Resource Manager API

SPECIFICATION V1.0

Disclaimer Note

About Accanto Systems

Headquartered in Finland with customers worldwide, Accanto Systems offer an advanced network orchestration and quality management platform, called StratOSS^{TM}. Focused on improving customer experience, StratOSS automates and optimizes the management of physical and virtual networks, enabling Service Providers to focus on use cases that deliver business value.

www.accantosystems.com

Contents

About this document

Document Scope

This document covers the high-level architecture for the Resource Manager API and the specification of the messages sent across this interface.

Intended Audience

This document is intended for use by teams implementing their own Resource Manager implementations that need to comply with this API specification.

1 Interface Architecture

The Resource Manager API is responsible for defining the interactions between a lifecycle manager and the resource managers used to manage resources within virtual (or physical) infrastructures.

1.1 High Level Overview

There are four groups of interface endpoints referenced within the Resource Manager API. These groups are responsible for the management or retrieval of different entities from the resource managers.

The following diagram shows the high-level functional architecture referenced in this interface specification.

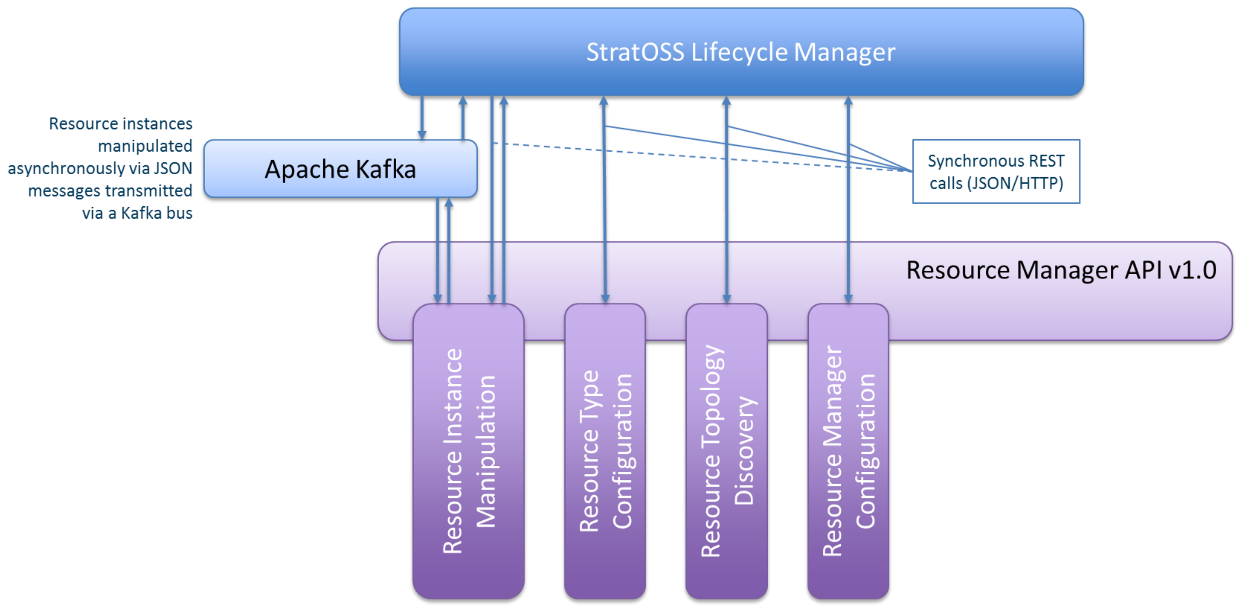
StratOSS Lifecycle Manager
Resource instances manipulated asynchronously via JSON messages transmitted via a Kafka bus
Apache Kafka
Synchronous REST calls (JSON/HTTP)
Resource Manager API v1.0
Resource Instance Manipulation
Resource Type Configuration
Resource Topology Discovery
Resource Manager Configuration

Figure 1 - High Level Interface Architecture

1.2 API Interaction Principles

All of the message descriptions (detailed in section 2) are in JSON format and should be submitted with the HTTP content-type header of “application/json”.

The REST endpoints can be secured using HTTPS, but there is no current provision for further authentication across the interface. Future changes could add support for HTTP basic authentication or the use of tokens (such as JWT, OAuth, etc…).

All of the REST endpoints should be accessible from the same root URL (e.g. http://localhost:8080/api/configuration and http://localhost:8080/api/types)

1.3 Interface Interaction Patterns

This section outlines the various potential interaction patterns with how the Lifecycle Manager intends to call the Resource Manager APIs.

1.3.1 Resource Manager Onboarding

When a new Resource Manager is being onboarded into the Lifecycle Manager, the following calls will be made.

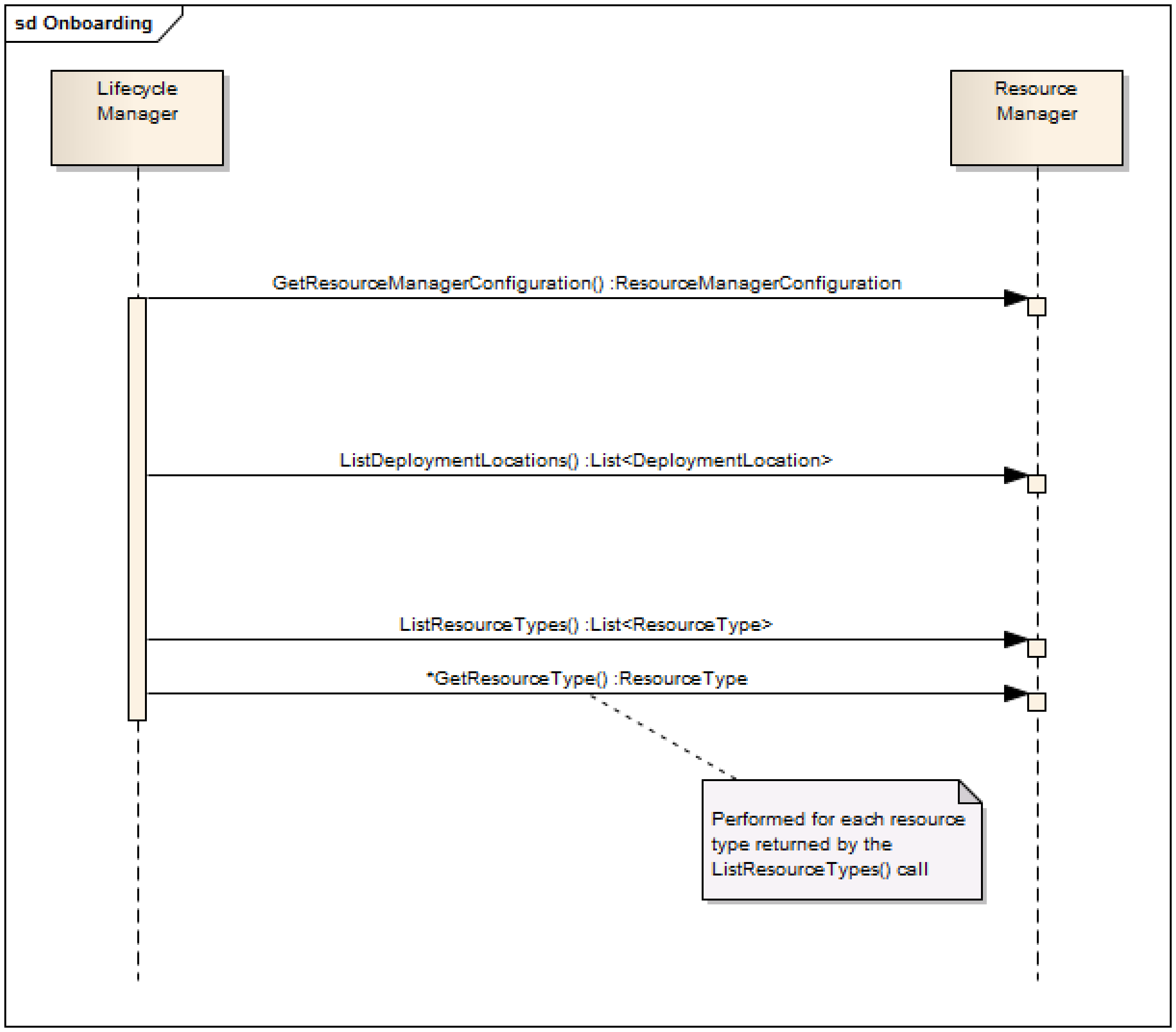
sd Onboarding
Lifecycle Manager
Resource Manager
GetResourceManagerConfiguration() :ResourceManagerConfiguration
ListDeploymentLocations() :List<DeploymentLocation>
ListResourceTypes() :List<ResourceType>
*GetResourceType() :ResourceType
Performed for each resource type returned by the ListResourceTypes() call

Figure 2 - Onboarding Message Interaction Pattern

1.3.2 Process Creation

When a new transition process is being created within the Lifecycle Manager the following calls may be made to a Resource Manager.

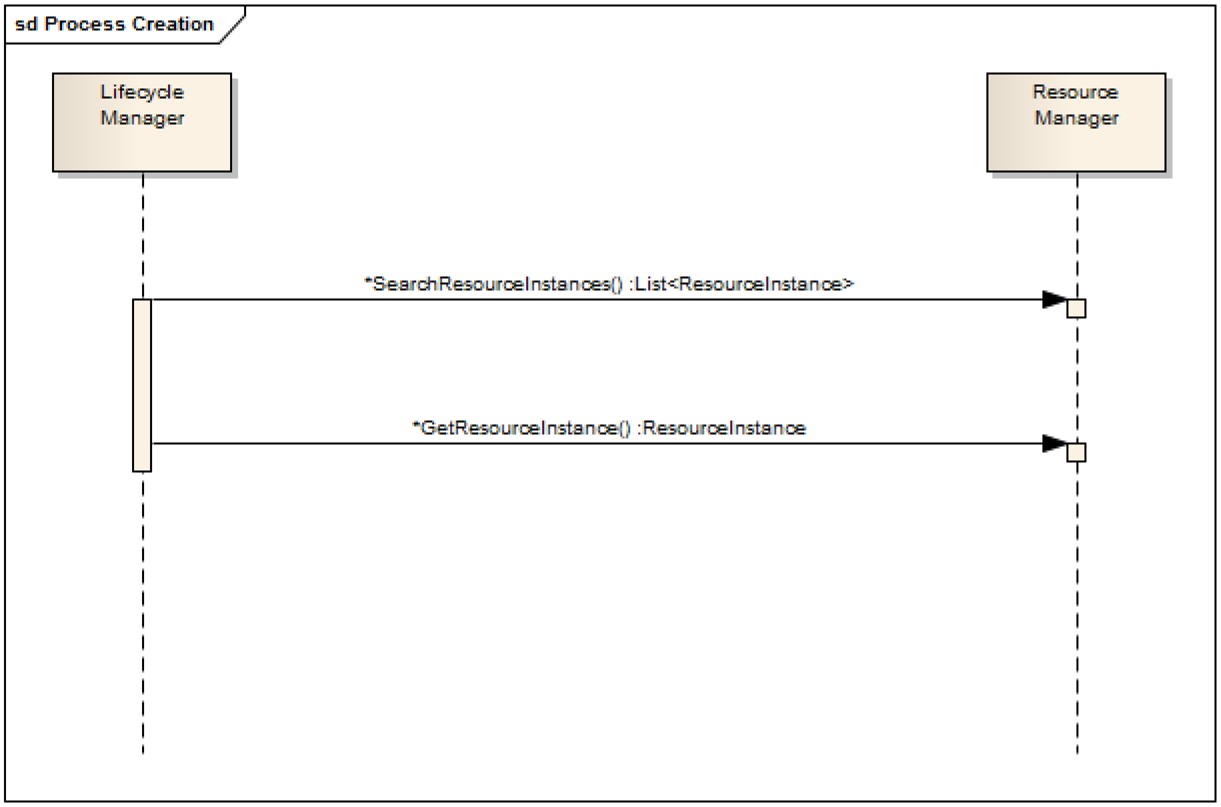
sd Process Creation
Lifecycle Manager
Resource Manager
*SearchResourceInstances() :List<ResourceInstance>
*GetResourceInstance() :ResourceInstance

Figure 3 - Process Creation Message Interaction Pattern

1.3.3 Resource Transitions (Synchronous)

As part of an assembly transition, the Lifecycle Manager may call out to a Resource Manager to transition or perform an operation on a Resource. This is the interaction pattern for this scenario when the Resource Manager does not support asynchronous response messages.

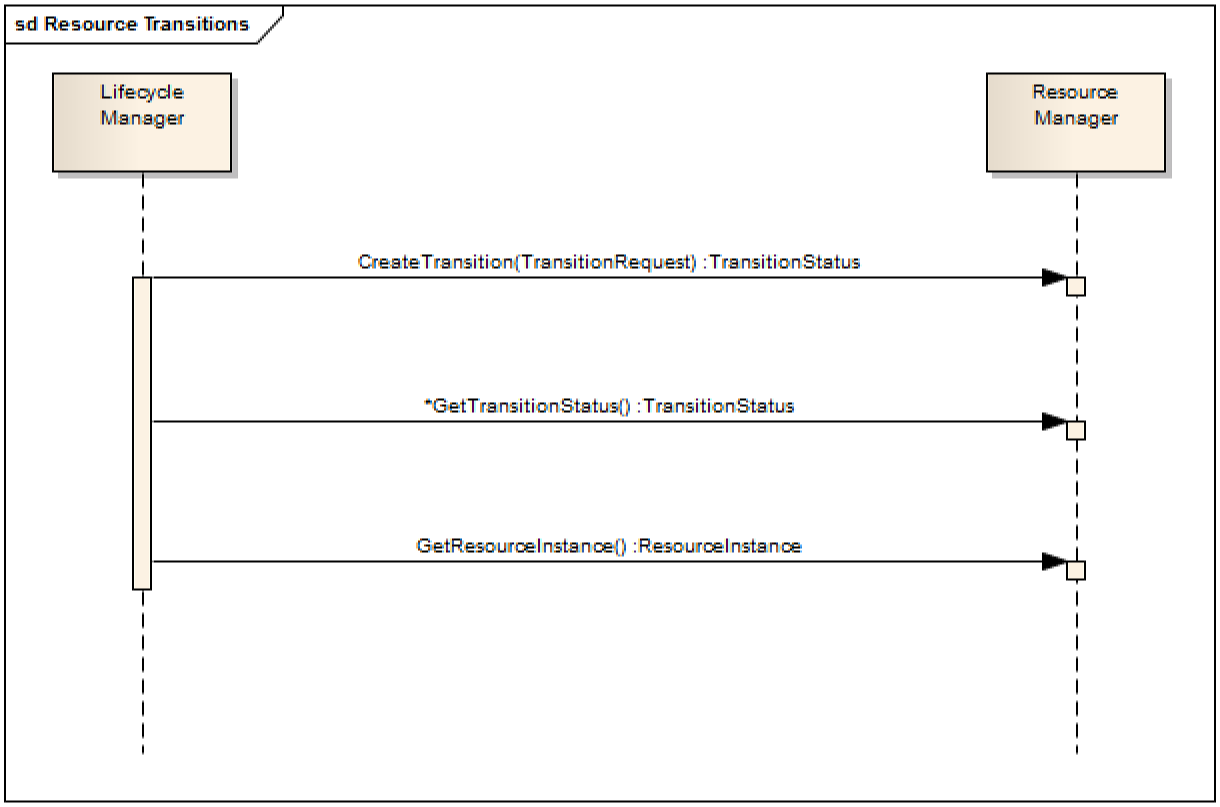
sd Resource Transitions
Lifecycle Manager
Resource Manager
CreateTransition(TransitionRequest) :TransitionStatus
*GetTransitionStatus() :TransitionStatus
GetResourceInstance() :ResourceInstance

Figure 4 - Resource Transition (Synchronous) Message Interaction Pattern

1.3.4 Resource Transitions (Asynchronous)

As part of an assembly transition, the Lifecycle Manager may call out to a Resource Manager to transition or perform an operation on a Resource. This is the interaction pattern for this scenario when the Resource Manager supports asynchronous response messages.

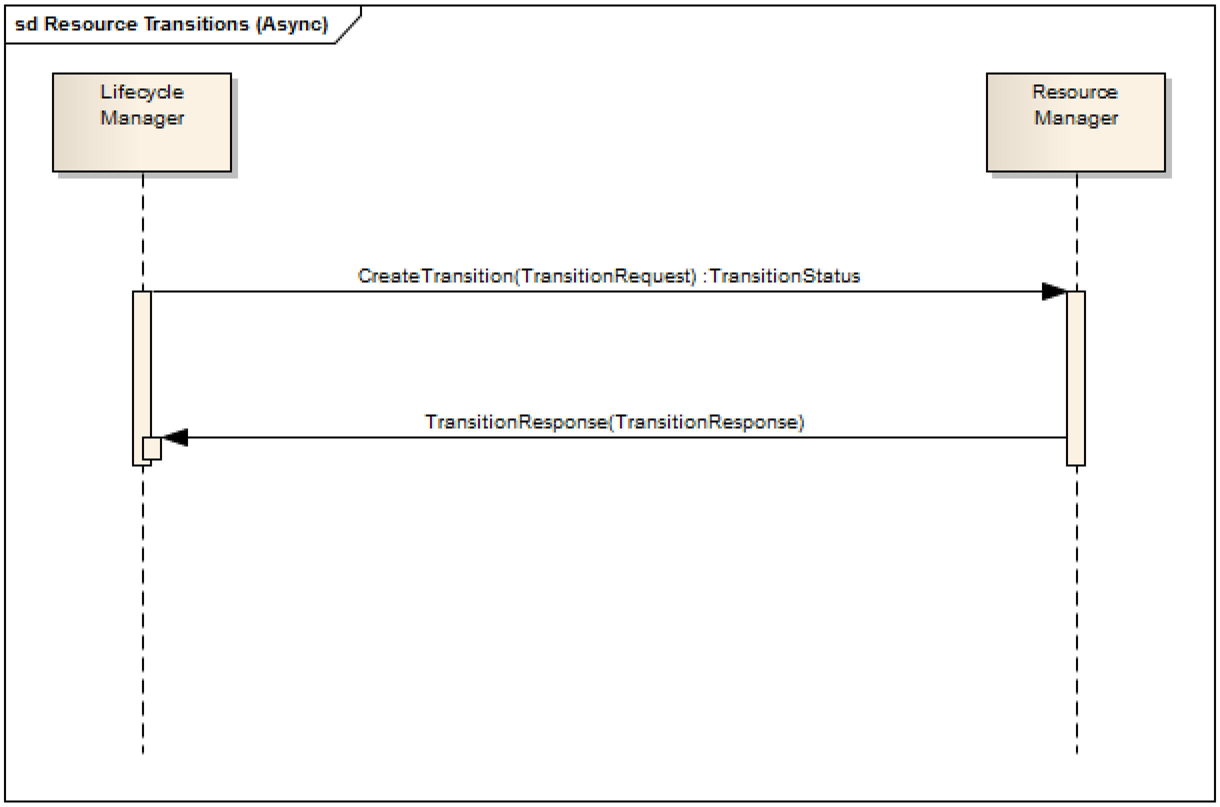
sd Resource Transitions (Async)
Lifecycle Manager
Resource Manager
CreateTransition(TransitionRequest) :TransitionStatus
TransitionResponse(TransitionResponse)

Figure 5 - Resource Transition (Asynchronous) Message Interaction Pattern

2 API Definition

2.1 Resource Manager Configuration

2.1.1 Get Resource Manager Configuration

Returns high-level information about the configuration of this Resource Manager.

This endpoint is called when onboarding the Resource Manager.

2.1.1.1 Request Format

Aspect	Value
Endpoint URL	/configuration
HTTP Method	GET
Parameters	None

2.1.1.2 Response Format

Aspect	Value
Return Code	200 OK

2.1.1.3 *Example Response*

```
{
        "name": "default-rm::dev",
        "version": "1.0.0",
        "supportedApiVersions": [ "1.0" ],
        "supportedFeatures": {
                "AsynchronousTransitionResponses": "true"
        },
        "properties": {
                "responseKafkaConnectionUrl": "zookeeper://localhost:2181",
                "responseKafkaTopicName": "lm-responses"
        }

}
```

2.2 Resource Type Configuration

2.2.1 List Resource Types

Returns a list of all resource types managed by this Resource Manager

NOTE: The descriptor is not returned in this list.

Allowable states for the resource types are as follows:

UNPUBLISHED, PUBLISHED, DELETED

2.2.1.1 *Request Format*

Aspect	Value
Endpoint URL	/types
HTTP Method	GET
Parameters	None

2.2.1.2 *Response Format*

Aspect	Value
Return Code	200 OK

2.2.1.3 *Example Response*

```
[
        {
                "name": "default-rm::Resource::Network",
                "state": "PUBLISHED",
                "createdAt": "2017-05-01T11:22:33Z",
                "lastModifiedAt": "2017-05-04T12:13:14+01:00"
        }
]
```

2.2.2 Get Resource Type

Returns information about a specific resource type, including its YAML descriptor.

2.2.2.1 Request Format

Aspect	Value
Endpoint URL	/types/{name}
HTTP Method	GET
Parameters	name – Unique name for the resource type requested

2.2.2.2 Response Format

Aspect	Value
Return Code	200 OK

2.2.2.3 Example Response

```
{
        “name”: “default-rm::Resource::Network”,
        “descriptor”: “YAML Descriptor for this resource type”,
        “state”: “PUBLISHED”,
        “createdAt”: “2017-05-01T11:22:33Z”,
        “lastModifiedAt”: “2017-05-04T12:13:14+01:00”
}
```

2.3 Resource Topology

2.3.1 List Deployment Locations

Returns a list of all deployment locations available to this Resource Manager

2.3.1.1 Request Format

Aspect	Value
Endpoint URL	/topology/deployment-locations
HTTP Method	GET
Parameters	None

2.3.1.2 Response Format

Aspect	Value
Return Code	200 OK

2.3.1.3 Example Response

```
[
        {
                "name": "dev-cloud",
                "type": "default-rm::Cloud"
        },
        {
                "name": "test-cloud",
                "type": "default-rm::Cloud"
        }
]
```

2.3.2 Get Deployment Location

Returns information for the specified deployment location

2.3.2.1 Request Format

Aspect	Value
Endpoint URL	/topology/deployment-locations/{name}
HTTP Method	GET
Parameters	name – Unique name for the deployment location requested

2.3.2.2 Response Format

Aspect	Value
Return Code	200 OK

2.3.2.3 Example Response

```
{
        "name": "dev-cloud",
        "type": "default-rm::Cloud"
}
```

2.3.3 Search for Resource Instances

Searches for resource instances managed within the specified deployment location.

The search can be restricted by the type of the resources to be returned, or a partial match on the name of the resources.

2.3.3.1 Request Format

Aspect	Value
Endpoint URL	/topology/deployment-locations/{name}/instances
HTTP Method	GET
Parameters	name – Unique name for the deployment location

	instanceType – Limits results to be of this resource type (optional, exact matches only) instanceName – Limits results to contain this string in the name (optional, partial matching)

2.3.3.2 Response Format

Aspect	Value
Return Code	200 OK

2.3.3.3 Example Response

```
[
        {
                "resourceId": "default-rm://c675e0bd-9c6c-43ca-84bf-2c061d439c6b",
                "resourceName": "dev-network",
                "resourceType": "default-rm::Resource::Network",
                "resourceManagerId": "default-rm::dev",
                "deploymentLocation": "dev-cloud",
                "properties": {
                        "propertyName": "propertyValue"
                },
                "createdAt": "2017-05-01T12:00:00Z",
                "lastModifiedAt": "2017-05-01T12:00:00Z"
        }
]
```

2.3.4 Get Resource Instance

2.3.4.1 Returns information for the specified resource instance

2.3.4.2 Request Format

Aspect	Value
Endpoint URL	/topology/instances/{id}
HTTP Method	GET
Parameters	id – Unique id for the resource instance

2.3.4.3 Response Format

Aspect	Value
Return Code	200 OK

2.3.4.4 Example Response

```
{
        "resourceId": "default-rm://c675e0bd-9c6c-43ca-84bf-2c061d439c6b",
        "resourceName": "dev-network",
        "resourceType": "default-rm::Resource::Network",
        "resourceManagerId": "default-rm::dev",
        "deploymentLocation": "dev-cloud",
        "properties": {
                "propertyName": "propertyValue"
        },
        "createdAt": "2017-05-01T12:00:00Z",
        "lastModifiedAt": "2017-05-01T12:00:00Z",
        "internalResourceInstances": [
                {
                        "id": "3cb7822b-fc44-46ab-8072-9c65cd778d1f",
                        "name": "MGMT-NETWORK",
                        "type": "OpenDaylight::PrivateNetwork"
                }
        ]
}
```

2.4 Resource Lifecycle Management

2.4.1 Create Resource Transition

Requests this Resource Manager performs a specific transition against a resource

2.4.1.1 Request Format

Aspect	Value
Endpoint URL	/lifecycle/transitions
HTTP Method	POST
Parameters	None

2.4.1.2 Example Request Body

```
{
        "resourceManagerId": "default-rm::dev",
        "deploymentLocation": "dev-cloud",
        "resourceType": "default-rm::Resource::Network",
        "transitionName": "Transition::Install",
        "resourceName": "dev-network-c675e0bd",
        "properties": {
                "propertyName": "propertyValue"
        },
        "context": {}
}
```

2.4.1.3 Response Format

Allowable states for the request state are as follows:

PENDING, IN_PROGRESS, COMPLETED, CANCELLED, FAILED

Aspect	Value
Return Code	202 ACCEPTED

2.4.1.4 *Example Response*

```
{
        "requestId": "80fc4a66-7e92-41f8-b4bb-7cb98193f5fa",
        "requestState": "PENDING",
        "context": {
              "AsynchronousTransitionResponses": "true"
        }
}
```

2.4.2 Get Resource Transition Status

Returns information about the specified transition or operation request

2.4.2.1 *Request Format*

Aspect	Value
Endpoint URL	/lifecycle/transitions/{id}/status
HTTP Method	GET
Parameters	id – Unique identifier for the resource transition

2.4.2.2 *Response Format*

Aspect	Value
Return Code	200 OK

2.4.2.3 *Example Response*

```
{
        "requestId": "80fc4a66-7e92-41f8-b4bb-7cb98193f5fa",
        "requestState": "COMPLETED",
        "requestStateReason": "Transition successfully completed in 324 msecs",
        "startedAt": "2017-05-01T12:00:00Z",
        "finishedAt": "2017-05-01T12:00:00Z"
        "context": {
              "AsynchronousTransitionResponses": "true"
        }
}
```

2.5 Resource Type Configuration (Asynchronous)

Optionally, a resource manager can emit events when the definition of a resource type changes, or a new resource type is created/deleted.

2.5.1 Resource Type Update Message Example

```
{
        “name”: “default-rm::Resource::Network”,
        “descriptor”: “YAML Descriptor for this resource type”,
        “state”: “PUBLISHED”,
        “createdAt”: “2017-05-01T11:22:33Z”,
        “lastModifiedAt”: “2017-05-04T12:13:14+01:00”
}
```

2.6 Resource Lifecycle Management (Asynchronous)

Optionally, a resource manager can choose to emit responses when transitions are completed (either successfully or not). It is recommended that this method is used to avoid the Lifecycle Manager needing to poll for the status periodically.

2.6.1 Resource Transition Response Message Example

```
{
        "requestId": "80fc4a66-7e92-41f8-b4bb-7cb98193f5fa",
        "resourceManagerId": "default-rm::dev",
        "deploymentLocation": "dev-cloud",
        "resourceType": "default-rm::Resource::Network",
        "transitionName": "Transition::Install",
        "resourceInstance": {
                "resourceId": "default-rm://c675e0bd-9c6c-43ca-84bf-2c061d439c6b",
                "resourceName": "dev-network",
                "resourceType": "default-rm::Resource::Network",
                "resourceManagerId": "default-rm::dev",
                "deploymentLocation": "dev-cloud",
                "properties": {
                        "propertyName": "propertyValue"
                },
                "createdAt": "2017-05-01T12:00:00Z",
                "lastModifiedAt": "2017-05-01T12:00:00Z",
                "internalResourceInstances": [
                        {
                                "id": "3cb7822b-fc44-46ab-8072-9c65cd778d1f",
                                "name": "MGMT-NETWORK",
                                "type": "OpenDaylight::PrivateNetwork"
                        }
                ]
        },
        "context": {},
        "requestState": "COMPLETED",
        "requestStateReason": "Transition successfully completed in 324 msecs",
        "startedAt": "2017-05-01T12:00:00Z",
        "finishedAt": "2017-05-01T12:00:00Z"
}
```