Development of WebLogic 12c Management tools

September 2017

AUTHOR:
Saúl Alonso Monsalve

SUPERVISOR:
Nicolas Marescaux

CERN openlab Summer Student Report

Abstract

The CERN WebLogic infrastructure (operated by the IT-DB group) consists of 250 applications, and it is used to deploy a broad range of critical applications in the administration and engineering sectors, so it is not easy to manage the resources of the infrastructure. However, WebLogic offers different administration solutions, among which the WebLogic RESTful Management Services stand out thanks to their simplicity and efficiency. Moreover, the CERN WebLogic CLI Tool is a client application developed by the CERN IT-DB group that allows a transparent deployment of applications in the WebLogic infrastructure by using the WebLogic RESTful management services. Unfortunately, the CERN WebLogic CLI Tool was outdated, so the IT-DB group was preparing the upgrade of the tool to WebLogic 12.2.1., the newest version. This version offers new interesting features like dynamic clusters, and its REST API also offers all the functionalities that were not available in the previous product. This Openlab Summer Student project aimed to upgrade the CERN WebLogic CLI Tool to WebLogic 12.2.1. version: adding the new functionalities and updating the previous ones. After two months of work, all the tasks have been completed, and the upgraded version of the tool is now in production. This report presents an overview of the project and describes the work performed during July and August 2017.

Table of contents

1. Introduction

The CERN WebLogic infrastructure (operated by the IT-DB group) consists of 250 applications, and it is used to deploy a broad range of critical applications in the administration and engineering sectors. Deploying applications and managing the resources of the CERN WebLogic infrastructure is not an easy task due to the complexity of the components involved: servers, clusters, applications, libraries, etc. However, WebLogic offers different administration solutions, among which the WebLogic RESTful Management Services stand out above the rest due to their simplicity and efficiency.

The CERN WebLogic CLI Tool [1, 2] is a client application developed by the CERN IT-DB group in 2016 that allows a transparent deployment of applications in the WebLogic infrastructure. The tool is based on the WebLogic RESTful management services. Before starting this Openlab Summer Student project, the CERN IT-DB group was preparing the upgrade to WebLogic 12.2.1. This version offers new exciting features like dynamic clusters, and the REST API also offers all the functionalities that were not available in the previous version.

This project aimed to upgrade the CERN WebLogic CLI Tool to WebLogic version 12.2.1: adding the new functionalities and updating the previous ones. During this two months, I have been working in the Infrastructure & Middleware Service section (IT-DB-IMS), while Oracle was the partner company of the project. At the end of the Openlab Summer Student Programme, all the tasks have been completed, and the upgraded version of the tool is now in production.

The rest of the document is organized as follows: Section 2 describes the Oracle WebLogic server, while Section 3 presents the project overview; Section 4 describes in detail the developed work; Section 5 shows the main results of the project; and finally, Section 6 concludes the document and presents some future work.

2. Oracle WebLogic Server

The Oracle WebLogic Server [3] is a server software application developed by Oracle that runs in a middle tier between the user applications - that can be executed from different kind of devices and platforms - and the backend tier - which contains the enterprise resources (see Figure 1).

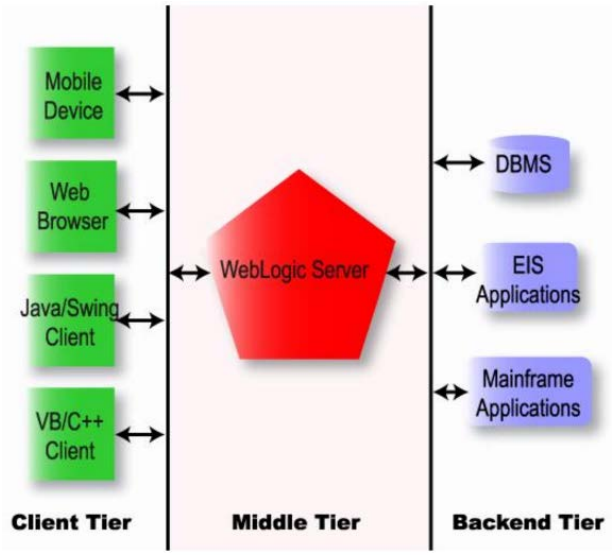

Figure 1. WebLogic server three-tier architecture [4].

Specifically, the Oracle WebLogic Server is a scalable, enterprise-ready Java Platform, Enterprise Edition (Java EE) application server [3]. The CERN WebLogic infrastructure, which is operated by the IT-DB group, consist of 250 applications. This infrastructure is used to deploy a wide range of critical applications in administration and engineering sectors. One example is the EDH application from the AIS group [5]. Figure 1 shows the overview of the CERN WebLogic infrastructure and its interaction with other agents. On the one hand, the developers create and package their applications, and deploy them into the WebLogic infrastructure, which consists of several logical servers and clusters virtualized on top of physical machines, currently located at the CERN Data Centre [6]. On the other hand, the final users consume the applications via the WebLogic infrastructure.

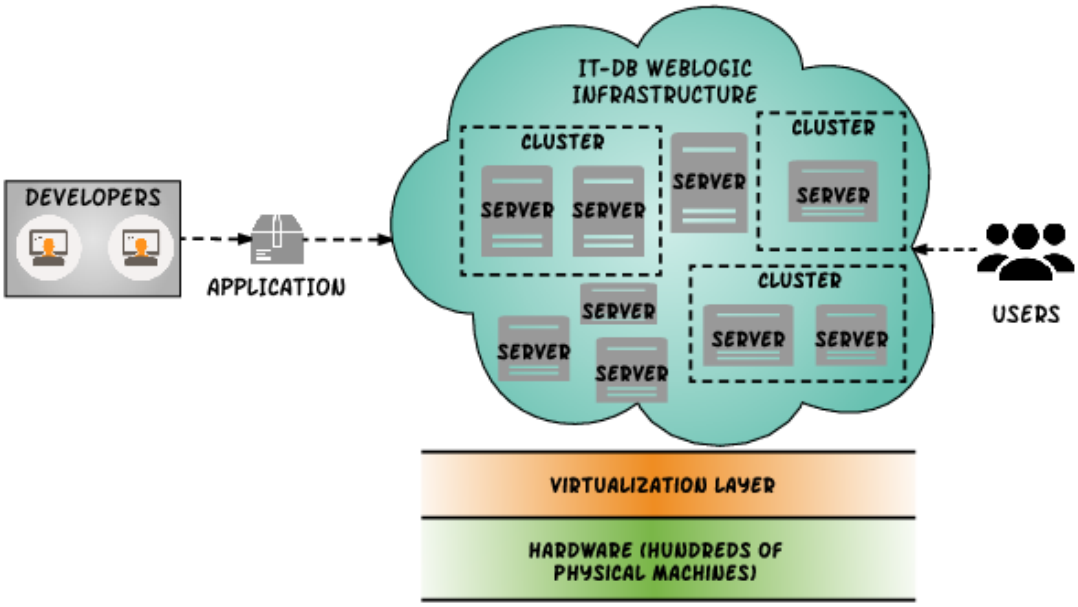

Figure 2. Overview of CERN WebLogic infrastructure.

3. Project overview

As explained in Section 2 (see Figure *2*), the CERN developers deploy their applications into the IT-DB WebLogic infrastructure, so the final users can consume them by accessing the infrastructure. Deploying applications and managing the resources of the WebLogic infrastructure is not an easy task. However, WebLogic offers three different administration solutions to achieve this issue:

- **WebLogic Server Administration Console**: it is the web-based management interface for WebLogic Server. It is the most traditional solution since the user is forced to use a web browser to perform all the management operations.
- **WebLogic Scripting Tool (WLST)**: it is a command-line scripting interface used to monitor the WebLogic infrastructure. It is based on the Java scripting interpreter, Jython.
- **WebLogic RESTful Management Services**: WebLogic Server also provides RESTful Management Services [7] that enable the users to monitor the WebLogic Server.

Table 1 compares the different features of the three solutions:

	Connection type	Execution	Previous configuration?
Administration console	Web browser	Slow	No
WLST	T3 (Specific protocol)	Slow	Yes (setting some libraries)
RESTful Management Services	HTTP	Fast	No

Table 1. Comparison of WebLogic Server administration alternatives.

It is easy to understand that the best option is to use the RESTful Management Services, because they are simple, lightweight and fast. This is the reason why the CERN IT-DB group developed in 2016 the CERN WebLogic CLI Tool, a command-line interface tool based on the WebLogic RESTful Management Services for administrating the CERN WebLogic infrastructure.

a. CERN WebLogic CLI Tool

The CERN WebLogic CLI Tool [1, 2] is a client application developed by the CERN IT-DB group in 2016 that allows CERN developers to manage their resources within the IT-DB WebLogic infrastructure. The tool is mainly written in Python and is based on the WebLogic RESTful management services. This software aims to provide a rich user experience, as well as maximize the productivity since the tool hides all the administration complexity by adding an abstraction layer to the REST interface.

Although dealing with RESTful Web Services is an easy and fast task, it could be irritating to define a different cURL request for every single operation or to have hundreds of sample cURL scripts. Besides, the JSON output returned always contains too much information, and most of it is useless for the users. Fortunately, the CERN WebLogic CLI Tool solves all these problems: it automatically generates the REST call from a simple user request, and then only shows the most relevant information of the JSON output to the user (see Table 2).

	cURL	CERN WebLogic CLI Tool

Request	`curl -v \`
	`  --user ${USERNAME}:${PASSWORD} \`
	`  -H X-Requested-By:MyClient \`
	`  -H Accept:application/json \`
	`  -H Content-Type:application/json \`
	`  -d "{`
	`  name: 'fairShare',`
	`  sourcePath: '/ORA/dbs01/oracle/product/Middleware_2.2.1.0_generic/wlserver/server/lib/wls-cat.war',`
	`  targets: [ { identity: [ 'clusters', 'devSAUL1_A_Cluster' ] } ],`
	`  versionIdentifier: "1.2"`
	`  }" \`
	`  -X POST http://wls-saul1-dev-admin.cern.ch:30000/management/weblogic/latest/edit/appDeployments`
		`wls-cli --url http://wls-saul1-dev-admin.cern.ch:30000 --deploy --appname app1 --file /ORA/dbs01/oracle/product/Middleware_2.2.1.0_generic/wlserver/server/lib/wls-cat.war --targets devSAUL1_A_Cluster --netrc`

Output	{
	"links": [{
	"rel": "job",
	"href": "http:\/\/wls-saul1-dev-admin.cern.ch:30000\/management\/weblogic\/latest\/domainRuntime\/deploymentManager\/deploymentProgressObjects\/fairShare"
	}],
	"identity": [
	"deploymentManager",
	"deploymentProgressObjects",
	"fairShare"
	],
	"rootExceptions": [],
	"endTimeAsLong": 1504258816848,
	"deploymentMessages": [
	"[Deployer:149191]Operation \"deploy\" on application \"fairShare\" is initializing on \"devSAUL1_A_1\".",
	"[Deployer:149191]Operation \"deploy\" on application \"fairShare\" is initializing on \"devSAUL1_A_2\".",
	"[Deployer:149192]Operation \"deploy\" on application \"fairShare\" is in progress on \"devSAUL1_A_1\".",
	"[Deployer:149192]Operation \"deploy\" on application \"fairShare\" is in progress on \"devSAUL1_A_2\".",
	"[Deployer:149194]Operation \"deploy\" on application \"fairShare\" has succeeded on \"devSAUL1_A_1\".",
	"[Deployer:149194]Operation \"deploy\" on application \"fairShare\" has succeeded on \"devSAUL1_A_2\"."
	],
	"name": "fairShare",
	"operationType": 3,
	"startTimeAsLong": 1504258812394,
	"state": "STATE_COMPLETED",
	"id": "120",
	"type": "DeploymentProgressObject",
	"targets": ["devSAUL1_A_Cluster"],
	"applicationName": "fairShare",
	"failedTargets": [],
	"progress": "success",
	"completed": true,
	"intervalToPoll": 1000,
	"startTime": "2017-09-01T11:40:12.394+02",
	"endTime": "2017-09-01T11:40:16.848+02"
	}
		2017-09-01 11:43:40,310 - Logger - INFO - Started parsing request.
		2017-09-01 11:43:40,312 - Logger - INFO - Operating request: 'deploy an app from local source' using: REST
		
		Deployed the application ‘fairshare’. : SUCCESS
		
		2017-09-01 11:43:42,492 - Logger - INFO - Finish.
		Status code: 0

Table 2. Comparison between cURL and CERN WebLogic CLI Tool for a deployment operation.

The workflow of the tool is the following (see Figure 3): (1) the tool first parses the user request, interpreting and validating all the arguments; (2) it then creates the strategy objects necessary to run the request, injecting the domain data (e.g., entity or servers); (3) then, the API data is injected (URL, cURL options); (4)

once the request is ready, it is executed against WebLogic; (5) finally, the output results of the executed request are also parsed and presented to the user in a fancy way.

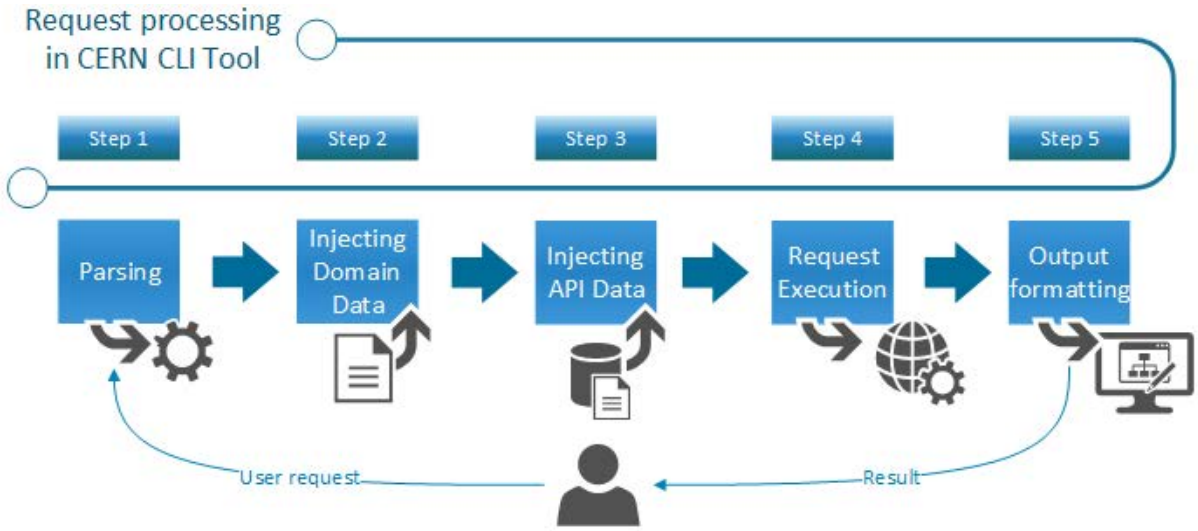

Figure 3. Workflow of the CERN WebLogic CLI Tool [1].

b. Limitations

Although the CERN WebLogic CLI Tool has been handy useful, it had some problems before starting this Openlab Summer Student project:

- The tool worked for WebLogic version 12.1.3, but the current WebLogic version is 12.2.1:
  - It was a problem because of the WebLogic 12.1.3. REST API was not complete, and some basic WebLogic operations were not available using the REST API (e.g., the creation of servers, clusters, or datasources), so developers had to use a more traditional solution, such as the Administration Console.
  - The current WebLogic REST API not only covers all the functionalities that were not available in the previous version (WebLogic 12.1.3. REST API), but also offers new intrinsic administration solutions of WebLogic version 12.2.1 (e.g., dynamic clusters).

- There was a lack of technical documentation of the tool:
  - This fact made very difficult to modify the behavior of the tool.

c. Motivation

Within the CERN WebLogic infrastructure, the resources are currently stored in the CERN Data Centre [6] but, in the future, the IT-DB group could decide to store the resources in other locations. Developers need to have a tool that allows them to manage their resources, and that is why the CERN WebLogic CLI Tool is essential. However, as stated before, the tool currently has some limitations – it is outdated, and it is difficult to maintain due to the lack of technical documentation – so it is really important to upgrade the tool to the newest version and, most importantly, make things easier for people who want to add or modify new functionalities in the future.

d. Objectives

The goal of this Openlab Summer Student project was to **upgrade the CERN WebLogic CLI Tool to WebLogic version 12.2.1**. We can also determine the following objectives from the main goal:

- **O1**: checking whether the current functionalities work for WebLogic version 12.2.1. and, if not, updating them to the current WebLogic version.
- **O2**: adding the new functionalities the WebLogic version 12.2.1. offers and are needed for the CERN IT-DB group.
- **O3**: writing the user and technical documentation of the work done, as well as some examples of how to integrate new functionalities within the tool.

4. Development

a. REST operations

The first step was to understand the WebLogic REST API [8]. All the WebLogic REST requests have the same cURL structure:

```
curl -v \
    --user {USERNAME}:{PASSWORD} \
    -H X-Requested-By:MyClient \
    -H Accept:application/json \
    -H Content-Type:application/json \
    [-d "{
       postfield1: value [, postfield2: value2] ...
    }" \
    -X {GET | POST | PUT | DELETE} http[s]://DOMAIN:PORT/URL
```

Where USERNAME and PASSWORD correspond to the username and password of the domain; GET, POST, PUT and DELETE correspond to the HTTP method; DOMAIN corresponds to the domain name (e.g., wls-saul1-dev-admin.cern.ch); PORT corresponds to the listen port (e.g., 3000), and URL corresponds with the URL bean (e.g., management/weblogic/latest/edit/clusters/devSAUL1_A_Cluster/dynamicServers).

Before starting this project, the CERN WebLogic CLI Tool integrated the most important available features of the WebLogic 12.1.3. REST API, such as the deployment of applications and libraries. However, other important features of WebLogic 12.1.3 were not available in the REST API of that version. Some examples of these unavailable management functionalities are all the editing operations, such as the creation of targets (servers, machines, clusters) and their configuration, or the management of datasources (e.g., the creation of datasources, JDBC Resources, JNDIs, etc.). In such cases, it was still necessary to use the Administration Console or the WebLogic Scripting Tool (as mentioned in Section 3).

After checking the WebLogic 12.2.1. REST API, all the unavailable functionalities of the previous version were available in the current one. In fact, the user should use the following URL to edit the management resources:

```
/management/weblogic/{version}/edit
```

We have created WebLogic 12.2.1. REST API scripts for every single request needed. For example, this is the structure of the cURL call that creates a server called devSAUL1_A_1, assigned to a machine called wls-devsaul1-a-1 and to a cluster called devSAUL1_A_Cluster, among others:

```
curl -v \
    --user {USERNAME}:{PASSWORD} \
```

```
    -H X-Requested-By:MyClient \
    -H Accept:application/json \
    -H Content-Type:application/json \
    [-d "{
       name:                 'devSAUL1_A_1'
       machine:              [ 'machines', 'wls-devsaul1-a-1' ],
       cluster:              [ 'clusters', 'devSAUL1_A_Cluster' ],
       listenAddress:        'wls-devsaul1-a-1.cern.ch'
       listenPortEnabled:    true,
       listenPort:           30006
    }" \
    -X POST http://wls-saul1-dev-admin.cern.ch:30000/management/
weblogic/latest/edit/servers
```

We have gathered all these scripts in a private CERN GitLab repository, which contains the following folders:

- **cluster**: scripts to create, start and stop clusters, as well as configure dynamic clusters.
- **datasource**: scripts to create and configure datasources.
- **deploy**: scripts to deploy and stop apps and libraries, as well as get their state.
- **edit**: scripts to start editing and activate changes.
- **machine**: scripts to create machines.
- **server**: scripts to create, start, and stop servers.
- **version**: scripts to check the current WebLogic version.
- **view**: scripts to query the bean trees.

After testing all the scripts against a development domain, we have validated all the different REST calls but one: the creation of Unix machines. According to the WebLogic 12.2.1. documentation [8], if you want to create a Unix machine using the WebLogic 12.2.1. REST API you should specify the type "UnixMachine" as follows:

```
curl -v \
    --user {USERNAME}:{PASSWORD} \
    -H X-Requested-By:MyClient \
    -H Accept:application/json \
    -H Content-Type:application/json \
    [-d "{
       name:                 'wls-devsaul1-a-1',
       machine:              'UnixMachine'
    }" \
    -X POST http://wls-saul1-dev-admin.cern.ch:30000/management/
weblogic/latest/edit/machines
```

However, this operation does not work. We contacted the Oracle Support [9] one month before finishing this project to report the issue but they are still looking for a solution, so we believe it is a bug. Nevertheless, all the remaining operations work perfectly.

b. Integration of functionalities into the tool

Once checking that all the scripts worked, it was time to update all the tool's previous functionalities and integrate the new ones. It was a complicated task since the CERN WebLogic CLI Tool has 163 source code files, almost 15,000 lines of code, and an insufficient technical documentation. Figure 4 shows de state machine diagram of the tool, which helps to understand the role of each file. The tool was developed in an object-oriented way and follows a Model-View-Controller (MVC) design pattern.

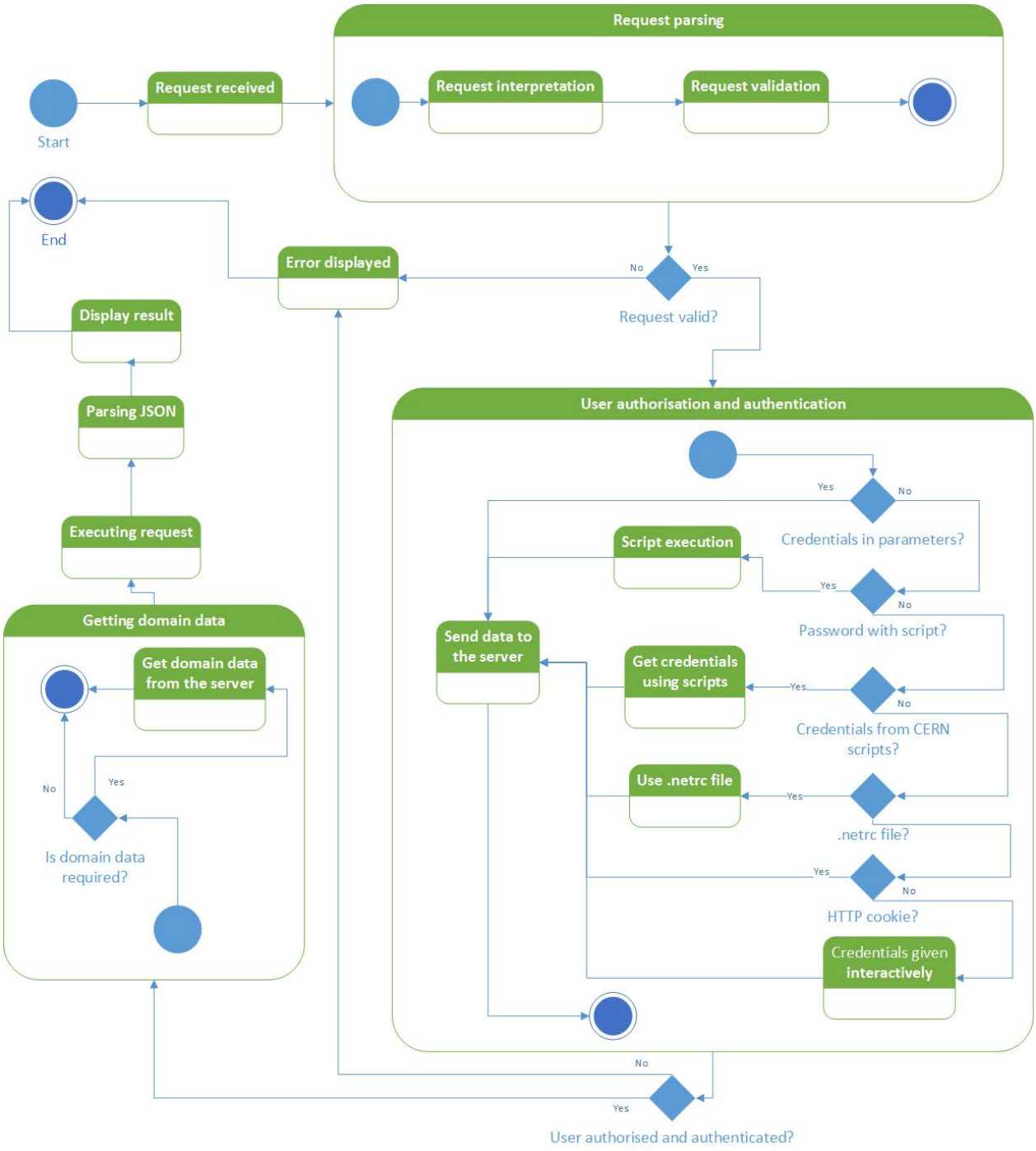

Figure 4. State machine diagram of the tool [10].

Understanding how the tool works was crucial to update and integrate functionalities. First of all, it was necessary to understand how the tool parses the user request: interpreting the arguments to identify the desired operation, and validating these arguments. Then, the user should be authenticated in the system. Once it is done, the tool prepares the agents and the strategy objects that are going to be responsible for executing the request. The CERN WebLogic CLI Tool uses the PyCurl library – a Python interface to libcurl - in order to create and execute the cURL request: first, you only need to define the data (command name, url context and other variables) necessary to create a cURL request:

```
def create_server(self, strategy_tuple):
        strategy_tuple.update({"NM":"EMPTY"})
        strategy_tuple.update({"REST":""})
        strategy_tuple.update({"HTTP":"POST"})
        strategy_tuple.update({"postfields":True})
        strategy_tuple.update({"url_context":"/management/weblogic/latest/edit/servers"}
        )
```

This is an example of the creation of a server; you only need to specify the HTTP method, whether the request needs postfields, and the URL. After that, you should define the postfields – in this example, only the name is required – and the HTTP header of the cURL call:

```
def set_create_server(self, data_wrapper):
        ''' setting deploy uploaded app / lib operation '''
        path = data_wrapper.source
        postfields = {"name": data_wrapper.target}
        data = json.dumps(postfields)
        self.curl_agent.setopt(pycurl.POST, 1)
        self.curl_agent.setopt(pycurl.POSTFIELDS, data)
        self.curl_agent.setopt(pycurl.HTTPHEADER, \
          ['X-Requested-By: MyClient','Content-Type: application/json' , \
          'Accept: application/json'])
        self.write_postfields(postfields)
```

After executing the request, the tool needs to handle the output and return the results to the user. During this Openlab Summer Student project, we have updated all the previous functionalities and added the newest ones (parsing the request for each operation, creating the cURL call, executing and parsing the results, etc.), as well as documenting and testing the tool.

5. Results

We have been working in two CERN GitLab [10] repositories:

- wls-rest-12.2: individual WebLogic 12.2.1. REST scripts.
- cerndb-infra-wls_rest: CERN WebLogic CLI Tool.

At the end of the project, the upgraded CERN WebLogic CLI Tool is ready to be used:

- All the previous functionalities have been updated to version 12.2.1.
- All the other desired administration functionalities are now available:
    - Creation of servers.
    - Creation of machines.
    - Creation of clusters.
    - Creation of datasources.
    - Creation of JDBC resources.
    - Creation of JNDI.
    - Configuration of datasources (driver, URL, username, password).
    - Allowance of asynchronous operations.
    - Writing request details into file.
    - Etc.
- We have created a new **RPM package**, and the tool is now **in production**.
- There are several of module and unit tests available in the own tool.

- The documentation of the tool is also available:
  - Technical and user documentation in the cerndb-infra-wls_rest GitLab repository.
  - User manual within the own tool.

6. Conclusion and Future Work

This report has introduced the Openlab Summer Student Project titled "*Development of WebLogic 12c Management Tools*". This project aimed to upgrade the CERN WebLogic CLI Tool to WebLogic version 12.2.1., and this document has presented the work done during July and August 2017. At the end of the project, we have met all the goals presented in the project overview (Section 3):

- **O1**: we have checked all the functionalities implemented in the tool and updated the ones that did not work.
- **O2**: we have integrated the new functionalities WebLogic version 12.2.1. offers and are needed for the CERN IT-DB group.
- **O3**: we have written the user and technical documentation of the tool, as well as some examples of how to integrate new functionalities within the tool.

At a personal level, I had no previous work experience in this field, so this project has been very challenging. Besides, I am happy with the result, because both I have learned a lot and the tool is now in production, so this means CERN developers are currently using it.

For future work, the IT-DB group is planning to locate part of the current resources outside the CERN Data Centre (e.g., in an Amazon cloud), and this would be an interesting scenario for evaluating the simplicity of the tool since the developers could deploy their applications without worrying where the physical machines are. It would also be interesting to analyze the WebLogic documentation further and decide whether to integrate other advanced features into the CERN WebLogic CLI Tool.

References

[1] Oracle, "Oracle WebLogic Server Technical Information," [Online]. Available: http://www.oracle.com/technetwork/middleware/weblogic/overview/index.html.

[2] Oracle, "What is WebLogic Server?," [Online]. Available: https://docs.oracle.com/cd/E13222_01/wls/docs61/intro/chap1.html. [Accessed 2 August 2017].

[3] Advanced Informatics Support (AIS), "CERN EDH - Web version," 2000. [Online]. Available: https://edh.cern.ch. [Accessed 15 August 2017].

[4] P. Andrade, T. Bell, J. van Eldik, G. McCance, B. Panzer-Steindel, M. Coelho dos Santos, S. Traylen and U. Schwickerath, "Review of CERN Data Centre Infrastructure," Journal of Physics: Conference Series, vol. 396, no. 4, p. 042002, 2012.

[5] L. Richardson and S. Ruby, Restful Web Services, First, 2007.

[6] K. A. Kaczkowski, "Databases at CERN blog," 27 November 2015. [Online]. Available: https://db-blog.web.cern.ch/blog/konrad-aleksander-kaczkowski/2015-11-cern-it-db-group-develops-new-weblogic-command-line.

[7] K. A. Kaczkowski, "Oracle Blogs," 26 March 2016. [Online]. Available: https://blogs.oracle.com/emeapartnerweblogic/cern-it-db-group-develops-a-new-weblogic-command-line-interface-tool-for-cern-java-middleware-services-by-konrad-aleksand.

[8] Oracle, "My Oracle Support," [Online]. Available: https://support.oracle.com/.

[9] Oracle, "RESTful Edit Reference for Oracle WebLogic Server," October 2015. [Online]. Available: https://docs.oracle.com/middleware/1221/wls/WLRER.pdf.

[10] CERN, "GitLab@CERN," [Online]. Available: https://gitlab.cern.ch/.