

**Cairo University**

**Faculty of Computer and Information**

**Computer Science – Information Technology**

**Fork**

**Dr. Haitham Hamza – Dr. Sherif Khattab**

|  |  |  |
| --- | --- | --- |
| **Name** | **ID** | **Department** |
| Islam Said Esawy | 20110019 | IT |
| Ahmed Emam Ahmed | 20110040 | IT |
| Ahmed Hamed Sayed | 20110046 | CS |
| Mohamed Ibrahim Mohamed | 20110277 | CS |
| Mohamed Waleed Mohamed | 20110335 | CS |

Contents

[Chapter 1: Introduction 3](#_Toc423396145)

[**Problem Definition** 3](#_Toc423396146)

[**Suggested Solution** 3](#_Toc423396147)

[Chapter 2: Idea and Current Work 5](#_Toc423396148)

[**Cacti** 5](#_Toc423396149)

[**Data Retrieval** 6](#_Toc423396150)

[**Data Storage** 7](#_Toc423396151)

[**Information Storage** 8](#_Toc423396152)

[**Current Work** 8](#_Toc423396153)

[Chapter 3: Design and Analysis 9](#_Toc423396154)

[**Functional Requirements** 9](#_Toc423396155)

[**Operational Requirements** 9](#_Toc423396156)

[**Configuration Requirements** 9](#_Toc423396157)

[**Non-Functional Requirements** 10](#_Toc423396158)

[**Usability** 10](#_Toc423396159)

[**Reliability** 10](#_Toc423396160)

[**Platform** 10](#_Toc423396161)

[**Performance** 10](#_Toc423396162)

[**Requirements Table** 10](#_Toc423396163)

[**Use Cases** 11](#_Toc423396164)

[**Architecture** 22](#_Toc423396165)

[**Package and Class Diagram** 24](#_Toc423396166)

[**Sequence Diagram** 29](#_Toc423396167)

[Chapter 4: Snapshots and Manual 31](#_Toc423396168)

[**Configuration and installation:** 31](#_Toc423396169)

[**Application screens explanation:** 32](#_Toc423396170)

[**Zones tab** 33](#_Toc423396171)

[**Scripts tab** 36](#_Toc423396172)

[**Rules tab** 39](#_Toc423396173)

[**Application background process:** 45](#_Toc423396174)

[Chapter 5: Conclusion 46](#_Toc423396175)

# Chapter 1: Introduction

With Network monitoring tool it is possible to gather virtually limitless types of data from the network. High performance real-time monitoring means that tens of thousands of servers, virtual machines and network devices can be monitored simultaneously. Along with storing the data, visualization features are available (overviews, maps, graphs, screens, etc.), as well as very flexible ways of analyzing the data for the purpose of alerting. Using Network monitoring tool such as (Cacti, Zenos, Zabbix … etc.), you can easily monitor servers, network devices and applications, gathering accurate statistics and performance data.

Monitoring performance indicators like CPU, memory, network, disk space and processes can be done easily with Network monitoring tool agent, which is available for Linux, UNIX and Windows platforms.

## **Problem Definition**

Any network in the world needs maintenance through devices monitoring to overcome any unexpected issues that might happen, so the network needs engineers with a lot experience to detect any events or issues all the time over network devices data. This devices data which comes from network monitoring tools such as Cacti, Zenos and Opnet helps the network engineers to take some actions to fix issues. The main point here that network maintenance is effort and time consuming.

## **Suggested Solution**

Building a cross-platform java application called Fork that connects to the network monitoring tool. This connection purpose is to pull the devices data and information needed to be helpful in controlling and troubleshooting network through discovering expected problems, some of these problems can be solved automatically using scripts instead of letting the engineers watching the devices graphs and data all the time to detect some sort issues.

Fork application will build rule based and scripts run if rule achieve which are being added by engineers based on their experience, poll device data from network monitoring tool, analysis device data, put device data in general structure and based on rules Fork application can suggest or take action by run script over device in network.

# Chapter 2: Idea and Current Work

How our idea will help network administrators?  
Network engineers are working with some rules to manage, control and monitoring the network performance and utilizations. They need permanent monitoring by installing and configuring any network monitoring tool. Then they have to watch the changes with their own eyes.

Our idea is to build an application to do that instead. The application lists all network devices, which is being monitored by the current monitoring system tool, to the user. After that the user have to build his own zones.  
Zone consists of several devices. These devices are chosen by the user. An example of zone system is represented by the Cairo university network which consists of many zones such as FCI and Faculty of Commerce.  
The application will ask the engineers to enter number of rules which consists of several conditions to be checked. The application will be fed by network monitoring tool data and information gathered from the network, then it will check for any rule to be fired. Firing a rule means that our system will notify the user with rule details.  
Rule details are represented in a name, description, and a certain number of scripts to run.  
Script is a set of lines of instructions and commands which can be run on network devices.

## **Cacti**

Cacti is an [open-source](https://en.wikipedia.org/wiki/Open_source), web-based [network monitoring](https://en.wikipedia.org/wiki/Network_monitoring) and graphing tool designed as a [front-end](https://en.wikipedia.org/wiki/Front_and_back_ends) application for the open-source, industry-standard [data logging](https://en.wikipedia.org/wiki/Data_logger#Data_logging_versus_data_acquisition) tool [RRDtool](https://en.wikipedia.org/wiki/RRDtool" \o "RRDtool). Cacti allows a user to [poll](https://en.wikipedia.org/wiki/Polling_(computer_science)) services at predetermined intervals and graph the resulting data. It is generally used to graph time-series data of metrics such as CPU load and network [bandwidth](https://en.wikipedia.org/wiki/Bandwidth_(computing)) utilization. A common usage is to monitor network traffic by polling a [network switch](https://en.wikipedia.org/wiki/Network_switch) or [router](https://en.wikipedia.org/wiki/Router_(computing)) interface via [Simple Network Management Protocol](https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol) (SNMP).

The front end can handle multiple users, each with their own graph sets, so it is sometimes used by [web hosting](https://en.wikipedia.org/wiki/Web_hosting_service) providers (especially [dedicated server](https://en.wikipedia.org/wiki/Dedicated_server), [virtual private server](https://en.wikipedia.org/wiki/Virtual_private_server), and collocation providers) to display bandwidth statistics for their customers. It can be used to configure the data collection itself, allowing certain setups to be monitored without any manual configuration of RRDtool. Cacti can be extended to monitor any source via [shell scripts](https://en.wikipedia.org/wiki/Shell_script) and executables.

Cacti can use one of two [back ends](https://en.wikipedia.org/wiki/Front_and_back_ends): "cmd.php", a [PHP](https://en.wikipedia.org/wiki/PHP) script suitable for smaller installations, or "Spine" (formerly Cactid), a [C](https://en.wikipedia.org/wiki/C_(programming_language))-based poller which can scale to thousands of hosts.

### **Data Retrieval**

How Network monitoring tool communicate with devices in the network?

Simple Network Management Protocol (SNMP) is an "[Internet-standard protocol](https://en.wikipedia.org/wiki/Internet_protocol_suite) for managing devices on [IP](https://en.wikipedia.org/wiki/Internet_Protocol) networks". Devices that typically support SNMP include routers, switches, servers, workstations, printers, modem racks and more. SNMP is used in [network management systems](https://en.wikipedia.org/wiki/Network_management_systems) to [monitor](https://en.wikipedia.org/wiki/Network_monitoring) network-attached devices for conditions that warrant administrative attention. SNMP is a component of the [Internet Protocol Suite](https://en.wikipedia.org/wiki/Internet_Protocol_Suite) as defined by the [Internet Engineering Task Force](https://en.wikipedia.org/wiki/Internet_Engineering_Task_Force) (IETF). It consists of a set of [standards](https://en.wikipedia.org/wiki/Technical_standard) for network management, including an [application layer](https://en.wikipedia.org/wiki/Application_layer) [protocol](https://en.wikipedia.org/wiki/Protocol_(computing)), a database [schema](https://en.wikipedia.org/wiki/Logical_schema), and a set of [data objects](https://en.wikipedia.org/wiki/Data_object).

SNMP exposes management data in the form of variables on the managed systems, which describe the system configuration. These variables can then be queried (and sometimes set) by managing applications.

### **Data Storage**

RRDtool assumes time-variable data in intervals of a certain length. This interval, usually named step, is specified upon creation of an RRD file and cannot be changed afterwards. Because data may not always be available at just the right time, RRDtool will automatically interpolate any submitted data to fit its internal time-steps.

The value for a specific step, that has been interpolated, is named a primary data point (PDP). Multiple PDPs may be consolidated according to a consolidation function (CF) to form a consolidated data point (CDP). Typical consolidation functions are [average](https://en.wikipedia.org/wiki/Average), minimum, maximum.

After the data have been consolidated, the resulting CDP is stored in a round-robin archive (RRA). A round-robin archive stores a fixed number of CDPs and specifies how many PDPs should be consolidated into one CDP and which CF to use. The total time covered by an RRA can be calculated as follows:

🡪 time covered = (#CDPs stored) \* (#PDPs per CDP) \* steps

After this time the archive will "wrap around": the next insertion will overwrite the oldest entry. This behavior in this context is referred to as "round-robin" and is the reason for the program's name. However this is different from the common computer science definition, which is a method of distributing resources among multiple consumers or processes.

To cover several timespans and/or use several consolidation functions, an RRD file may contain multiple RRAs. The data retrieval function of RRDtool automatically selects the archive with the highest resolution that still covers the requested timespan. This mechanism is also used by RRDtool's graphing subsystem.

### **Information Storage**

Cacti uses Mysql database to store the information needed to run cacti such as data sources, templates, snmp configuration and other settings.

## **Current Work**

Dsad

# Chapter 3: Design and Analysis

## **Functional Requirements**

### **Operational Requirements**

#### **Firing rules**

The system notifies the engineer with any fired rules in a given network.

#### **Running a script**

The system makes the user able to perform an action from different scripts such as shutting down an interface of a router.

#### **Polling data**

The system polls data from Cacti, by querying cacti mysql database and rrd’s.

#### **Saving cacti data in rdf model**

After polling data from cacti, the system stores devices data and information in rdf model.

#### **Storage of rules and scripts**

The system stores the user’s rules and scripts in a SQLite database.

### **Configuration Requirements**

#### **Adding Zone**

The user have to add several zones which consists of various devices.

#### **Adding Rule**

The user can add a specific network rule to the system so that the system can use them to detect events.

#### **Adding Script**

The user can add a specific script that can be run within a certain device under user confirmation.

#### **Delete or Edit Script**

The user can delete or edit a specific script.

#### **Connectivity Configuration**

The user can modify the configuration management of the connectivity of the system and the monitoring tool like Mysql authentication.

## **Non-Functional Requirements**

### **Usability**

This application is developed for network engineers. The application is very user-friendly. The application will need 10 hours from the engineer to fully understand the application scenarios.

### **Reliability**

Crashsafe. This system should be crash safe in 95% of its runtime.

### **Platform**

The Application is a cross platform. Tested on Linux and Windows.

### **Performance**

The system takes from 4 to 10 seconds to build the rdf and SQLite database.

## **Requirements Table**

|  |  |  |
| --- | --- | --- |
| ID | Description | Complexity |
| 1 | User should first login with cacti’s MySQL username and password and check that it’s true and Apache work. | 1 |
| 2 | System should get devices information from cacti MySQL and poll data for each interface from device.  Store and update device data in RDF model. | 5 |
| 3 | System should update data each specific period. | 2 |
| 4 | User should see all devices that aren’t belong to zone and user can select list of devices and enter zone name and add zone.  System saves zone information to RDF model.  User can remove zone.  User can select several zones to remove.  System removes zone information from RDF model. | 4 |
| 5 | User should be able to add scripts files.  User should be able to delete scripts files.  System saves scripts in database. | 3 |
| 6 | User can add rules. Each rule can be several conditions. Each condition is consisting of device name, interface name and minimum-maximum limits for each data source in interface.  User can add scripts that run when rule happen. | 4 |
| 7 | System should check for each rule if it happen or not, if it happen show message contain rule information and scripts that related to it.  System runs scripts chosen by user. | 4 |

## **Use Cases**

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 1 | |
| Use Case Name: | Cacti MySQL login | |
| Actors: | User – System. | |
| Pre-conditions (Use Case ID): | None. | |
| Post-conditions: | Application open successfully. | |
| Flow of events: | User Action | System Action |
|  | System Verify data stored in database. |
| If it’s not true, User Enter Username and password. |  |
|  | System Verify data. |
|  | System saves data in database. |
|  | System closes login view and open application. |
| Exceptions: | User cancels login.  Apache server doesn’t work. | |

|  |  |
| --- | --- |
| Use Case ID: | 2 |
| Use Case Name: | System Life Cycle |
| Actors: | User – System. |
| Pre-conditions (Use Case ID): | 1 |
| Post-conditions: | Data updated in application. |
| Flow of events: | System Action |
| System blocks GUI. |
| System updates data using cacti MySQL database. |
| System release GUI. |
| Exceptions: | User closes application. |

|  |  |
| --- | --- |
| Use Case ID: | 3 |
| Use Case Name: | Poll Device Data |
| Actors: | User - System. |
| Pre-conditions (Use Case ID): | 2 |
| Post-conditions: | Device Data polled successfully. |
| Flow of events: | System Action |
| System checks that Fuseki server works. |
| System poll device information (ID, IP, Host Name) from cacti MySQL database. |
| System poll device interface information (Interface Name, Device ID, rrd files paths) from cacti MySQL database. |
| System poll device interface data from rrd’s files. |
| Exceptions: | User closes application.  Fuseki server doesn’t work.  Apache server stops working. |

|  |  |
| --- | --- |
| Use Case ID: | 4 |
| Use Case Name: | Store Device Data |
| Actors: | User – System. |
| Pre-conditions (Use Case ID): | 3 |
| Post-conditions: | Device Data stored in RDF model. |
| Flow of events: | System Action |
| System updates Device information in RDF. |
| System updates Device interface information in RDF. |
| System updates Device interface data in RDF. |
| Exceptions: | User closes application.  Fuseki server stops working.  Apache server stops working. |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 5 | |
| Use Case Name: | List Free Devices | |
| Actors: | User – System. | |
| Pre-conditions (Use Case ID): | 1, 2 | |
| Post-conditions: | List Free Devices to user. | |
| Flow of events: | User Action | System Action |
| User select Zone panel. |  |
|  | System poll data that don’t belong to zone from RDF. |
|  | System shows device in GUI. |
| Exceptions: | User closes application.  Fuseki server stops working. | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 6 | |
| Use Case Name: | Add Zone | |
| Actors: | User – System. | |
| Pre-conditions (Use Case ID): | 5 | |
| Post-conditions: | User adds zone. | |
| Flow of events: | User Action | System Action |
| User select list of devices. |  |
| User enter zone name |  |
| User click on add zone button |  |
|  | System validate that user enter zone name and list of device contain at least 1 device. |
|  |  | System adds zone information to RDF. |
| Exceptions: | User closes application.  Fuseki server stops working. | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 7 | |
| Use Case Name: | Remove Zone | |
| Actors: | User – System. | |
| Pre-conditions (Use Case ID): | 1, 2 | |
| Post-conditions: | User removes zone. | |
| Flow of events: | User Action | System Action |
| User click on remove zone button. |  |
|  | System gets all zones from RDF. |
|  |  | System shows list of zones in GUI. |
|  | User select list of zones. |  |
|  | User click ok button. |  |
|  |  | System delete zones list from RDF. |
| Exceptions: | User closes application.  User close confirmation frame.  Fuseki server stops working. | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 8 | |
| Use Case Name: | Add Script | |
| Actors: | User – System. | |
| Pre-conditions (Use Case ID): | 1 | |
| Post-conditions: | User adds script | |
| Flow of events: | User Action | System Action |
| User select Script tap. |  |
|  | System gets all scripts files name from database and show it in list. |
| User select add script button. |  |
| User select script file. |  |
| User click ok button. |  |
|  | System store file name in database. |
| Exceptions: | User closes application.  User close confirmation frame. | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 9 | |
| Use Case Name: | Remove Script | |
| Actors: | User – System. | |
| Pre-conditions (Use Case ID): | 1 | |
| Post-conditions: | User removes script. | |
| Flow of events: | User Action | System Action |
| User clicks on script tab. |  |
|  | System gets scripts files name stored in database. |
| User selects list of scripts. |  |
|  | User click remove button. |  |
|  |  | System remove selected scripts files name from database. |
| Exceptions: | User closes application. | |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 10 | |
| Use Case Name: | Add Rule | |
| Actors: | User – System. | |
| Pre-conditions (Use Case ID): | 1 | |
| Post-conditions: | User adds rule. | |
| Flow of events: | User Action | System Action |
| User selects rule tab. |  |
|  | Systems get zones list with its devices and show it. |
|  | User select zone than device in this zone. |  |
|  |  | System open new dialog contains data source name for interface chosen and text area to enter limits of condition. |
|  | User selects data source. |  |
|  | User puts limits. |  |
|  | User clicks ok button. |  |
|  |  | System adds condition to rule. |
|  | User enters rule name. |  |
|  | User selects list of scripts to rule. |  |
|  | User clicks ok button. |  |
|  |  | System validates and adds rule to database. |
| Exceptions: | User closes application.  Fuseki server stops working. | |

|  |  |
| --- | --- |
| Use Case ID: | 11 |
| Use Case Name: | Check rules |
| Actors: | User – System. |
| Pre-conditions (Use Case ID): | 1, 2 |
| Post-conditions: | System check rules and user take action. |
| Flow of events: | System Action |
| System convert rule to query. |
| Run query over RDF. |
| If it happen show dialog to user with scripts that selected with rule. |
| System allows user to run script. |
| Exceptions: | User closes application.  Fuseki server stops working. |

## **Architecture**

**Cacti Adapter**

Devices Poller

Data Poller

Parser

**Expert System Core**

**Persistence Layer**

**Output Controller**

Jena RDF

SQLite

**User Interface**

The architecture contains the following packages

1. Cacti Adapter

This package is used for connecting the system with any monitoring tool like Cacti and Zenos to poll devices data and information required to perform core operations. It includes the following components

* 1. Device Poller: It is a component wherein the developer will implement the function needed to poll devices information from any monitoring tool.
  2. Data Poller: It is a component wherein the developer will implement the function needed to poll devices data from any monitoring tool.
  3. Parser: It is a component wherein the developer will implement the function needed to parse the devices data.

1. Expert System Core

The core layer which is responsible for rule checking and firing

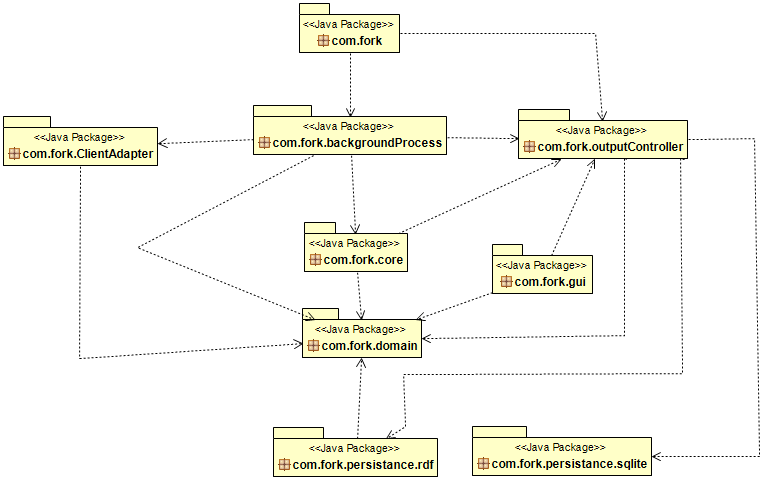
1. Persistence Layer
   1. Jena RDF
   2. SQLite Database
2. Output Controller

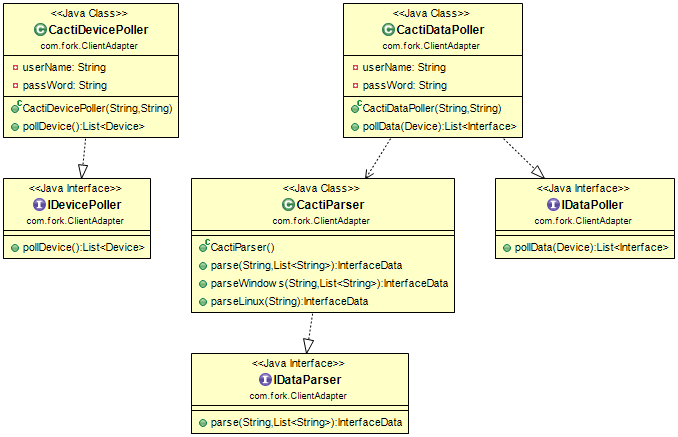
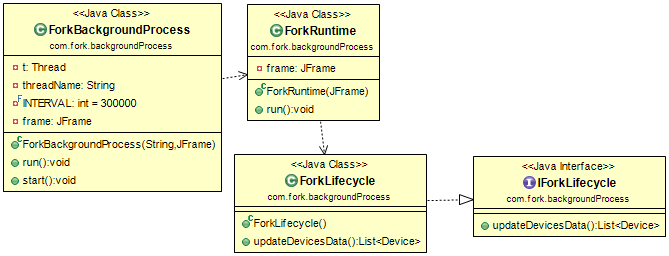
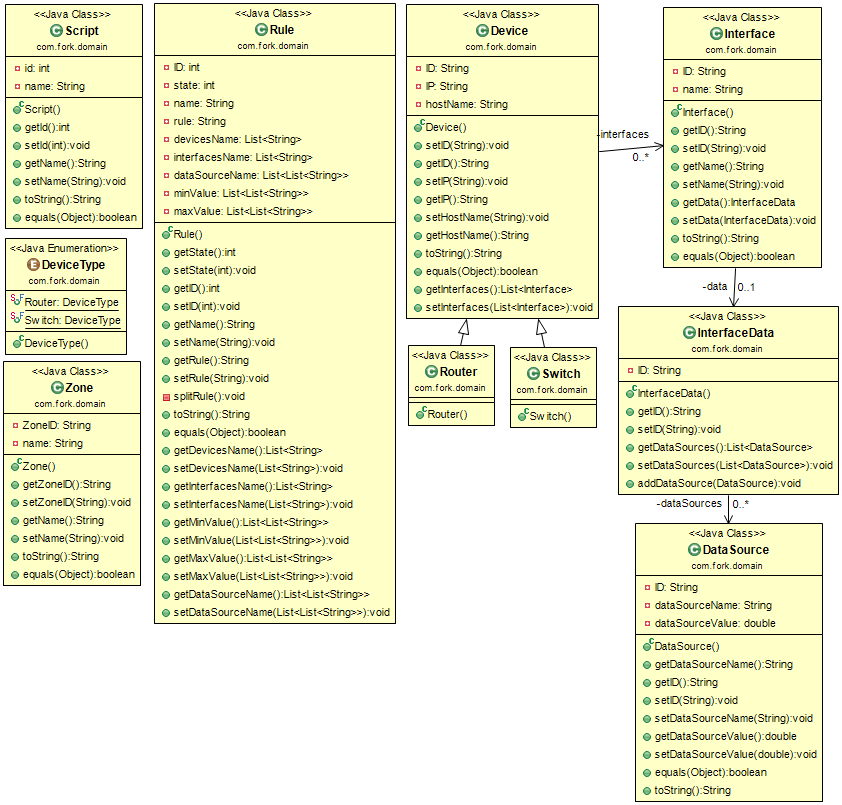
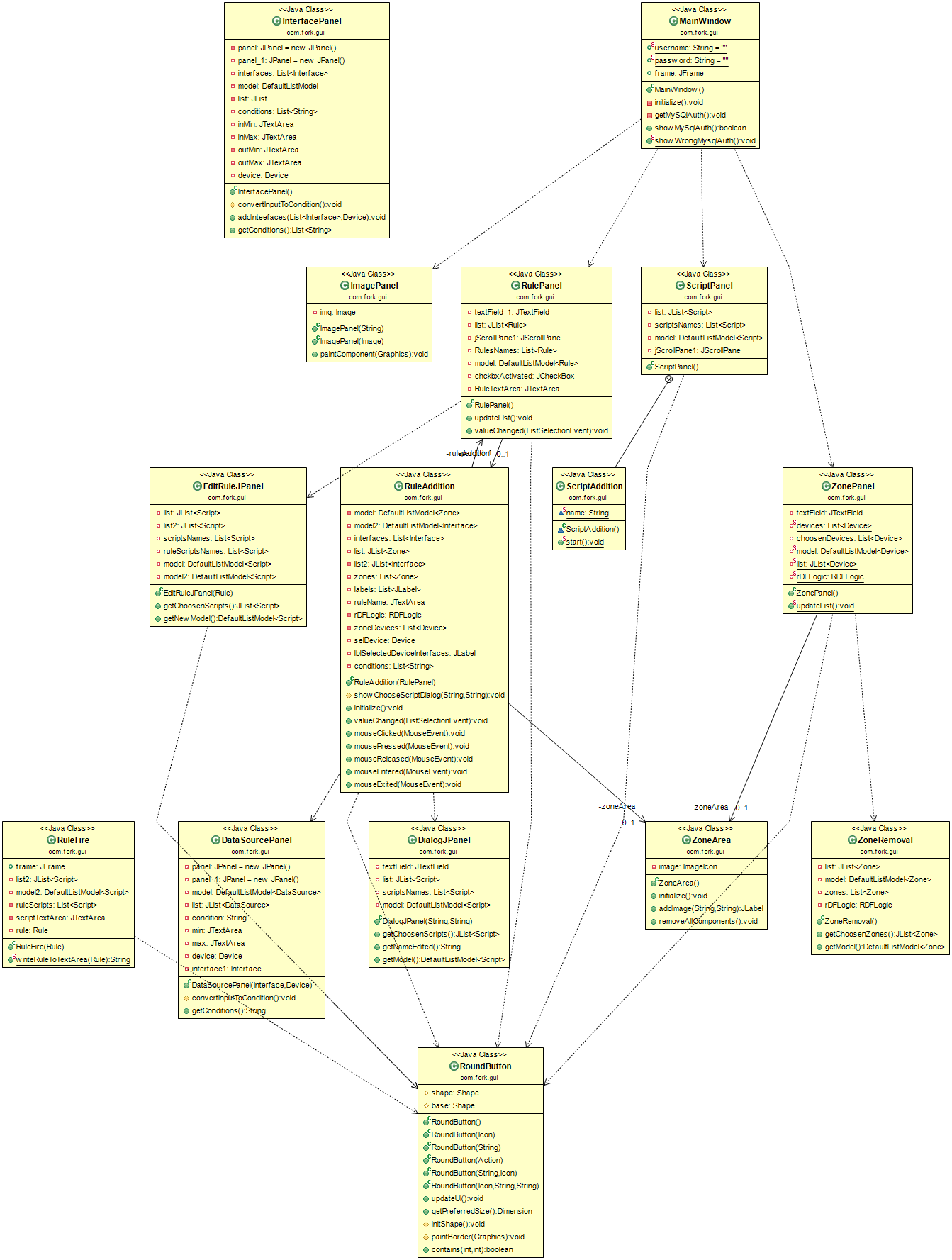
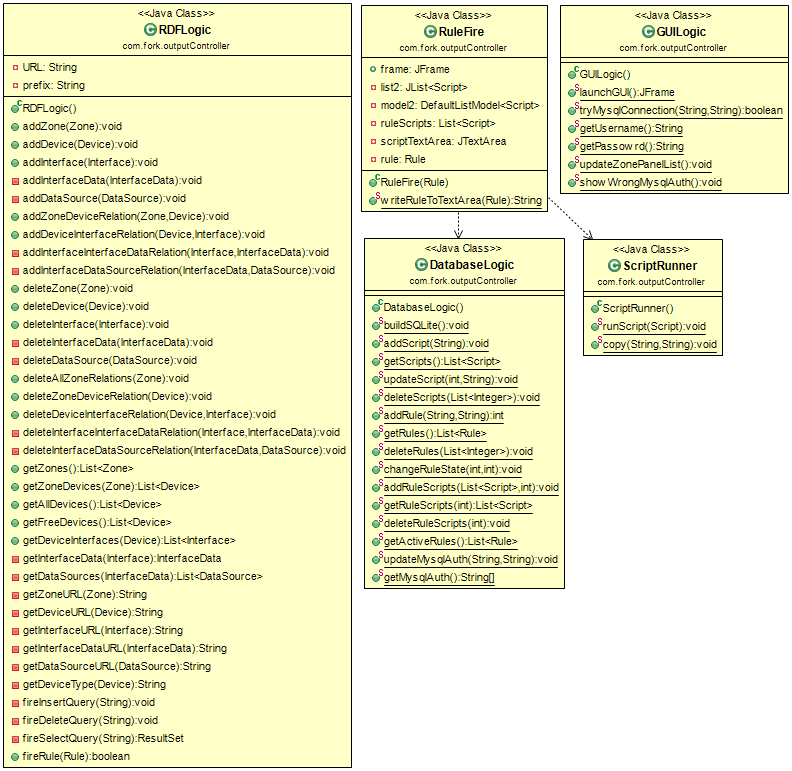
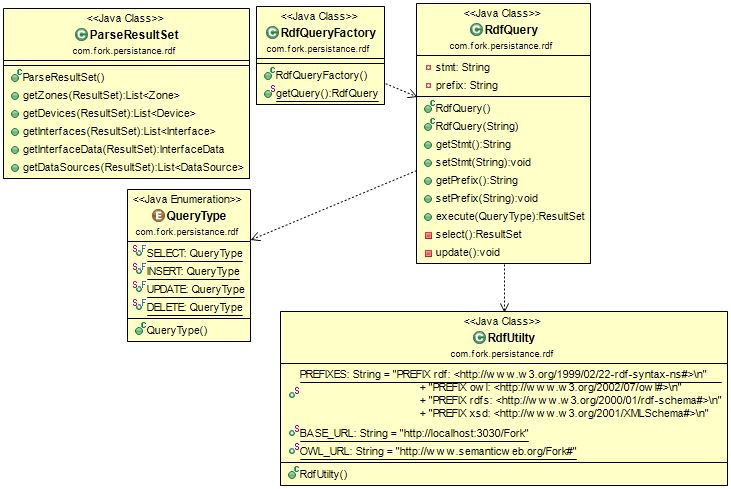
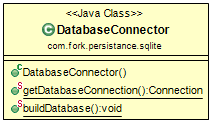
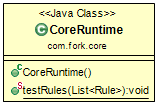
It is the brain of the system, it connects with most of the layers like Jena, SQLite, core, and interface.

1. User Interface

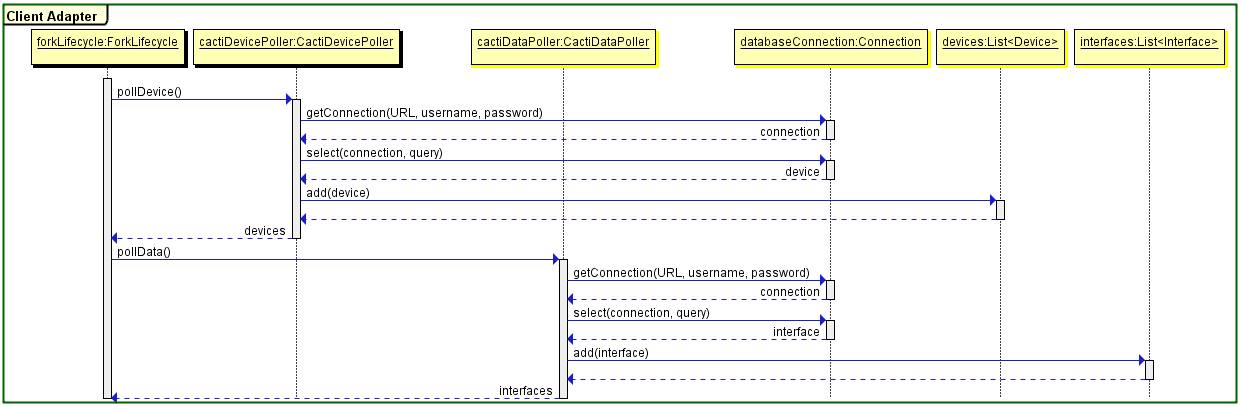
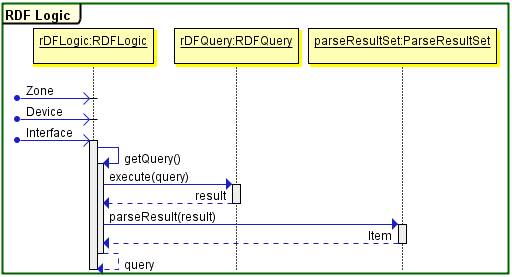
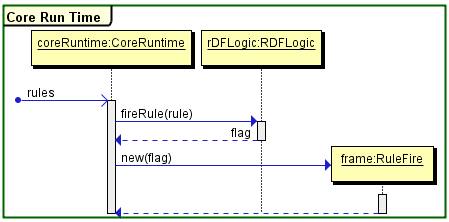
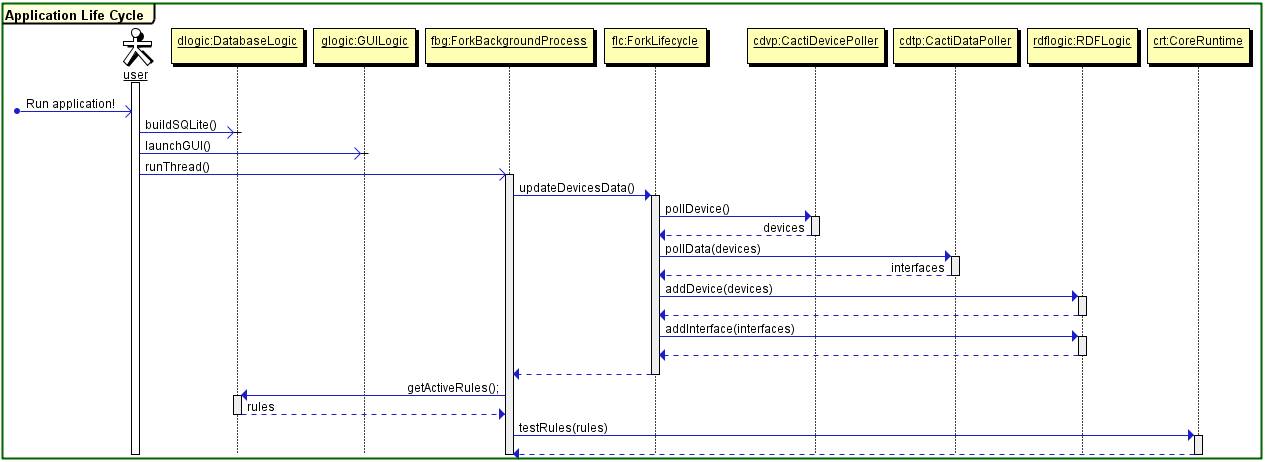
This component displays the system outputs to user through a graphical user interface. And making the user interact with the system.

## **Package and Class Diagram**



1. Client Adapter Package  
   
2. BackgroundProcess Package
3. Domain Package  
   
4. GUI Package  
   
5. Output Controller Package  
   
6. Persistence rdf Package  
   
7. Persistence SQLite Package  
   
8. Core Package  
   

## **Sequence Diagram**



# Chapter 4: Snapshots and Manual

Fork is a useful expert system for help network experts to automate the process of detecting common problems by providing some of rules that previously happened.

## **Configuration and installation:**

1. Application prerequisites :

You have to install **Cacti** network monitoring system that is responsible for drawing required charts for a given device whether it is a router or switch or pc, also you have to configuring SNMP protocol in your operating system so it will be used by cacti for polling data from monitored device to be displayed in a visual chart.

Do not forget to remember username and password for MYSQL cacti user, because they will be used by fork system.

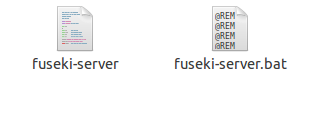
You must run Apache fuseki server before run Fork application.

You have to setup a java runtime environment to run Fork application.

1. Application files :

You will be provided with the application in a folder contains Fork.jar file and apache-fuseki server.

To open the fork application you have to start fuseki server so you must enter the folder of apache-fuseki and run the file according to below figure



Linux: run fuseki-server file

Windows: run fuseki-server.bat file

So now if the server running successfully without any problem, you are ready now to start Fork application.

1. How to run the application :

To run Fork application, as mentioned above you will be provided with a jar file called Fork.jar so double click on this file to run Fork application whether you have Linux or Windows operating system.

## **Application screens explanation:**



Figure

When you run Fork application for the first time the system will ask you about Mysql Cacti username and password as mentioned above, you have to provide this information to be able to continue use the system.

Once the application run you will find three tabs to browse between.

### **Zones tab**

The left panel contains all non-zoned devices in cacti application, so you can add a new zone by enter the zone name as shown in (Figure 1.2,1.3) and add the required device in the zone panel (the right hand side panel) and a confirm message will be appeared immediately if the zone added successfully   
(Figure 1.4)

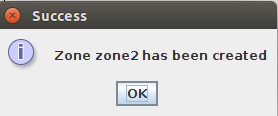


Figure 2

You can also remove a zone by press into the (-) button as shown in figure 1.2 and a new dialog will appear as shown in (Figure 1.5) with the added zones , you should select one of them and press remove button.

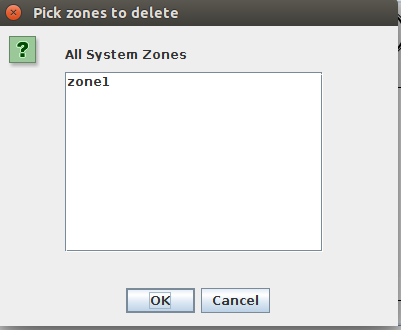
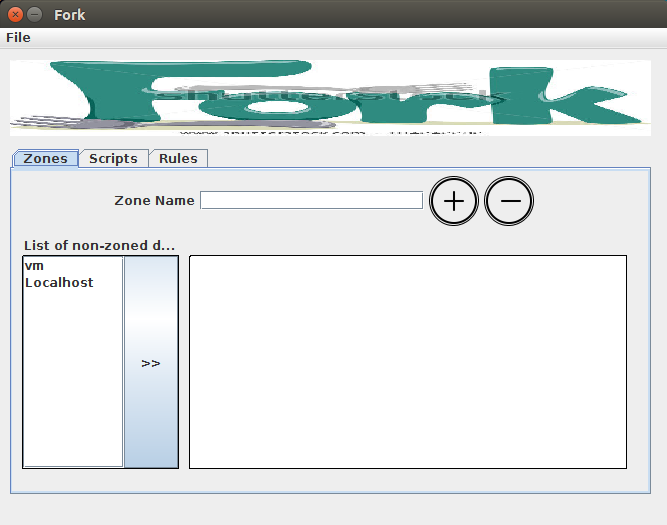
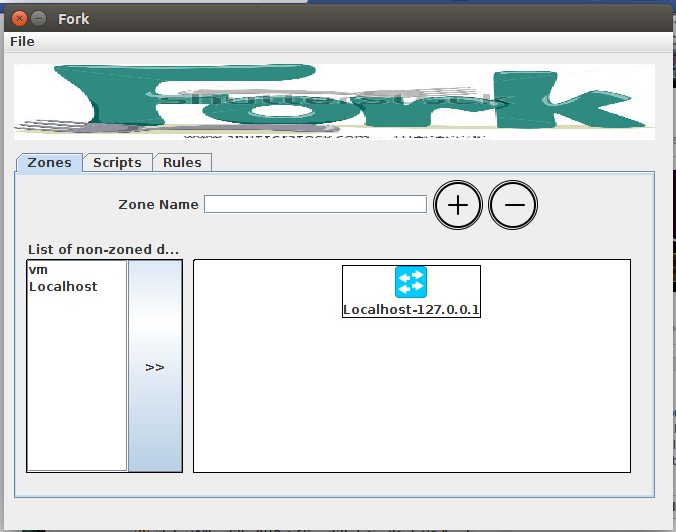


Figure 3

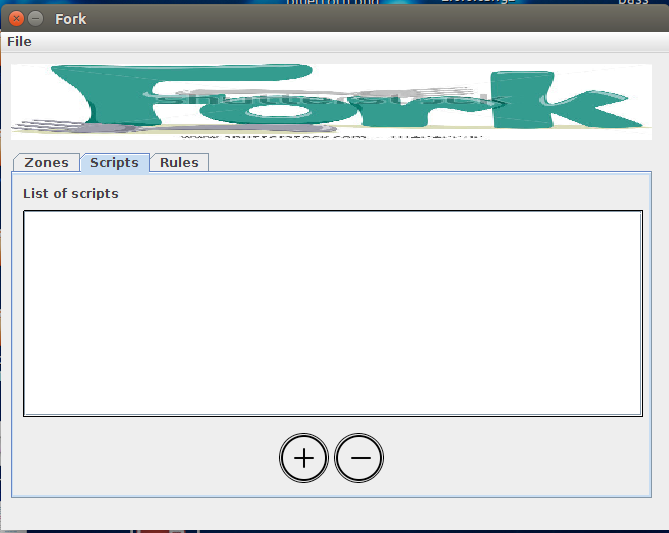


Figure



Figure

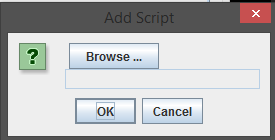
### **Scripts tab**



Figure

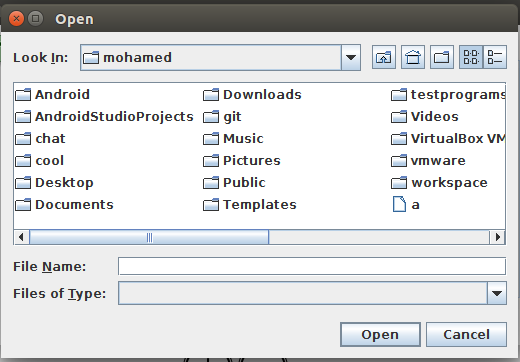
Here you will find a list of all scripts that you want to add to save in Fork application for executing this scripts when one of the rules match with a real condition.

If you want to add a new script you have to write your script in a file and press (+) button a new dialogue will appear (Figure 1.7)



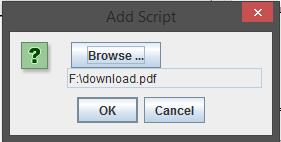
Figure

Then Click browse button (Figure 1.8)



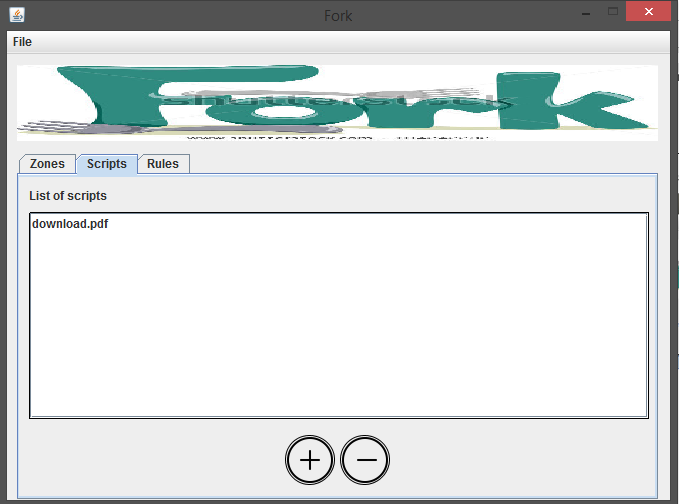
Figure

Select you script file a and click open then the previous dialogue will appear with the path and name of your script file (Figure 1.9)



Figure

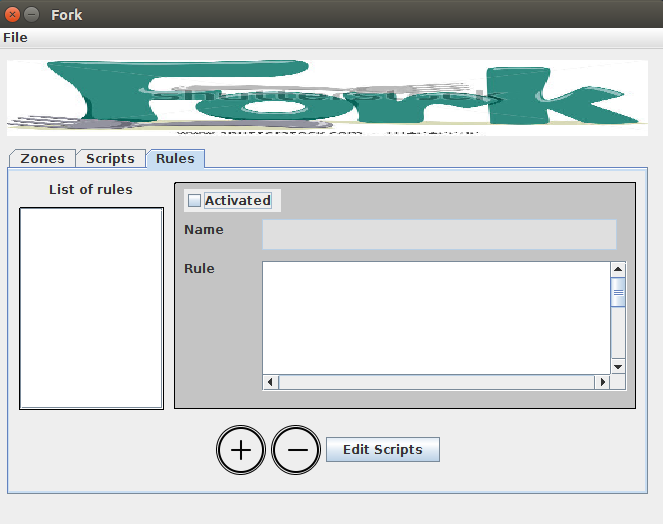
Then press ok button and your script will be add to Fork database and will be appeared in list of scripts in the centered panel as shown in (Figure 1.10).



Figure

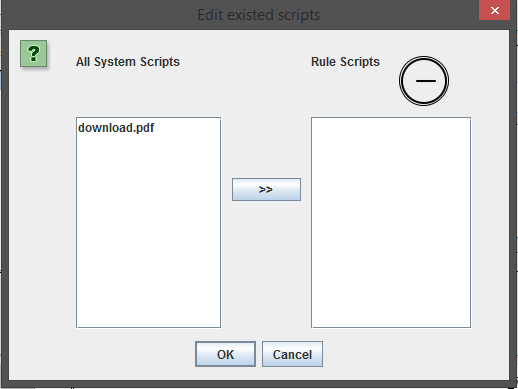
Also you can remove a script by select a script from the available scripts as shown in (Figure 1.10) and click the (-) button.

### **Rules tab**



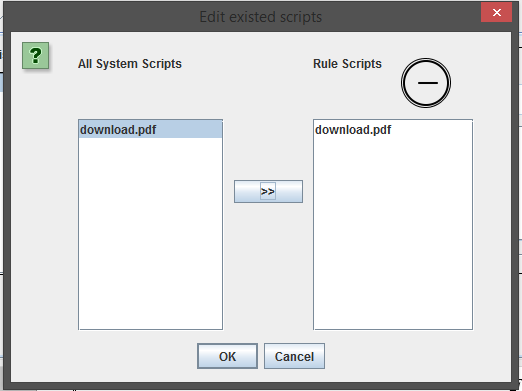
Figure

* This tab is responsible for adding the required rule in Fork system from the network expert.
* In the left panel you have the list of saved rules.
* In the right panel you have the information about the selected rule from the left panel such as (name of the rule and the rule itself as a syntax) .
* You can activate and deactivate the rule form the check box **Activated** in (Figure 1.11) and also you can modify the rule from the big text box **Rule**.
* You can associate a script from your saved scripts to an existing rule to be executed after the rule matches, this is done by press **Edit Scripts** button so a new dialogue will appear as shown in (Figure 1.12)



Figure

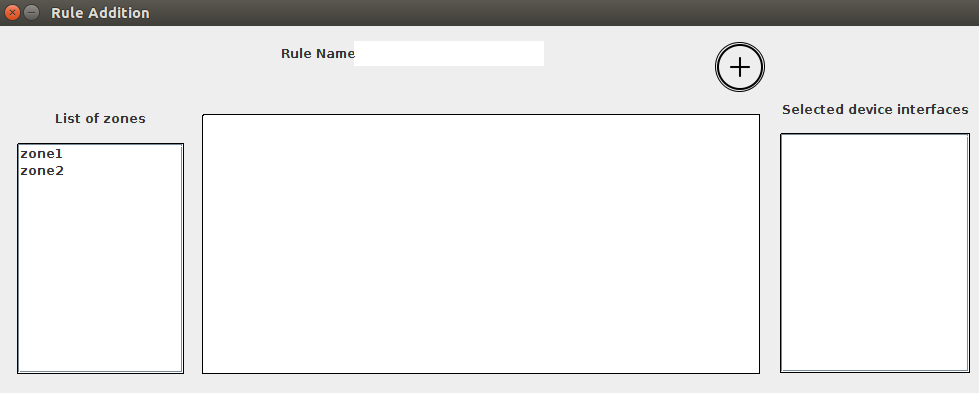
* You have to add the required scripts form the list of available scripts and press ok.
* You can remove any existed scripts from the selected rule by clicking on (-) button



Figure

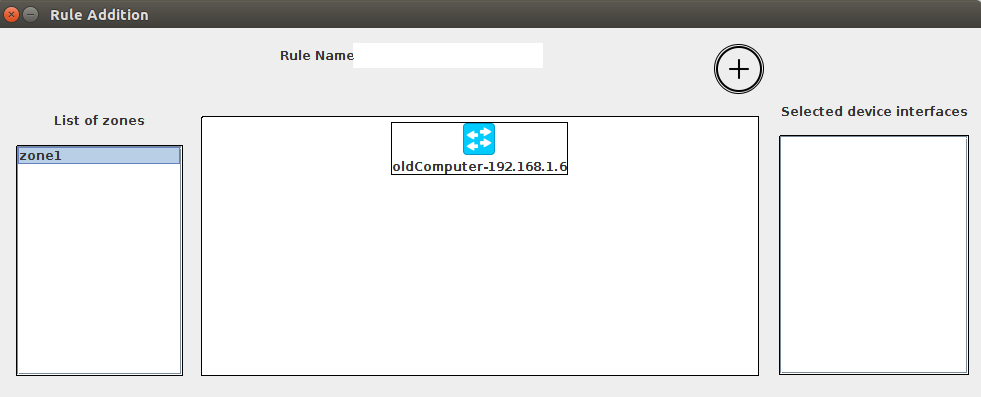
* The most important thing in the application is rule addition and this is done by the (+) button shown in (Figure 1.11)

And new dialogue will appear



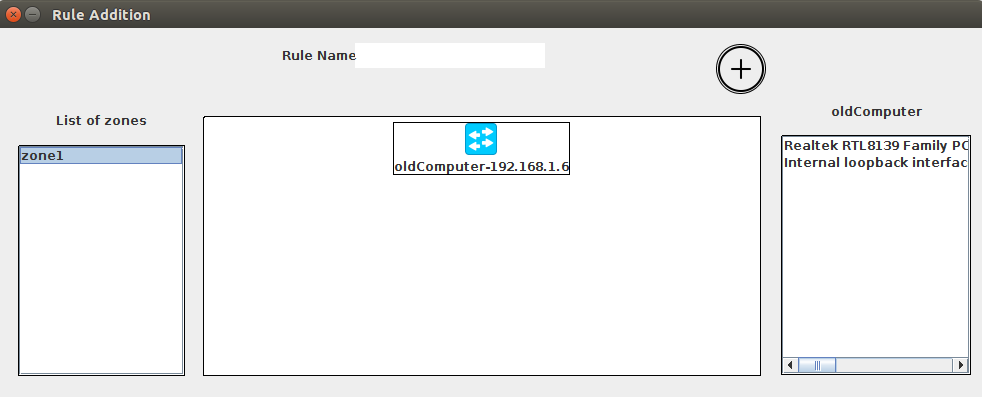
Figure

* A new dialogue contains a three panels, the left panel contains a list of saved zones that the network expert previously created in the application, when you press one of the zones the centered panel will updated with the devices that is attached with the selected zone as shown in (Figure 1.15).



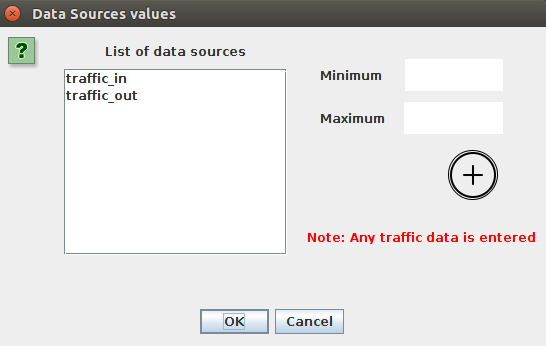
Figure

Then if you select one of the shown devices the right panel will updated with all available interfaces for the selected device as shown in (Figure 1.16).



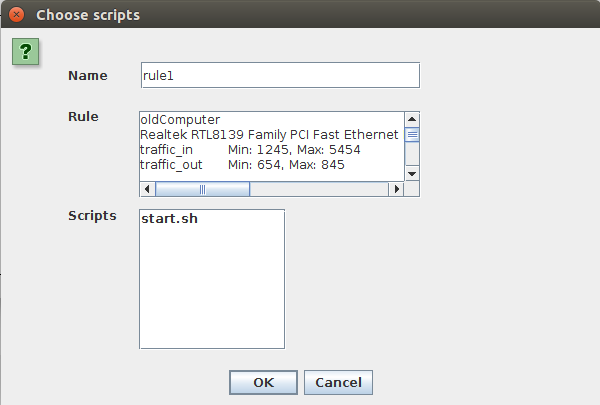
Figure

* Select one of the interfaces in the right panel and a new dialogue appear (Figure 1.17)



Figure

* In Figure 1.17 the right panel contains all the data sources available for that device, so you have to select the data source that will be involved in the rule conditions and enter the its minimum and its maximum and press (+) button, the values will entered in bits/seconds then press ok, then you can repeat this step for any interface and device to form your rule conditions then press (+) button in (Figure 1.16).
* When you pressed on the (+) button in previous step , a new dialogue will appear (Figure 1.18)



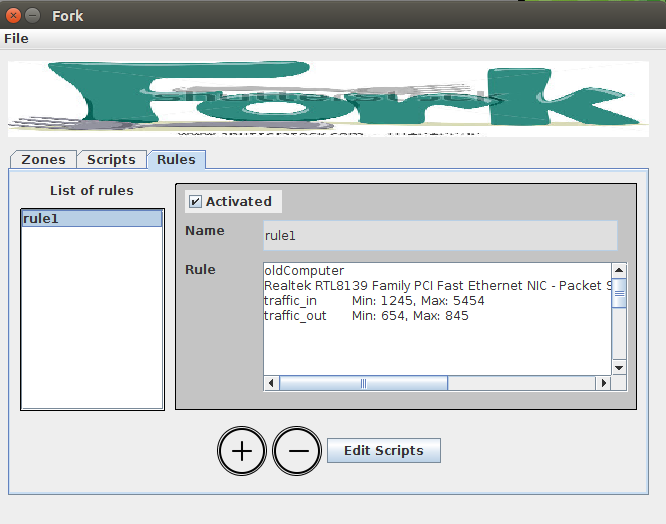
Figure

* In this dialogue you have to confirm that this is your rule that you formed and you want to add

The first box is the rule name then the rule itself as syntax and you have to select a script from the available scripts that you associate with this rule to run after the rule is fired. Then press ok to add this rule to the Fork system.

You can edit the rule manually and rule name from that dialogue but you have to watch out for the syntax.

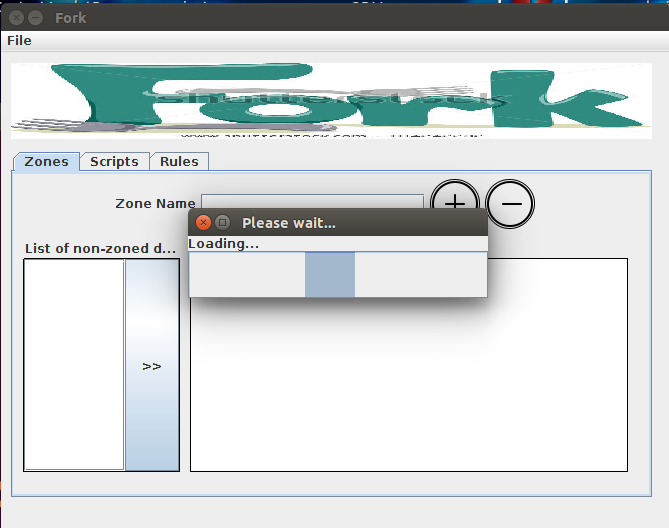
* After adding the rule the rule information will be displayed for editing when you select it as show in (Figure 1.19).



Figure

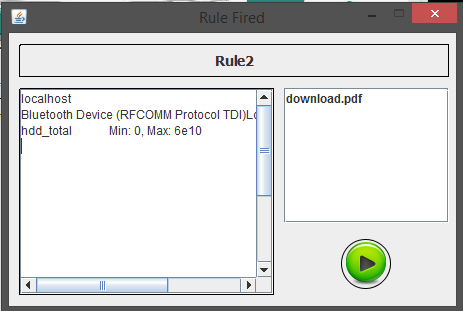
Also you can select a rule and remove it from (-) button.

### **Application background process:**

* Fork system use a background process for retrieving data from network monitoring system after specific time interval so this is a task that need for synchronization and more security. So every specific time you will find your application load the data and update the application but for security reasons we blocked all the interactions with the user interface to guarantee that the data is safe and the application will keep safe so the GUI will tell you that as shown in (Figure 1.20).

Figure

* In Figure (1.21), a certain rule has been fired. The system should open a new frame that contains rule name, description and scripts to run when the user clicks on the run button.



Figure

# Chapter 5: Conclusion