

# **RSA** | NetWitness Suite

Ixia CloudLens Integration

Copyright © 2017 EMC Corporation. All Rights Reserved.

## **Trademarks**

RSA, the RSA Logo and EMC are either registered trademarks or trademarks of EMC Corporation in the United States and/or other countries. All other trademarks used herein are the property of their respective owners. For a list of EMC trademarks, go to [www.emc.com/legal/emc-corporation-trademarks.htm](http://www.emc.com/legal/emc-corporation-trademarks.htm).

## **License Agreement**

This software and the associated documentation are proprietary and confidential to EMC, are furnished under license, and may be used and copied only in accordance with the terms of such license and with the inclusion of the copyright notice below. This software and the documentation, and any copies thereof, may not be provided or otherwise made available to any other person.

No title to or ownership of the software or documentation or any intellectual property rights thereto is hereby transferred. Any unauthorized use or reproduction of this software and the documentation may be subject to civil and/or criminal liability. This software is subject to change without notice and should not be construed as a commitment by EMC.

## **Third-Party Licenses**

This product may include software developed by parties other than RSA.

## **Note on Encryption Technologies**

This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when using, importing or exporting this product.

## **Distribution**

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license. EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED "AS IS." EMC CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

# Contents

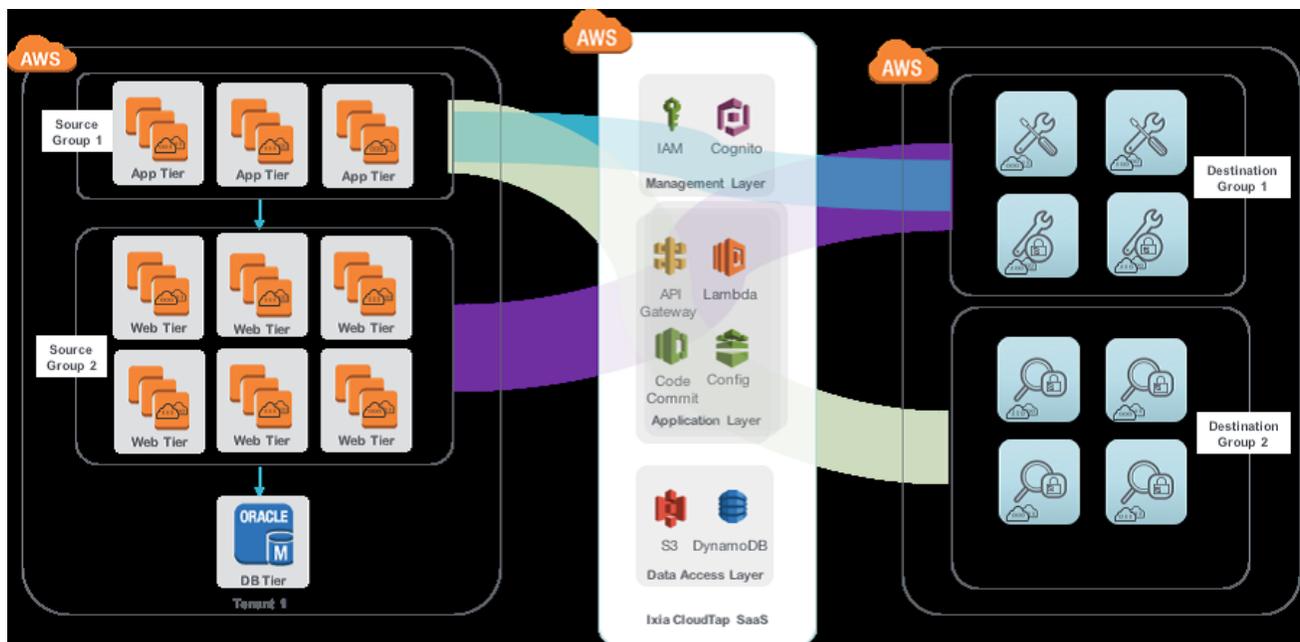
---

<b>RSA NetWitness Suite Integration with Ixia CloudLens .....</b>	<b>4</b>
Prerequisites .....	4
Integration Steps .....	5
<b>Prepare AWS Environment .....</b>	<b>6</b>
<b>Create CloudLens Project .....</b>	<b>7</b>
<b>Install Docker Container on Decoder .....</b>	<b>9</b>
<b>Install Docker Container on Clients .....</b>	<b>10</b>
<b>Create Mapping between NetWitness Decoder and Ixia Clients .....</b>	<b>11</b>
<b>Validate CloudLens Packets Arriving at Decoder .....</b>	<b>13</b>
<b>Set the Interface in the Packet Decoder .....</b>	<b>14</b>

# RSA NetWitness Suite Integration with Ixia CloudLens

Ixia CloudLens™ is the industry’s pioneer Software as a Service solution that moves traditional visibility controls down to each workload—enabling each workload to have its own microcosm packet brokering. Because of its Software as a Service architecture, Ixia’s solution can span across multiple cloud instantiations and has the capability to provide visibility across private, public and hybrid cloud.

CloudLens Public container is a wire data collection system designed specifically to function seamlessly in public cloud deployments. To do this, it must cope effortlessly with rapid elastic scale, it must be a self-serve install for tenants in the cloud with no help required from the provider, it must operate without network constraints on the customer’s network, and it must be out of band, not inline. And CloudLens does this without relying on hypervisor- or provider- specific features, in public or private cloud, for both virtualized network interfaces. CloudLens even works in on-premises bare metal.



## Prerequisites

You need the following items before you begin the integration process:

- Ixia account (<https://login.ixiacom.com/>)
- Access to AWS console
- Network rout-able (and proper AWS Security Groups) for the containers to transfer data to the RSA NetWitness Suite Decoder.

## Integration Steps

The remainder of this document discusses the steps for integrating the RSA NetWitness Suite Decoder with Ixia CloudLens.

1. [Prepare AWS Environment](#)
2. [Create CloudLens Project](#)
3. [Install Docker Container on Decoder](#)
4. [Install Docker Container on Clients](#)
5. [Create Mapping between NetWitness Decoder and Ixia Clients](#)
6. [Validate CloudLens Packets Arriving at Decoder](#)
7. [Set the Interface in the Packet Decoder](#)

## Prepare AWS Environment

---

Prepare the AWS environment by performing the following procedures:

1. Deploy a NetWitness Decoder instance in AWS. For details, see [AWS Deployment Guide](#).
2. Deploy client machines onto which you want to route the traffic to NetWitness Decoder.
  - See the Ixia CloudLens documentation for specifications needed for supported client machines. Ixia public documentation is available at <https://store.ixiacom.com/product/cloudlens-public>.
  - See the AWS documentation to deploy AWS on the supported client machines. The AWS Deployment Guide for 10.6.3 is available here: <https://community.rsa.com/docs/DOC-75925>.

## Create CloudLens Project

---

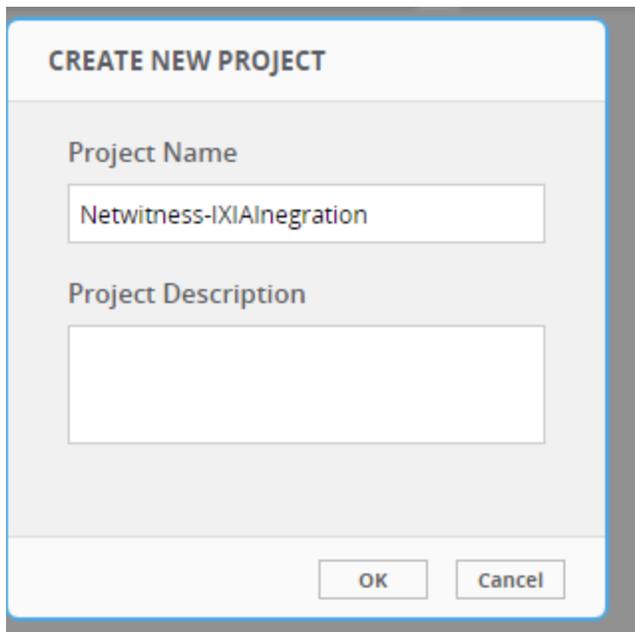
Before you access the CloudLens site for the first time, create an Ixia login account (if you don't already have one) at <https://login.ixiacom.com/>. Send your Ixia login account email to [cloudlens@ixiacom.com](mailto:cloudlens@ixiacom.com) so they can provide you with access to the Sandbox.

Access the CloudLens public site at <https://www.ixia-sandbox.cloud/> and login with your Ixia account credentials.

### To create a new project and get your project key:

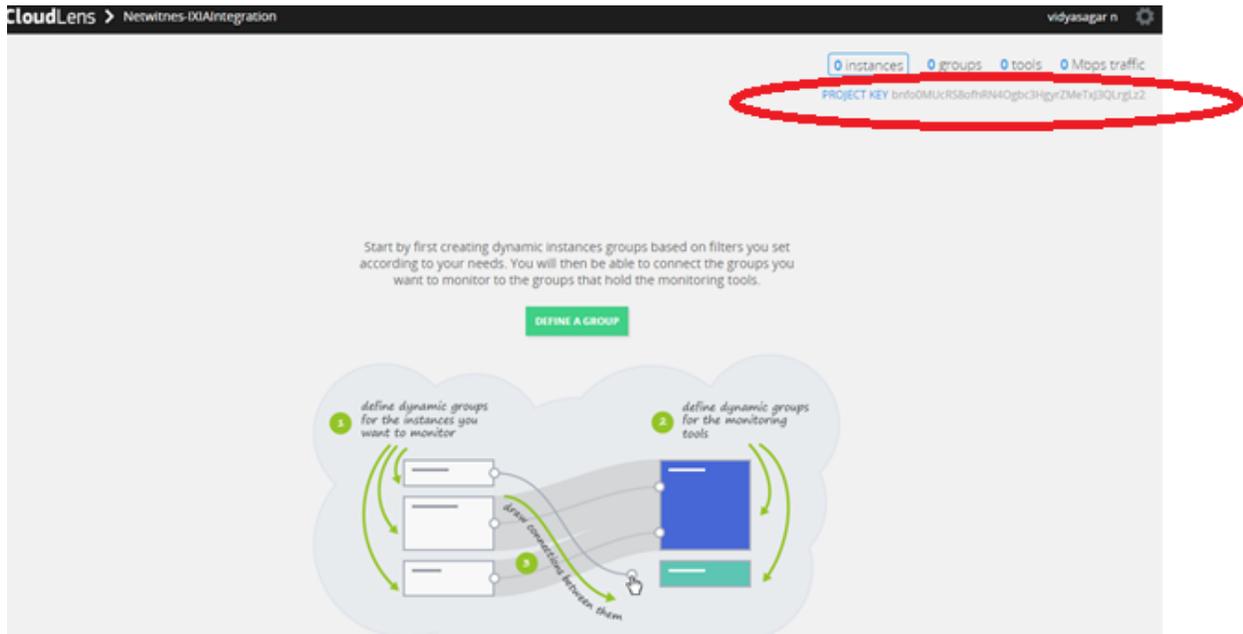
1. Click the + Icon to create a new project.

In the following image, the project name is **Netwitness-IxiaIntegration**, but you can enter any name you wish.



The image shows a dialog box titled "CREATE NEW PROJECT". It has two input fields: "Project Name" and "Project Description". The "Project Name" field contains the text "Netwitness-IXIAIntegration". The "Project Description" field is empty. At the bottom of the dialog box, there are two buttons: "OK" and "Cancel".

- Click on your newly created project and make note of your Project Key. You will need this later for the **Host & Tool** agents.



## Install Docker Container on Decoder

---

Next, you need to install the Docker container onto the NetWitness Decoder.

1. Open an SSH connection the public IP address of the RSA NetWitness Suite Decoder.
2. Install the Docker RPM onto the Decoder:

```
#rpm -iUvh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
# yum -y install docker-io
```

**Note:** Make sure to enable the required repository, so that you can install it.

3. Start the Docker service:

```
# service docker start
```

4. Use the `docker run` command to access the Ixia repository and obtain the `cloudlens-sandbox-agent` container. Replace the variable **ProjectKeyFromIxiaProjectPortal**, which identifies your project key in Ixia portal, with the Project Key you created in the previous section.

```
docker run --name ca \
-v /:/host \
-d --restart=always \
--net=host \
--privileged \
ixiacom/cloudlens-sandbox-agent \
--server agent.ixia-sandbox.cloud \
--accept_eula y \
--apikey ProjectKeyFromIxiaProjectPortal
```

## Install Docker Container on Clients

You need to install the Docker Container onto the client machines for which you want to route the traffic to the NetWitness Decoder.

1. Open an SSH connection the Public IP address of the AWS Client instance.
2. You must enable root access to the OS CLI (for example, `sudo su -`).
3. Install Docker by running the following command:

```
# yum -y install docker
```

**Note:** Make sure to enable the required repository, so that you can install it.

4. Start the Docker service:

```
# service docker start
```

5. Use the `docker run` command to access the Ixia repository and obtain the **cloudlens-sandbox-agent** container. Replace the variable **ProjectKeyFromIxiaProjectPortal**, which identifies your project key in Ixia portal, with the Project Key you created in the previous section.

```
docker run --name ca \  
-v /:/host \  
-d --restart=always \  
--net=host \  
--privileged \  
ixiacom/cloudlens-sandbox-agent \  
--server agent.ixia-sandbox.cloud \  
--accept_eula y \  
--apikey ProjectKeyFromIxiaProjectPortal
```

**Warning:** If you cut and paste commands from a PDF, it is a good idea to first paste into a test editor such as Notepad, and then confirm the syntax before pasting into the OS CLI. We have seen issues where direct cut and paste between PDF and CLI can contain dashes or other special characters that should not be part of the commands.

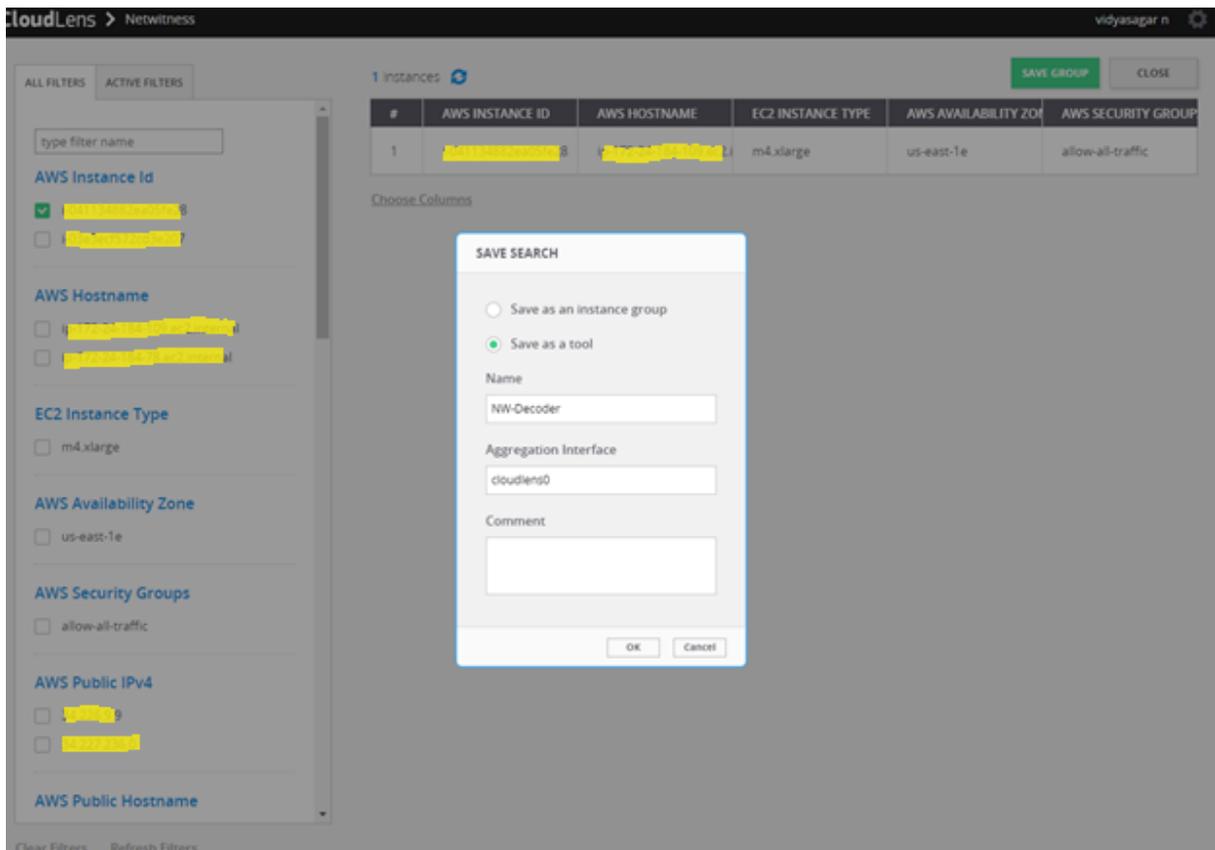
## Create Mapping between NetWitness Decoder and Ixia Clients

Next, you need to create a mapping between the NetWitness Decoder and the client machines for which you want to route the traffic to the Decoder.

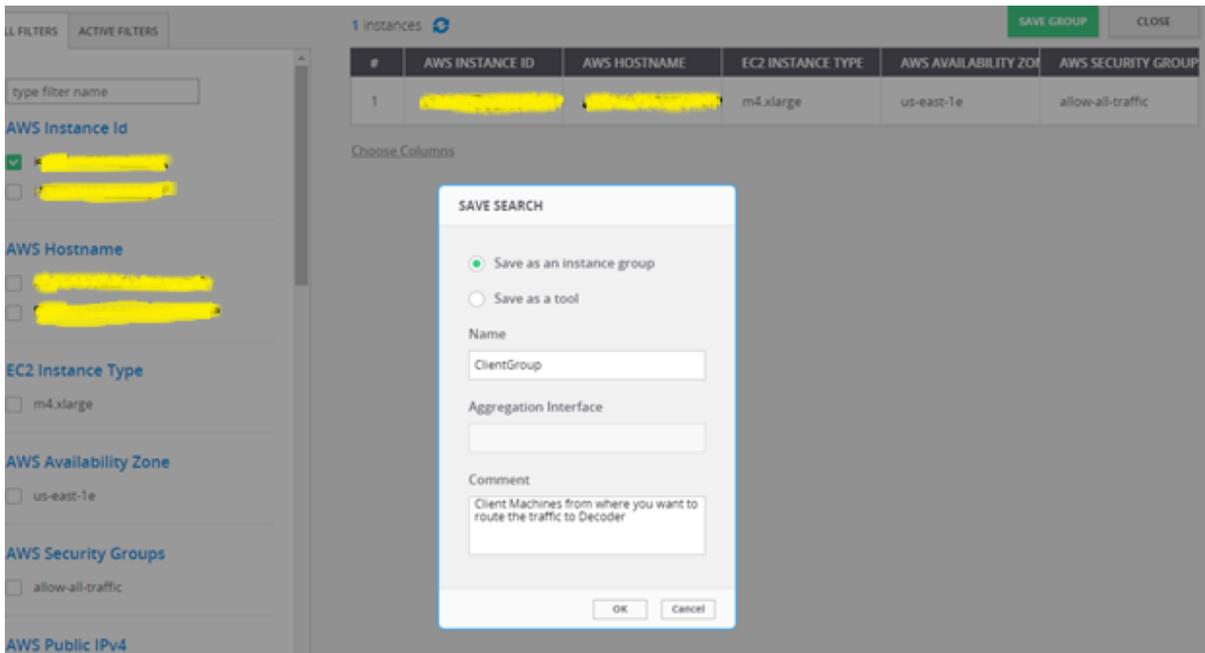
1. From the CloudLens site, open your project by double-clicking it.
  - a. Click the **Define Group** button or the Instances count.

You should see two instances listed, one for your decoder and the other for the client machines.
  - b. Filter for the decoder instance and click **Save Search**.
  - c. Choose **Save as a tool**.

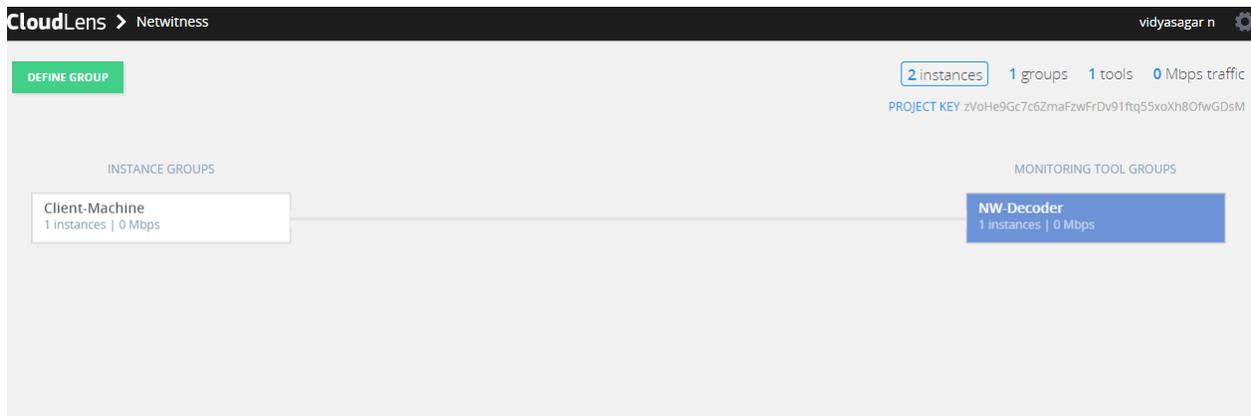
You will be asked to specify a name for the tool, and also the **Aggregation Interface**.
  - d. Give the Aggregation Interface a meaningful name, for example **CloudTAP**. This is a virtual interface that will appear in the OS where your Tool is installed. You will later need to instruct your tool to ‘listen’ to that interface.



- e. Filter the client host instance from the list, and click **Save Search**.



2. Navigate back to the top-level view of the project.  
You will see your client machine and Decoder instances.
3. Drag a connection between them to allow the flow of packets.



## Validate CloudLens Packets Arriving at Decoder

Once you have the mapping between your clients and the NetWitness Decoder, the next step is to validate that packets are actually arriving at the Decoder.

1. Open an SSH connection the public IP address of the Decoder.
2. Run the `ifconfig` command.

You will see the new aggregation interface which you created.

```
[root@ip-172-24-104-100 ~]# ifconfig
cloudlens0 Link encap:Ethernet HWaddr 82:07:07:04:00:00
inet6 addr: fe80::207:707:fe04:0000/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:9100 Metric:1
RX packets:6 errors:0 dropped:0 overruns:0 frame:0
TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:468 (468.0 b) TX bytes:468 (468.0 b)
```

3. Generate traffic from the client OS instance CLI. For example, run `wget http://www.google.com/`.

```
[root@ip-172-24-104-100 ~]# wget https://172.24.104.100 --no-check-certificate
--2017-06-19 14:33:05-- https://172.24.104.100/
Connecting to 172.24.104.100:443... connected.
WARNING: cannot verify 172.24.104.100's certificate, issued by 欵稩N=Puppet CA: cc4bfb66-8746-4b2f-88ee-3f82862c7069欵?
Unable to locally verify the issuer's authority.
WARNING: certificate common name 欵稩c4bfb66-8746-4b2f-88ee-3f82862c7069欵? doesn't match requested host name 欵? 72.24.214欵?
HTTP request sent, awaiting response... 302 Found
location: https://172.24.104.100/login [following]
--2017-06-19 14:33:05-- https://172.24.104.100/login
Reusing existing connection to 172.24.104.100:443.
HTTP request sent, awaiting response... 200 OK
length: unspecified
Saving to: 欵稩index.html.7欵?

index.html.7 [ <=> ] 2.01K --.-KB/s in 0s

2017-06-19 14:33:05 (246 MB/s) - 欵稩index.html.7欵? saved [2062]
```

4. Open an SSH connection the public IP address of the Decoder to go to your Decoder instance CLI, and look for suitable results in the `tcpdump` by running the following command:

```
tcpdump -i Cloudlens0
```

```
74 packets dropped by kernel
root@ip-172-24-104-100 ~]# tcpdump -i cloudlens0
tcpdump: WARNING: cloudlens0: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on cloudlens0, link-type EN10MB (Ethernet), capture size 65535 bytes

4:37:11.408308 IP 175.2.141.156 > ip-172-24-104-100.ec2.internal: ICMP echo request, id 132, seq 32849, length 8
4:37:11.408318 IP ip-172-24-104-100.ec2.internal > 175.2.141.156: ICMP echo reply, id 132, seq 32849, length 8
4:37:11.781923 IP 175.2.141.156 > ip-172-24-104-100.ec2.internal: ICMP 175.2.141.156 protocol 1 unreachable, length 36
```

## Set the Interface in the Packet Decoder

Finally, in your RSA NetWitness Suite Packet Decoder, set the interface to use for the Ixia integration.

1. SSH to Decoder and restart decoder service.

```
$ sudo restart nwdecoder
```

The user should now be set to capture the network traffic in Decoder.

2. Log onto NetWitness and in the RSA NetWitness menu, select **Administration > Services**.
3. In the Admin Services view, select a Decoder service, and select **View > Explore**.
4. Expand **decoder** and then click **config** to view the configuration settings.
5. For the **capture.selected** parameter, set it to the following:

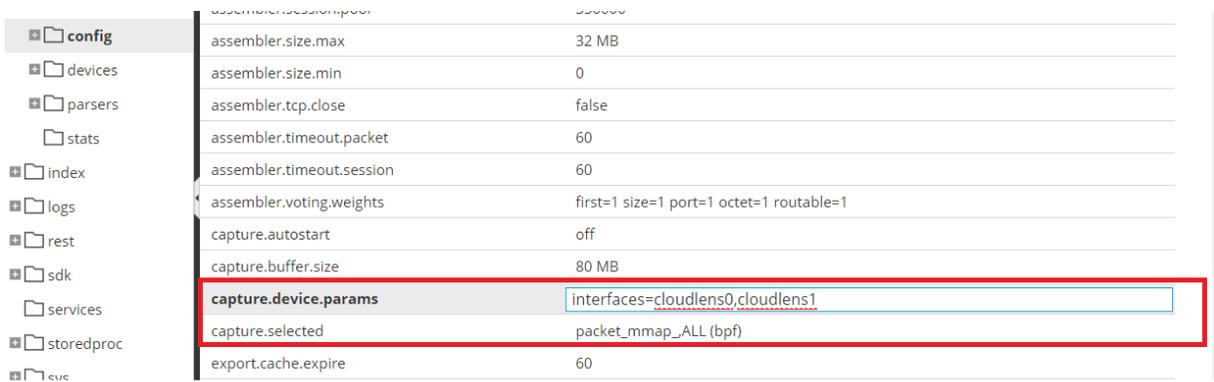
```
packet_mmap_,cloudlens0(bpf)
```

The screenshot shows the configuration page for the decoder service. The left sidebar contains a tree view with folders for 'connections', 'database', 'decoder', 'config', 'recovery', 'rules', 'devices', and 'parsers'. The 'config' folder is expanded, and the 'capture.selected' parameter is highlighted with a red box. The value for this parameter is 'packet\_mmap\_,cloudlens0 (bpf)'. A tooltip for this parameter reads: 'Current capture device and interface. Change takes effect immediately.' Other parameters visible include 'capture.buffer.size' (32 MB), 'capture.autostart' (off), and 'export.packet.enabled'.

assembler.timeout.packet	ou
assembler.timeout.session	60
assembler.voting.weights	first=1 size=1 port=1 octet=1 routable=1
capture.autostart	off
capture.buffer.size	32 MB
capture.device.params	
capture.selected	packet_mmap_,cloudlens0 (bpf)
export.cache.expire	
export.packet.enabled	

6. (Conditional) - If you have multiple capture interfaces on the Packet Decoder, set your parameters as follows:
  - a. Set the **capture.device.params** and the **capture.selected** parameters. For example, if you have two interfaces, you could set the parameters as follows:
 

```
capture.device.params --> interfaces=cloudlens0,cloudlens1
capture.selected --> packet_mmap_,All
```



assembler.session.timeout	300000
assembler.size.max	32 MB
assembler.size.min	0
assembler.tcp.close	false
assembler.timeout.packet	60
assembler.timeout.session	60
assembler.voting.weights	first=1 size=1 port=1 octet=1 routable=1
capture.autostart	off
capture.buffer.size	80 MB
<b>capture.device.params</b>	<input type="text" value="interfaces=cloudlens0,cloudlens1"/>
capture.selected	packet_mmap_ALL (bpf)
export.cache.expire	60

- b. Restart Decoder service after you set the **capture.selected** parameter.

This completes the RSA NetWitness integration with Ixia CloudLens.