Lifecycle Management

Module Objectives

In this module, you will look at:

- Application lifecycle management
- Controlling Applications
- Controlling Application elements

Continuuity Reactor Lifecycle Management

Lifecycle management means:

- Deploying, updating, promoting or deleting Continuuity Reactor Applications; and
- Managing the lifecycle of Flows, Procedures, MapReduce jobs and Workflows

Lifecycle management is performed using the Reactor Client HTTP API

Deploy an Application

To deploy an Application from your local file system, submit an HTTP POST request:

```
POST <base-url>/apps
```

with the name of the JAR file as a header:

```
X-Archive-Name: <JAR filename>
```

and its content as the body of the request:

```
<JAR binary content>
```

Updating an Application

Invoke the same command to update an Application to a newer version:

```
POST <base-url>/apps
```

However, stop all of its Flows, Procedures and MapReduce jobs before updating the Application

To list all of the deployed applications, issue an HTTP GET request:

```
GET <base-url>/apps
```

- This will return a JSON String map that lists each Application with its name and description
- Can be pretty-printed using `json_reformat`

List of Deployed Applications

```
[
    {
        "type": "App",
        "id": "PurchaseHistory",
        "name": "PurchaseHistory",
        "description": "Purchase history app"
    },
    {
        "type": "App",
        "id": "ResponseCodeAnalytics",
        "name": "ResponseCodeAnalytics",
        "description": "HTTP response code analytics"
    },
    {
        "type": "App",
        "id": "TrafficAnalytics",
        "name": "TrafficAnalytics",
        "description": "HTTP request counts on an hourly basis"
    }
]
```

Promoting an Application

Promote moves an Application from a local Continuuity Reactor to a Sandbox Continuuity Reactor

- Send a POST request with the host name of your Sandbox in the request body
- Must include the API key for the Sandbox in the request header

---

Example of Promoting an Application

Promote the Application *HelloWorld* from a Local Reactor to a Sandbox:

```
POST <base-url>/apps/HelloWorld/promote
```

with the API Key in the header:

```
X-Continuuity-ApiKey: <api-key> {"hostname":"<sandbox>.continuuity.net"}
```

Where:

`<api-key>`:	Continuuity Reactor API key, obtained from an account at Continuuity Accounts (http://accounts.continuuity.com)
`<sandbox>`:	Sandbox located on `continuuity.net`

Deleting an Application

To delete an Application together with all of its Flows, Procedures and MapReduce jobs, submit an HTTP DELETE:

```
DELETE <base-url>/apps/<app-name>
```

Where:

`<app-name>`:	Name of the Application to be deleted

- The `<app-name>` in this URL is the name of the Application as configured by the Application Specification; this is not necessarily the same as the name of the JAR file that was used to deploy the Application
- This does not delete the Streams and DataSets associated with the Application because they belong to your account, not the Application

---

Application Lifecycle Management

After an Application is deployed, you can:

- Start and stop its Flows, Procedures, MapReduce jobs and Workflows
- Query for an element's status

Use HTTP POST and GET methods:

```
POST <base-url>/apps/<app-id>/<element-type>/<element-id>/<operation>

GET <base-url>/apps/<app-id>/<element-type>/<element-id>/status
```

Where:

`<app-id>`:	Name of the Application being called
`<element-type >`:	Name of the element (*Flow*, *Procedure*, *MapReduce*, or *WorkFlow*) being called
`<operation>`:	One of `start` or `stop`

Lifecycle Management Examples

Start a Flow *WhoFlow* in the Application *HelloWorld*:

```
POST <base-url>/apps/HelloWorld/flows/WhoFlow/start
```

Stop the Procedure *RetrieveCounts* in the Application *WordCount*:

```
POST <base-url>/apps/WordCount/procedures/RetrieveCounts/stop
```

Get the status of the Flow *WhoFlow* in the Application *HelloWorld*:

```
GET <base-url>/apps/HelloWorld/flows/WhoFlow/status
```

Specifying Runtime Arguments

When starting an element, you can optionally specify runtime arguments as a JSON map in the request body:

```
POST <base-url>/apps/HelloWorld/procedures/Greeting/start
```

with the arguments as a JSON string in the body:

```
{"greeting":"Good Morning"}
```

These runtime arguments are used only for this single invocation of the element

---

Saving and Retrieving Runtime Arguments

To save the runtime arguments so that the Reactor will use them every time you start the element, issue an HTTP PUT with the parameter `runtimeargs`:

```
PUT <base-url>/apps/HelloWorld/procedures/Greeting/runtimeargs
```

with the arguments as a JSON string in the body:

```
{"greeting":"Good Morning"}
```

To retrieve the runtime arguments saved for an Application's element, issue an HTTP GET request to the element's URL using the same parameter `runtimeargs`:

```
GET <base-url>/apps/HelloWorld/procedures/Greeting/runtimeargs
```

This will return the saved runtime arguments in JSON format

Module Summary

You should now be able to:

- Describe Application lifecycle management
- Control a Reactor Application using the REST API
- Control Application elements using the REST API

Module Completed