AMWA NMOS Automated Testing
Andrew Bonney – BBC R&D
Gareth Sylvester-Bradley – Sony Europe

Introduction
• What is NMOS?
• Why do we need a testing tool?
• What does it test and how?
• How do you use it?
• What’s next?
What is NMOS?

• A family of specifications with the collective aim to make the use of IP production systems as easy, or easier than, their SDI counterparts
• Including key foundations of:
  – Discovery and registration
  – Connection management
• Agnostic to:
  – Data type, media transport, resolutions, rates and codecs
• Built upon proven Internet technologies
  – HTTP, DNS, TLS, OAuth

Why do we need a testing tool?

• NMOS specifications originally used paper test plans at workshops
• Many detailed tests were too time consuming to perform
• When something doesn’t work in the wild, how do you know which implementation is at fault?

• It is critical that both vendors and end users can confirm adherence to specifications outside of scheduled industry testing events
Is it for me?

• Yes! It’s for vendors, system integrators and end users
• Use it:
  – During software development
  – During QA ahead of releases
  – Ahead of multi-vendor trials or customer ‘PoC’
  – Before you purchase new equipment
  – When you upgrade existing equipment
• If you find a fault it didn’t identify, let us know!

“The reason for this prerequisite is the strong intention from the industry to move to PICS-driven self-certification model”

– Ievgen KOSTIUKEVYCH, Senior IP Media Technology Architect, EBU
How did it come about?

• Collaboration between end users and vendors
  – Various original tools were written by BBC, Riedel and Streampunk
  – A new effort began with BBC R&D in late 2018, aiming to integrate existing tools where possible
  – Sony and other AMWA members have contributed heavily since then

• Test coverage of the core NMOS Specifications is now high
  – Architecture enables straightforward testing of future specifications which use the same design patterns

What does it test?

- NMOS Nodes
- NMOS Registries
- Secure Comms
- Natural Grouping
- Audio Channel Mapping
- Event & Tally
- NMOS Clients e.g. Broadcast Controller
- NMOS System
- Network Controller
- Authorization Server
What does it test?

• Schema conformance
• API structure
• API behaviour
• API discovery
• Common faults and inconsistencies

• As well as REQUIRED items, RECOMMENDED and OPTIONAL items are covered, with explanations of the circumstances in which they are important.
How does it work?

- Downloads each NMOS specification using Git
- Parses the RAML API definition and JSON schemas in order to construct basic tests automatically
- Merges automatically constructed tests with manually defined ones
- Launches a web interface to run tests from

How do I use it?

- Open source
- Windows or Linux
- Python 3 (including the pip package manager)
- Git

```
$ pip3 install -r requirements.txt
$ python3 nmos-test.py
```
NMOS Test

This test suite is under active development and does not yet provide 100% coverage of specifications. We recommend regularly re-testing implementations as new tests are developed.
NMOS Test

Result for test suite IS-04 Registry APIs on: http://172.29.80.65:80/x-nmos/registration/v1.2/, http://172.29.80.65:80/x-nmos/query/v1.2/

<table>
<thead>
<tr>
<th>Test</th>
<th>Pass</th>
<th>Description</th>
<th>Reason</th>
<th>Completion Time</th>
<th>Time Elapsed</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>init</em></td>
<td>Not Applicable</td>
<td>Test initialization</td>
<td></td>
<td>16:19:13.352</td>
<td>1.308s</td>
</tr>
<tr>
<td>run_tests</td>
<td>Not Applicable</td>
<td>Test setup</td>
<td></td>
<td>16:19:13.363</td>
<td>0.003s</td>
</tr>
<tr>
<td>auto_query_1</td>
<td>Pass</td>
<td>GET /x-nmos</td>
<td></td>
<td>16:19:13.369</td>
<td>0.006s</td>
</tr>
<tr>
<td>auto_query_2</td>
<td>Pass</td>
<td>GET /x-nmos/query</td>
<td></td>
<td>16:19:13.375</td>
<td>0.006s</td>
</tr>
<tr>
<td>auto_query_3</td>
<td>Pass</td>
<td>GET /x-nmos/query/v1.2</td>
<td></td>
<td>16:19:13.382</td>
<td>0.006s</td>
</tr>
<tr>
<td>auto_query_4</td>
<td>Pass</td>
<td>GET /x-nmos/query/v1.2/devices</td>
<td></td>
<td>16:19:13.395</td>
<td>0.011s</td>
</tr>
<tr>
<td>auto_query_5</td>
<td>Could Not Test</td>
<td>GET /x-nmos/query/v1.2/devices/01dev01</td>
<td>No resource found to perform this test</td>
<td>16:19:13.395</td>
<td>0.000s</td>
</tr>
<tr>
<td>auto_query_6</td>
<td>Pass</td>
<td>GET /x-nmos/query/v1.2/flows</td>
<td></td>
<td>16:19:14.415</td>
<td>0.041s</td>
</tr>
<tr>
<td>auto_query_7</td>
<td>Could Not Test</td>
<td>GET /x-nmos/query/v1.2/flows/001dev01</td>
<td>No resource found to perform this test</td>
<td>16:19:14.415</td>
<td>0.000s</td>
</tr>
</tbody>
</table>

This test suite is under active development and does not yet provide 100% coverage of specifications. We recommend regularly re-testing implementations as new tests are developed.
Example: IS-04 Nodes

• One of the more challenging areas to test
• Mock DNS Server
  – Integrates a DNS server which is populated with records to thoroughly test IS-04 discovery
• Mock Registries
  – Launches multiple IS-04 Registration APIs to test how Nodes interact with them
External Integrations

- **SDPoker**
  - Calls out to SDPoker with pre-defined command line arguments to test every SDP file a device exposes

- **TestSSL**
  - Calls out to TestSSL with pre-defined command line arguments to ensure TLS support is enabled, uses the correct versions and ciphers, and otherwise conforms with AMWA BCP-003-01

- **OpenSSL**
  - Includes a certificate authority and scripting to generate TLS certificates for testing purposes

Advanced Usage

- **Continuous Integration**

  ```
  $ python3 nmos-test.py suite IS-04-01 --host 128.66.12.5 --port 80 --version v1.2 --output results.xml
  ```
Continuous Integration: Case Study

• Sony nmos-cpp
  – Open-Source Software
    • [https://github.com/sony/nmos-cpp](https://github.com/sony/nmos-cpp)
  – NMOS Registry & Virtual Node
    – IS-04, IS-05, IS-07, IS-09, BCP-003-01, ...
• Travis CI integration
  – AMWA NMOS Testing Tool
• Docker container for interop testing
  – Published by Richard Hastie, Mellanox
    • [https://hub.docker.com/r/rhastie/nmos-cpp](https://hub.docker.com/r/rhastie/nmos-cpp)

JT-NM Tested

• The testing tool was key in the TR-1001-1 test plan
• It enabled us to perform over 100 tests per device without manual intervention
• It identified a number of additional areas which would benefit from automated testing
• A release is available on GitHub matching the code used for this event
What’s next?

• Testing of clients
• Increased coverage of specifications
  – Such as IS-07, and IS-09/BCP-003-02 for Nodes
• Potential integrations with other tools such as EBU LIST
More information

- Downloads and documentation:
  - https://github.com/AMWA-TV/nmos-testing
  - https://amwa-tv.github.io/nmos-testing/

- If you have a question or experience a problem:
  - https://github.com/AMWA-TV/nmos-testing/issues

- We recommend the testing tool as the first port of call if you experience an issue with an implementation

Thank you
Andrew.Bonney@bbc.co.uk
Gareth.Sylvester-Bradley@sony.com

Thank you to our Media Partners