Quali



IxNetwork Controller Shell

Release date: January 2019

Shell version 1.6.1

Document version A

***Legal notice***

*Information in this document is subject to change without notice. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Quali Ltd. Quali may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except if expressly provided in any written license agreement from Quali, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property. Quali, CloudShell, CloudShell Authoring, CloudShell Resource Manager, CloudShell Remote Runner, CloudShell Runtime, CloudShell Monitor, CloudShell Spy, CloudShell Portal, the Quali logo, the CloudShell logo, and the CloudShell application logos, and all other Quali product names and logos are trademarks or registered trademarks of Quali Ltd. The absence of a trademark from this list does not constitute a waiver of Quali intellectual property rights concerning that trademark.*

*All other trademarks, brand and product names are property of their respective holders.*

*© 2016 Quali Ltd. All rights reserved.*

Contents

[Overview 3](#_Toc536343574)

[About IxNetwork Controller Shell 3](#_Toc536343575)

[Standard version 3](#_Toc536343576)

[Supported OS 3](#_Toc536343577)

[Supported IxNetwork 3](#_Toc536343578)

[Supported CloudShell 3](#_Toc536343579)

[Requirements 3](#_Toc536343580)

[Downloading the Shell 3](#_Toc536343581)

[Import and Configure the Shell 4](#_Toc536343582)

[Importing the Shell into CloudShell 4](#_Toc536343583)

[Offline installation of a Shell 4](#_Toc536343584)

[Configuring a new service 5](#_Toc536343585)

[Updating offline Python dependencies 6](#_Toc536343586)

[Updating online Python dependencies 6](#_Toc536343587)

[Typical workflow 7](#_Toc536343588)

[Scenario 1 – Use a controller to run IxNetwork traffic 7](#_Toc536343589)

[References 12](#_Toc536343590)

[Release notes 13](#_Toc536343591)

# Overview

A Shell implements integration of a device model, application or other technology with CloudShell. A shell consists of a data model that defines how the device and its properties are modeled in CloudShell, along with automation that enables interaction with the device via CloudShell.

## About IxNetwork Controller Shell

This Shell provides you with connectivity and management capabilities such as loading configuration, running traffic and getting results for IxNetwork application.

## Standard version

The IxNetwork Controller Shell 1.5.0 is based on the Traffic Shell standard version 3.0.0. This shell works with Ixia Chassis Shell 2G resources.

For detailed information about the Shell’s structure and attributes, see the Traffic Shell standard on [cloudshell-standards repository](https://github.com/QualiSystems/shell-traffic-standard) in GitHub.

## Supported OS

* Windows

## Supported IxNetwork

* IxNetwork API Server – 8.0.1 GA and up
* IxNetwork Connection Manager – 8.40-EA and up

## Supported CloudShell

* CloudShell version 8.1 and above

## Requirements

* NA

## Downloading the Shell

The IxNetwork Controller Shell is available from the [Quali Developer Center](http://community.quali.com/spaces/12/index.html?__hstc=46213176.aaafbe5adb338215377a985e0c025079.1467146361756.1471392182746.1471395614692.11&__hssc=46213176.1.1471395614692&__hsfp=2437115919)[.](https://support.qualisystems.com/entries/87063688-Solution-Pack-Download-Center) Download the files into a temporary location on your local machine.

The Shell comprises:

|  |  |
| --- | --- |
| ixia\_ixnetwork\_controller.zip | The Shell Package. |
| ixia\_ixnetwork\_controller\_offline\_requirements.zip | Shell Python dependencies (**for offline installation only**) |
| IxNetwork Controller Shell Doc.pdf | Documentation |

# Import and Configure the Shell

This section describes how to import, configure and modify the IxNetwork Controller Shell.

## Importing the Shell into CloudShell

Use the following procedure to import the downloaded Shell:

**To import the Shell into CloudShell:**

1. Download the Shell from the [Quali Developer Center](http://community.quali.com/spaces/12/index.html?__hstc=46213176.aaafbe5adb338215377a985e0c025079.1467146361756.1471392182746.1471395614692.11&__hssc=46213176.1.1471395614692&__hsfp=2437115919)[.](https://support.qualisystems.com/entries/87063688-Solution-Pack-Download-Center)
2. Back up your database.
3. Log in to **CloudShell Portal** as administrator and access the relevant domain.
4. From the **User** menu, select **Import Package.**



5. Browse to the location of the downloaded Shell file, select the relevant .zip file and click **Open**. Alternatively, drag the shell’s .zip file into CloudShell Portal.

## Offline installation of a Shell

**Note:** Offline installation instructions apply only if Cloudshell Execution Server has no access to PyPi. You can skip this section if your execution server has access to PyPi. *For additional information, see the online help topic on offline dependencies.*

The Shell uses a variety of Python packages.

**To work in offline mode:**

1. Download the ixia\_chassis\_shell\_offline\_requirments.zip file (see *Downloading the Shell*).
2. Unzip it to a local repository. Make sure the Execution Server has access to this folder.
3. On the Execution Server machine, in the customer.configfile, add the following key:

|  |  |
| --- | --- |
| <add key="PythonOfflineRepositoryPath" value="repository | |
| full path"/> |  |

Make sure to update the value with the path to the repository containing the unzipped file.

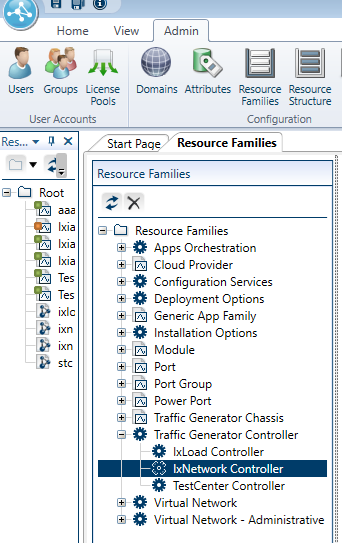
1. Restart the Execution Server.

## Configuring a new service

Perform this procedure to configure the service after importing the Shell.

**To configure the service:**

1. Go to the **Admin** tab and select the **Resource Families**.
2. Under **Traffic Generators Controllers**, select **IxNetwork Controller**.



1. Define default values for the **IxNetwork Controller** service.

|  |  |
| --- | --- |
| Name | Description |
| Controller Address | The IP address of the API Server / Connection Manager. Default is localhost. |
| Controller TCP Port | The TCP port of the API Server / Connection Manager. Default is 11009. |
| Use | IGNORE – for future use. |
| Password | IGNORE – for future use. |

Updating Python Dependencies for Shells

This section guides through on how to update your Python dependencies folder. This is required when you upgrade a Shell, driver that has new/updated dependencies. It applies to both online and offline dependencies.

## Updating offline Python dependencies

**To update offline Python dependencies:**

1. Download the latest Python dependencies package zip file locally.
2. Extract the zip file to the suitable offline package folder(s).
3. Restart any Execution Server that has a live instance of the relevant driver or script. This requires running the TestShell Execution Server's configuration wizard, as explained in the Configure the TestShell Execution Server topic of the CloudShell Suite Installation Guide - see the [CloudShell Docs & Training](http://www.quali.com/community/training/) page.

## Updating online Python dependencies

In online mode, the Execution Server automatically downloads and extracts the appropriate dependencies file to the online Python dependencies repository every time a new instance of the driver or script is created.

**To update online Python dependencies:**

* If there is a live instance of the Shell's driver or script, restart the execution server, as explained above. If an instance does not exist, the execution server will download the Python dependencies the next time a command of the driver or script runs.

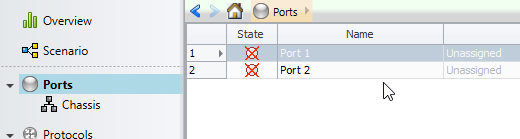
# Typical workflow

## Scenario 1 – Use a controller to run IxNetwork traffic

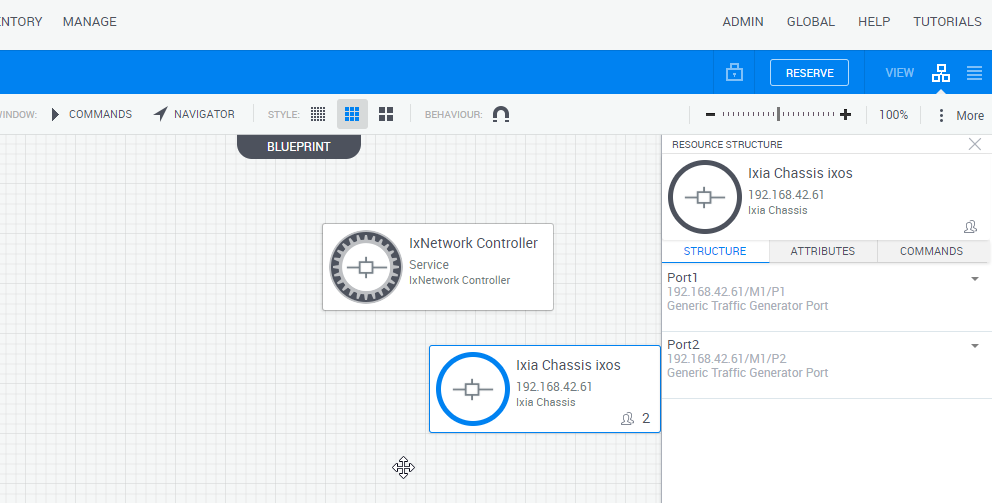
* Create Blueprint

Create a Blueprint with IxNetwork controller service and Ixia chassis resource ports. Number of Ixia ports in the blueprint should match the number of ports in the IxNetwork configuration.

For example, if we have configuration with two ports:



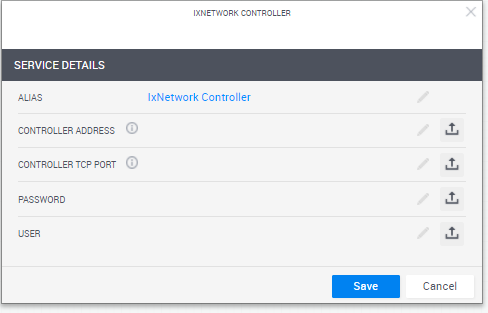
We create Blueprint with two IxNetwork ports



* Reserve Sandbox

Create a Sandbox from the Blueprint.

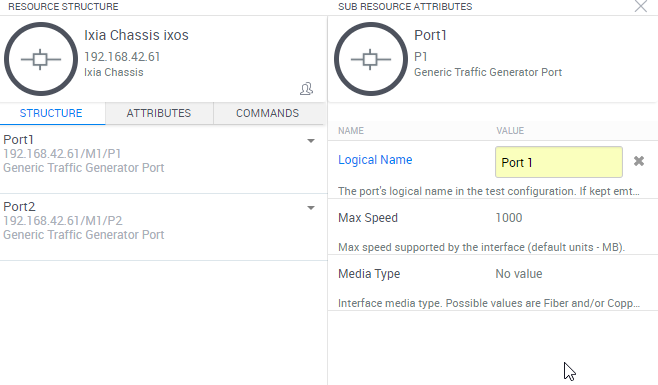
Edit IxNetwork Controller Service parameters if required.



See ‘Configuring a new service’ above.

* Map configuration ports to Sandbox ports

For each port in the IxNetwork configuration assign physical port from the ports in the sandbox. Open the attributes tab and set the Logical Name to the port name in the configuration.



* Controller Commands

|  |  |  |  |
| --- | --- | --- | --- |
| Command | Description | Parameters | |
|  |  | Parameter | Description |
| Load Configuration | Load configuration and reserve ports | Ixia config file name | Full path to Ixia configuration file name |
| Start ARP/ND | Send ARP/ND for all protocols |  |  |
| Start Protocols | Start all protocols |  |  |
| Stop Protocols | Stop all protocols |  |  |
| Start Traffic | Start L2-3 Traffic | Blocking | True - return after traffic finish to run, False - return immediately |
| Stop Traffic | Stop L2-3 Traffic |  |  |
| Get Statistics | Get view statistics | View Name | Port Statistics, Traffic Item Statistics, Flow Statistics, etc. |
| Output Type | CSV or JSON. If CSV. The statistics will be attached to the reservation csv file. |
| Run Quick Test | Run quick test | Quick Test Name | Name of quick test to run |

* Hidden developer commands

|  |  |  |  |
| --- | --- | --- | --- |
| Command | Description | Parameters | |
|  |  | Parameter | Description |
| get\_session\_id | Returns the REST session. This ID can be used to run any IxNetwork REST command directly. |  |  |
| get\_children | Returns list of all children of a specific type of the requested object.  The root object reference can be retrieved from the session ID. | obj\_ref | Requested object reference. |
| child\_type | Requested child type |
| get\_attributes | Returns dictionary of all <attribute: value> of the requested object. | obj\_ref | Requested object reference. |
| set\_attribute | Sets value of specific attribute of the requested object. | obj\_ref | Requested object reference. |
| attr\_name | Requested attribute name |
| attr\_value | Value to set |

* Following is a simple code snippet demonstrating the hidden commands:

The code bellow assumes that reservation\_id holds the reservation ID and session holds CS session.

# Get session ID

session\_id = session.ExecuteCommand(reservation\_id, *'IxNetwork Controller'*, *'Service'*, *'get\_session\_id'*)

# Get root object reference

root\_obj = '{}ixnetwork'.format(session\_id.Output[1:-1]

# Get all children of type globals of the root object.

globals = session.ExecuteCommand(reservation\_id, 'IxNetwork Controller', 'Service', 'get\_children' [InputNameValue('obj\_ref', root\_obj), InputNameValue('child\_type', 'globals')])

# There is only one globals object so we take it.

globals\_obj = json.loads(globals.Output)[0]

# Get all children of type preferences of the globals object.

prefs = session.ExecuteCommand(reservation\_id,

*'IxNetwork Controller'*, *'Service'*, *'get\_children'*, [InputNameValue(*'obj\_ref'*, globals\_obj),

InputNameValue(*'child\_type'*, *'preferences'*)])

# Again, there is only one preferences object so we take it.

prefs\_obj = json.loads(prefs.Output)[0]

# Get attributes of preferences object.

prefs\_attrs = session.ExecuteCommand(reservation\_id, *'IxNetwork Controller'*, *'Service'*, *'get\_attributes'*,

[InputNameValue(*'obj\_ref'*, prefs\_obj)])

# Set *connectPortsOnLoadConfig* == True

session.ExecuteCommand(reservation\_id, *'IxNetwork Controller'*, *'Service'*, *'set\_attribute'*, [InputNameValue(*'obj\_ref'*, prefs\_obj),

InputNameValue(*'attr\_name'*, *'connectPortsOnLoadConfig'*),

InputNameValue(*'attr\_value'*, *'True'*)])

# References

Additional technical documentation is available in the [Quali's Developer Center](http://community.quali.com/spaces/12/index.html?__hstc=46213176.aaafbe5adb338215377a985e0c025079.1467146361756.1471392182746.1471395614692.11&__hssc=46213176.1.1471395614692&__hsfp=2437115919)[.](https://support.qualisystems.com/entries/22858046-download-center)

For Quali discussion forums, click [here](http://community.quali.com/spaces/13/index.html?__hstc=46213176.aaafbe5adb338215377a985e0c025079.1467146361756.1471392182746.1471395614692.11&__hssc=46213176.1.1471395614692&__hsfp=2437115919)[.](https://support.qualisystems.com/)

# Release notes

**What’s new:**

* Support for IxNetwork version 8.5 and up.

**Known issue:**

* Performance
  + The REST API performance is very poor. Loading configuration and reserving ports can take up dozens of seconds depending on the specific setup. It is advised to start idle connections on the Connection Manager to reduce startup time.
* No available connection on connection manager
  + In case there is no available connection on the connection manager the user must login to the Connection Manager and close zombie connections or create new connections.
* Licensing
  + If license server on Connection Manager / API server is not configured Load Configuration might success but ports will be in Down state and any further operation will fail.
* Reserved ports
  + If ports are reserved by other users Load Configuration might success but ports will be in Down state and any further operation will fail.