Lifecycle Manager API Management Interface

Lifecycle Manager

API Management Interface
Version 2019.1.20
September, 2020

Perforce Software

Perforce Software
400 First Avenue North #200
Minneapolis, MN 55401
612.517.2100
http://www.perforce.com
info@perforce.com

Contents

Chapter 1 | Overview

The Lifecycle Manager product provides the capability to automate and govern production of APIs into the API Portal via integration with Policy Manager and Community Manager. Lifecycle Manager models services exposed in the API Portal as a “Service” representing the actual underlying service (SOAP or HTTP) with one or more “APIs” that each represent the Service within a specific Community Manager Tenant.

Chapter 2 | Resources

The Lifecycle Manager product exposes a REST API utilizing three resources for working with Services, APIs and configuring the API publishing behavior. These resources are as follows:

Services

The Services resource provides operations for creating updating and deleting Services and their associated APIs. Requests to create and update services may optionally include API information allowing a single-call update of a Service and its API(s). Alternatively the Services resource may be used to maintain Services independent of their associated APIs. In the latter case, the APIs resource (described in the following section) would be used to independently maintain APIs.

Operations

The base URI for the following operations is:

http://<host:port>/lm/custom/rest/<library name>/services.

The Create Service and Update Service operations take a JSON request body that is described in the Metadata section of this document.

Note that the creation, update and deletion of Service and API entities in Policy Manager and Community Manager resulting from these operations is asynchronous by default.

Create Service

This method is used to create a new Service, associated API(s) and supporting entities into Lifecycle Manager and Policy Manager and Community Manager.

HTTP Operation:

POST

Path:

…/services

Request Content Type

multipart/mixed

Parameters:

- **synchronous**
  Passing "true" for the synchronous query parameter results in the operation not returning until the service is published to Policy Manager and all Community Manager APIs (if specified) are created or until a Lifecycle Manager-configured timeout is reached (this timeout is defaulted to 120 seconds).

Request Details

The request should consist of a JSON part describing the Service metadata as defined in the Service Metadata section, with File parts for each "by-value" **Document** referenced in the metadata section.

Response Codes:

- **200** – Service (and APIs) successfully created
- **400** – Service could not be created due the request information being invalid
  The response body will be text containing details about the error.
- **500** – Service could not be created due an either a configuration error or an unexpected error within the server.
  The response body will be text containing details about the error.

Response

The response body will be of type JSON with the following properties:

- "service-id"
  The id of the Lifecycle Manager Service asset (this is the id required by update and delete operations).
- "api-info"
  An array of api-info properties for each API specified with the Service
  Each api-info instance will specify the following:
  - "tenant-name"
    The name of the Community Manager tenant the API is assigned to
  - "api-id"
    The id of the Lifecycle Manager API asset.
  - "cm-api-version-id"
    The id of the associated API version entity in Community Manager. Note that this property will only be present if the "synchronous" query parameter is set to "true".

Example Response:

```
{
  "api-info": [
          {
                    "tenant-name": "Community Manager",
                    "cm-api-version-id": "08d31c4b-d248-41a1-bf4a-459483a403fc.enterpriseapi",
                    "api-id": "1.0_1437704830024726896585"
          }
  ],
  "service-id": "1.0_1437704826884726896585"
}
```

Update Service

This method is used to update an existing Service, associated API(s) and associated entities in Lifecycle Manager, Policy Manager and Community Manager.

HTTP Operation:

POST

Path

…services/<service id>
Where <service id> is the service-id returned from the Create Service operation

Request Content Type

multipart/mixed

Parameters*:*

- **synchronous**
  Passing "true" for the synchronous query parameter results in the operation not returning until the service is updated in Policy Manager and all Community Manager APIs (if specified) are updated or until a Lifecycle Manager-configured timeout is reached (this timeout is defaulted to 120 seconds).

Request Details

The request should consist of a JSON part describing the Service metadata as defined in the Service Metadata section, with File parts for each "by-value" **Document** referenced in the metadata section.

Response Codes:

- **200** – Service successfully updated
- **404** – The specified Service asset was not found
- **400** – Service could not be updated due the request information being invalid
  The response body will be text containing details about the error.
- **500** – Service could not be updated due an either a configuration error or an unexpected error within the server.
  The response body will be text containing details about the error.

Response:

The response body will be of type JSON. If the update resulted in new APIs being created the response will contain the following properties:

- "api-info"
  An array of api-info properties for each new API specified with the update of the Service
  Each api-info instance will specify the following:
  - "tenant-name"
    The name of the Community Manager tenant the API is assigned to
  - "api-id"
    The id of the Lifecycle Manager API asset.
  - "cm-api-version-id"
    The id of the associated API version entity in Community Manager. Note that this property will only be present if the "synchronous" query parameter is set to "true".

Example Response:

```
{
        "api-info": [
                {
                        "tenant-name": "Community Manager",
                        "cm-api-version-id": "08d31c4b-d248-41a1-bf4a-459483a403fc.enterpriseapi",
                        "api-id": "1.0_1437704830024726896585"
                }
        ]
}
```

Delete Service

This method is used to delete an existing Service, its associated APIs and supporting entities from Lifecycle Manager, Policy Manager and Community Manager.

HTTP Operation:

DELETE

Path

…services/<service id>

Where <service id> is the service-id returned from the Create Service operation

Request Content Type

n/a

Parameters

None

Request Details

No request body is expected.

Response Codes:

- **200** – Service successfully deleted
- **404** – The specified Service asset was not found
- **500** – Service could not be deleted due an either a configuration error or an unexpected error within the server.
  The response body will be text containing details about the error.

Response:

None

Get Service

This method is used to retrieve information about an existing Service.

HTTP Operation:

GET

Path

…services/<service id>

Where <service id> is the service-id returned from the Create Service operation

Request Content Type

n/a

Request Details

No request body is expected.

Response Codes:

- **200** – Service retrieved
- **404** – The specified Service asset was not found
- **500** – Service could not be retrieved due an either a configuration error or an unexpected error within the server.
  The response body will be text containing details about the error.

Response:

The response body will be of type JSON with the following properties:

- “service-name”
  The name of the Service.
- “service-version”
  The version of the Service.
- “service-description”
  The description of the Service.

- "service-group"
  The name of the Lifecycle Manager Group the service is assigned to.
- "api-info"
  An array of api-info properties for each API associated with the Service
  Each api-info instance will specify the following:
  - "tenant-name"
    The name of the Community Manager tenant the API is assigned to
  - "api-id"
    The id of the Lifecycle Manager API asset.
  - "cm-api-version-id"
    The id of the associated API version entity in Community Manager. Note that this property is only present if the API asset has been published to Community Manager.

Example Response:

```
{
        "apis": [
                {
                        "service-id": "1.0_14377470430931058952902",
                        "tenant-name": "Community Manager",
                        "api-name": "RAMLServiceAPI",
                        "api-version": "1.0",
                        "api-description": "the description",
                        "api-group": "Test Group",
                        "cm-api-version-id": "e00e39f6-7508-4f76-b175-cba5286f39cc.enterpriseapi"
                }
        ],
        "service-name": "RAMLService",
        "service-version": "1.0",
        "service-description": "the description",
        "service-group": "Enterprise Group"
}
```

APIs

The APIs resource provides operations for creating, updating and deleting APIs for particular Community Manager Tenants independently of their underlying Service.

Operations

The base URI for the following operations is:

http://<host:port>/lm/custom/rest/<library name>/apis.

The Create API and Update API operations take a JSON request body that is described in the Metadata section of this document.

Note that the creation, update and deletion of API entities in Policy Manager and Community Manager resulting from these operations is asynchronous by default.

Create API

This method is used to create a new API for a specified Tenant and supporting entities into Lifecycle Manager, Policy Manager and Community Manager.

HTTP Operation:

POST

Path:

…/apis

Request Content Type

multipart/mixed

Parameters

- **synchronous**
  Passing "true" for the synchronous query parameter results in the operation not returning until the API and its underlying virtual services are published to Policy Manager and Community Manager or until a Lifecycle Manager-configured timeout is reached (this timeout is defaulted to 120 seconds).

Request Details

The request should consist of a JSON part describing the API metadata as defined in the API Metadata section, with File parts for each "by-value" **Document** referenced in the metadata section. The request body must specify the id of the service to associate the API with in the "service-id" property.

Response Codes:

- **200** – API successfully created
- **400** – API could not be created due the request information being invalid
  The response body will be text containing details about the error.
- **500** – API could not be created due an either a configuration error or an unexpected error within the server.
  The response body will be text containing details about the error.

Response

The response body will be of type JSON with the following properties:

- "api-id"
  The id of the Lifecycle Manager API asset.
- "cm-api-version-id"
  The id of the associated API version entity in Community Manager. Note that this property will only be present if the "synchronous" query parameter is set to "true".

Example Response:

```
{
        "cm-api-version-id": "2858544c-7150-43b8-9907-0b8441e04518.enterpriseapi",
        "api-id": "1.0_1437756185596-328509575"
}
```

Update API

This method is used to update an existing API and associated entities in Lifecycle Manager, Policy Manager and Community Manager.

HTTP Operation:

POST

Path

…apis/<api id>

Where <api id> is the api-id returned from the Create API operation

Request Content Type

multipart/mixed

Parameters

- **synchronous**
  Passing “true” for the synchronous query parameter results in the operation not returning until the API and its underlying virtual services are published to Policy Manager and Community Manager or until a Lifecycle Manager-configured timeout is reached (this timeout is defaulted to 120 seconds).

Request Details

The request should consist of a JSON part describing the API metadata as defined in the API Metadata section, with File parts for each “by-value” **Document** referenced in the metadata section.

Response Codes:

- **200** – API successfully updated
- **404** – The specified API asset was not found
- **400** – API could not be updated due the request information being invalid
  The response body will be text containing details about the error.
- **500** – API could not be updated due an either a configuration error or an unexpected error within the server.
  The response body will be text containing details about the error.

Response:

None

Delete API
This method is used to delete an existing API and supporting entities from Lifecycle Manager, Policy Manager and Community Manager.

HTTP Operation:
DELETE

Path
…apis/<api id>
Where <api id> is the api-id returned from the Create API operation

Request Content Type
n/a

Parameters
None

Request Details
No request body is expected.

Response Codes:
- **200** – API successfully deleted
- **404** – The specified API asset was not found
- **500** – API could not be deleted due an either a configuration error or an unexpected error within the server.
  The response body will be text containing details about the error.

Response:
None

Get API
This method is used to retrieve information about an existing API.

HTTP Operation:
GET

Path
…apis/<api id>
Where <api id> is the api-id returned from the Create API operation

Request Content Type
n/a

Request Details
No request body is expected.

Response Codes:
- **200** – API retrieved
- **404** – The specified API asset was not found
- **500** – API could not be retrieved due an either a configuration error or an unexpected error within the server.
  The response body will be text containing details about the error.

Response:
The response body will be of type JSON with the following properties:

- "service-id"
  The id of the associated Service asset.
- "tenant-name"
  The name of the Community Manager tenant the API is assigned to
- "api-name"
  The name of the API.
- "api-version"
  The version of the API.
- "api-description"
  The description of the API.
- "api-group"
  The name of the Lifecycle Manager Group the API is assigned to.
- "cm-api-version-id"
  The id of the associated API version entity in Community Manager. Note that this property is only present if the API asset has been published to Community Manager.

Example Response:

```
{
	"service-id": "1.0_14377561519811058952902",
	"tenant-name": "Community Manager",
	"api-name": "RAMLServiceAPI",
	"api-version": "1.0",
	"api-description": "the new description",
	"api-group": "Enterprise Group",
	"cm-api-version-id": "2858544c-7150-43b8-9907-0b8441e04518.enterpriseapi"
}
```

Tenants

The Tenants resource provides operations for creating, updating and deleting configuration information for Community Manager Tenants within Lifecycle Manager.   The configuration for a Tenant includes connection info for the Community Manager Tenant as well as configurations for the endpoints of services published into the tenant. These configurations (known as Runtime Configuration assets) affect how the virtual services in Policy Manager that support an API's endpoints are deployed.

Operations

The base URI for the following operations is:

http://<host:port>/lm/custom/rest/<library name>/tenants.

The Create Tenant and Update Tenant operations take a JSON request body that is described in the Metadata section of this document.

Create Tenant

This method is used to create a new Tenant configuration into Lifecycle Manager.

HTTP Operation:

POST

Path:

…/tenants

Request Content Type

application/json

Parameters

None

Request Details

The request should consist of a JSON body as described in the Tenant Metadata section of this document. The request body must specify the name to assign to the tenant in the "tenant-name" property.

Response Codes:

- **200** – Tenant successfully created
- **400** – Tenant could not be created due the request information being invalid
  The response body will be text containing details about the error.

- **500** – Tenant could not be created due an either a configuration error or an unexpected error within the server.
  The response body will be text containing details about the error.

Response

None

Update API

This method is used to update an existing Tenant Configuration in Lifecycle Manager.

HTTP Operation:

POST

Path

…tenants/<tenant name>

Where <tenant name> is the tenant name used in the Create Tenant operation.

Request Content Type

application/json

Parameters

None

Request Details

The request should consist of a JSON body as described in the Tenant Metadata section of this document.

Response Codes:

- **200** – Tenant successfully updated
- **404** – The specified Tenant asset was not found
- **400** – Tenant could not be updated due the request information being invalid
  The response body will be text containing details about the error.
- **500** – Tenant could not be updated due an either a configuration error or an unexpected error within the server.
  The response body will be text containing details about the error.

Response:

None

Delete Tenant

This method is used to delete an existing Tenant configuration from Lifecycle Manager.

HTTP Operation:
DELETE

Path
…tenants/<tenant name>

Where <tenant name> is the tenant name used in the Create Tenant operation.

Request Content Type
n/a

Parameters
None

Request Details
No request body is expected.

Response Codes:
- **200** – Tenant successfully deleted
- **404** – The specified Tenant asset was not found
- **500** – Tenant could not be deleted due an either a configuration error or an unexpected error within the server.
  The response body will be text containing details about the error.

Response:
None

Get Tenant
This method is used to retrieve information about an existing Tenant configuration.

HTTP Operation:
GET

Path
…tenants/<tenant name>

Where <tenant name> is the tenant name used in the Create Tenant operation.

Request Content Type
n/a

Request Details

No request body is expected.

Response Codes:

- **200** – Tenant retrieved
- **404** – The specified Tenant asset was not found
- **500** – Tenant could not be retrieved due an either a configuration error or an unexpected error within the server.
  The response body will be text containing details about the error.

Response:

The response body will be of type JSON with the following properties:

- "tenant-name"
  The name of the tenant configuration
- "tenant-description"
  The description of the Tenant.
- "endpoint"
  The endpoint of the associated Community Manager Tenant.
- "community-manage-name"
  The name of the underlying Community Manager federated-system in the Lifecycle Manager configuration.
- "service-qualifier"
  The suffix to use in Policy Manager virtual service names associated with this tenant.
- "admin-role-filter"
  Role Filter within Lifecycle Manager to use in assigning administrators to the API in Community Manager.
- "application-group"
  Default Lifecycle Manager Group to own Application assets propagated from Community Manager for this Tenant

Example Response:

```
{
        "user": "support@roch.soa.local",
        "endpoint": "http://tenant2.soa.local:9900",
        "community-manager-name": "Community Manager",
        "tenant-name": "TenantFromAPI2",
        "tenant-description": "Auto=created Community Manager Tenant",
        "service-qualifier": "tenantfromapi2",
        "admin-role-filter": "admin-roles",
        "application-group": "External Apps"
}
```

Chapter 3 | Metadata

The operations for creating and updating Services, APIs and Tenants expect request metadata to be provided as a JSON document in the body of the requests.   This section describes the expected properties for these JSON requests. Note that many of the concepts that appear in the Service and API metadata originate from the Policy Manager and Community Manager products and will not be described in depth in this document.

Service Metadata

service-name

The name of the Service. Note that this will also be the default name for any associates APIs.

Example:

"service-name": "MyService"

Create Semantics: Required.

Update Semantics: Optional. If not specified the Service name will not be modified.

service-version

The version of the Service. Note that this will also be the default name for any associates APIs.

Example:

"service-version": "1.0"

Create Semantics: Required.

Update Semantics: Optional. If not specified the Service version will not be modified.

service-description

The description of the Service. Note that this will also be the default description for any associates APIs and will be propagated to the API and API version descriptions in Community Manager.

Example:

"version-description": "This is the description of MyService"

Create Semantics: Required.

Update Semantics: Optional. If not specified the Service description will not be modified.

service-group

The name of the owning group in Lifecycle Manager for created Service asset. This will also be used as the default owning group for any associated API assets. Note, that this will also determine the Policy Manager organization for the target Physical Service representing the API.

The group must pre-exist in Lifecycle Manager

Example:

"service-group": "API Production Group"

Create Semantics: Optional, if not specified owning group will default to the Enterprise Group for the library.

Update Semantics: N/A.

target-namespace

This property is used to specify the target namespace to use for producing the WSDL to represent the Service in Policy Manager. The standard LM configuration defaults the namespace based on the original name and version of the API. Setting this field allows the default generated namespace to be overridden.

Example:

"target-namespace": "http://www.akana.com/MyAPI_1_0"

Create Semantics: Optional (in standard LM configuration). Default is determined by the LM configuration.

Update Semantics: Optional. If not specified the target namespace will not be modified.

service-classifiers

This field is used to set arbitrary classifiers onto the LM Service asset. The LM configuration may use these classifiers in determining the governance flow for the asset or map them to categorizations on the associated target service in Policy Manager, thus allowing policies to be automatically assigned to the target service.

The service-classifiers property consists of an array of classifier properties that each require a "name" and an array of values. The classifiers used in this property must correspond to those defined in the LM configuration.

Example:

```
"service-classifiers": [
        {
                "name": "business-domain",
                "values": [
                        "Shipping|Request"
                ]
        }
]
```

Create Semantics: Optional.

Update Semantics: Optional. If the property is not specified the classifiers will not be modified. Specifying a classifier in the aservice-classifiers array that has no values will cause existing values for the classifier to be removed.

descriptor-document

This is a document that defines the operations of the API. Current supported formats are:

- RAML
- WADL
- WSDL
- Swagger 1.2 API Declaration

This property is represented as a ***Document***[1].

Example using an attached file:

```
"descriptor-document": {
                "name": "PlotResource.json"
        }
```

[1] A document can specify either a "url" property or a "name" property that corresponds to a file part on the request itself. In the case of a url, the url must be accessible from LM. In the case of a file, a part must exist on the multi-part request with a file name that matches the specified "name".

Example using a url:

```
"descriptor-document": {
                    "url": “http://www.mycompany.com/apis/PlotResource.json”
          }
```

Create Semantics: Required.

Update Semantics: Optional. If not specified the Service definition will not be modified.

service-artifacts

This property specifies ***Documents*** to be associated with the Service as artifacts in LM. The LM configuration may specify that these artifacts be propagated to associated APIs by default. The service-artifacts property contains an array of “artifacts” which contain a “category” property as well as a “document” property.

Example:

```
"service-artifacts": [
                    {
                              "category": "requirements",
                              "document": {
                                        "name": "Requirements.pdf"
                              }
                    }
          ]
```

Create Semantics: Optional.

Update Semantics: Optional. If the property is not specified the Service artifacts will not be modified. Specifying an artifact in the service-artifacts array that has no document property will cause an existing document with matching category to be removed.

orchestrated-service

This is a Boolean property used to indicate that the target service in Policy Manager should be a virtual service in expectation that it will be implemented using Policy Manager’s orchestration wizard.

Example:

```
"orchestrated-service": "true"
```

Create Semantics: Optional. Defaults to “false” (physical target service).

Update Semantics: N/A.

submit

This is a Boolean property used to indicate that the Service asset in Lifecycle Manager should be submitted for publishing at the conclusion of this call.

Example:

"submit": "true"

Create Semantics: Optional. Defaults to "true".

Update Semantics: N/A.

apis

The apis property holds an array of api properties as described in the following section on API Metadata. This property can be used to create or update associated APIs for a service in the same call as the service create/update. Including an api property on a Create Service call is the logical equivalent to following the Create Service call with a Create API call for the specified api. In the case of an Update Service call, api properties matching existing API assets will result in updates to those API assets, while new api properties will be treated as Create API calls. Note that unmatched API assets will not be removed.

Example Create Service Metadata

The following is a sample body for a Create Service request that does not specify API data:

```
{
        "service-name": "OSAgents",
        "service-version": "1.0",
        "service-description": "the description",
        "service-group": "Services Group",
        "target-namespace": "http://www.akana.com",
        "service-classifiers": [
                {
                        "name": "business-domain",
                        "values": [
                                "Shipping|Request"
                        ]
                }
        ],
        "descriptor-document": {
                "name": "os-agents.wadl"
        },
        "service-artifacts": [
                {
                        "category": "requirements",
                        "document": {
                                "name": "service-requirements.txt"
                        }
                }
        ]
}
```

This example shows a Create Service call that also specifies an API:

```
{
	"service-name": "OSAgents",
	"service-version": "1.0",
	"service-description": "the description",
	"service-group": "Services Group",
	"target-namespace": "http://www.akana.com",
	"service-classifiers": [
		{
			"name": "business-domain",
			"values": [
				"Shipping|Request"
			]
		}
	],
	"descriptor-document": {
		"name": "os-agents.wadl"
	},
	"service-artifacts": [
		{
			"category": "requirements",
			"document": {
				"name": "service-requirements.txt"
			}
		}
	],
	"apis": [
		{
			"tenant-name": "Community Manager",
			"api-visibility": "private",
			"use-licenses": true,
			"api-keywords": [
				"keyword1",
				"keyword2"
			],
			"api-classifiers": [
				{
					"name": "business-domain",
					"values": [
						"Shipping|Request"
					]
				}
			],
			"avatar": {
				"name": "avatar.jpg"
			},
			"api-artifacts": [
				{
					"category": "usage-guide",
					"document": {
						"name": "UsageGuide.pdf"
					}
				},
				{
					"category": "api-config",
					"document": {
```

```
                        "name": "api-config.json"
                    }
                }
            ],
            "legal-agreements": [
                {
                    "name": "Legal Agreement",
                    "version": "1",
                    "id": null
                }
            ],
            "production-endpoint": {
                "path": "prod-path",
                "virtual-host": "prod-apis",
                "consumption-auto-approved": false
            },
            "operations": [
                {
                    "name": "operation1",
                    "scopes": [
                        "Scope1",
                        "Scope2"
                    ]
                },
                {
                    "name": "operationToReplace2",
                    "scopes": [
                        "Scope2"
                    ]
                }
            ],
            "api-access": {
                "group-access": [
                    {
                        "group": {
                            "name": "group a"
                        },
                        "licenses": [
                            {
                                "name": "License1"
                            },
                            {
                                "name": "License2"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

API Metadata

api-name

The name of the API.

Example:

"api-name": "MyAPI"

Create Semantics: Optional, defaults to target Service name if not specified.

Update Semantics: Optional. If not specified the API name will not be modified.

api-version

The version of the API.

Example:

"api-version": "1.0"

Create Semantics: Optional, defaults to target Service version if not specified.

Update Semantics: Optional. If not specified the API version will not be modified.

api-description

The description of the API.

Note that this description will be used for the API and API version descriptions in Community Manager.

Example:

"api-description": "This is the description of MyAPI"

Create Semantics: Optional, defaults to target Service description if not specified.

Update Semantics: Optional. If not specified the API description will not be modified.

api-group

The name of the owning group in Lifecycle Manager for the API asset.

The group must pre-exist in Lifecycle Manager

Example:

"api-group": "API Production Group"

Create Semantics: Optional, defaults to target Service group if not specified.

Update Semantics: N/A.

api-visibility

This field is used to set the "Visibility" field of the API in Community Manager.

Valid values for this field are: "Public", "Private", and "Registered Users". See Community Manager documentation for additional details.

Example:

"api-visibility": "Private"

Create Semantics: Optional. Default is based on LM configuration.

Update Semantics: Optional. If not specified the API visibility will not be modified.

use-licenses

This field is used to indicate that licenses should be enabled for the API in Community Manager.

See Community Manager documentation on licenses for additional details.

Example:

"use-licenses": "true"

Create Semantics: Optional. Default is based on LM configuration.

Update Semantics: Optional. If not specified the API use licenses setting will not be modified.

scopes

These are the license scopes defined in Community Manager that are to be assigned to all operations on the API. Specified scopes must correspond to scopes defined under the "Scopes" configuration tab in Community Manager. See Community Manager Documentation on Scopes for additional details.
Example:

```
"scopes": [
                    "Scope1",
                    "Scope2"
          ],
```

Create Semantics: Optional, may only be set when use-licenses property is set to “true”.

Update Semantics: Optional, may only be set when use-licenses property is set to “true”.

If the scopes property is not specified, API-wide scopes will not be modified.

api-keywords

This field is used to set values for the “Tags“ field of the API in Community Manager.

Example:

```
"api-keywords": [
		"keyword1",
		"keyword2"
]
```

Create Semantics: Optional.

Update Semantics: Optional. If the property is not specified the API tags will not be modified. If the property is specified with an empty array, existing tags will be removed from the API.

api-classifiers

This field is used to set arbitrary classifiers onto the LM API asset. The LM configuration may use these classifiers in determining the governance flow for the asset or map them to categorizations on the associated Virtual Service in Policy Manager, thus allowing policies to be automatically assigned.

The api-classifiers property consists of an array of classifier properties that each require a “name” and an array of values. The classifiers used in this property must correspond to those defined in the LM configuration.

Example:

```
"api-classifiers": [
		{
			"name": "business-domain",
			"values": [
				"Shipping|Request"
			]
		}
]
```

Create Semantics: Optional.

Update Semantics: Optional. If the property is not specified the classifiers will not be modified. Specifying a classifier in the api-classifiers array that has no values will cause existing values for the classifier to be removed.

avatar

This is an image to be used as the avatar of the API in Community Manager. It should be an image file such as a jpg or png that is equal or less than 75x75 pixels.

This property is represented as a ***Document***.

Example:

```
"avatar": {
                    "name": "avatar.jpg"
          }
```

Create Semantics: Optional.

Update Semantics: Optional. If not specified the API avatar will not be modified.

api-artifacts

This property specifies ***Documents*** to be associated with the API as artifacts in LM. The LM configuration can specify certain artifact categories to be published as documents on the API in Community Manager. The api-artifacts property contains an array of "artifacts" which contain a "category" property as well as a "document" property.

Example:

```
"api-artifacts": [
                    {
                              "category": "usage-guide",
                              "document": {
                                        "name": "UsageGuide.pdf"
                              }
                    },
                    {
                              "category": "api-config",
                              "document": {
                                        "name":"api-config.json"
                              }
                    }
          ]
```

Create Semantics: Optional.

Update Semantics: Optional. If the property is not specified the API artifacts will not be modified. Specifying an artifact in the api-artifacts array that has no document property will cause an existing document with matching category to be removed.

legal-agreements

Legal agreements are stored as separate assets within LM. These assets (which must pre-exist) can be specified by name/version or by asset-id within a legal-agreement element.

Example (specified by name/version):

```
"legal-agreements": [
                    {
                            "name": "Legal Agreement",
                            "version": "1"
                    }
            ]
```

Example (specified by asset-id):

```
"legal-agreements": [
                    {
                            "id": " 1.0_1429894704190190640092l"
                    }
            ]
```

Create Semantics: Optional.

Update Semantics: Optional. If the property is not specified the legal agreements for the API will not be modified. Specifying the legal-agreement property with an empty array will cause existing legal agreements to be removed.

production-endpoint

The production-endpoint property is used to specify Service and API configuration specific to the production endpoint of the API. It consists of a number of individual properties:

path

The path property on an endpoint specifies the path of the managed Virtual Service that represents the endpoint in Policy Manager. This becomes the path of the published production endpoint for the API in Community Manager. For example: http://<host>:<port>/***<path>*** . Paths for each API production endpoint should be unique. See HTTP Details "location" in Policy Manager documentation for additional details.

Example:

```
"path": "MyAPI"
```

Create Semantics: Optional but recommended unless LM is configured to generate a unique path.

Update Semantics: Optional. If the property is not specified the path for the endpoint will not be modified.

virtual-host

The virtual-host property on an endpoint specifies a virtual host to be used in the published endpoint for the API. For example: http:*//<virtual-host>*:<port>/<path> . The virtual host is specified on the managed Virtual Service that represents the endpoint in Policy Manager with the assumption that calls to the endpoint will be routed to the actual managed virtual service endpoint. See HTTP Details “virtual host” in Policy Manager documentation for additional details.
Example:

"virtual-host": "mycompany"

Create Semantics: Optional.

Update Semantics: Optional. If the property is not specified the virtual-host for the endpoint will not be modified.

consumption-auto-approved

This Boolean property corresponds inversely to the “Approval Required” endpoint field in Community Manager. If set to *true*, “Approval Required” will be set to *false* on the API endpoint in Community Manager. Note that a value of *true* is only valid if the *use-licenses* property for the API is set to *false*.
Example:

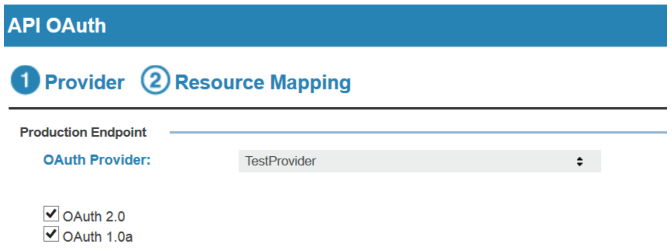
"consumption-auto-approved": false

Create Semantics: Optional, default value is dependent on LM configuration

Update Semantics: Optional. If the property is not specified the setting on the API will not be modified.

oauth-details

This property corresponds to the endpoint OAuth configuration in Community Manager:


API OAuth
1 Provider 2 Resource Mapping
Production Endpoint
OAuth Provider: TestProvider
☑ OAuth 2.0
☑ OAuth 1.0a

The OAuthProvider property is used to specify the OAuth provider while Boolean fields "OAuth10aSupported" and "OAuth20Supported" indicate the version of OAuth supported. See Community Manager documentation for additional details on OAuth configuration. Note that the provider specified must correspond to one that is available in the target Community Manager tenant.

Example:

```
"oauth-details": {
                                        "OAuthProvider": "TestProvider",
                                        "OAuth10aSupported": true,
                                        "OAuth20Supported": true
}
```

Create Semantics: Optional, defaults to no provider if not provided.

Update Semantics: Optional. If the property is not specified the endpoint OAuth settings will not be modified.

Example of complete production-endpoint property:

```
"production-endpoint": {
             "path": "MyAPI",
             "virtual-host": "mycompany",
             "consumption-auto-approved": false,
             "oauth-details": {
                          "OAuthProvider": "TestProvider",
                          "OAuth10aSupported": true,
                          "OAuth20Supported": true
             }
}
```

sandbox-endpoint

The sandbox-endpoint property is of the same format as the production-endpoint. It is used to configure the sandbox endpoint for the API, if applicable.

Example of complete sandbox-endpoint property:

```
"sandbox-endpoint": {
             "path": "MyAPI",
             "virtual-host": "sandbox-apis",
             "consumption-auto-approved": true,
             "oauth-details": {
                          "OAuthProvider": "TestProvider",
                          "OAuth10aSupported": true,
                          "OAuth20Supported": true
             }
}
```

sandbox-physical-endpoint

This property is used to specify a target endpoint of the physical service supporting the API's sandbox endpoint. The value must be a valid url. Specifying this property results in a physical service being generated in Policy Manager with the specified target endpoint.

Example:

```
"sandbox-physical-endpoint": "http://www.mycompany.com/sandbox/MyAPIService"
```

Create Semantics: Optional.

Update Semantics: Optional. If specified the endpoint for the physical service will be updated. If not specified, an existing physical service will not be modified.

operations

The operations property is used to specify operation-specific settings for the API. It consists of an array of operation properties which each consist of an operation name and properties that apply to that operation. Operation names must correspond to operation names specified in the API definition.

Example:

```
{
        "name": "getPlot",
        "scopes": [
                "Scope1",
                "Scope2"
        ],
        "oauth-resources": [
                "OauthScope1"
        ]
}
```

In addition to the "name" property, which is mandatory, operations may contain these properties:

scopes

These are the license scopes defined in Community Manager that are to be assigned to the operation. Specified scopes must correspond to scopes defined under the "Scopes" configuration tab in Community Manager. See Community Manager Documentation on Scopes for additional details.

Example:

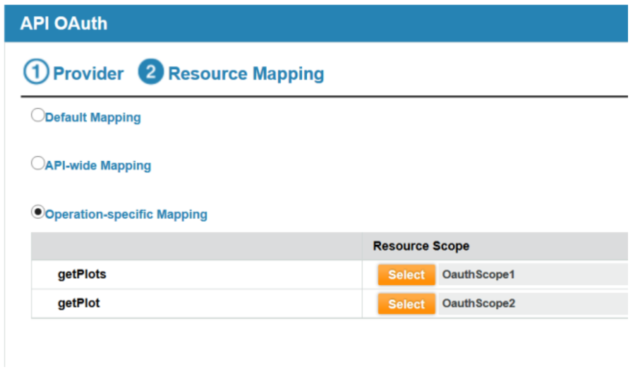
```
"scopes": [
                "Scope1",
                "Scope2"
        ]
```

Create Semantics: Optional.

Update Semantics: Optional. If the scopes property is not specified, scopes for the operation will not be modified.

oauth-resources

The oauth-resources property allows operation-specific resource scopes to be assigned. This information corresponds to the "operation specific mapping" section of the Community Manager OAuth wizard:



Default Mapping

API-wide Mapping

● Operation-specific Mapping

	Resource Scope
getPlots	Select OauthScope1
getPlot	Select OauthScope2

Specified scopes must correspond to scopes defined for the chosen endpoint OAuth provider in Community Manager[2]. See Community Manager Documentation on OAuth configuration for additional details.

Example:

```
"oauth-resources": [
                    "OauthScope1"
          ]
```

Create Semantics: Optional.

Update Semantics: Optional. If the oauth-resources property is not specified, the OAuth scopes for the operation will not be modified.

api-access

The api-access property allows access to the API to be configured for specified Community Manager Groups and is applicable only when the api-visibility property is set to “Private”. The api-access property consists of a “group-access” array with each property specifying the name or id of a group defined in Community Manager and the name or id of licenses[3] assigned to that group. The api-access property corresponds to the Visibility/Groups API edit tab in Community Manager.

Example (using group and license names):

```
"api-access": {
          "group-access": [
                    {
                              "group": {
                                        "name": "group a"
                              },
                              "licenses": [
                                        {
                                                  "name": "License1"
```

[2] Note that with the current version of Community Manager if different OAuth providers are selected for production and sandbox endpoints, the scopes defined for those providers must match.
[3] Note that the specified licenses must correspond to license scopes assigned to the API or its methods. See Community Manager documentation on Licenses and Scopes for additional details.

```
                        },
                        {
                                "name": "License2"
                        }
                ]
        }
    ]
}
```

Example (using group and license ids):

```
"api-access": {
    "group-access": [
        {
            "group": {
                "id": "e68eff8f-a0f5-4ced-8f14-d45b611b9a5e.enterpriseapi"
            },
            "licenses": [
                {
                    "id": "1965c22f-96e6-43a1-8cdf-303756988c32.enterpriseapi"
                }
            ]
        }
    ]
}
```

A group may be granted unrestricted access to the API by setting the “unrestricted” property to “true”:

Example:

```
"group-access": [
        {
            "group": {
                "name": "group a"
            },
            "unrestricted": “true”
        }
```

Create Semantics: Required if api-visibility property is set to “Private”.

Update Semantics: Optional. If the api-access property is not specified, group access for the operation will not be modified. If the api-access property is specified, all existing group access information will be replaced.

Example Create API Metadata

The following is a sample body for a Create API request for an existing Service

```
{
            "tenant-name": "Community Manager",
            "api-visibility": "private",
            "use-licenses": true,
            "api-keywords": [
                "keyword1",
                "keyword2"
            ],
"api-classifiers": [
        {
                "name": "business-domain",
                "values": [
                        "Shipping|Request"
                ]
        }
],
"avatar": {
        "name": "avatar.jpg"
},
"api-artifacts": [
        {
                "category": "usage-guide",
                "document": {
                        "name": "UsageGuide.pdf"
                }
        },
        {
                "category": "api-config",
                "document": {
                        "name": "api-config.json"
                }
        }
],
"legal-agreements": [
        {
                "name": "Legal Agreement",
                "version": "1",
                "id": null
        }
],
"production-endpoint": {
        "path": "prod-path",
        "consumption-auto-approved": false,
        "oauth-details": {
                "OAuthProvider": "TestProvider",
                "OAuth10aSupported": true,
                "OAuth20Supported": true
        }
},
"sandbox-endpoint": {
        "path": "sandbox-path",
        "consumption-auto-approved": false,
        "oauth-details": {
                "OAuthProvider": "TestProvider",
                "OAuth10aSupported": true,
                "OAuth20Supported": true
        }
},
"sandbox-physical-endpoint": "http://www.sandbox.com",
"operations": [
        {
                "name": "anOperation",
                "scopes": [
                        "Scope1",
                        "Scope2"
                ],
                "oauth-resources": [
                        "OAuthScope1"
								]
							},
							{
								"name": "anotherOperation",
								"scopes": [
									"Scope2"
								],
								"oauth-resources": [
									"OAuthScope1",
									"OAuthScope2"
								]
							}
						],
						"api-access": {
							"group-access": [
								{
									"group": {
										"name": "group a"
									},
									"licenses": [
										{
											"name": "License1"
										},
										{
											"name": "License2"
										}
									]
								}
							]
						}
}
```

Tenant Metadata

community-manager-name

This is the name of the Community Manager federated-system in the Lifecycle configuration that this Tenant is to be associated with. This is an optional property and defaults to “Community Manager”. In a standard configuration with a single target Policy Manager/Community Manager target the property need not be set.
See the Lifecycle Manager API Governance Document for additional details.

Example:

"community-manager-name": "Community Manager"

Create Semantics: Optional, defaults to “Community Manager”.

Update Semantics: N/A

tenant-name

The name of the Tenant configuration within Lifecycle Manager. Note that this need not match the actual tenant name in Community Manager.

Example:

"tenant-name": "MyTenant"

Create Semantics: Required.

Update Semantics: Optional. If not specified the API name will not be modified.

tenant-description

The description of the Tenant.

Example:

"tenant-description": "This is the description of MyTenant"

Create Semantics: Optional.

Update Semantics: Optional. If not specified the API description will not be modified.

endpoint

The endpoint (URL) of the Community Manager tenant.

Example:

"endpoint": "http://mytenant.mycompany.com:9900"

Create Semantics: Required.

Update Semantics: Optional. If not specified the endpoint for the tenant will not be modified.

user

The user id (generally an email address) to use in publishing to the Community Manager tenant.

Example:

"user": "integrationuser@mycompany.com"

Create Semantics: Required.

Update Semantics: Optional. If not specified the user for the tenant will not be modified.

password

The password for the specified user within the Community Manager tenant.

Example:

"password": "mypassword"

Create Semantics: Required.

Update Semantics: Optional. If not specified the password for the tenant will not be modified.

admin-role-filter

The name of a Role Filter within Lifecycle Manager to use in assigning users as administrators for an API in Community Manager. See the Lifecycle Manager API Production Governance document for additional details.

Example:

"admin-role-filter": "admin-roles"

Create Semantics: Optional.

Update Semantics: Optional. If not specified the admin role filter for the tenant will not be modified.

application-group

The name of a Group within Lifecycle Manager to be the default owner for Application assets propagated from Community Manager for this Tenant. See the Lifecycle Manager API Production Governance document for additional details.
Example:

"application-group": "External Apps"

Create Semantics: Optional, defaults to Enterprise Group.

Update Semantics: Optional. If not specified the application group for the tenant will not be modified.

runtime-configuration

The runtime-configuration property holds the configuration information for the endpoints of APIs published to this Tenant. At least one “production-endpoint-configuration” property is required, while the “sandbox-endpoint-configuration” property is optional and used only if the APIs are to support sandbox separate sandbox endpoints. The properties for endpoint-configurations are described below.

name

The name of a particular endpoint configuration. This name must be unique within the tenant configuration.

Example:

> *"name": "Production Endpoint Configuration"*

Create Semantics: Required.

Update Semantics: Optional. If not specified the name for the endpoint configuration will not be modified.

virtual-host

The virtual-host property on an endpoint configuration specifies the default virtual host to be used in the published endpoints for APIs published with this endpoint configuration. For example: http***://<virtual-host>***:<port>/<path> . The virtual host is specified on the managed Virtual Service that represents the endpoint in Policy Manager with the assumption that calls to the endpoint will be routed to the actual managed virtual service endpoint. See HTTP Details “virtual host” in Policy Manager documentation for additional details.

Example:

> *"virtual-host": "MyCompany"*

Create Semantics: Optional.

Update Semantics: Optional. If not specified the default virtual-host for the endpoint configuration will not be modified.

container-name

This is the name of the container that will host the virtual services representing the API endpoints associated with this endpoint configuration. If containers are clustered the container representing the cluster should be specified. See Policy Manager documentation for additional details on containers.

Example:

> *"container-name": "Network Director"*

Create Semantics: Optional. If not specified, API endpoint virtual services will not be deployed to a container and the resulting APIs will not display endpoints.

Update Semantics: Optional. If not specified the container name assigned to the endpoint configuration will be removed.

policy-categorization

This is an array of references to predefined policy representation assets in Lifecycle Manager. Each reference must correspond to the policy-categorization classifier of an existing Technical Policy asset.

Example:

```
"policy-categorization": [
        "Audit|Standard",
        "Consumer Credentialing|Standard Internal"
]
```

Create Semantics: Optional. If not specified, no predefined policy categorizations will be assigned to the API endpoints using this configuration.

Update Semantics: Optional. If not specified the policy categorizations for this endpoint configuration will not be updated.

classifiers

This is an array of classifier properties indicating classifier values that are to be associated with an API endpoints using this endpoint configuration. Commonly these classifiers will be mapped to Policy Manager categorizations in the Lifecycle Manager configuration, allowing the classifiers specified on the endpoint configuration to determine Policy Manager policies applied to the virtual service representing the associated API endpoint. For example the production endpoint configuration may include classifiers that result in different policies being applied to production endpoint virtual services than for sandbox endpoint virtual services.
See the Service Metadata section for additional details on the classifier property format.

Example:

```
"classifiers": [
    {
        "name": "transaction-volume",
        "values": ["high"]
    }
]
```

Create Semantics: Optional.

Update Semantics: Optional. If the classifiers property is not specified the classifiers for this endpoint configuration will not be updated.

api-filter-name

This is the name of an asset-filter in Lifecycle Manager used to determine which APIs should be assigned this endpoint configuration. The filter name must correspond to an asset-filter defined in the Library Process Configuration (LPC) document in Lifecycle Manager or to an implicit asset-filter e.g. "asset-

type:API".
Asset filters assigned to endpoint configurations should be distinct by endpoint type (production or sandbox). The use of asset filters allows a Tenant Configuration to specify different production/sandbox endpoint configurations based on the classifiers of an API being published.

Example:

"asset-filter-name": "High Volume APIs"

Create Semantics: Optional. If not specified, api-filter-name defaults to "asset-type:API" meaning that all APIs published to the Tenant will be assigned this endpoint configuration.

Update Semantics: Optional. If not specified the api-filter-name will not be modified.

virtual-service-suffix

This property determines the suffix that will be assigned to virtual services in Policy Manager representing the API endpoint associated with this endpoint configuration. The virtual-service-suffix must be unique across endpoint configurations for a Tenant.

Example:

"virtual-service-suffix": "_API"

Create Semantics: Optional. If not set suffix will default to that defined in the Lifecycle Manager configuration for the endpoint type.

Update Semantics: Optional. If not specified the virtual service suffix for the endpoint configuration will not be modified.

physical-service-suffix

In the case where the endpoint configuration represents a sandbox endpoint, this property determines the suffix that will be assigned to physical service in Policy Manager representing the API's sandbox endpoint. The physical-service-suffix must be unique across endpoint configurations for a Tenant.

Example:

"physical-service-suffix": "_SANDBOX"

Create Semantics: Optional. If not set the physical service suffix will default to "_SAND".

Update Semantics: Optional. If not specified the physical service suffix for the endpoint configuration will not be modified.

default-oauth-details

This property represents the default OAuth security settings for API endpoints associated with this endpoint configuration. The structure of the property corresponds to the endpoint OAuth configuration in Community Manager.   This property is useful in specifying API-wide OAuth settings for APIs by default.

Example:

```
"default-oauth-details": {
                              "OAuthProvider": "AnotherProvider",
                              "OAuth10aSupported": true,
                              "OAuth20Supported": true
                    }
```

Create Semantics: Optional.

Update Semantics: Optional. If not specified the default OAuth settings for the endpoint configuration will not be modified.

Sample Create Tenant Metadata

The following is a sample body for a Create Tenant request:

```
{
          "tenant-name":"ExternalAPIs",
          "community-manager-name": "Community Manager",
          "user": "support@roch.soa.local",
          "password": "password",
          "endpoint": "http://tenant2.soa.local:9900",
          "admin-role-filter": "admin-roles",
          "application-group": "Enterprise Group",
          "runtime-configuration": {
                    "production-endpoint-configuration": [
                              {
                                        "name": "External Production Endpoint Configuration",
                                        "virtual-host": "MyProductionHost",
                                        "container-name": "Network Director",
                                        "policy-categorization": [
                                                  "Audit|Standard",
                                                  "Consumer Credentialing|External Business Partner"
                                        ],
                                        "classifiers": [
                                                  {
                                                            "name": "transaction-volume",
                                                             "values": ["high "]
                                                   }
                                        ],
                                        "default-oauth-details": {
                                                  "OAuthProvider": "MyProductionProvider",
                                                  "OAuth10aSupported": true,
                                                  "OAuth20Supported": true
                                        }
                              }
                    ],
                    "sandbox-endpoint-configuration": [
                              {
                                        "name": "External Sandbox Endpoint Configuration",
                                        "virtual-host": "MySandboxHost",
                                        "container-name": "Network Director"
```

```
                                    }
                        ]
            }
}
```

Chapter 4 | LifeCycle Manager Configuration

The operations described in this document require that the target Lifecycle Manager library be configured with full Policy Manager and Community Manager integration enabled. How certain API metadata such as documents are mapped to APIs in Community Manager is also dependent on the Lifecycle Manager configuration. Since this document does not cover Lifecycle Configuration, it is recommended that customers work with Akana to implement the necessary library configuration.