logo

RDS Status Service Design

Table of Contents

[Document Header 2](#_Toc444439524)

[1.1 Document Information 2](#_Toc444439525)

[1.1.1 General 2](#_Toc444439526)

[1.1.2 Revisions 2](#_Toc444439527)

[2. Feature overview 3](#_Toc444439528)

[3. Proposed Solution 3](#_Toc444439529)

[3.1 RDS Status service 3](#_Toc444439530)

[3.2 RDS Status CLI 4](#_Toc444439531)

[4. Sequences 4](#_Toc444439532)

[4.1 Special cases 4](#_Toc444439533)

[4.2 Sequence diagrams 4](#_Toc444439534)

[5. ReST APIs 5](#_Toc444439535)

[5.1 Consts 5](#_Toc444439536)

[5.2 API documentation 7](#_Toc444439537)

[6. DB Structure 10](#_Toc444439538)

[6.1 Zone\_Resource\_Type\_Status Table 10](#_Toc444439539)

[7. Testing 11](#_Toc444439540)

[8. Open Issues 11](#_Toc444439541)

# Document Header

## Document Information

General

|  |  |
| --- | --- |
| Author/s |  |
| Date Created | 24.2.2016 |

Revisions

|  |  |  |  |
| --- | --- | --- | --- |
| Revision No. | Date | Author | Description |
| 1 | 24.2.2016 |  | DRAFT, Partial |
| 2 | 28.2.2016 | Kobi Mansharov | Update after review |

Changes description

|  |  |  |
| --- | --- | --- |
| Date | Name | Description |
| 2/3/2016 | Kobi | Added CLI command documentation, added clarification for non-valid resource-type from LCP. |

# Feature overview

* This design covers ORM RDS Status service, it’s ReST API and database structure.
* The scope of this document is the following USs: US618237, 612434, 620130, 618234, 618235.

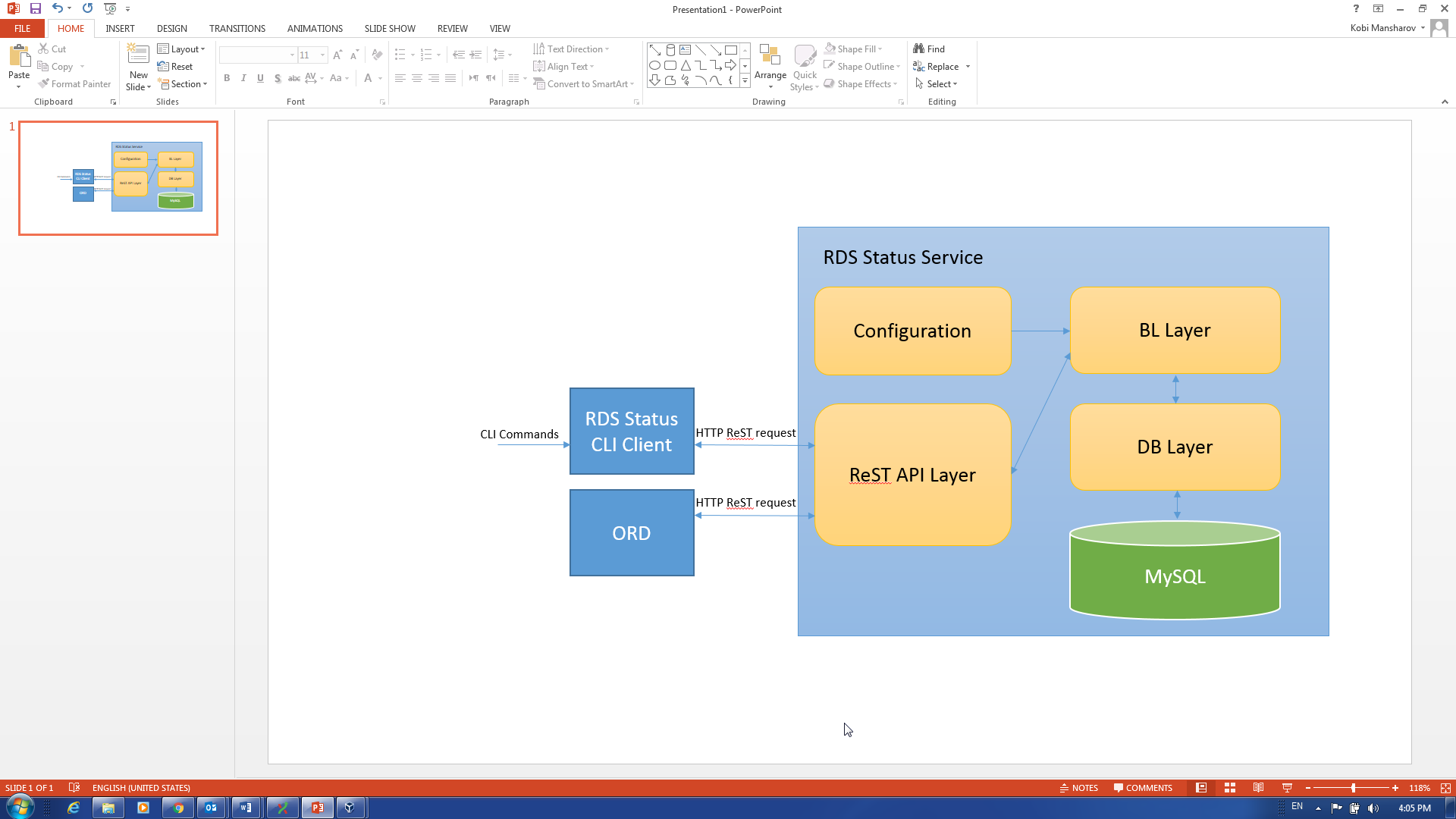
# Proposed Solution

The proposed solution consists of 2 projects:

## RDS Status service

This is a pecan based service that handles the ReST requests from Clients such as CLI or any ReST client (testing), and ORD on LCP.

The service structure consists of the OpenStack project structure and utilizes some of OpenStack modules.



* **Configuration** - this module will handle configuration file, read configuration values and will serve the various modules.
* **BL Layer** – this module will be the controller and will be the mid layer between DB and ReST Layer
* **ReST API Layer** – this layer handles the ReST requests from various clients.
* **DB Layer** – this is an abstraction over SQL DB.
* **SQL DB** – We will use **MySQL** DB for storing the resource type status data.

## RDS Status CLI

This is a CLI client that its structure is as of other OpenStack CLI clients.

# Sequences

## Special cases

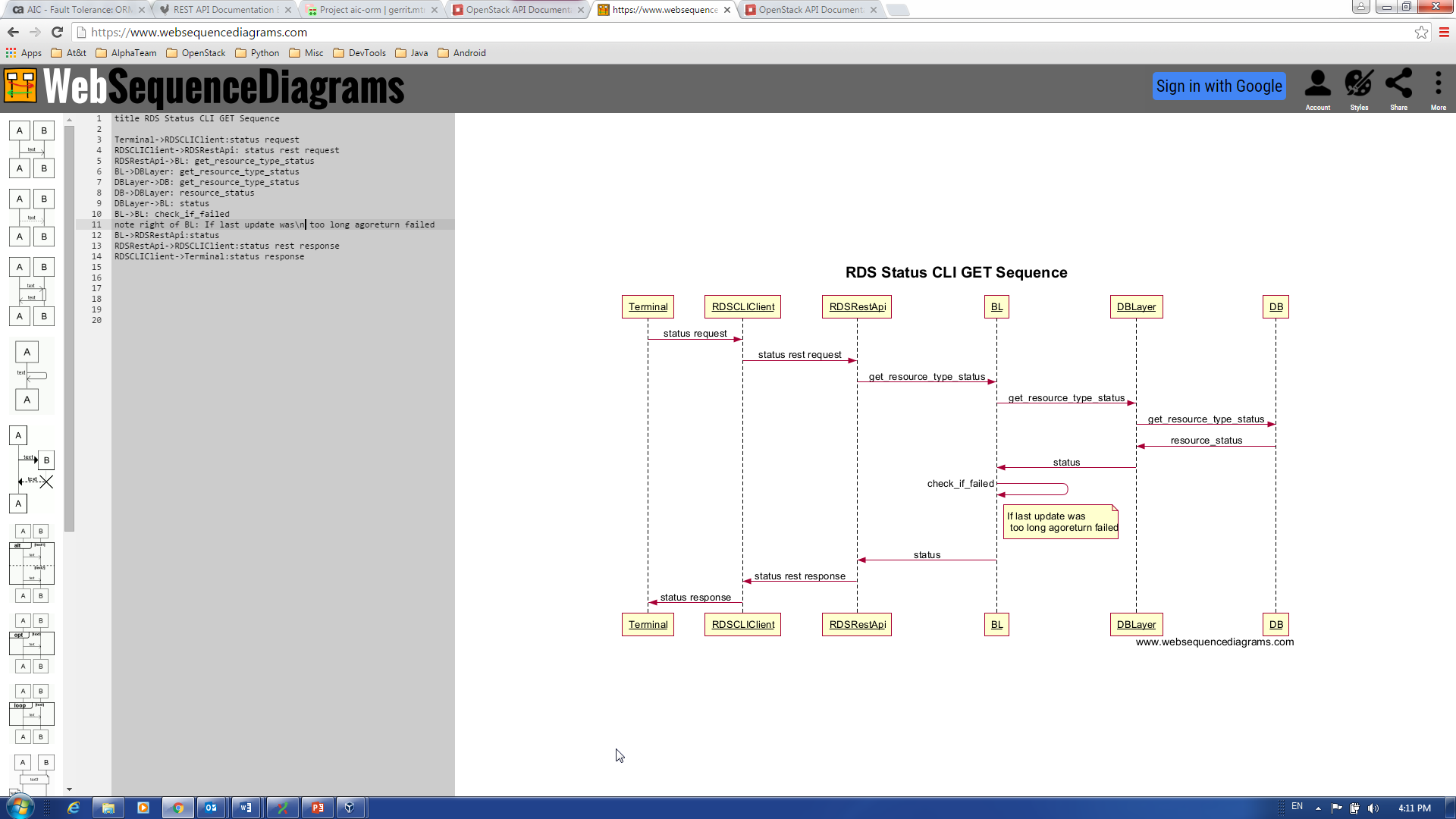
* When LCP/ORD sends a non-periodic update we should do the following:
  + We should save a non-periodic updated
  + We should add a periodic update with a new timestamp.
* When a Get resource type status message received, **only** then we check the period from last update and in case it is too long, FAILED will be returned.

The FAILED value will not be stored in DB.

* LCP/ORD can send an update with a non-valid resource-type, this is valid.

The update will be stored in DB, but Get request on this resource will return error.

## Sequence diagrams



WebSequence code:

title RDS Status CLI GET Sequence

Terminal->RDSCLIClient:status request

RDSCLIClient->RDSRestApi: status rest request

RDSRestApi->BL: get\_resource\_type\_status

BL->DBLayer: get\_resource\_type\_status

DBLayer->DB: get\_resource\_type\_status

DB->DBLayer: resource\_status

DBLayer->BL: status

BL->BL: check\_if\_failed

note right of BL: If last update was\n too long agoreturn failed

BL->RDSRestApi:status

RDSRestApi->RDSCLIClient:status rest response

RDSCLIClient->Terminal:status response

# ReST APIs

## Consts

### Resource types

One of the following resource types:

* images
* tenants
* flavors
* users

### Resource status values

One of the following resource types:

* NA – no status is available for this resource type
* FAILED - Resource type was not updated for too long (TBD).
* UP\_TO\_DATE – The resource type is up to date
* PENDING – Zone updated that the resource type is waiting for update, or in update progress.

### Errors

* 400 - Bad Request for malformed requests
* 401 - Unauthorized for requests without the proper authentication
* 403 - Forbidden for un-authorized access
* 404 - Not Found for GET of unknown resource-type

### Zone ids

Zone is equivalent to LCP.

The list of available Zones should be requested from AIC whenever needed.

## API documentation

### Get resource type status in zone

|  |  |
| --- | --- |
| Name | Get resource type status |
| Type | GET |
| Path | /v1/status/zone/{zone\_id}/type/{resource\_type} |
| Description | Get the latest status of a given resource type in a given zone |
| Normal Response Code | 200 OK |
| Error Response Code | See 5.1.4 for errors. |

Request Parameters:

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| resource\_type | Plain | String | See 5.1.2 for resource types. |
| zone\_id | Plain | String | Return resource type status for the given zone. |

Request Example:

**/v1/status/zone/12345/type/images**

Response Parameters:

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| status | Plain | String | See 5.1.3 for resource status values. |
| timestamp | Plain | Long | Time of status update(UTC format) |

Response Example:

{

“status”: “UP\_TO\_DATE”,

“timestamp”: “1/1/2016 12:35:00”

}

### Get status in zone for all resource-types

|  |  |
| --- | --- |
| Name | Get all resource types status |
| Type | GET |
| Path | /v1/status/zone/{zone\_id} |
| Description | Get the latest status of all resource types in a given zone |
| Normal Response Code | 200 OK |
| Error Response Code | See 5.1.4 for errors. |

Request Parameters:

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| zone\_id | Plain | String | Return resource type status for the given zone. |

Request Example:

**/v1/status/zone/12345**

Response Parameters:

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| status | Plain | String | See 5.1.3 for resource status values. |
| timestamp | Plain | DateTime | Time of status update in UTC format |

Response Example:

{

“status”: [

{

“type”: “images”,

“status”: “UP\_TO\_DATE”,

“timestamp”: “1/1/2016 12:35:00”

},

{

}

]

}

### Update resource-type status

|  |  |
| --- | --- |
| Name | Update resource type status |
| Type | POST |
| Path | /v1/status/zone/{zone\_id}/type/{resource\_type} |
| Description | Update resource type status for zone |
| Normal Response Code | 201 Created |
| Error Response Code | See 5.1.4 for errors. |

Request Parameters:

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Style | Type | Description |
| resource\_type | Plain | String | See 5.1.2 for resource types. |
| zone\_id | Plain | Number | The zone from which the update came from |
| status | Plain | String | See 5.1.3 for resource status values. |
| is\_periodic | Plain | Boolean | True, False |
| timestamp | Palin | Long | Time of status update in miliseconds since 1/1/1970 |

Request example:

/v1/status/zone/54321/type/images

{

“status”: “UP\_TO\_DATE”,

“is\_periodic”: “True”,

“timestamp”: “12345678”

}

# DB Structure

## Zone\_Resource\_Type\_Status Table

Description:

This table holds resource type status updates as received from zones.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Key type | Attribute type | Description |
| id | Primary | number | Id of the update |
| zone\_id |  | string | Id of the zone the status refers to |
| timestamp |  | Long | Timestamp of the status update in miliseconds since 1/1/1970 |
| is\_periodic |  | boolean | Is the status periodic update |
| resource\_type |  | string | Resource type |
| status |  | string | Resource type status |

# CLI client

After setting up the CLI client, in order to use it we simply run the commands in terminal :

## Get Resource Type Status

Command:

orm status –zone {zone\_id} –resource-type {resource-type}

Example:

orm status –zone 3 –resource-type images

Output:

{“status”: “UP\_TO\_DATE”, “timestamp”: 1456911916000}

# Testing

# Open Issues

* We will also need the error codes for CLI, errors to string translations.
* Authentication and security was not handled