr' request parameter in conjunction with the HTTP GET method. **[PR-08]**

Format	Description	Example
?SelectAttr=attr1,attr2,…	Comma (",") separated attribute names of the resource to return. If an attribute is not part of the resource, an HTTP 4XX status code SHALL be returned. **[PR-09]**	Assembly132?SelectAttr=name,description,tags Would access only "name", "description", "tags" attributes of Assembly132.

Table 6-1: Request Parameters

The "SelectAttr" query parameter MAY appear more than once (separated by an "&"). **[PR-10]** The Consumer SHALL URL encode the request parameters. **[PR-11]**

When one or more request parameters are specified for a PUT request, a Consumer SHALL NOT include attributes in the request entity body that are not specified in the request parameter. **[PR-12]** Upon receiving such a request the Provider SHALL respond with a 400 status code. **[PR-13]**

6.6 POST Body Parameters

Parameters MAY be included when performing a POST request on any resource with a parameterDefinitionsUri attribute defined. **[PR-14]** Supported parameters are defined by the parameterDefinitions resource referenced by the parameterDefinitionsUri attribute of the resource handling the POST request.

Example of a POST Parameter:

```
POST /<assembly-template-resource-url> HTTP/1.1
Host: example.org
Content-Type: application/json
Content-Length: ...

{ "EXAMPLE:someParameter": "bar" }

HTTP/1.1 201 Created
Location: http://example.org/paas/assembly/1
Content-Type: ...
Content-Length: ...
```

6.6.1 Parameter Handling

Parameters allow customizing Resources upon creation. Parameters MAY have the same name as an Attribute on the Resource. **[PR-15]** In such cases the Provider SHOULD set the Attribute to take the value of the Parameter OR clearly document alternate behavior. **[PR-16]** The parameterExtensionUri MAY be used to reference the Extension which documents how the parameter is handled. **[PR-17]**

If a POST request body does not contain a value for a required parameter, a "400 Bad Request" response SHALL be returned with an Error Response Message Resource. **[PR-18]**

If a POST request body does not contain an acceptable value for a parameter, a "400 Bad Request" response SHALL be returned with an Error Response Message Resource. **[PR-19]**

6.7 Response Headers

Responses returned by the Provider make standard use of HTTP headers. All HTTP responses that return representation of a resource SHOULD use strong Etag response header field indicating the current value of the entity tag for the resource. **[PR-20]**

6.8 HTTP Status Codes

The API returns standard HTTP response codes.

6.9 Mutability of Resource Attributes

Consumers SHALL NOT send a request that changes the value of a resource attribute that is declared with a constraint of 'Mutable=false' or 'Consumer-mutable=false'. **[PR-21]** On receiving such a request the Provider SHALL generate an HTTP response with 403 HTTP status code. **[PR-22]**

6.10 Updating Resources

Attributes of the resources defined with "consumerMutable: true" can be modified by Consumers in two ways. Consumers MAY use the HTTP PUT method to replace the representation of a resource, in its entirety, with a new representation that adds, omits or replaces the values for some of the attributes. **[PR-23]** Alternatively, Consumers MAY use the [HTTP PATCH] method and the "application/json-patch" media type [JSON Patch] to add, delete, or replace specific attributes. **[PR-24]**

6.10.1 Updating with PUT

HTTP PUT requests are requests for complete replacement of the resource identified by the request URL. If a resource attribute is present on a resource and if an HTTP PUT request omits that attribute, it SHOULD be treated by the Provider as a request to delete the attribute. **[PR-25]**

6.10.2 Updating with JSON Patch

JSON Patch [JSON Patch] defines a JSON document structure for expressing a sequence of operations to apply to a JSON document, suitable for use with the HTTP PATCH method. The "application/json-patch" media type is used to identify such patch documents.

Providers SHALL support the HTTP PATCH method in conjunction with the "application/json-patch" media type with the following, additional provisions with respect to the operations defined in section 4 of the JSON Patch specification [JSON Patch]: **[PR-26]**

- Providers SHALL support the 'add', 'remove', and 'replace' operations. **[PR-27]**
- Providers MAY support the 'move', 'copy, and 'test' operations. **[PR-28]**

6.11 Registering an Application

As indicated in Section 3.2,"Creating an Assembly Template from a PDP", registering an application moves it to the deployed state. There are two ways to register a PDP: by POSTing the entire PDP to the Platform resource (by value) or by POSTing the URI of the PDP to the Platform resource (by reference). Similarly, there are two ways to register an application using a DP: by POSTing the entire DP to the Platform resource (by value) or by POSTing the URI of the DP to the Platform resource (by reference). All of these methods are described below. A Provider supports registering a PDP using the ZIP [ZIP], TAR [TAR], and GZIP [GZIP] compressed TAR format. The media types associated with the various formats SHALL be as follows:

- ZIP: "application/x-zip" **[PR-29]**
- TAR: "application/x-tar" **[PR-30]**
- GZIP compressed TAR: "application/x-tgz" **[PR-31]**

Since the DP is a YAML file, when registering an application using a DP the associated media type SHALL be "application/x-yaml". **[PR-32]**

6.11.1 Registering an Application by Reference

To register an application by reference, a Consumer sends a POST HTTP request to the Platform URL. The entity body of the request contains the URI that identifies the PDP or the DP that is being registered. If the URI that identifies the PDP or the DP is a relative URI, its base URI is the Platform URI. When registering a PDP the JSON serialization of the HTTP request entity body is:

```
{"pdpUri" : "<uri-of-the-pdp>"}
```

When registering a DP the JSON serialization of the HTTP request entity body is:

```
{"dpUri" : "<uri-of-the-dp>"}
```

Where, the value of *pdpUri* is the URI of the PDP to be registered and the value of *dpUri* is the URI of the DP to be registered. The JSON object MAY contain additional name-value pairs that are not defined in this specification. **[PR-33]** On successful registration of the application, the Provider creates an AssemblyTemplate resource and sends a 201 Created HTTP status code with the *Location* header in the HTTP response. The *Location* header points to the newly created AssemblyTemplate resource. The Provider also updates the *assemblyTemplates* attribute of the Platform resource to include a reference to the newly created AssemblyTemplate. When a PDP is used to register the application, the Provider SHALL include the pdpUri attribute, which identifies the PDP from which the template was created, in the newly created AssemblyTemplate resource. **[PR-34]** When a DP is used to register the application, the Provider SHALL include the dpUri attribute, which identifies the DP from which the template was created, in the newly created AssemblyTemplate resource. **[PR-35]**

An example HTTP request-response is as follows:

```
POST /<platform-url> HTTP/1.1
Host: example.org
Content-Type: application/json
Content-Length: ...

{"pdpUri": "/paas/pdp/1"}

HTTP/1.1 201 Created
Location: http://example.org/paas/asm_template/1
Content-Type: ...
Content-Length: ...

...
```

6.11.2 Registering an Application by Value

To register an application by value, a Consumer sends a POST HTTP request to the Platform URL. The entity body of the request contains the PDP or the DP that is being registered. On successful registration of the PDP, the Provider creates an AssemblyTemplate resource and sends a 201 Created HTTP status code with the *Location* header in the HTTP response. The *Location* header points to the newly created AssemblyTemplate resource. The Provider also updates the *assemblyTemplates* attribute of the Platform resource to include a reference to the newly created AssemblyTemplate. When a PDP is used to register the application, the Provider SHALL include the pdpUri attribute, which identifies the PDP from which the template was created, in the newly created AssemblyTemplate resource. **[PR-36]** When a DP is used to register the application, the Provider SHALL include the dpUri attribute, which identifies the DP from which the template was created, in the newly created AssemblyTemplate resource. **[PR-37]**

For example, the *pdpUri* can point to an entry in the repository used by the Platform to manage its deployment artifacts.

An example HTTP request-response is as follows:


```
POST /<platform-url> HTTP/1.1
Host: example.org
Content-Type: application/x-zip
Transfer-Encoding: chunked
Content-Transfer-Encoding: binary
...

... binary PDP ZIP octets ...

HTTP/1.1 201 Created
Location: http://example.org/paas/asm_template/12
Content-Type: application/json

...
```

For large PDPs, the Consumer can use existing HTTP facilities like chunked transfer encoding. Please note that, unlike registration by reference, it is not possible to include additional parameters when registering by value.

6.12 Instantiating an Application

Once the application is in the deployed state, a Consumer can instantiate the application by sending a POST HTTP request to the corresponding AssemblyTemplate URL. The entity body of the request can be empty. Interpretation of a non-empty entity body of the request is implementation-dependent. On success the server creates an Assembly resource and sends a 201 Created HTTP status code with the *Location* header in the HTTP response. The *Location* header points to the newly created Assembly resource. The server also updates the *AssemblyInstances* attribute of the Platform resource to include a reference to the newly created assembly.

An example HTTP request-response is as follows:

```
POST /paas/asm_template/1 HTTP/1.1
Host: example.org

HTTP/1.1 201 Created
Location: .org/paas/assembly/1
Content-Type: ...
Content-Length: ...

...
```

6.13 Suspending and Resuming an Application

To suspend, or resume an application, a client sends a POST HTTP request to the assembly resource URL. The entity body of the request contains the value of the new state for the application. The JSON serialization of the HTTP request entity body is:

```
{"new_state" : "<new-state-value>"}
```

Where, *new_state* specifies the new desired value for the application state. This specification defines two such values: "suspend", and "resume," whose semantics are as defined in Section 0. A Provider MAY have additional state values that it allows. **[PR-38]** The JSON object MAY contain additional name-value pairs that are not defined in this specification. **[PR-39]** An implementation can define additional state values or name-value pairs, to allow clients to specify the scale at which the application is suspended and resumed.

An example HTTP request-response is as follows:


```
POST /<assembly-resource-url> HTTP/1.1
Host: example.org
Content-Type: application/json
Content-Length: ...

{"new_state": "suspend"}

HTTP/1.1 200 OK
```

6.14 Deleting an Application Instance and a Deployed Application

To delete an application instance (an Assembly), a client sends a DELETE HTTP request to the Assembly resource URL. Similarly, to delete a deployed application, a client sends a DELETE HTTP request to the Assembly template URL.

7 Extensions

Features provided by this specification can be extended to provide additional information and functionality. Using Requirements and Capabilities is RECOMMENDED instead of Extensions, if possible. **[EX-01]** Extensions MAY be added by registering the new functionality in the Extensions resource. **[EX-02]** Extensions SHALL NOT change or remove any features or functionality of this specification. **[EX-03]** Each Extension SHALL satisfy all criteria in the Conformance section and SHALL NOT contradict any normative statements in this document. **[EX-04]** The following extensions are allowed:

Category	Functionality	Description
API Extension	New HTTP Request Verbs	Support for additional HTTP Request Verbs that are not used by this specification, such as HEAD.
API Extension	HTTP Header Handlers	Processing of specific HTTP headers provided by clients. For example, an API Extension may require an authentication token header.
API Extension	New Resources	Addition of new resources that MAY handle HTTP requests such as POST or PUT to create instantiations of a new resource type.
API Extension	New Resource Methods	Allow the creation of new methods or actions that may cause different sequences of state changes than happen by default.
PDP Extension	New Metadata in the PDP	Additional metadata provided in the PDP to allow for more sophisticated handling of the bundled data.
Resource Extension	New Resource Types	Addition of new resource types.
Resource Extension	New Resource Attributes	Addition of new attributes to existing resources.
Resource Extension	New States in any Application Lifecycle	Addition of new application states, such as an intermediate state between the states defined by the specification.

Table 7-1: Extension Categories and Functionality

7.1 Unique Name Requirement

Entities
• Resources

• Attributes • Methods • PDP Metadata Keys

Table 7-2: Entities

Entities are enumerated in Table 7-2. The Extension Developer SHALL use a unique name for new Entities within an existing namespace. **[EX-05]** Entities added by an Extension SHALL NOT interfere with names of existing entities, including any added by another Extension. **[EX-06]**

NOTE: Each resource has its own namespace. It is acceptable to create a resource named *example.org:Foo*, and another resource named *example.org:Bar*, where both resources have an attribute named *fooBar*.

The use of your registered ICAAN Internet domain name followed by a colon (":") character as a prefix to all your entity names is RECOMMENDED to comply with these requirements. **[EX-07]**

Example: New Attribute "foo" added by Example Organization

```
example.org:foo
```

Example: New Attribute "foo" added by Example Inc.

```
EXAMPLE-INC:foo
```

Extension Category	New Entity	Exception
API Extension	Adding HTTP Request Verbs	Unique name not required for HTTP verbs
API Extension	Adding HTTP Header Handlers	Unique name not required for HTTP headers

Table 7-3: Unique Name Exceptions

A unique name is not required for entities listed in Table 7-3.

NOTE: RFC-3986 identifies Unreserved Characters that may be used in a URI without any encoding. Percent-Encoding allows any character to be represented in a URI. Special characters such as ":" and "." have specific meanings in scripting languages such as JavaScript. Special characters must be properly escaped in order to use them as part of a name string. Your data serialization format may not escape all problematic characters, so you may need to add logic to your clients to escape special characters to enable interaction with an Extension.

7.2 Extensions Resource

The Extensions resource contains an array of Links to Extension resources. It allows the identification of Extensions. The Extensions resource is represented as:

```
{
  "uri": URI,
  "name": String,
  "type": "types",
  "description": String, ?
  "tags": String[], ?
  "representationSkew": String ?,
  "extensionLinks": Link[]
}
```

Instances of the Extensions contain the following attribute:

7.2.1 extensionLinks

Type: Link[]

Required: true

Mutable: false

This attribute contains Links to Extension resources that contain information about Extensions available on the Platform. For every Extension available, there SHALL be an Extension resource Link that represents the Extension. **[EX-08]** The Platform resource SHALL provide a Link to the Extensions resource in the required attribute named extensionsUri. **[EX-09]**

Example of an extensionLinks value:

```
[
  {
    "targetName" : "EXAMPLE:Auth",
    "href": "http://example.org/paas1/extension/1"
  },
  {
    "targetName" : "EXAMPLE:PDPforFooLang",
   "href" : "http://example.org/paas1/extension/2"
   }
…
]
```

7.3 Extension Resource

An Extension resource represents new functionality added to the Platform. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "extension",
  "description": String,
  "version": String,
  "documentation": URI ?
}
```

The Extension resource contains the following attributes:

7.3.1 version

Type: String

Required: true

Mutable: false

This attribute contains a string identifier of the version of this Extension.

7.3.2 documentation

Type: Link

Required: false

Mutable: false

This attribute is a Link to a human readable document that describes the Extension in depth.

7.4 Extending Existing Resources

New attributes MAY be added to an existing resource using an Extension if the Unique Name Requirement in 0 is met. **[EX-10]** A new resource type is not required in order to add new attributes.

Example of an Extended Extension Resource:


```
{
  "uri": URI,
  "name": String,
  "type": "extension",
  "description": String,
  "version": String,
  "documentation": URI ?,
  "acme.com:foo": String ?
}
```

Note that in the above example, the new attribute “acme.com:foo” was added, and the type attribute remained set to the original value “extension”.

8 Conformance

There are three conformance targets defined in this specification:

- CAMP Provider
- CAMP Consumer
- Platform Deployment Package

8.1 CAMP Provider

An implementation claiming to conform to the requirements of a CAMP Provider defined in this specification SHALL comply with all of the mandatory statements listed in Appendix C.1, "Mandatory Statements", related to CAMP Provider or Provider.

8.2 CAMP Consumer

An implementation claiming to conform to the requirements of a CAMP Consumer defined in this specification SHALL comply with all of the mandatory statements listed in Appendix C.1, "Mandatory Statements", related to CAMP Consumer or Consumer.

8.3 Platform Deployment Package

For a document to be a valid PDP, it SHALL comply with all mandatory statements listed in Appendix C.1, "Mandatory Statements", related to the PDP.

Appendix A. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Roshan Agrawal	Rackspace Hosting, Inc.
Michael Behrens	US Department of Defense (DoD)
Bhaskar Reddy Byreddy	Software AG, Inc.
Mark Carlson	Oracle
Martin Chapman	Oracle
Francesco D'Andria	Cloud4SOA
Jacques Durand	Fujitsu Limited
Panagiotis Gouvas	Cloud4SOA
Keith Grange	JumpSoft
Alex Heneveld	Cloudsoft Corporation Limited
Gershon Janssen	Individual Member
David Jilk	Standing Cloud, Inc.
Duncan Johnston-Watt	Cloudsoft Corporation Limited
Anish Karmarkar	Oracle
Tobias Kunze	Red Hat
Eugene Luster	US Department of Defense (DoD)
Ashok Malhotra	Oracle
Rich Miller	Cloudsoft Corporation Limited
Jeff Mischkinsky	Oracle
Adrian Otto	Rackspace Hosting, Inc.
Derek Palma	Vnomic
Gilbert Pilz	Oracle
Krishna Raman	Red Hat
Tom Rutt	Fujitsu Limited
Zhexuan Song	Huawei Technologies Co., Ltd.
Charles Tupitza	JumpSoft
Jeffrey West	Oracle
Prasad Yendluri	Software AG, Inc.

Appendix B. Glossary

Application – a set of components that act together to provide useful functions and are typically exposed as a service to Application end-users. An application is represented by different resources (e.g. Assembly Template, Assembly) throughout its lifecycle.

Application Component – a collection of code and/or, resources (optionally accompanied by metadata) that either provides a set of related services or functionality or contains a set of related information. Code examples include Ruby gems, Java libraries, and PHP modules. Examples of resources include data sets, identity sets (i.e. collections of user account and attribute information), and collections of graphical images.

Application Component Capability – a management resource that represents an Application Component's capabilities.

Application Component Requirement – a management resource that represents a requirement on an Application Component, expressed with attributes that may have value ranges.

Application Component Template - a management resource that represents an unrealized Application Component and includes a reference to the executable code as well as metadata for configuring the Application Component and referencing its platform and other components.

Application Development Environment (ADE) – a developer tool used to create an application (can be an offline tool installed locally or part of the platform offering itself).

Assembly – a management resource that represents a running application.

Assembly Template - a management resource that represents an unrealized Assembly and includes a reference to the Application Component Templates used within the Application as well as metadata for configuring the Application Component and referencing its platform and other components.

Deploy – the step of creating one or more management resources on the platform. Deployment can be done through the API for individual management resources (i.e via a POST to a URI), or can be done as part of the import of a Platform Deployment Package.

Deployment Plan - packaging management meta-data that provides a description of the artifacts that make up an application, the services that are required to execute or utilize those artifacts, and the relationship of the artifacts to those services.. Deployment Plans are an essential a required part of a Platform Deployment Package.

Extension - a systematic representation of additional features and functionality added by an Extension Developer.

Platform – The collection of management resources that constitute the consumer visible view of the Platform as a Service offering. The Platform management resource is an aggregation and discovery point for all the Applications and their dependencies currently deployed and running.

Platform as a Service (PaaS) - A type of cloud computing in which the service provider offers customers/consumers access to one or more instances of a running application computing platform or application service stack.

Platform Component – a management resource that represents an application's use of a realized and running Platform Component.

Platform Component Capability – a management resource that represents a Platform Component's capabilities.

Platform Component Requirement – a management resource that represents a requirement on a Platform Component, expressed with attributes that may have value ranges.

Platform Component Template - a management resource that represents an unrealized Platform Component and includes references to metadata for configuring an instance of that Platform Component.

Platform Deployment Package (PDP) - an archive of executable images, dependency descriptions and metadata (Management Resources serialized into a Deployment Plan) that can be used to move an Application and its Components from Platform to Platform, or between an Application Development

Environment and a Platform (e.g. a storefront application with component binaries, database images and all the configurations needed to install and run).

Supported Formats - one or more data serialization format for data representation. JSON format is required, but other data serialization formats are also allowed. The Platform resource identifies all Supported Formats in the optional supportedFormatsUri attribute. If the supportedFormatsUri attribute is absent from the Platform resource, then only JSON is supported.

Appendix C. Normative Statements

C.1 Mandatory Statements

Tag	Statement
[CO-01]	An Application Component Template SHALL be referenced by a single Assembly Template.
[CO-02]	An Assembly Template SHALL NOT be instantiated until all of its Application Component Templates are successfully instantiated.
[PDP-02]	A Provider SHALL support the following archive formats for a PDP: • A PDP as a ZIP archive [ZIP]
[PDP-03]	A Provider SHALL support the following archive formats for a PDP: • A PDP as a TAR archive [TAR]
[PDP-04]	A Provider SHALL support the following archive formats for a PDP: • A PDP as a GZIP [GZIP] compressed TAR archive
[PDP-10]	The format of the manifest file and the certificate file SHALL be as defined by the OVF specification [OVF].
[PDP-11]	A Platform Deployment Package (PDP) SHALL contain a single Deployment Plan.
[PDP-12]	The Deployment Plan SHALL be located at the root of the PDP archive.
[PDP-13]	The Deployment Plan file SHALL be named "`camp.yaml`" and SHALL consist of a well-formed YAML 1.1 [YAML 1.1] file that conforms to the description provided in this section.
[PDP-17]	A Deployment Plan SHALL contain a single instance of a DeploymentPlan node.
[PDP-18]	This value SHALL be the Specification Version String of the CAMP specification to which this Deployment Plan conforms.
[PDP-19]	For Deployment Plans that conform to this document, the value of this node SHALL be "CAMP 1.1" as defined in Section **Error! Reference source not found.** "Specification Version".
[PDP-20]	Providers SHALL NOT regard the order of the ArtifactSpecifications within this array as semantically significant.
[PDP-21]	Providers SHALL NOT treat the order of ServiceSpecifications within this array as semantically significant.
[PDP-23]	Providers SHALL NOT treat the order of RequirementSpecifications within this array as semantically significant.
[PDP-24]	Providers SHALL NOT treat the order of CharacteristicSpecifications within this array as semantically significant.
[PDP-25]	Content Specifications SHALL declare either a String attribute "href" that references the content or a String attribute "data" whose value is the data or, but not both.
[PDP-26]	For IANA-assigned URI schemes (e.g. "http", "https", "ftp", etc.) the Provider SHALL engage the protocol as per the relevant spec.

[PDP-27]	Providers SHALL support the “http” and “https” URI schemes.
[PDP-29]	Providers SHALL understand this delimiter and SHALL NOT resolve any content if the archive format is unsupported.
[RE-01]	The following attributes SHALL be present in a Link.
[RE-04]	Consumers SHALL NOT change the value of this attribute.
[RE-05]	Though both attributes of this type are optional, a valid RequirementTemplateLinks type SHALL have at least one of the following attributes:
[RE-06]	If the Required boolean constraint for an attribute of a resource type has a value of "true", then an instance of that resource type SHALL have the attribute present.
[RE-07]	This boolean indicates the mutability of the attribute’s value(s). “false” indicates that the value of the attribute, once set, SHALL NOT change for the lifetime of the resource.
[RE-09]	“false” indicates that the value(s) of the attribute SHALL NOT be changed by the Consumers.
[RE-11]	If present, `representationSkew` SHALL have one of the following values: • “CREATING” – describes a resource that is in the process of being created. The client can expect that the resource will have a skew of “NONE” once this process has completed. • “NONE” – is an assertion by the CAMP server that the information in the resource is an accurate representation of the underlying platform implementation. Absent some action by the client or some other event (e.g. platform shutdown), a resource with a skew of NONE can be expected to remain in synch with the platform implementation. • “UNKNOWN” – indicates that the CAMP server cannot accurately depict the aspect of the platform implementation represented by this resource. Users can attempt to address the underlying issues(s) by manipulating this and/or other resources as specified by the API. • “DESTROYING” – describes a resource that is in the process of being destroyed. The client can expect that the resource will cease to exist once this process has completed.
[RE-12]	The following table lists the methods that SHALL be supported for each `representationSkew` value.

representationSkew value	Methods Available
CREATING	GET, DELETE
NONE	All supported methods for that resource.
UNKNOWN	All supported methods for that resource.
DESTROYING	GET

[RE-14]	However, it SHALL NOT contradict the HTTP status code that is returned with the request.
[RE-16]	Because of the unique function of this resource, future versions of the CAMP specification SHALL NOT make non-backwards compatible changes to this resource.
[RE-18]	This array SHALL contain at least one PlatformEndpoint Resource.

[RE-19]	References between the resources (PlatformEndpoints, PlatformEndpoint, and Platform) SHALL be self-consistent.
[RE-20]	Each PlatformEndpoint Resource SHALL refer to exactly one Platform Resource, and indicate the versions supported by the Platform.
[RE-21]	Because of the unique function of this resource, future versions of the CAMP specification SHALL NOT make non-backwards compatible changes to this resource.
[RE-22]	For Platforms that implement this version of the CAMP specification, the value of this attribute SHALL be “CAMP 1.1” as defined in section 1.7.
[RE-23]	The values in this array SHALL be the Specification Version Strings of previous CAMP specification versions.
[RE-24]	PlatformEndpoint resources that reference Specification Version “CAMP 1.1” Platform Resources SHALL NOT include this attribute because no previous versions are compatible.
, “Specification Version”. **[RE-26]**	For Platforms that implement this version of the CAMP specification, the value of this attribute SHALL be “CAMP 1.1” as defined in Section 1.7, “Specification Version”.
[RE-27]	The value of this attribute SHALL exactly match the value of the specificationVersion attribute of any PlatformEndpoint resource that references this Platform Resource.
[RE-29]	The value of this attribute SHALL exactly match the value of the implementationVersion attribute of any PlatformEndpoint resource that references this Platform Resource.
[RE-30]	The Platform resource SHALL indirectly reference a parameterDefinition resource that describes the pdpUri parameter.
[RE-31]	Consumers SHALL NOT change the value of this attribute.
[RE-33]	but Providers SHALL NOT instantiate an Assembly using an AssemblyTemplate that does not reference at least one ApplicationComponentTemplate.
[RE-34]	The ParameterDefinitions resource referenced by this attribute SHALL define parameters to allow setting the ‘name’, ‘description’, and ‘tags’ attributes of any new resource created in the course of interacting with this resource.
[RE-35]	This attribute SHALL be present.
[RE-36]	The specified value for each component attribute SHALL be in the range defined in the corresponding Platform Component Capability.
[RE-37]	The ParameterDefinitions resource referenced by this attribute SHALL define parameters to allow setting the ‘name’, ‘description’, and ‘tags’ attributes of any new resource created in the course of interacting with this resource.
[RE-38]	If this attribute is present in an instance of this resource, Providers SHALL support the POST method on that instance in addition to the methods defined in Section 0, “HTTP Method Support”.
[RE-39]	An Assembly resource SHALL have at least one reference to an ApplicationComponent resource.
[RE-40]	For every format that the Platform supports, there SHALL be a Format resource Link that represents such a format.

[RE-41]	The Required JSON Format Resource SHALL be listed first in the formatLinks array.
[RE-42]	The name, mimeType, version, and documentation attribute values for the JSON Format Resource SHALL reflect the above values.
[RE-43]	The Platform resource SHALL provide a Link to the TypeDefinitions resource in the required attribute named `typeDefinitionsUri`.
[RE-44]	If the array is non-empty, for every resource type that the Platform supports, there SHALL be a TypeDefinition resource Link that represents such a resource type.
[RE-45]	For every attribute of the type, there SHALL be an AttributeDefinition resource Link that represents the attribute.
[RE-53]	Providers SHALL support the HTTP GET, PUT, and PATCH methods on all of the resources defined in this section.
[RE-55]	In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP POST method on the Platform resource as described in Section 6.11, "Registering an Application".
[RE-56]	In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP POST and DELETE methods on instances of the AssemblyTemplate resource.
[RE-57]	In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP DELETE method on instances of the ApplicationComponentTemplate resource.
[RE-58]	In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP DELETE method on instances of the ApplicationComponentRequirement resource.
[RE-59]	In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP DELETE method on instances of the ApplicationComponentCapability resource.
[RE-60]	In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP DELETE method on instances of the PlatformComponentRequirement resource.
[RE-61]	In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP DELETE methods on instances of the Assembly resource as described in Section 6.14, "Deleting an Application Instance and a Deployed Application".
[RE-62]	In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP DELETE method on instances of the ApplicationComponent resource.
[RE-63]	In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP DELETE method on instances of the PlatformComponent resource.
[RE-64]	In addition to the methods defined in Section 0, "HTTP Method Support", Providers SHALL support the HTTP POST method on instances of the Operation resource.
[RE-65]	Consumers and Providers SHALL express Timestamps in UTC (Coordinated Universal Time), with the special UTC designator ("Z").
[PR-01]	Providers SHALL provide representations of all available resources in JSON.

[PR-02]	Both Consumers and Providers SHALL NOT transmit JSON objects that contain duplicate keys.
[PR-05]	Supported Formats SHALL be applied uniformly for all resources defined by this specification.
[PR-07]	If the *If-Match* header field value in the request does not match the one on the server-side, the Provider SHALL send back a '412 Precondition Failed' status code.
[PR-09]	If an attribute is not part of the resource, an HTTP 4XX status code SHALL be returned.
[PR-11]	The Consumer SHALL URL encode the request parameters.
[PR-12]	When one or more request parameters are specified for a PUT request, a Consumer SHALL NOT include attributes in the request entity body that are not specified in the request parameter.
[PR-13]	Upon receiving such a request the Provider SHALL respond with a 400 status code.
[PR-18]	If a POST request body does not contain a value for a required parameter, a “400 Bad Request” response SHALL be returned with an Error Response Message Resource.
[PR-19]	If a POST request body does not contain an acceptable value for a parameter, a “400 Bad Request” response SHALL be returned with an Error Response Message Resource.
[PR-21]	Consumers SHALL NOT send a request that changes the value of a resource attribute that is declared with a constraint of 'Mutable=false' or 'Consumer-mutable=false'.
[PR-22]	On receiving such a request the Provider SHALL generate an HTTP response with 403 HTTP status code.
[PR-26]	Providers SHALL support the HTTP PATCH method in conjunction with the “application/json-patch” media type with the following, additional provisions with respect to the operations defined in section 4 of the JSON Patch specification [JSON Patch]:
[PR-27]	Providers SHALL support the ‘add’, ‘remove’, and ‘replace’ operations.
[PR-29]	The media types associated with the various formats SHALL be as follows: • ZIP: "application/x-zip"
[PR-30]	The media types associated with the various formats SHALL be as follows: • TAR: "application/x-tar"
[PR-31]	The media types associated with the various formats SHALL be as follows: • GZIP compressed TAR: "application/x-tgz"
[PR-32]	Since the DP is a YAML file, when registering an application using a DP the associated media type SHALL be "application/x-yaml".
[PR-34]	When a PDP is used to register the application, the Provider SHALL include the *pdpUri* attribute, which identifies the PDP from which the template was created, in the newly created AssemblyTemplate resource.
[PR-35]	When a DP is used to register the application, the Provider SHALL include the *dpUri* attribute, which identifies the DP from which the template was created, in the newly created AssemblyTemplate resource.
[PR-36]	When a PDP is used to register the application, the Provider SHALL include the *pdpUri* attribute, which identifies the PDP from which the template was created, in the newly created AssemblyTemplate resource.

[PR-37]	When a DP is used to register the application, the Provider SHALL include the *dpUri* attribute, which identifies the DP from which the template was created, in the newly created AssemblyTemplate resource.
[PR-41]	TLS 1.1 [RFC4346] SHALL be implemented by the Provider.
[PR-45]	Consumers SHALL NOT make assumptions about the layout of the URIs or the structure of the URIs of the resources.
[EX-03]	Extensions SHALL NOT change or remove any features or functionality of this specification.
[EX-04]	Each Extension SHALL satisfy all criteria in the Conformance section and SHALL NOT contradict any normative statements in this document.
[EX-05]	The Extension Developer SHALL use a unique name for new Entities within an existing namespace.
[EX-06]	Entities added by an Extension SHALL NOT interfere with names of existing entities, including any added by another Extension.
Error! Reference source not found.	For every Extension available, there SHALL be an Extension resource Link that represents the Extension.
Error! Reference source not found.	The Platform resource SHALL provide a Link to the Extensions resource in the required attribute named `extensionsUri`.

C.2 Non-Mandatory Statements

Tag	Statement
[PDP-01]	A PDP archive MAY include other files related to the application including, but not limited to, language-specific bundles, resource files, application content files such as web archives, database schemas, scripts, source code, localization bundles, and icons; and metadata files such as manifests, checksums, signatures, and certificates.
[PDP-05]	Providers MAY support additional archive formats for the PDP.
[PDP-06]	A PDP MAY contain a manifest file, named `camp.mf`, at the root of the archive.
[PDP-07]	A Provider SHOULD reject a PDP if any digest listed in the manifest does not match the computed digest for that file in the package.
[PDP-08]	A PDP MAY contain a certificate, named `camp.cert`, at the root of the archive.
[PDP-09]	A Provider SHOULD reject any PDP for which the signature verification fails.
[PDP-14]	Providers MAY reflect the value of this attribute in the names of any resources that are created in the processing the Deployment Plan.
[PDP-15]	Providers MAY reflect the value of this attribute in the descriptions of the resources that are in the processing the Deployment Plan.
[PDP-16]	Providers MAY reflect the values of this attribute in the tags of the resources that are created in the processing of the Deployment Plan.

[PDP-22]	The artifact MAY be contained within the PDP or MAY exist in some other location.
[PDP-28]	A Provider MAY support additional URI schemes.
[RE-02]	Other attributes, not defined in this specification, MAY also be present.
[RE-03]	The value of this attribute MAY be changed by the Provider.
[RE-08]	“true” indicates that the value of the attribute MAY change due to the actions or activity of either the provider or the consumer.
[RE-10]	A value of “true” indicates that consumers MAY change the value of the attribute.
[RE-13]	For each `representationSkew` value, CAMP Providers MAY support HTTP methods in addition to those listed in the corresponding row of Table 5-1.
[RE-15]	A Provider MAY concurrently offer multiple instances of the CAMP API.
[RE-17]	A Provider MAY expose the PlatformEndpoints and corresponding PlatformEndpoint resources in a way that allows for version discovery before the client has authenticated.
[RE-25]	Multiple implementations of the same CAMP specification MAY be offered concurrently.
[RE-28]	A Provider MAY choose to offer multiple implementations of the same CAMP specification.
[RE-32]	An AssemblyTemplate MAY have zero references to ApplicationComponentTemplate resources
Error! Reference source not found.	Multiple resources MAY reference the same ParameterDefinitions Resource.
[RE-47]	The Operation MAY require content in the body of the POST, such as parameters.
[RE-48]	The response to a POST request on an Operation resource SHOULD indicate what changes were made on the target resource.
[RE-49]	For asynchronous operations, the response SHOULD indicate how to track the progress of the request operation.
[RE-50]	The documentation SHOULD describe the behavior of the operation, the form of the body expected in POST requests, and the semantics and form of the response to such requests.
[RE-51]	When a “value” attribute is supplied, any timestamp provided in this attribute SHOULD correspond to when that value was observed.
[RE-52]	Extensions MAY be defined to govern common sensor management operations, such as enabling, disabling, adjusting collection frequency, specifying the history of values which should be remembered, or collecting immediately.
[RE-54]	Providers MAY elect to support additional HTTP methods in addition to those described here.
[PR-03]	If a Consumer sends a Provider a request containing duplicate keys in a JSON object, the Provider SHOULD reject the request by sending back a ‘400 Bad Request’ status code.
[PR-04]	If a Provider sends a Consumer a response containing duplicate keys in a JSON object, the Consumer SHOULD raise an error to the user indicating the response from the server was malformed.

[PR-06]	All PUT requests that update a resource SHOULD contain the *If-Match* header field with a single entity tag value.
[PR-08]	To retrieve a subset of the attributes in a resource, the Consumer MAY use the 'SelectAttr' request parameter in conjunction with the HTTP GET method.
[PR-10]	The "SelectAttr" query parameter MAY appear more than once (separated by an "&").
[PR-14]	Parameters MAY be included when performing a POST request on any resource with a parameterDefinitionsUri attribute defined.
[PR-15]	Parameters MAY have the same name as an Attribute on the Resource.
[PR-16]	In such cases the Provider SHOULD set the Attribute to take the value of the Parameter OR clearly document alternate behavior.
[PR-17]	The parameterExtensionUri MAY be used to reference the Extension which documents how the parameter is handled.
[PR-20]	All HTTP responses that return representation of a resource SHOULD use strong *Etag* response header field indicating the current value of the entity tag for the resource.
[PR-23]	Consumers MAY use the HTTP PUT method to replace the representation of a resource, in its entirety, with a new representation that adds, omits or replaces the values for some of the attributes.
[PR-24]	Alternatively, Consumers MAY use the [HTTP PATCH] method and the "application/json-patch" media type [JSON Patch] to add, delete, or replace specific attributes.
[PR-25]	If a resource attribute is present on a resource and if an HTTP PUT request omits that attribute, it SHOULD be treated by the Provider as a request to delete the attribute.
[PR-28]	Providers MAY support the 'move', 'copy, and 'test' operations.
[PR-33]	The JSON object MAY contain additional name-value pairs that are not defined in this specification.
[PR-38]	A Provider MAY have additional state values that it allows.
[PR-39]	The JSON object MAY contain additional name-value pairs that are not defined in this specification.
[PR-40]	Therefore, requests sent from Consumers across unsecured networks SHOULD use the HTTPS protocol.
[PR-42]	TLS 1.2 [RFC5246] is RECOMMENDED.
[PR-43]	When TLS is implemented, the following cipher suites are RECOMMENDED to ensure a minimum level of security and interoperability between implementations: • TLS_RSA_WITH_AES_128_CBC_SHA (mandatory for TLS 1.1/1.2)
[PR-44]	When TLS is implemented, the following cipher suites are RECOMMENDED to ensure a minimum level of security and interoperability between implementations: • TLS_RSA_WITH_AES_256_CBC_SHA256 (addresses 112-bit security strength requirements)
[EX-01]	Using Requirements and Capabilities is RECOMMENDED instead of Extensions, if possible.
[EX-02]	Extensions MAY be added by registering the new functionality in the Extensions resource.
[EX-07]	The use of your registered ICAAN Internet domain name followed by a colon (":") character

	as a prefix to all your entity names is RECOMMENDED to comply with these requirements.
[EX-10]	New attributes MAY be added to an existing resource using an Extension if the Unique Name Requirement in 0 is met.

Appendix D. Example Database Platform Component (Non-Normative)

One important Platform Component that can be provided by many platforms is a database. The following sections illustrate how database components could be provided by extending the model defined in this specification. The material in these sections is non-normative.

D.1 Model

A Database Platform Component provides four sub-classed resources as shown in the below diagram:

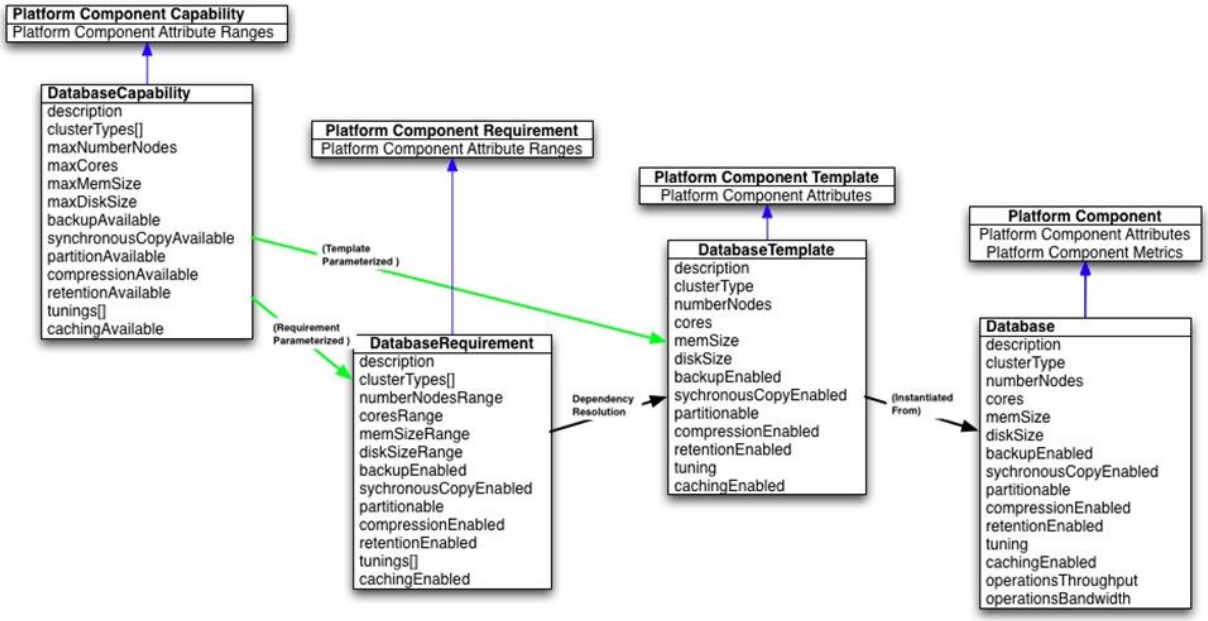

Figure D-1: Database Platform Component Model

D.2 DatabaseCapability

For an Application Administrator, a Database Capability represents the definition of a database platform component and its range of capabilities. This resource has the following, general representation:


```
{
  "uri": URI,
  "name": String,
  "type": "databaseCapability",
  "description": String,
  "representationSkew": String, ?
  "tags": [ String, + ], ?
  "clusterTypes": [
    String, +
  ],
  "maxNumberNodes": String,
  "maxCores": String,
  "maxMemSize": String,
  "maxDiskSize": String,
  "backupAvailable": Boolean,
  "synchronousCopyAvailable": Boolean,
  "partitionAvailable": Boolean,
  "compressionAvailable": Boolean,
  "retentionAvailable": Boolean,
  "tunings": [
    String, +
  ],
  "cachingAvailable": Boolean,
}
```

Each type of DatabasePlatformComponent implements this class and populates the attributes in the list below.

D.2.1 clusterTypes

Type: String[]

Required: true

Mutable: true

Consumer-mutable: false

An array of supported cluster types. Values include: “active” and “passive”.

D.2.2 maxNumberNodes

Type: String

Required: true

Mutable: true

Consumer-mutable: false

Expresses the maximum number of supported nodes for scaling purposes.

D.2.3 maxCores

Type: String

Required: true

Mutable: true

Consumer-mutable: false

Expresses the maximum number of supported cores for scaling purposes.

D.2.4 maxMemSize

Type: String

Required: true

Mutable: true

Consumer-mutable: false

Expresses the maximum size of supported memory for an instance.

D.2.5 maxDiskSize

Type: String

Required: true

Mutable: true

Consumer-mutable: false

Expresses the limit to the size of a disk for an instance.

D.2.6 backupAvailable

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: false

true if backup can be enabled.

D.2.7 synchronousCopyAvailable

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: false

true if synchronous copy can be enabled

D.2.8 partitionAvailable

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: false

true if partitioning can be enabled

D.2.9 compressionAvailable

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: false

true if compression can be enabled

D.2.10 retentionAvailable

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: false

true if retention can be enabled

D.2.11 tunings

Type: String[]

Required: true

Mutable: true

Consumer-mutable: false

An array of supported tuning types. Values include: “OLAP”, “DataMining”, and “Spatial”

D.2.12 cachingAvailable

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: false

true if caching can be enabled

D.3 DatabaseRequirement

For an Application Administrator, a Database Requirement represents an Application Component’s requirements on a database platform component and its required range of capabilities. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "databaseRequirement",
  "description": String,
  "representationSkew": String, ?
  "tags": [ String, + ], ?
  "clusterTypes": [ String, + ],
  "numberNodesRange": String,
  "coreRange": String,
  "memSizeRange": String,
  "diskSizeRange": String,
  "backupEnabled": Boolean,
  "synchronousCopyEnabled": Boolean,
  "partitionEnabled": Boolean,
  "compressionEnabled": Boolean,
  "retentionEnabled": Boolean,
  "tunings": [ String, + ],
  "cachingEnabled": Boolean
}
```

Each type of DatabaseRequirement implements this class and populates the attributes in the list below.

D.3.1 clusterTypes

Type: String[]

Required: true

Mutable: true

Consumer-mutable: true

An array of required cluster types. Values include: “active” and “passive”.

D.3.2 numberNodesRange

Type: String

Required: true

Mutable: true

Consumer-mutable: true

Expresses the range of the number of nodes for scaling purposes

D.3.3 coreRange

Type: String

Required: true

Mutable: true

Consumer-mutable: true

Expresses the required range of number of cores.

D.3.4 memSizeRange

Type: String

Required: true

Mutable: true

Consumer-mutable: true

Expresses the range of required memory sizes.

D.3.5 diskSizeRange

Type: String

Required: true

Mutable: true

Consumer-mutable: true

Expresses the range if disk sizes required.

D.3.6 backupEnabled

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: true

true if backup is required.

D.3.7 synchronousCopyEnabled

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: false

true if synchronous copy is required.

D.3.8 partitionEnabled

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: true

true if partitioning is required.

D.3.9 compressionEnabled

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: true

true if compression is required.

D.3.10 retentionEnabled

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: true

true if retention is required.

D.3.11 tunings

Type: String[]

Required: true

Mutable: true

Consumer-mutable: true

An array of required tuning types. Values include: “OLAP”, “DataMining” and “Spatial”.

D.3.12 cachingEnabled

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: true

true if caching is required.

D.4 DatabaseTemplate

A Database Template represents the desired configuration of a Database Platform Component with specific values for the component capabilities. The specified value for each component attribute shall be in the range defined in the corresponding Database Capability. Some PaaS offerings might only offer a fixed number of Database Template instances that cannot be modified (read-only, no new instances) indicating pre-tuned and configured pools of database resources from which these draw. Other PaaS offerings may allow newly created Database Template instances with values in any combination. This resource has the following, general representation:


```
{
  "uri": URI,
  "name": String,
  "type": "databaseTemplate",
  "description": String,
  "representationSkew": String, ?
  "tags": [ String, + ], ?
  "clusterType": String,
  "numberNodes": String,
  "cores": String,
  "memSize": String,
  "diskSize": String,
  "backupEnabled": Boolean,
  "synchronousCopyEnabled": Boolean,
  "partitionEnabled": Boolean,
  "compressionEnabled": Boolean,
  "retentionEnabled": Boolean,
  "tuning": String,
  "cachingEnabled": Boolean
}
```

Each type of DatabaseTemplate implements this class and populates the attributes in the list below.

D.4.1 clusterType

Type: String

Required: true

Mutable: true

Consumer-mutable: false

The desired cluster type. Values include: “active” and “passive”

D.4.2 numberNodes

Type: String

Required: true

Mutable: true

Consumer-mutable: false

The desired number of nodes for an instance

D.4.3 cores

Type: String

Required: true

Mutable: true

Consumer-mutable: false

The desired number of cores for an instance

D.4.4 memSize

Type: String

Required: true

Mutable: true

Consumer-mutable: false

The desired size of memory for an instance

D.4.5 diskSize

Type: String

Required: true

Mutable: true

Consumer-mutable: false

The desired size of a disk for an instance.

D.4.6 backupEnabled

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: false

true if backup is desired for an instance.

D.4.7 synchronousCopyEnabled

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: false

true if synchronous copy is desired for an instance.

D.4.8 partitionEnabled

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: false

true if partitioning is desired for an instance

D.4.9 compressionEnabled

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: false

true if compression is desired for an instance

D.4.10 retentionEnabled

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: false

true if retention is desired for an instance

D.4.11 tuning

Type: String

Required: true

Mutable: true

Consumer-mutable: false

The desired tuning types. Values include: “OLAP”, “DataMining” and “Spatial”

D.4.12 cachingEnabled

Type: Boolean

Required: true

Mutable: true

Consumer-mutable: false

true if caching is desired for an instance

D.5 Database

A Database represents the runtime instance of a Database Template and its configuration of component attributes as well as metrics associated with those attributes. Each Application’s use of a Database is represented by an instance of this resource. This resource has the following, general representation:

```
{
  "uri": URI,
  "name": String,
  "type": "database",
  "description": String,
  "representationSkew": String, ?
  "tags": [ String, + ], ?
  "externalManagementResource": String,
  "clusterType": String,
  "numberNodes": String,
  "cores": String,
  "memSize": String,
  "diskSize": String,
  "backupEnabled": Boolean,
  "synchronousCopyEnabled": Boolean,
  "partitionEnabled": Boolean,
  "compressionEnabled": Boolean,
  "retentionEnabled": Boolean,
  "tuning": String,
  "cachingEnabled": Boolean,
  "operationsThroughput": String,
  "operationsBandwidth": String
}
```

Each type of Database implements this class and populates the attributes in the list below.

D.5.1 externalManagementResource

Type: URI

Required: false

Mutable: false

A URI to an external management interface to manage this platform component (such as an IaaS API to manage the virtual machines that make up this database). This is platform dependent and requires external documentation to understand its meaning.

D.5.2 clusterType

Type: String

Required: true

Mutable: false

The actual cluster type. Values include: “active” and “passive”

D.5.3 numberNodes

Type: String

Required: true

Mutable: false

The actual number of nodes for an instance.

D.5.4 cores

Type: String

Required: true

Mutable: false

The actual number of cores for an instance.

D.5.5 memSize

Type: String

Required: true

Mutable: false

The actual size of memory for an instance.

D.5.6 diskSize

Type: String

Required: true

Mutable: false

The actual size of a disk for an instance.

D.5.7 backupEnabled

Type: Boolean

Required: true

Mutable: false

true if backup is enabled for this instance.

D.5.8 synchronousCopyEnabled

Type: Boolean

Required: true

Mutable: false

true if synchronous copy is enabled for this instance.

D.5.9 partitionEnabled

Type: Boolean

Required: true

Mutable: false

true if partitioning is enabled for this instance.

D.5.10 compressionEnabled

Type: Boolean

Required: true

Mutable: false

true if compression is enabled for this instance.

D.5.11 retentionEnabled

Type: Boolean

Required: true

Mutable: false

true if retention is enabled for this instance.

D.5.12 tuning

Type: String

Required: true

Mutable: false

The actual tuning type of the database instance. Values include: “OLAP”, “DataMining” and “Spatial”

D.5.13 cachingEnabled

Type: Boolean

Required: true

Mutable: false

true if caching is enabled for this instance.

D.5.14 operationsThroughput

Type: String

Required: true

Mutable: true

Consumer-mutable: false

The billable operations per second.

D.5.15 operationsBandwidth

Type: String

Required: true

Mutable: true

Consumer-mutable: false

The billable MegaBytes per second.

Appendix E. Revision History

Revision	Date	Editor	Changes Made
1	2012-12-04	Anish Karmarkar	Applied OASIS template to version 1.0
2	2012-12-18	Anish Karmarkar	Included resolutions for issues 7, 12, 13, 15, 24, 33.
3	2013-02-05	Anish Karmarkar	Included resolutions for issues 2, 6, 10, 14, 25, 35
4	2013-02-12	Adrian Otto	Included resolutions for 19, 38
5	2013-02-13	Adrian Otto	Included resolutions for 1, 49
6	2013-02-27	Adrian Otto	Included resolutions for 36, 48, 53
7	2013-02-27	Adrian Otto	Included resolutions for 34, 52
8	2013-03-13	Adrian Otto	Included resolution for 40
9	2013-04-29	Anish Karmarkar	Included resolutions for 50, 57
10	2013-05-01	Adrian Otto	Included resolution for 30
11	2013-06-05	Adrian Otto	Included resolution for 58
12	2013-06-13	Gilbert Pilz, Tom Rutt	Included resolutions for issues 17, 67 and 68
13	2013-06-27	Gilbert Pilz	Included resolution for 3.
14	2013-07-01	Tom Rutt	Updated Figures for 3, kept revision marks from 12
15	2013-07-08	Gilbert Pilz	Included resolution for 4.
16	2013-07-10	Gilbert Pilz, Tom Rutt	Included resolution for 9. Miscellaneous, non-material changes and cleanups.
17	2013-07-11	Gilbert Pilz, Tom Rutt	Included resolution for 65. Miscellaneous, non-material changes and cleanups.
18	2013-07-19	Gilbert Pilz, Tom Rutt	Includes resolution for 55. Miscellaneous, non-material changes and cleanups.
19	2013-07-25	Gilbert Pilz	Included resolutions for 45, 56, 61, 75, 76, and 78. Miscellaneous, non-material changes and cleanups.
20	2013-07-26	Gilbert Pilz	Added names to Appendix A, “Acknowledgements”. Tagged normative statements that were missed in WD19. Homogenized captions for figures and tables. Homogenized and corrected cross-references. Miscellaneous editorial cleanups.

