CloudPortal Client API

The CloudPortal Client gets all of its data from asynchronous (AJAX) calls over HTTP through a REST interface to the CloudPortal Server. The data returned will always be JSON-formatted documents. This page describes that API. **Note:** The goal of the API is *convenience* for the client, and this goal takes priority over any pedantic approach to REST-purity.

> **This Specification will Change!**
> The documentation on this page is *initial thoughts and ideas* and not a complete specification. We will change this over the next few sprints and update this page accordingly.

REST CRUD Operations

REST is great at CRUD, since REST implies *resources*, and operations on resources are CRUD. We are planning on using the Express-resource middleware to Node.js to do the dirty work. For a given resources, **xyz**, the following calls are supported:

- `GET /api/xyz` - Returns all instances of the resource. Parameters include:
  - `max` - The total number of results to return. Defaults to 20.
  - `offset` - The position of the first result to return after sorting has taken place
  - `sort` - The name of the field to sort the results by, as in "name" or "title"
  - `order` - Can either be "asc" (ascending) or "desc" (descending)
- `POST /api/xyz` - Creates a new instance. The body of the post contains a JSON formatted object containing the values.
- `GET /api/xyz/ID` - Returns a single instance based on the value of the `ID` parameter. While this is normally a small number, don't expect IDs to be in anything less than a **long**.
- `PUT /api/xyz/ID` - Changes **some** values for an instance referenced by `ID`. The body contains changed values, and attributes not specified, will not be changed.
- `DELETE /api/xyz/ID` - Deletes an instance referenced by `ID`.

Typical Behavior

The following are typical behaviors that each REST operation request adheres.

POST Body

When giving a `POST` (or `PUT`) to create a new instance, keep in mind that the bulk of the body is assigned to a `data` key field. For instance, to create a new `system-service`, you would specify the body like:

```
{
  "data": {
      "name": "blingbling",
      "description": "Bling Bling Container",
      "srvversion": "1.0"
  }
}
```

Note: Do not specify an `id` key field, as that will be generated and returned for the `POST` commands.

Error Messages

Each request will contain a key, `success`, which is either `true` if all went well, or `false` if an error occurred.

If the request ended in error, the response will contain details of the situation in a `message` key field, as in:

```
{
  "success":false,
  "message":"Object with id=3 not found"
}
```

Multiple Results

If the request returns multiple results (which is typical when issuing a `GET` request without an specific `ID` value), the total number of results *returned* is assigned to the value of `count`. For instance, if you ask for all `runtime` instances, you might get the following result:

```
{
  "success": true,
  "data": [
    {
      "id": 1,
      "name": "node"
      "description": "Node.js",
      "fwversion": "0.4.5",
    },
    {
      "id": 2,
      "name": "java"
      "description": "Java 6",
      "fwversion": "1.6",
    },
    {
      "id": 3,
      "name": "ruby18"
      "description": "Ruby 1.8.7",
      "fwversion": "1.8.7",
    },
    {
      "id": 4,
      "name": "ruby19"
      "description": "Ruby 1.9",
      "fwversion": "1.9.2p180",
    }
  ],
  "count": 4
}
```

REST Resources

This section contains all of the *resources* that are currently available for clients.

Team

Example: `http://localhost:8080/cloudportal/api/team/1`

Body:


```
{
  "success": true,
  "data": {
    "id": 1,
    "name": "CloudEco Internal Team",
    "description": "A team that contains all of the team members as administrators.",
    "accounts": [ 1, 3, 8 ],
    "applications": [ 1, 6, 12 ],
    "services": [ 1, 4 ]
  }
}
```


Runtime

Example to get all entries: `http://localhost:8080/cloudportal/api/runtime/`

Results:


```
{
  "success": true,
  "data": [
    {
      "id": 1,
      "name": "node"
      "description": "Node.js",
      "fwversion": "0.4.5",
    },
    {
      "id": 2,
      "name": "java"
      "description": "Java 6",
      "fwversion": "1.6",
    },
    {
      "id": 3,
      "name": "ruby18"
      "description": "Ruby 1.8.7",
      "fwversion": "1.8.7",
    },
    {
      "id": 4,
      "name": "ruby19"
      "description": "Ruby 1.9",
      "fwversion": "1.9.2p180",
    }
  ],
  "count": 4
}
```


Application

Example to get a specific application with an ID of 1: `http://localhost:8080/cloudportal/api/application/1`

```
{
  "success": true,
  "data": {
    "id": 1,
    "name": "mongrue",
    "description": "A simple, free-form REST interface to a MongoDB database instance.",
    "instances": 1,
    "memory": -1,
    "runtime": 1,
    "url": "mongrue"
    "team": 1,
  }
}
```


SystemService

A number of available services are available for a given customer, to get a list of all of these, you would issue:

`http://localhost:8080/cloudportal/api/system-service`

Which would return:


```
{
  "success": true,
  "data": [
    {
      "id": 1,
      "name": "mongodb",
      "description": "MongoDB NoSQL store",
      "srvversion": "1.8"
    },
    {
      "id": 2,
      "name": "mysql",
      "description": "MySQL database service",
      "srvversion": "5.1"
    },
    {
      "id": 3,
      "name": "postgresql",
      "description": "PostgreSQL database service (vFabric)",
      "srvversion": "9.0"
    },
    {
      "id": 4,
      "name": "rabbitmq",
      "description": "RabbitMQ messaging service",
      "srvversion": "2.4"
    },
    {
      "id": 5,
      "name": "redis",
      "description": "Redis key-value store service",
      "srvversion": "2.2"
    }
  ],
  "count": 5
}
```


ProvisionedService

Once a customer has provisioned a service and made it available to her applications, you could get information about a particular one via:

`http://localhost:8080/cloudportal/api/service/1`

Which would return the following:

```
{
  "success": true,
  "data": {
    "id": 1,
    "name": "mongodb-800ab",
    "service": 1,
    "team": 1
  }
}
```

Accounts

To view information about a particular user account, for instance, with an ID of 1, use:

`http://localhost:8080/cloudportal/api/account/1`

```
{
  "success": true,
  "data": {
    "id": 1,
    "team": 1,
    "username": "howard"
    "name": "Howard Abrams",
    "description": "Maker of mudpies",
    "accountExpired": false,
    "accountLocked": false,
    "addresses": [ 1 ],
    "admin": true,
    "authorities": [ 1 ],
    "emails": [ 1 ],
    "enabled": true,
    "password": "focker",
    "passwordExpired": false,
    "phones": [ 1, 2 ],
    "preferences": null,
  }
}
```

Technical Details

The resources exposed through the REST API are described in the domain classes located in `ce-portal-server/grails-app/domain/com/cloudeco/portal`. These classes are tagged as being exposed through the REST interface with the following code:

```
class Xyz {
   static expose = 'xyz'
   // ...
```

Exposing the domain classes through a REST API is currently being done via a Grails plugin.

Note: Since this plugin is quite *automatic*, it may not be sufficient for all our needs, and we may end up creating controllers for each class to expose the data as REST in a way that we can control it better.