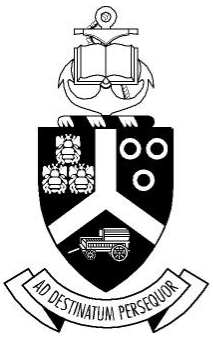


UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS 301 - Software Engineering

Amazon
Network Visualizer
Demo 3

Not-Like-This

Authors:	*Student number:*
Jedd Schneier	u13133064
Daniel King	u13307607
Muller Potgieter	u12003672

September 11, 2016

Contents

1 Vision

The Network Visualizer is intended to be used by registered Amazon Web Services (AWS) users. The visualizer is primarily aimed at consumers of AWS, in order to provide a simple and clear representation of the various networks' structures.

In order to access the visualizer, the user must first submit their AWS password and secret password. The visualizer then attempts to access the server, using the provided passwords. If this is successful, the visualizer will scan the specified network and log the nodes (such as instances and VPC's) and their relationships. It uses this information to construct a tree-esque representation of the network.

Using this representation, it is then translated into HTML. Making use of the vis.js library, the structure of the network is presented in a clear, visual hierarchial structure. The page also allows the user to specify which region they wish to be scanned and represented.

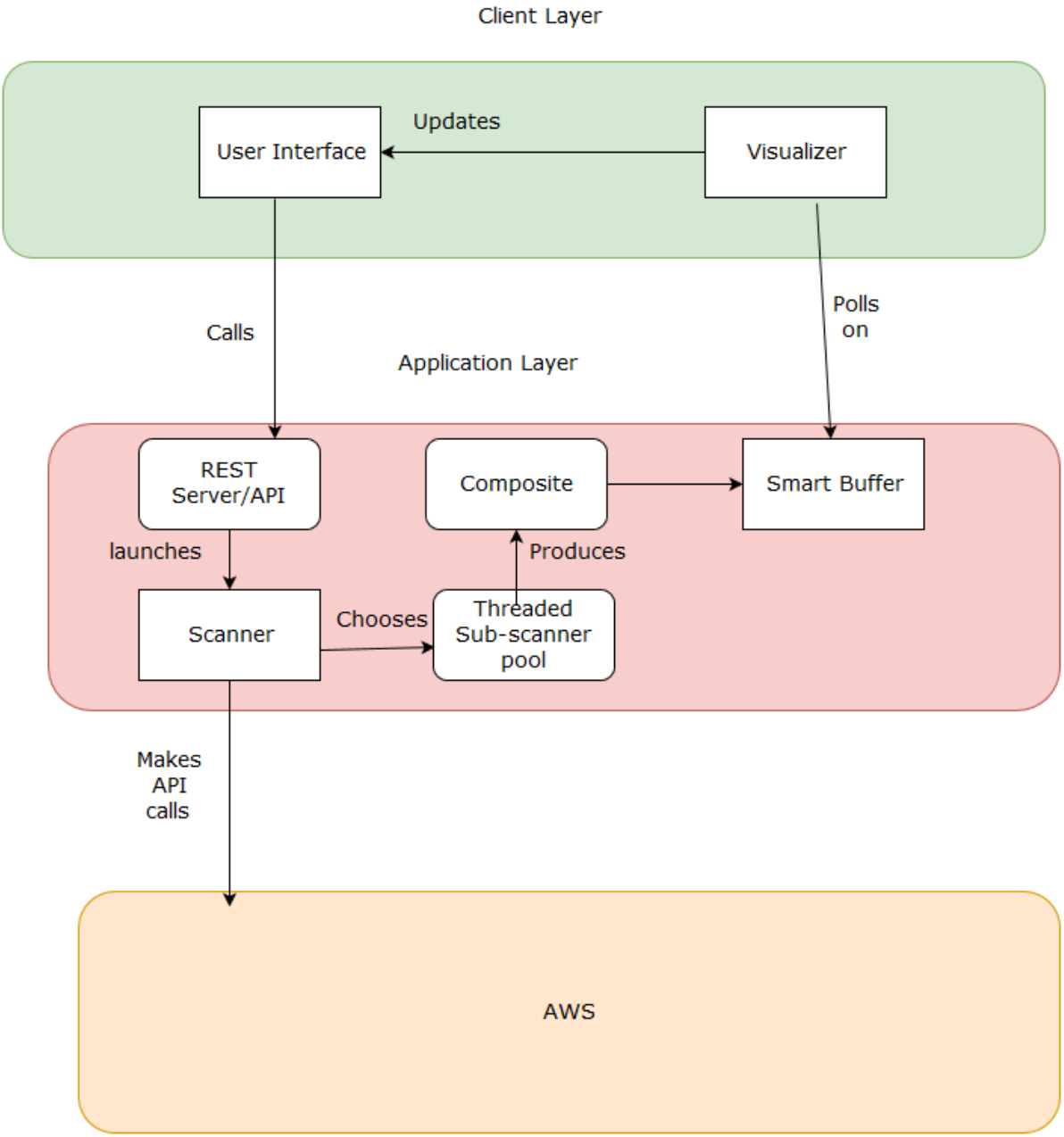
2 Scope

- The user requires a valid Amazon Web Services account (or access to one), in order to make use of the visualizer.
- The user requires an internet capable device and a modern browser to access the visualizer's page.

3 Architectural Requirements

3.1 Architecture Scope

Layered Architecture: RESTFUl Architecture Threading Pool tactic:



3.2 Quality Requirements

3.2.1 Critical

1. Performance: The first and perhaps most important requirement. The system is working with massive amounts of data. The system is redundant if it can not process and show the data quickly. The majority of our architectual design has been done to improve performance. To keep it within its 3 second output requirement.

2. Scalability: This is one of the major quality requirements for the service, as stated before. The system will have to work with very large networks in future and thus, the algorithms within will need to scale well, such that they can work with these large networks and so that performance costs are minimized. So far, our performance requirements matches our scale.

3.2.2 Important

1. Security: From our own experince, we know how important security is with this system. As we are working with sensitive client info (links to credit cards), it is imperative that we make sure the system is secure and does not allow for this information to fall in the hands of unwanted 3rd parties or be compromised in any way.

2. Usability: Usability is very important, as ease of use is imperative to the client's experience. However, it is not as important as perfrormance or scalabilty; as at higher levels, regardless of usability the sytem will be required to meet those requirements. The system is going to be used by the Amazon clients who will not necessarily have programming knowledge, thus the end product needs to cater to these users.

3.2.3 Nice To Have

Maintainability: The system is going to be used by the Amazon clients who will not necessarily have programming knowledge, thus the end product needs to be catered to these users such that they are able to use the program efficiently.

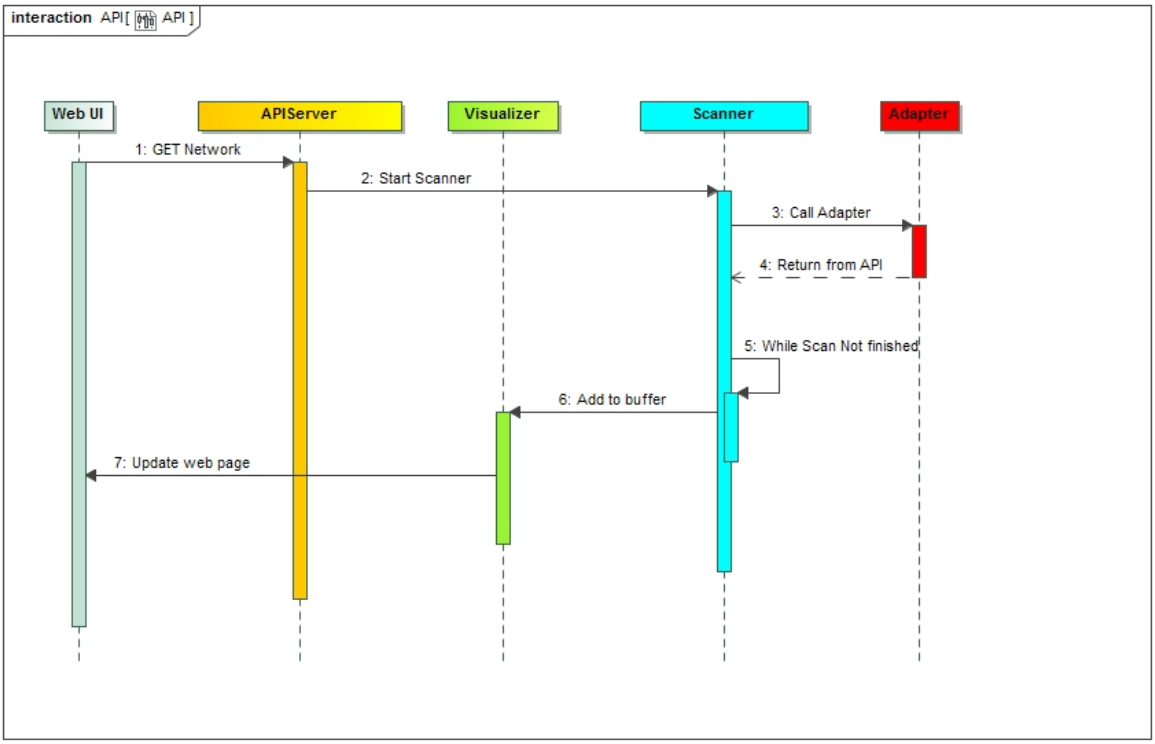
3.3 Architectural Constraint

The user must have an existing and valid account with the Amazon services.The user must use a virtual network. The must have a valid credit card to set up their account. The user must have a relitavely stable, preferably uncapped, internet connection. The user must also set up an access key and secret key on their end.

3.4 Access and integration channels

The information regarding the networks is taken via the Amazon API services.

- The user should login via the existing Amazon site and the visualizer should integrate into the site itself.
- The Front end must intergrate through the REST server with the backend.
- The scanner must intergrate with the Shared Buffer.
- The Scanner mustt intergrate with AWS.
- The interface must intergrate on all browsers and with standard web technologies



4 Architectural Design

4.1 Layered Architecture

Our core architectural design follows a standard Client-Applciation-Server layered architecture. As seen below, the "Server" (not to be confused with the Application server) is the AWS server we connect to through their API. The product does not persist to only read from it and we have no control over its functionality. It will be left for an intergration requirment and will not be disccuesed furhter in this section. Additional information on the AWS sytem and their API can be found here: https://aws.amazon.com/documentation/. Previousl,y we used a bridging layer , Adapter, to join the Application and Server layer, but it has since been incorporated into the Scanner, to reduce communciation overhead and facilitate threading tactics.The primary backend of the system lies on the Application layer, which is broken down into the follwoing components:

1. Scanner: The most important component, constructs a scan based on differnt priorities. It forms a Producer-Consumer relationship with the Visualizer. Once launced, it will continue to produce network trees (see below) for the buffer until it receives a pause/stop command, or completes a scan of the network.

2. Composite: Following the composite design pattern, this object represents the hierarchial Network tree, either in part in whole.

3. Threaded sub-Scanners: Each logical part of an AWS network has an assigned threaded scanner to it. Depemding on what is a priority to scan first, the scanner will launch a number of threads for each part, which will in turn, launch its on sub-scan threads; based on perfromance requirments. Each threaded SubScanner will scann 100 of its children, place them buffer and then continue to scan another 100.

4. Smart buffer: The link between the Visualizer and the Scanner as well as the "brain" of the system. The individual scanner threads will add to the buffer as soon as they have finished constructing a tree, then go back to scanning. The trees arrive disjointed and possisbly overlap. The smart buffer constructs the entire network tree from the disjointed trees it recieves accurately. It stores the most up to date construct for the Visualizer.

5. REST API and Server: The REST server joins the backend to the front end. See architecture below.

The client layer consists of a Single page Application and the Visualizer. The SPA is the user interface with all the scanning options and controll. and the visualized graph and scanned information. The Visualizer polls on the REST server requesting latest tree from the smart buffer, rendering it and presenting it to the user.

4.2 RESTFUL Design

User interaction is mapped from the user interface onto diffennt API calls, on the server. Each call launches a backend method to fullfill the request. The API calls are:

1. POST, ConnectToAccount: Takes the users' access and private key and creates a conenction to AWS for the remainder of the session.
2. GET, ScanNetwork: Launches a scan of the entire network, Visualizer polls on results as they stream in.
3. GET, StopScan: Cancels the current scan.
4. GET, PauseScan: Pauses the current scan.
5. GET, ResumeScan:Resumes the current scan. Visualiser polls on buffer.
6. GET, ScanFrom: Performs a subscan starting from the given point and scanning around its general vicinity.
7. GET, ScanUP: Scans the next logical area above what the ScanFrom scanned.
8. GET, ScanDown: Scans the next logical area below what the ScanFrom scanned.

4.3 Threading Tactic

Each network component (Region,VPC,Subnetwork,Instance) has a threaded scanner associated with it. It is up to the Scanner, depending on what the user is searching for, to construct the scan from the subscanners. By defualt, The scanner will thread a RegionScanner for each region and each one will have one scanner for the other parts. However, if the scan is narrowed, then scanner will thread on the most cost and performance effective way to compelte the scan . For example, if the scan was to only scan a single VPC then other Regions would not need to be scanned and thus only one RegionalScanner will be launced, one VPCScanner will be launced and it with thread on collections of its subnetworks. We trying to keep number of threads needed to a mninimum, but they are invaluabel to speeding up the scanning.

5 Functional Requirements

This section specifies the functional specifications for the AWS Network Visualizer system. It defines the user-system interaction and relationship between users and the product. It will provide the expected functionality for all user cases as well as the activity processes for the system.

5.1 Description of AWS

Amazon web services provide a cloud based service for hosting a clients network. There is a lack of information regarding ones network, specifically the logical representation on the system for the client to make sense of their network. The network visualizer aims to improve clients understanding of their own network, how AWS works and possible insights in the managing of their network.

5.2 Required Functionality

The system must be able to:

1. Be accessible to registered, valid AWS customers.
2. Scan the networks located in different regions.
3. Provide an interactive hierarchical and visual representation of the networks.
4. Give additional information on the network statistics, construction, etc.
5. Provide a clear image of the clients' virtual networks.
6. Improve the AWS clients' experience.

5.3 Use case

The usage of the system will typically consist of passing credentials to the system in the form of users access and private key for their AWS account. If valid the system will ask the user to choose a region to be scanned. All artefacts in that region will be scanned and a network representation will be constructed that will be the displayed to the user. The representation can also be interacted with to expand nodes and request further information.

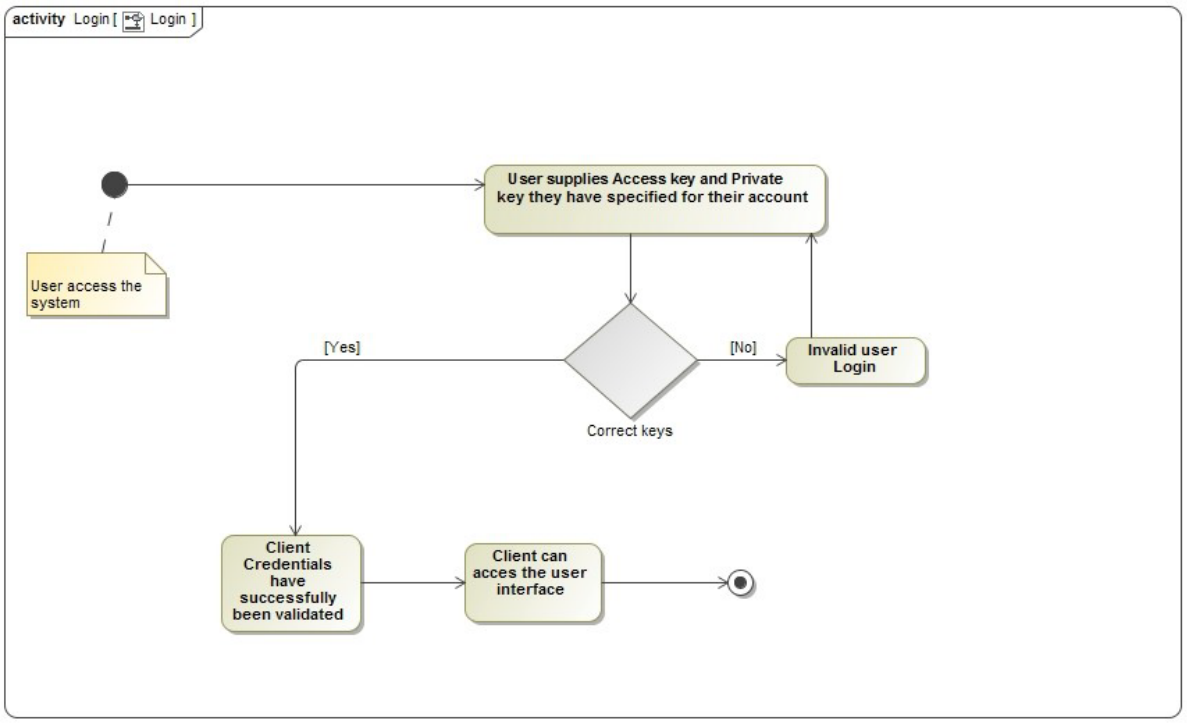
5.4 Use case/Service contracts

Use Case	Pre Condition	Post Condition	Description
Viewing the Hierarchy	The application must be connected to the server and able to read in network data. The server must be active and have acess to AWS.	The web page continuosly updates with the new data being loaded.	This use case forms the core of the project, as the primary purpose of the application is to visualise the structure of the virtual network.
Select a region	The application must be connected to the server and able to read in network data. The server must be active and have acess to AWS.	The previous region's visualisation is replaced by the selected region.	There are a number of AWS regions. For simplicity's sake, the visualiser only visualises one at a time, but allows for switching betweeen them.
Zooming in/out. Moving the hierarchy	The application must be connected to the server and able to read in network data. The server must be active and have acess to AWS.	The hierarchy's position or level of zoom is altered.	Since the visualised network may be large, the user can move it about and zoom in, for a better view.
Clicking a node.	The application must be connected to the server and able to read in network data. The server must be active and have acess to AWS.	The selected node is highlighted. Node related information is displayed below the hierarchy.	Each node in the network has a number of attributes that can provide valuable information. This way, the information can be displayed in a neat manner.
Hovering over a node.	The application must be connected to the server and able to read in network data. The server must be active and have acess to AWS.	Edges connecting nodes of different levels are rendered invisible. Edges connecting nodes on the same level are made vidible.	It is possible for nodes on the same level of the network to have relationships. They are normally hidden, in order to present a cleaner hierarchy.

5.5 Using the System

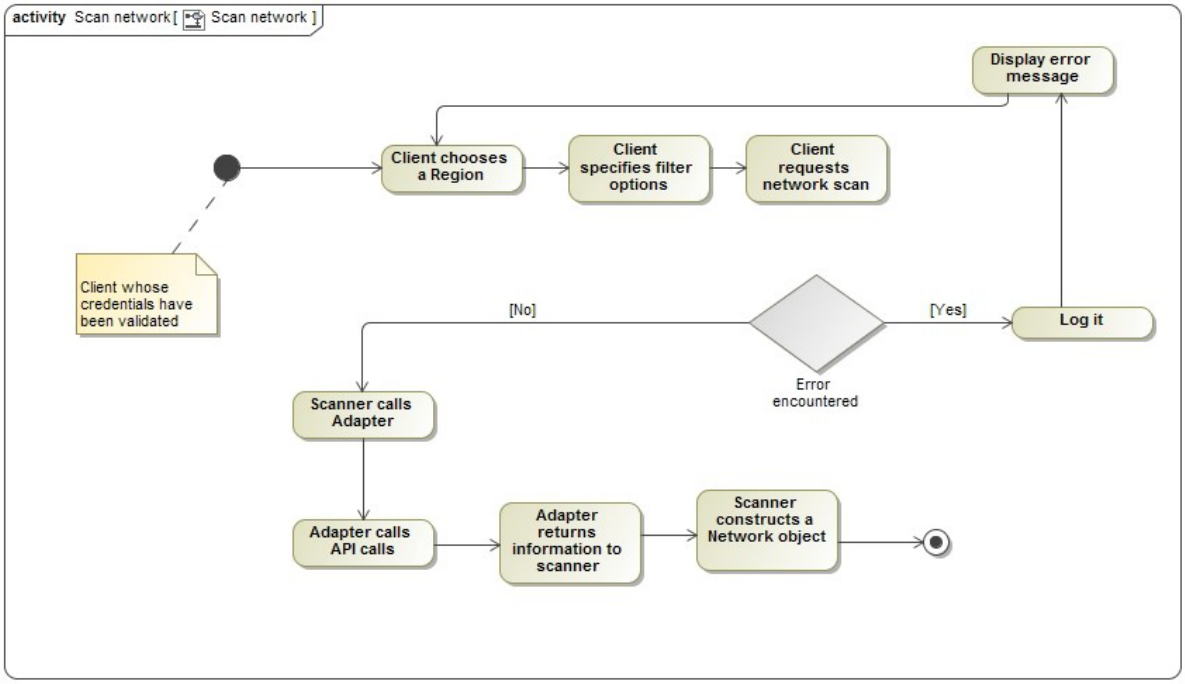
5.5.1 Login

Description
This function validates a clients credentials to allow them to scan an accounts network.
Inputs
Clients access key and clients private key.
Processing
The access key and private key is passed to a Credentials object. The credentials are validated by trying to connect to an account with those keys.
Outputs
If validation successful:
Access to the visualization interface.
If invalid:
Error message and retry prompt.



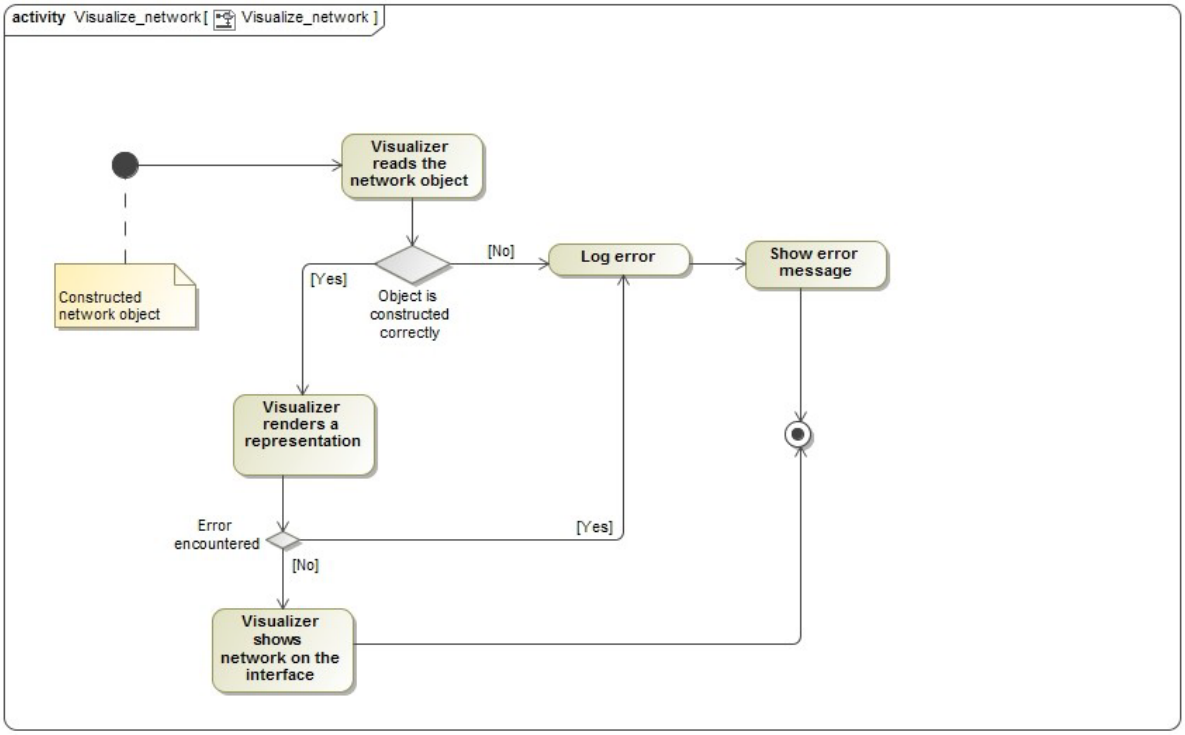
5.5.2 Scan Network

Description
This function scans the users networks in a specified region. **Inputs**
Specified region and filtering options to limit the data desired.
Processing
The scanner connects through the adapter interface, makes the desired API calls and constructs a representation object of the network
Outputs
If no exceptions:
The visualization application will be passed a Network object to render.
If exception thrown:
Error message, log the problem take user back to interface.



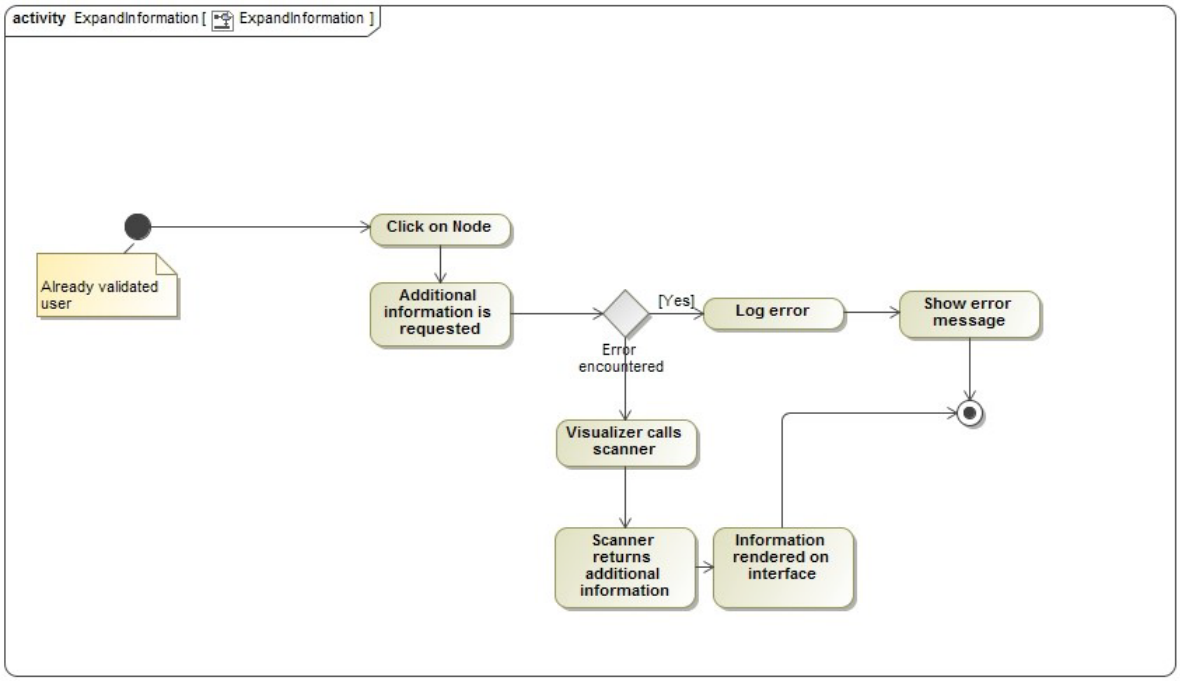
5.5.3 Visualise Network

Description
This function graphically renders the network. **Inputs**
Constructed Network object from scanner.
Processing
The visualizer recursively traverses the Network object, creating a tree hierarchy and rendering it for the user.
Outputs
If no exceptions:
A fully rendered representation of the network.
If exception thrown:
Error message, log the problem take user back to interface.



5.5.4 Expand Information

Description
This function produces a dialogue of additional information for selected node. **Inputs**
User clicks desired node.
Processing
The system will retrieve additional information on the desired node and return it to the interface.
Outputs
If no exceptions:
Additional information will be shown
If exception thrown:
Dialogue shows no additional information, log the error, show an error message in dialogue.



6 Specification Update

The specification has remained largely unchanged. The initial task of creating a high speed network scanner has not changed, however it has been reworked as a plugin, which will allow for easier deployment.

7 Testing

Sadly, our system is not yet at a point where it can be properly tested