



# Module 7: Automating Administration with the MMC REST API



# Goals

The screenshot displays the MuleSoft API Client interface. On the left, a sidebar shows a collection named "ESPOps MMC REST API" with 4 requests. The "Collections" tab is active, listing three GET requests and one POST request, "Add Max Server to MMC". The main workspace shows the details of the selected POST request to "http://localhost:8585/mmc-3.7.0/api/servers". The request body is in raw JSON format, containing server details. Below the request, the response is shown in "Pretty" JSON format, indicating a "201 Created" status with a response time of 5799 ms. The response body contains a detailed object with fields like name, id, agentUrl, groups, href, version, and status.

Search

Builder Runner Import

Get Server ... Get Server I... New tab Get Server ... Get All Serv... Add Max Ser... Add Max Ser... No en

History Collections

ESPOps MMC REST API  
3 Aug at 4:11PM • 4 requests

GET Get Server Info

GET Get Server Groups

GET Get All Servers

POST Add Max Server to MMC

POST http://localhost:8585/mmc-3.7.0/api/servers Params Send

Authorization Headers (2) Body Pre-request script Tests

☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON (application/json)

```
1 {  
2   "name": "Max",  
3   "agentUrl": "http://localhost:7777/mmc-support",  
4   "groupIds": ["cf412459-32a5-4a3f-89bb-01bb35cd9b42"]  
5 }
```

Body Cookies Headers (5) Tests Status 201 Created Time 5799 ms

Pretty Raw Preview JSON

```
1 {  
2   "name": "Max",  
3   "id": "local$64e0bfc4-f7c2-4fee-b477-fcef240826a8",  
4   "agentUrl": "https://localhost:7777/mmc-support",  
5   "groups": [  
6     {  
7       "name": "Development",  
8       "href": "http://localhost:8585/mmc-3.7.0/api/serverGroups/cf412459-32a5-4a3f-89bb-01bb35cd9b42"  
9     },  
10  ],  
11   "href": "http://localhost:8585/mmc-3.7.0/api/servers/local$64e0bfc4-f7c2-4fee-b477-fcef240826a8",  
12   "version": "3.7.0",  
13   "status": "OK",  
14   "runningServices": 1
```

# Objectives

- Use the MMC REST API to:
  - Create server groups and add servers to a server group
  - Deploy applications from the MMC repository
  - Deploy a deployment to Mule servers
- Use the MMC REST API to:
  - Manage Mule servers
  - Manage Mule servers to list all servers registered with an MMC server
  - Add and remove servers to an MMC server via the MMC REST API

# Topics

- Introduction to REST API for MMC
- Adding a Server
- Deploying an Application
- Undeploying an Application
- Restarting the Server
- Working with Files

# Requirements

- These tools are needed if you wish to try out the examples in this module:
  - **Postman** for Chrome or a similar add-on for your browser to make REST calls
  - A text editor or notepad application

# REST API for MMC - Overview



# Introduction to the REST API

- You can programmatically access the MMC's functionality using REST APIs:
  - Add /api/ to the URL to automate a set of services:
    - Server management
    - Deployments and applications management
    - Cluster management
    - Gathering of information and statistics
- The API is meant to automate all the GUI features:
  - If there is a button, a menu or a tab, there should be an API call
  - Unfortunately there are areas where this is not the case yet:
    - Flow analyzer and business events don't have an API yet.

# Module Outline

- Replace MMC GUI operations with REST requests
- We won't cover the whole API here:
  - It would take days to go through it, and would be extremely boring
  - Instead, we'll cover a small subset and then look at how to read the docs
- Some of the REST calls will have links to the Mule Documentation
  - <http://www.mulesoft.org/documentation/display/current/REST+API+Reference>



# Adding a Server



# The REST API URL

HTTP://host:port/mmc/api/[module]/[resourceID]/[action]/[parameters]

- **Module** is the functional area you want to address:
  - clusters
  - servers
  - serverGroups
  - deployments
  - repository
  - usergroups
- **ResourceID** is a UUID that points to a resource:
  - E.g.: A specific server, or a specific deployment
  - Some modules/actions do not require a Resource ID (e.g. to list servers)
- **Action** will tell MMC what to do with the module or resource
- Optional **parameters** will customize the action

# Adding a Mule Server

To add/register a server to MMC using Postman:

1. Set the URL to **http://localhost:8585/mmc-3.7.0/api/servers**
2. Set the HTTP method to **POST**
3. Add a header with “**Content-Type : application/json**”
4. In the request body, define your **server's name**, **URL** and the server **group** which it will belong to (e.g. Development or Test) in JSON format
5. Fill the Basic Authentication with **Username=admin** and **Password=admin**
  - This is the authentication requested to log in to MMC as described in Module 2

# Postman - Add Server request

<http://www.mulesoft.org/documentation/display/current/Servers#Servers-PAIR/REGISTER>

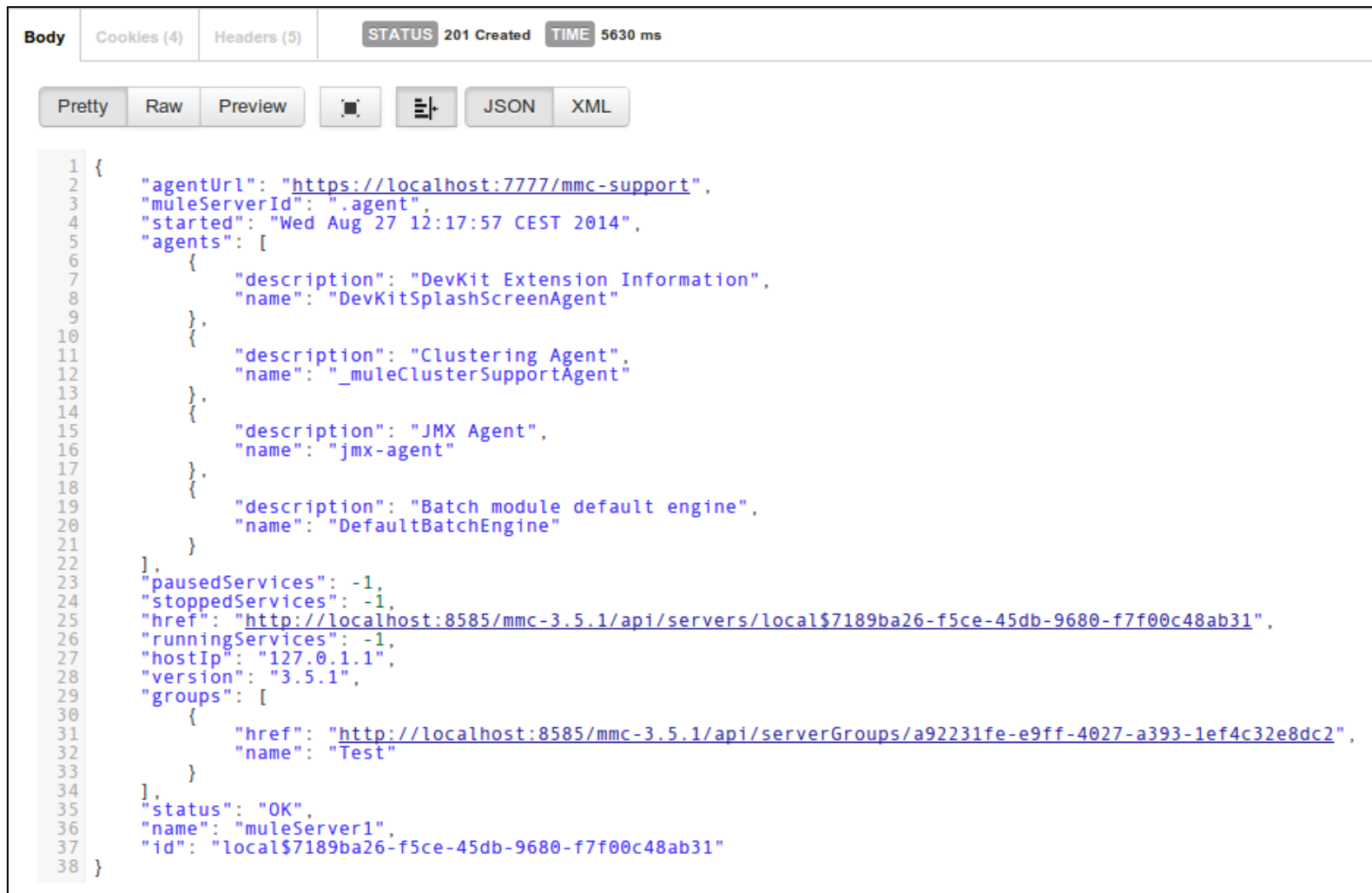
The image shows the Postman interface for configuring a Basic Auth request. The interface is divided into several sections:

- Authentication:** The 'Basic Auth' tab is selected. The 'Username' field contains 'admin' and the 'Password' field contains '\*\*\*\*\*'. A blue number '5' is next to the password field. A 'Refresh headers' button is below the password field. A 'Note' on the right states: 'The authorization header will be generated and added as a custom header.'
- Request URL and Method:** The URL is 'http://localhost:8585/mmc-3.5.1/api/servers' (marked with a blue number '1'). The method is 'POST' (marked with a blue number '2'). There are buttons for 'URL params' and 'Headers (1)'.
- Content-Type:** The 'Content-Type' is set to 'application/json' (marked with a blue number '3'). There is a 'Manage presets' button.
- Body:** The 'Body' tab is selected, showing a JSON payload (marked with a blue number '4'):

```
1 {
2   "name" : "muleServer1",
3   "agentUrl" : "http://localhost:7777/mmc-support",
4   "groupIds" : ["a92231fe-e9ff-4027-a393-1ef4c32e8dc2"]
5 }
```
- Buttons:** At the bottom, there are buttons for 'Send', 'Preview', 'Add to collection', and 'Reset'.

# Postman - Add Server reply

<http://www.mulesoft.org/documentation/display/current/Servers#Servers-PAIR/REGISTER>



Body Cookies (4) Headers (5) STATUS 201 Created TIME 5630 ms

Pretty Raw Preview JSON XML

```
1 {
2   "agentUrl": "https://localhost:7777/mmc-support",
3   "muleServerId": ".agent",
4   "started": "Wed Aug 27 12:17:57 CEST 2014",
5   "agents": [
6     {
7       "description": "DevKit Extension Information",
8       "name": "DevKitSplashScreenAgent"
9     },
10    {
11      "description": "Clustering Agent",
12      "name": "_muleClusterSupportAgent"
13    },
14    {
15      "description": "JMX Agent",
16      "name": "jmx-agent"
17    },
18    {
19      "description": "Batch module default engine",
20      "name": "DefaultBatchEngine"
21    }
22  ],
23  "pausedServices": -1,
24  "stoppedServices": -1,
25  "href": "http://localhost:8585/mmc-3.5.1/api/servers/local$7189ba26-f5ce-45db-9680-f7f00c48ab31",
26  "runningServices": -1,
27  "hostIp": "127.0.1.1",
28  "version": "3.5.1",
29  "groups": [
30    {
31      "href": "http://localhost:8585/mmc-3.5.1/api/serverGroups/a92231fe-e9ff-4027-a393-1ef4c32e8dc2",
32      "name": "Test"
33    }
34  ],
35  "status": "OK",
36  "name": "muleServer1",
37  "id": "local$7189ba26-f5ce-45db-9680-f7f00c48ab31"
38 }
```

# MMC after using Postman

- We see that with the REST API call we did on Postman, we registered our server “muleServer1” to MMC
- Also notice that “muleServer1” was also added to the server group Test, which came from the “groupId” field

The screenshot shows the MuleSoft MMC interface. The top navigation bar includes tabs for Dashboard, Servers, Deployments, Applications, Flows, Flow Analyzer, Business Events, Alerts, and Administration. The 'Servers' tab is active, and the 'Test Servers' section is displayed. On the left, a sidebar shows a tree view of server groups: All (1), Development (0), Production (0), Staging (0), Test (1), and Unregistered (1). The 'Test (1)' group is selected. The main panel shows a table of servers. The table has columns for Name, Version, Location, Flows, and Groups. One server is listed: 'muleServer1' with version '3.5.1' and location '127.0.1.1', associated with the 'Test' group.

Name	Version	Location	Flows	Groups
muleServer1	3.5.1	127.0.1.1	- / - / -	Test

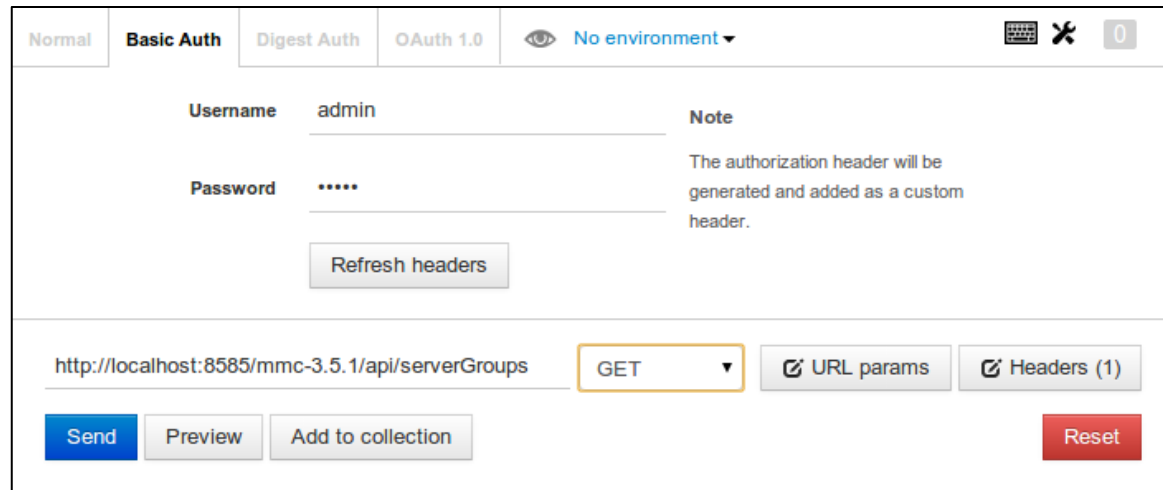
# Adding a Mule Server - Details

- Note that:
  - For this API call to work, your Mule Server instance and Tomcat instance need to be running
  - The port number (e.g. 7777) in the agentUrl field is the port that your current Mule Server instance is using
  - The groupIds field represents an array of server group IDs:
    - This can be left empty if you do not want to assign a group to your server
    - The following slides will demonstrate how to obtain the group ID used in our example

# Obtaining a Server Group ID

To get a server group ID using Postman:

1. Set the URL to **http://localhost:8585/mmc-3.7.0/api/serverGroups**
2. Set the HTTP method to **GET**
3. Fill the Basic Authentication with **Username=admin** and **Password=admin**



The screenshot shows the Postman interface with the following configuration:

- Auth:** Basic Auth
- Environment:** No environment
- Username:** admin
- Password:** \*\*\*\*\*
- Note:** The authorization header will be generated and added as a custom header.
- URL:** http://localhost:8585/mmc-3.5.1/api/serverGroups
- Method:** GET
- Buttons:** Send, Preview, Add to collection, Reset



# Walkthrough 7-1: MMC REST API

- List MMC Registers servers
- Register a new server into the MMC GUI
- Create a new Server Group
- Remove a server from MMC

# Deploying an Application



# Deployment Requirements

- To deploy an application, we need to POST this:  
<http://localhost:8585/mmc-3.7.0/api/deployments/{deploymentId}/deploy>
- {deploymentId} indicates that we need to create a deployment before we actually deploy it
- To create a deployment, we would need:
  - the ID of the server(s) we are deploying on
  - the ID of the application(s) and their respective version we want to deploy:
    - this implies that we need to upload our applications to the MMC repository beforehand

# Step-by-step

- These are the steps to do before we can deploy:
  1. Register a server to MMC (done in previous section)
  2. Get the server ID (store it in your notepad)
  3. Upload an application to the MMC repository
  4. Get the application version ID (store it in your notepad)
  5. Create a deployment
  6. Get the deployment ID (store it in your notepad)
  7. Deploy the deployment

## Step 4 - Get Application Version ID

- We could have taken the version ID from the reply when we uploaded the application. However, we can use another REST API call to get it.
- To obtain the version ID using Postman:
  1. Set the URL to **http://localhost:8585/mmc-3.7.0/api/repository**
  2. Set the HTTP method to **GET**
  3. Fill the Basic Authentication with **Username=admin** and **Password=admin**

## Step 4 –Request to get Application Version ID

- The request:

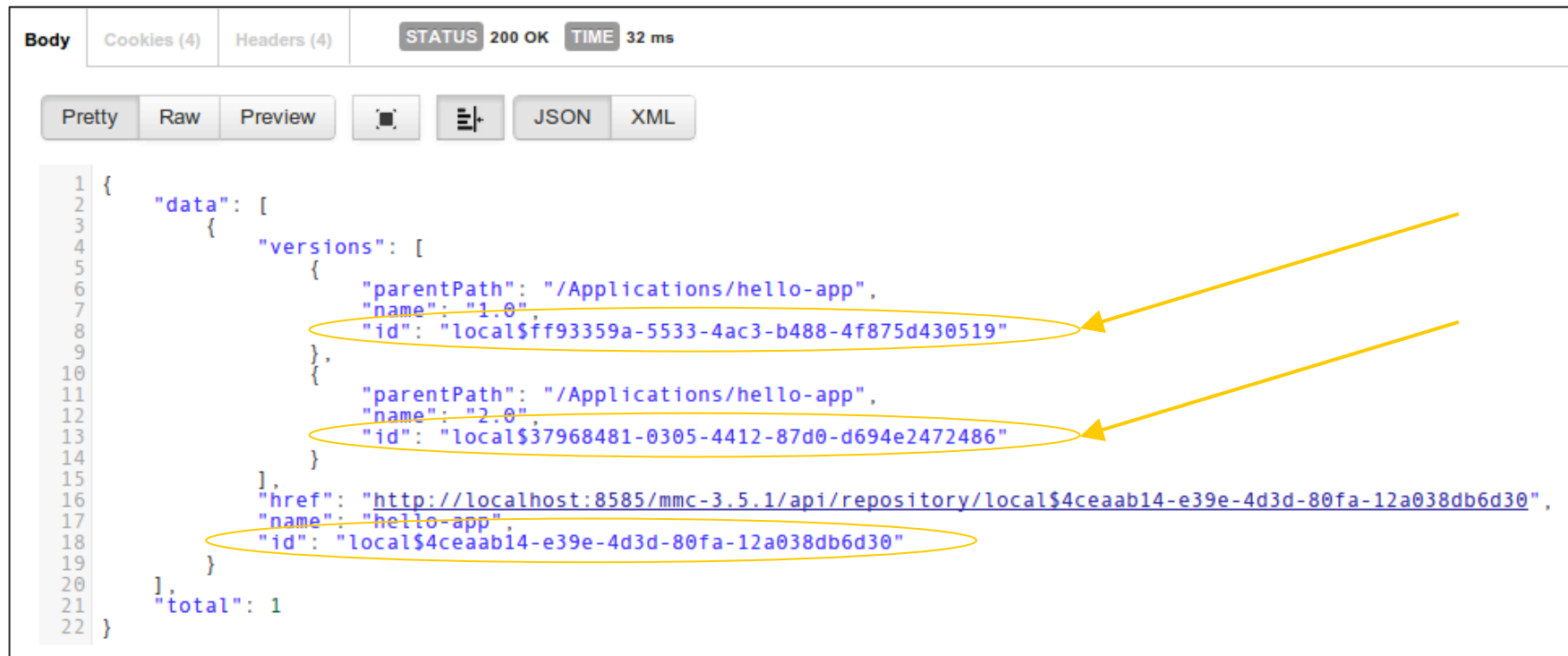
The screenshot displays the MuleSoft API Client interface. At the top, there are tabs for 'Normal', 'Basic Auth' (selected), 'Digest Auth', and 'OAuth 1.0'. To the right of these tabs is an eye icon, the text 'No environment', and a dropdown arrow. Further right are icons for a keyboard, a wrench, and a counter showing '0'.

Under the 'Basic Auth' tab, there are two input fields: 'Username' with the value 'admin' and 'Password' with masked characters '\*\*\*\*\*'. To the right of these fields is a 'Note' section containing the text: 'The authorization header will be generated and added as a custom header.' Below the password field is a 'Refresh headers' button.

At the bottom of the interface, there is a URL bar containing 'http://localhost:8585/mmc-3.5.1/api/repository', a method dropdown menu set to 'GET', and two buttons: 'URL params' and 'Headers (0)'. Below the URL bar are three buttons: 'Send' (highlighted in blue), 'Preview', and 'Add to collection'. On the far right, there is a red 'Reset' button.

## Step 4 - Get Application Version ID - JSON

- The application ID is common for both versions
- To deploy an application, we need the version ID



The screenshot displays a REST client interface with a JSON response. The status is 200 OK and the time taken is 32 ms. The JSON structure is as follows:

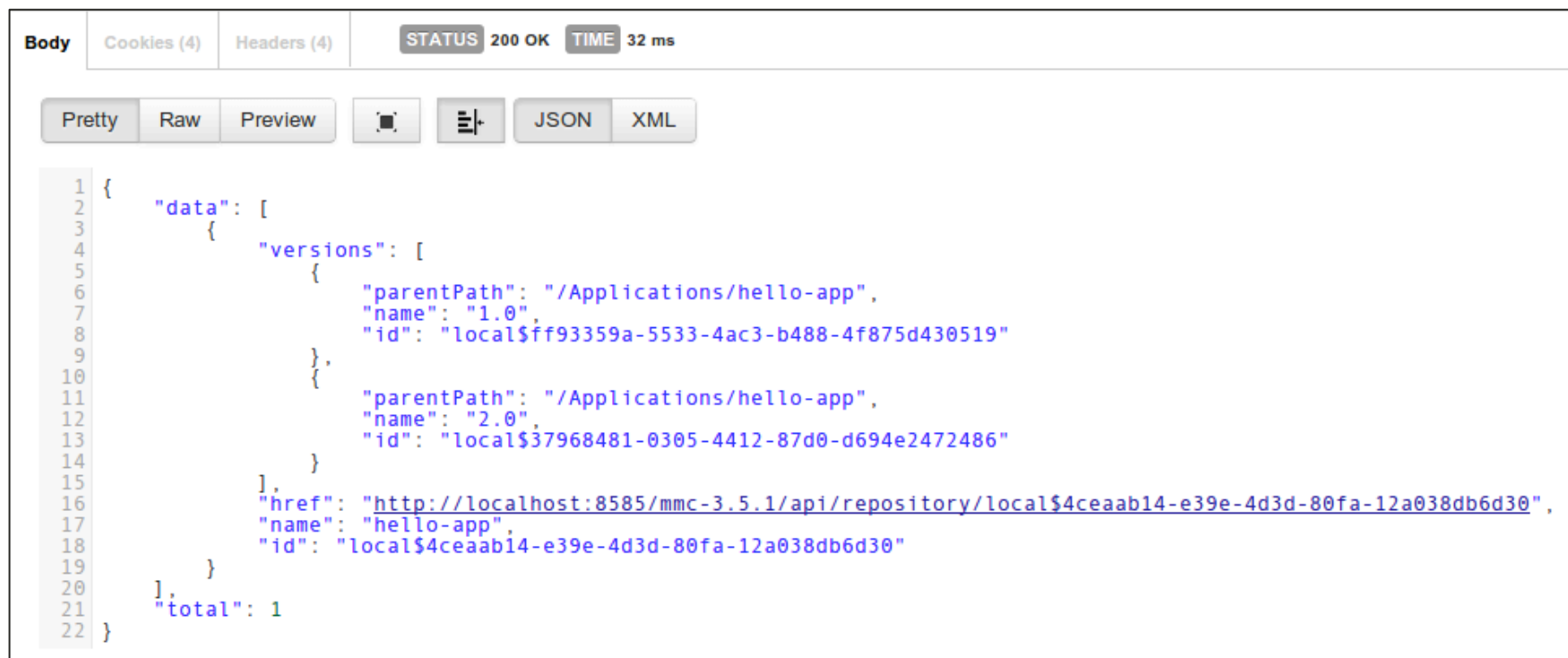
```
1 {
2   "data": [
3     {
4       "versions": [
5         {
6           "parentPath": "/Applications/hello-app",
7           "name": "1.0",
8           "id": "local$ff93359a-5533-4ac3-b488-4f875d430519"
9         },
10        {
11          "parentPath": "/Applications/hello-app",
12          "name": "2.0",
13          "id": "local$37968481-0305-4412-87d0-d694e2472486"
14        }
15      ],
16      "href": "http://localhost:8585/mmc-3.5.1/api/repository/local$4ceaab14-e39e-4d3d-80fa-12a038db6d30",
17      "name": "hello-app",
18      "id": "local$4ceaab14-e39e-4d3d-80fa-12a038db6d30"
19    }
20  ],
21  "total": 1
22 }
```

Yellow ovals and arrows highlight the version IDs and the application ID:

- Version 1.0 ID: `local$ff93359a-5533-4ac3-b488-4f875d430519`
- Version 2.0 ID: `local$37968481-0305-4412-87d0-d694e2472486`
- Application ID: `local$4ceaab14-e39e-4d3d-80fa-12a038db6d30`

# Walkthrough 7-2: MMC REST API

- Deploy and application from the MMC Repository
- Deploy and re-deploy and prepared deployment



Body Cookies (4) Headers (4) STATUS 200 OK TIME 32 ms

Pretty Raw Preview JSON XML

```
1 {
2   "data": [
3     {
4       "versions": [
5         {
6           "parentPath": "/Applications/hello-app",
7           "name": "1.0",
8           "id": "local$ff93359a-5533-4ac3-b488-4f875d430519"
9         },
10        {
11          "parentPath": "/Applications/hello-app",
12          "name": "2.0",
13          "id": "local$37968481-0305-4412-87d0-d694e2472486"
14        }
15      ],
16      "href": "http://localhost:8585/mmc-3.5.1/api/repository/local$4ceaab14-e39e-4d3d-80fa-12a038db6d30",
17      "name": "hello-app",
18      "id": "local$4ceaab14-e39e-4d3d-80fa-12a038db6d30"
19    }
20  ],
21  "total": 1
22 }
```



# Java Management Extensions (JMX)

- Simple and standard management and monitoring interface
  - Default JMX support agent
    - <jmx-default-config> config element
  - RMI registry agent
    - rmi://localhost:1099
  - Remote JMX access
    - service:jmx:rmi://jndi/rmi://localhost:1099/server
  - JMX notification agent
  - Log4J notification agent
  - MX4J adapter

# Example: JMX Server Configuration

- Set credentials

```
<management:jmx-server >

  <management:connector-server url="service:jmx:rmi:///jndi/
rmi://localhost:1099/server" rebind="false" />

  <management:credentials>

    <spring:entry key="jsmith" value="foo" />
    <spring:entry key="dthomas" value="bar" />

  </management:credentials>

</management:jmx-server>
```

# Summary

- MMC Console exposes a REST API
- Perform most administrative tasks via REST requests
  - Register/Unregister servers with MMC
  - List, Create Server Groups
  - List Servers
  - Create User Groups
  - Upload Applications to the MMC Repository
  - Create, View, Update Deployments
  - Deploy deployments

