U5040BSCB/U5040MULB/ U5040MDLB REST API Programming

Reference Guide

Notices

Edition

Edition 1.2, May 2022

Keysight Technologies, 1900 Garden of the Gods Rd, Colorado Springs, CO 80907, United States

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

U.S. Government Rights

The Software is "commercial computer software," as defined by Federal Acquisition Regulation ("FAR") 2.101. Pursuant to FAR 12.212 and 27.405-3 and Department of Defense FAR Supplement

("DFARS") 227.7202, the U.S. government acquires commercial computer software under the same terms by which the software is customarily provided to the public. Accordingly, Keysight provides the Software to U.S. government customers under its standard commercial license, which is embodied in its End User License Agreement (EULA), a copy of which can be found at: http://www.keysight.com/find/sweula. The license set forth in the EULA represents the exclusive authority by which the U.S. government may use, modify, distribute, or disclose the Software. The EULA and the license set forth therein, does not require or permit, among other things, that Keysight: (1) Furnish technical information related to commercial computer software or commercial computer software documentation that is not customarily provided to the public; or (2) Relinquish to, or otherwise provide, the government rights in excess of these rights customarily provided to the public to use, modify, reproduce, release, perform, display, or disclose commercial computer software or commercial computer software documentation. No additional government requirements beyond those set forth in the EULA shall apply, except to the extent that those terms, rights, or licenses are explicitly required from all providers of commercial computer software pursuant to the FAR and the DFARS and are set forth specifically in writing elsewhere in the EULA. Keysight shall be under no obligation to update, revise or otherwise modify the Software. With respect to any technical data as defined by FAR 2.101, pursuant to FAR 12.211 and 27.404.2 and DFARS 227.7102, the U.S. government acquires no greater than Limited Rights as defined in FAR 27.401 or DFAR 227.7103-5 (c), as applicable in any technical data.

Warranty

THE MATERIAL CONTAINED IN THIS DOCUMENT IS PROVIDED "AS IS," AND IS SUBJECT TO BEING CHANGED, WITHOUT NOTICE, IN FUTURE EDITIONS. FURTHER, TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, KEYSIGHT DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED WITH REGARD TO THIS MANUAL AND ANY INFORMATION CONTAINED HEREIN, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. KEYSIGHT SHALL NOT BE LIABLE FOR ERRORS OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, USE, OR PERFORMANCE OF THIS DOCUMENT OR ANY INFORMATION CONTAINED HEREIN. SHOULD KEYSIGHT AND THE USER HAVE A SEPARATE WRITTEN AGREEMENT WITH WARRANTY TERMS COVERING THE MATERIAL IN THIS DOCUMENT THAT CONFLICT WITH THESE TERMS, THE WARRANTY TERMS IN THE SEPARATE AGREEMENT WILL CONTROL.

Safety Notices

CAUTION

A CAUTION notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a CAUTION notice until the indicated conditions are fully understood and met.

WARNING

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

REST API Programming Reference Guide – At a glance

This help file contains the programming reference information for the Keysight U5040B Open RAN Studio REST API and the associated functions in the .NET Client Assembly.

More information

For the latest product information and updated U5040B Open RAN Studio software, refer to http://www.keysight.com/find/S5040A.

For more information, contact Keysight Technologies.

Product information and descriptions in this help file are subject to change without notice.

To understand the product functionality, refer to:

- Keysight S5040A Open RAN Studio Player and Capture Appliance User’s Guide
- Keysight U5040B Open RAN Studio Software Startup Guide
- Keysight U5040B Open RAN Studio Software User Guide

Contents

1 Getting started with REST API

Overview

NOTE **The U5040MULB (Uplink test) / U5040MDLB (Downlink test) licenses enable programming using the REST APIs. However, installing the U5040BSCB license is mandatory to launch the U5040B Open RAN Studio software GUI.**

The Open RAN Studio provides two different API interfaces, which are described below. This document describes the Open RAN Studio REST API only.

1. Open RAN Studio API - This API is used for the following functions:
   - Stimulus Generation - creating PCAP files from Signal Studio Waveform definitions (.scp files)
   - IQ Extraction - Extracting IQ information from PCAP files
2. Open RAN Studio REST API - This API is used for the following functions:
   - Managing and configuring hardware modules
   - Stimulus Player Control – Controls playing of the Stimulus (Ethernet O-RAN packets) from Open RAN Studio to the O-RU being tested
   - Capture Recorder Control – Records traffic between Open RAN Studio and the O-RU being tested
   - Player and recorder control

All API functions are REST-based and can be run remotely. Also, a .NET Client API has been provided.

The Open RAN Studio API functions have been covered in detail in the Keysight U5040B Open RAN Studio O-RAN Radio Unit (O-RU) Testing and Validation Online Help. This document focuses on the Player/Recorder REST API.

Features & Limitations

Features

Open RAN Studio provides a “Player/Recorder REST API”, which allows automated tests to access the following features:

- Automatic connection to the O-RAN Unit testing and Validation Hardware, when the service starts.
- Configure the Hardware (same as *Instrument Configuration* in the Open RAN Studio GUI)
  - Port Settings, Time Sync, Stimulus, Triggering and other settings
  - Configured via XML in the same format as that in the file *<user-directory>/Documents/Keysight/Open RAN Studio/InstrumentConfig.xml*
- Stimulus Player
  - Upload Stimulus into the Player on the Hardware.
  - Play on Open RAN Studio generated Stimulus file with a Single repetition or continuous radio frames
- Capture Recorder
  - Capture a fixed number of radio frames or until capture memory is filled (‘Until Full’)
  - Download Captured packets from the Recorder in the Hardware into a PCAP file
  - Upload/download files (related to Stimulus and Capture)

Limitations

The following features can be accessed using the Open RAN Studio API only (that is, they cannot be accessed using the REST API):

- Stimulus creation (that is, CU Plane Builder)
- IQ Extraction
- Explorer

API - Architecture

The diagram below shows the architecture for the REST API. This API runs as a Windows service on the O-RAN testing & validation hardware. A .NET assembly provides the client side interface. You may also use other REST clients, such as CURL, REST Libraries, and so on.

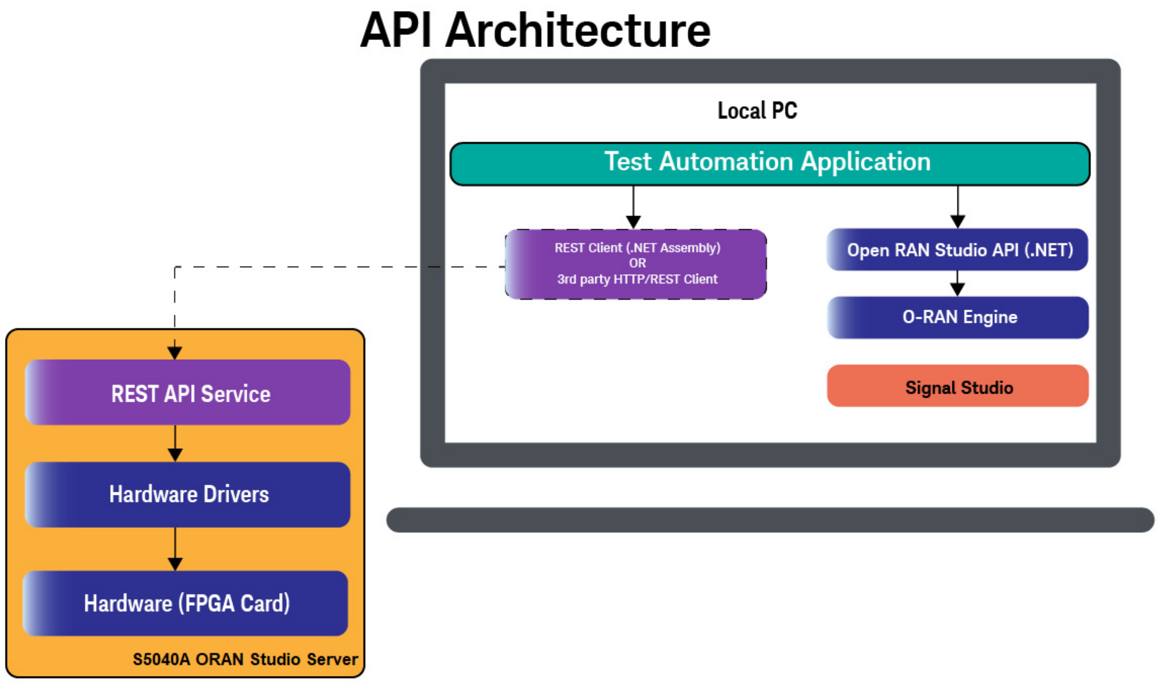

API Architecture
Local PC
Test Automation Application
REST Client (.NET Assembly)
OR
3rd party HTTP/REST Client
Open RAN Studio API (.NET)
O-RAN Engine
Signal Studio
REST API Service
Hardware Drivers
Hardware (FPGA Card)
S5040A ORAN Studio Server

Figure 1 Block diagram showing architecture of the REST API

Requirements

Licensing for Open RAN Studio REST API

- U5040BSCB - base software license (GUI / REST API)
- U5040MDLB - Manufacturing Downlink testing license (REST API only)
- U5040MULB - Manufacturing Uplink testing license (REST API only)

NOTE When you launch the U5040B Open RAN Studio software after installing it on the S5040A Appliance, the GUI shall prompt you for firmware updates, whenever required. Beginning with version 2.1, the firmware update can be performed, even without the U5040BSCB license installed. Following the firmware update, the S5040A system must be shutdown (to a power off state). After powering back the system on, you cannot launch the GUI until you install the U5040BSCB license.

Installing the Open RAN Studio REST API

The Open RAN Studio Remote Player REST API runs as a Windows service.

- The service is automatically installed with a complete installation of the U5040B Open RAN Studio software on the O-RAN Hardware service.
  - For the latest version of the software, navigate to the U5040B Open RAN Studio software download page.
- A client side interface (C# .NET assembly) is provided.

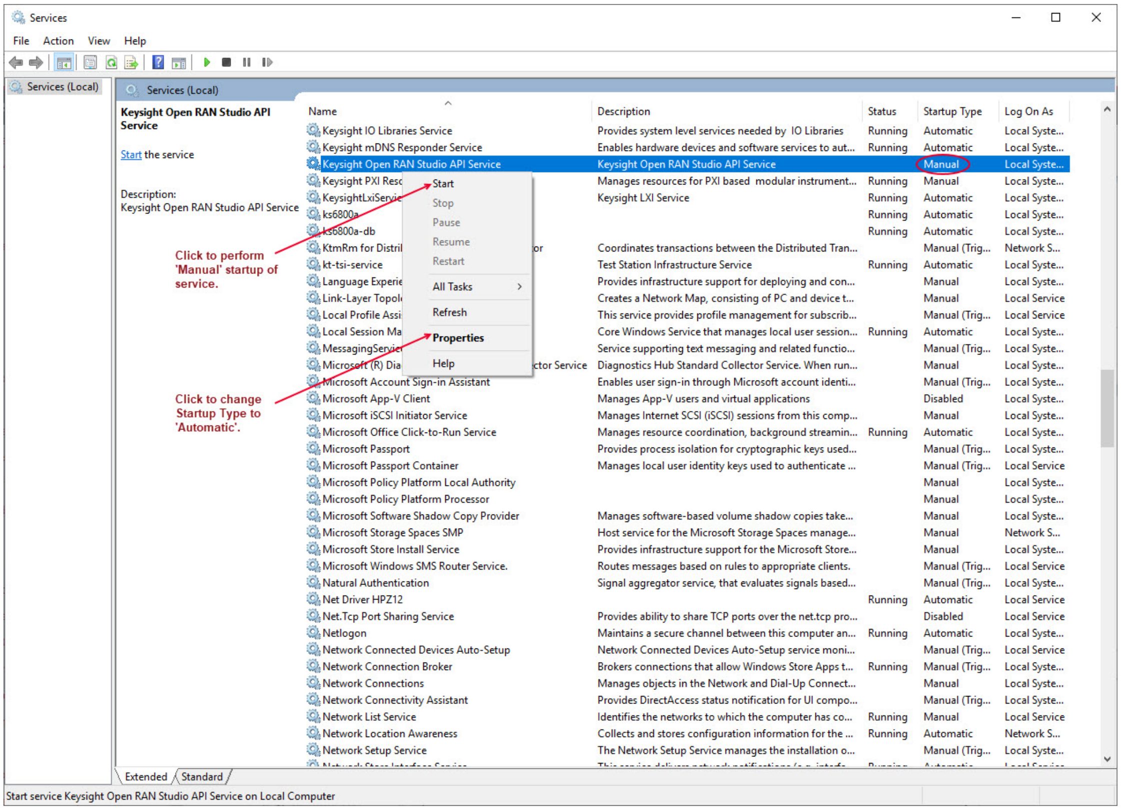
Note that REST API uses TCP Port 9000. Any firewall that is configured on your machine must be configured to allow incoming connections to TCP port 9000 for this service.

Starting the Open RAN Studio API service

NOTE When the Open RAN Studio REST API service is running, it takes control of the FPGA hardware. The U5040B Open RAN Studio Software cannot run at the same time. If the software is launched while the Open RAN Studio REST API service is running, the software will display an error message and exit.

The Windows service for the Open RAN Studio Remote Player REST API should be configured to start automatically.

1 From the **Start** menu, go to **Services**.



Keysight Open RAN Studio API Service

Start the service

Description:
Keysight Open RAN Studio API Service

Name	Description	Status	Startup Type	Log On As
Keysight IO Libraries Service	Provides system level services needed by IO Libraries	Running	Automatic	Local Syste...
Keysight mDNS Responder Service	Enables hardware devices and software services to aut...	Running	Automatic	Local Syste...
Keysight Open RAN Studio API Service	Keysight Open RAN Studio API Service		Manual	Local Syste...
Keysight PXI Resc	Manages resources for PXI based modular instrument...	Running	Manual	Local Syste...
KeysightLxiServic	Keysight LXI Service	Running	Automatic	Local Syste...
ks6800a		Running	Automatic	Local Syste...
ks6800a-db		Running	Automatic	Local Syste...
KtmRm for Distril ... or	Coordinates transactions between the Distributed Tran...		Manual (Trig...	Network S...
kt-tsi-service	Test Station Infrastructure Service	Running	Automatic	Local Syste...
Language Experie	Provides infrastructure support for deploying and con...		Manual	Local Syste...
Link-Layer Topol	Creates a Network Map, consisting of PC and device t...		Manual	Local Service
Local Profile Assi	This service provides profile management for subscrib...		Manual (Trig...	Local Service
Local Session Ma	Core Windows Service that manages local user session...	Running	Automatic	Local Syste...
MessagingServic	Service supporting text messaging and related functio...		Manual (Trig...	Local Syste...
Microsoft (R) Dia ... ctor Service	Diagnostics Hub Standard Collector Service. When run...		Manual	Local Syste...
Microsoft Account Sign-in Assistant	Enables user sign-in through Microsoft account identi...		Manual (Trig...	Local Syste...
Microsoft App-V Client	Manages App-V users and virtual applications		Disabled	Local Syste...
Microsoft iSCSI Initiator Service	Manages Internet SCSI (iSCSI) sessions from this comp...		Manual	Local Syste...
Microsoft Office Click-to-Run Service	Manages resource coordination, background streamin...	Running	Automatic	Local Syste...
Microsoft Passport	Provides process isolation for cryptographic keys used...		Manual (Trig...	Local Syste...
Microsoft Passport Container	Manages local user identity keys used to authenticate ...		Manual (Trig...	Local Service
Microsoft Policy Platform Local Authority			Manual	Local Syste...
Microsoft Policy Platform Processor			Manual	Local Syste...
Microsoft Software Shadow Copy Provider	Manages software-based volume shadow copies take...		Manual	Local Syste...
Microsoft Storage Spaces SMP	Host service for the Microsoft Storage Spaces manage...		Manual	Network S...
Microsoft Store Install Service	Provides infrastructure support for the Microsoft Store...		Manual	Local Syste...
Microsoft Windows SMS Router Service.	Routes messages based on rules to appropriate clients.		Manual (Trig...	Local Service
Natural Authentication	Signal aggregator service, that evaluates signals based...		Manual (Trig...	Local Syste...
Net Driver HPZ12		Running	Automatic	Local Service
Net.Tcp Port Sharing Service	Provides ability to share TCP ports over the net.tcp pro...		Disabled	Local Service
Netlogon	Maintains a secure channel between this computer an...	Running	Automatic	Local Syste...
Network Connected Devices Auto-Setup	Network Connected Devices Auto-Setup service moni...		Manual (Trig...	Local Service
Network Connection Broker	Brokers connections that allow Windows Store Apps t...	Running	Manual (Trig...	Local Syste...
Network Connections	Manages objects in the Network and Dial-Up Connect...		Manual	Local Syste...
Network Connectivity Assistant	Provides DirectAccess status notification for UI compo...		Manual (Trig...	Local Syste...
Network List Service	Identifies the networks to which the computer has co...	Running	Manual	Local Service
Network Location Awareness	Collects and stores configuration information for the ...	Running	Automatic	Network S...
Network Setup Service	The Network Setup Service manages the installation o...		Manual (Trig...	Local Syste...

Start service Keysight Open RAN Studio API Service on Local Computer

Figure 2 Setting up Windows Services to enable API programming

2 Locate **Keysight Open RAN Studio API Service**.

3 By default, the *Startup Type* is set to ‘Manual’. From the right-click menu, perform one of the following actions:

   i Click *Start* to start the service manually. Note that you must perform this step each time you want to access the Open RAN Studio API service.

   ii Click *Properties* to change the Startup Type to ‘Automatic’. The Open RAN Studio API service will start automatically each time you access the desktop on your machine.

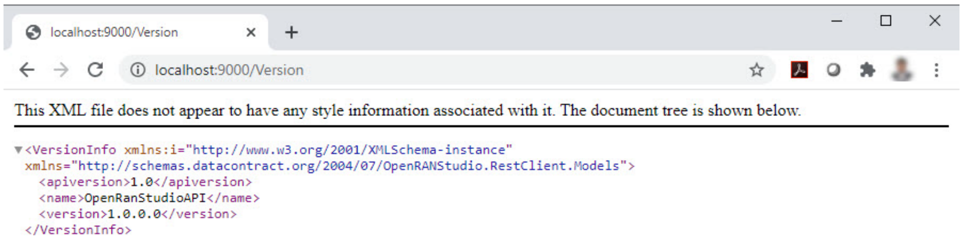
Testing the Open RAN Studio REST API service

The REST API runs as a ‘RESTful’ web service on HTTP port 9000.

In order to test whether or not the Open RAN Studio REST API service is running or not, enter the following URL on any browser on your machine.

http://localhost:9000/Version

The API version is returned in XML format.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<VersionInfo xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://schemas.datacontract.org/2004/07/OpenRANStudio.RestClient.Models">
   <apiversion>1.0</apiversion>
   <name>OpenRanStudioAPI</name>
   <version>1.0.0.0</version>
 </VersionInfo>
```

Figure 3 Validating running of REST API service

Module referencing

NOTE The REST API currently supports a single Module only on the Keysight S5040A Appliance hardware. However, module IDs are included in the API to allow multiple modules to be referenced in the future versions of the Open RAN Studio.

Generally, most APIs involving a Module will use “**Modules/<id>/<function>**”. However, when using module 0, you can omit the “**Modules/<id>/**” prefix and use a shorthand URL.

For example, the following API URLs are equivalent:

Full format	Shorthand (Module 0)
Modules/0/Configure	Configure
Modules/0/Player	Player
Modules/0/Recorder	Recorder

2 Structure and Functions of the REST API

Hierarchy of the REST API Structure

The following hierarchy displays the REST API Structure.

- Version/
  - GET - returns API version information
- Modules/
  - GET - returns list of modules
- Modules/<id>/
  - Configure
    - POST multipart/formdata - sends XML document to configure module
  - Player/
    - GET - returns status of Player
    - PUT - start / stop player
  - Player/Pcap
    - POST multipart/formdata - sends PCAP file to
  - Recorder/
    - GET - returns status of Recorder
    - PUT - start / stop player
  - Recorder/Pcap
    - GET - retrieve last captured PCAP file
- BER / BLER Computation

The API functions are described in the following sections.

Get Version Function

Get Version

HTTP GET /Version

Description

Returns version information for the web service in JSON format.

Table 1 Returned parameters

Parameter	Value	Description
name	"OpenRANStudioAPI"	Identifies this REST Web service
version	"Major.Minor.build.patch"	Corresponds to the application version that is installed
apiversion	"Major.Minor"	Current API version is "1.0" ▪ A major version change indicates the API may not be compatible with prior versions. ▪ A minor version change indicates new APIs or extensions may have been added but are backward compatible.

Table 2 HTTP Response Codes

HTTP Response Code	Message	Description
200 (OK)	–	Indicates success
500	exception details	Indicates an exception processing the request

Example in CURL script

```
curl -s -w "\nResponse: %{http_code}\n" -X GET http://localhost:9000/Version

{"name":"OpenRanStudioAPI","version":"1.0.0.0","apiversion":"1.0"}

Response: 200
```

.NET Client

GetVersionAsync()

Example in .NET REST Client

```
using OpenRANStudio.RestClient;

using OpenRANStudio.RestClient.Models;

RestClient client = new RestClient("127.0.0.1", 9000);

VersionInfo ver = await client.GetVersionAsync();
```

Module Functions

The following API functions are covered in this section:

- Get Module List
- Configure Module

Get Module List

HTTP GET /Modules

Description

Gets a list of modules in the system. Keysight recommends that this API be called to validate that an “S5040A” model exists.

NOTE **Only a single module is currently supported, with module ID 0.**

Table 3 Returned parameters

Returns a JSON array of records with the following fields:

Parameter	Value	Description
id	0 based integer	Identifies this REST Web service
model	“S5040A”	Identifies that the model is S5040A
address	(example) “PXI0::2-0.0::INSTR”	Identifies VISA address for the module

Table 4 HTTP Response Codes

HTTP Response Code	Message	Description
200 (OK)	–	Indicates success
500	exception details	Indicates an exception processing the request

Example in CURL script

```
curl -s -w "\nResponse: %{http_code}\n" -X GET http://localhost:9000/Modules
[{"id":0,"model":"S5040A","address":"PXI0::2-0.0::INSTR"}]
Response: 200
```

.NET Client

GetModulesAsync()

Example in .NET REST Client

```
using OpenRANStudio.RestClient;
using OpenRANStudio.RestClient.Models;
RestClient client = new RestClient("127.0.0.1", 9000);
List<ModuleInfo> moduleList = await client.GetModulesAsync();
```

Configure Module

Configures the Keysight S5040A Appliance hardware module using an XML configuration file. Also, this function effectively resets the driver and module. After configuration, Keysight recommends a wait duration of 30 seconds before either starting / stopping player stimulus, capturing recordings or performing both actions. If PTP synchronization is being used as the PTP Master/Slave, it will be reset and will have to be re-synchronized.

See Instrument Configuration File Format on page 36 for details on the XML format of the configuration file.

HTTP POST /Modules/<id>/Configure

Description

POST with content-type multipart/form-data, provides the configuration file for the module. This is an XML Document in the same format as that in *Documents/Keysight/Open RAN Studio/InstrumentConfig.xml*.

NOTE *InstrumentConfig.xml* files in BittWare are not compatible with the S5040A hardware because the QSFP ports on the S5040A Appliance are numbered beginning with 1 whereas those on the BittWare hardware begin with 0.

Table 5 HTTP Response Codes

HTTP Response Code	Message	Description
200 (OK)	–	Indicates success
500	exception details	Indicates an exception processing the request

Example in CURL script

```
curl -s -w "\nResponse: %{http_code}\n" -F "data=@InstrumentConfig.xml"
"http://localhost:9000/Modules/0/Configure"
```

```
Response: 200
```

.NET Client

ConfigureAsync()

Example in .NET REST Client

```
using OpenRANStudio.RestClient;

using OpenRANStudio.RestClient.Models;

RestClient client = new RestClient("127.0.0.1", 9000);

int moduleId = 0;

bool result = await client.ConfigureAsync(moduleId, "InstrumentConfig.xml");
```

Player Functions

The following API functions are covered in this section:

- Get Player State
- Start/Stop Player
- Load PCAP file into Player

Get Player State

HTTP GET /Modules/<id>/Player

Description

Returns the current state of the player.

Table 6 Returned parameters

Returns a JSON array of records with the following fields:

Parameter	Value	Description
state	▪ on ▪ off ▪ error ▪ not present	▪ Player is currently playing stimulus ▪ Player is currently stopped ▪ An error occurred when playing stimulus ▪ The player has not been loaded with stimulus
radioframes	0…n	The number of radio frames that was specified to play.

Table 7 HTTP Response Codes

HTTP Response Code	Message	Description
400 (BadRequest)	{"Message":"Module Not Found"}	The specified module ID was not found
500	exception details	Indicates an exception processing the request

Example in CURL script

```
EXEC: curl -s -w "\nResponse: %{http_code}\n" -X GET "http://localhost:9000/Modules/0/Recorder"
{"state":"on","radioframes":0}
```

.NET Client

GetPlayerStateAsync()

Example in .NET REST Client

```
using OpenRANStudio.RestClient;
using OpenRANStudio.RestClient.Models;
RestClient client = new RestClient("127.0.0.1", 9000);
int moduleId = 0;
PlayerInfo playerInfo = await client.GetPlayerStateAsync(moduleId);
// playerInfo.state contains the state: "on", "off", "error", "not present"
```

Start/Stop Player

HTTP PUT /Modules/<id>/Player

Description

Starts or stops the player. A JSON document must be included with the following parameters:

- state – “on” / “off”
- radioframes – 0 (continuous) / 1 (single radio frame)

Table 8 Returned parameters

Parameter	Value	Description
state	▪ on ▪ off ▪ error ▪ not present	▪ Player is currently playing stimulus ▪ Player is currently stopped ▪ An error occurred when playing stimulus ▪ The player has not been loaded with stimulus
radioframes	0...n	The number of radio frames that was specified to play.

Table 9 HTTP Response Codes

HTTP Response Code	Message	Description
200 (OK)	–	Indicates success
400 (BadRequest)	{"Message":"Module Not Found"}	The specified module ID was not found
400 (BadRequest)	{"Message":"No Stimulus is loaded for Play request"}	A Stimulus (.PCAP) file must be uploaded prior to issuing a request with "state":"on"
400 (BadRequest)	{"Message":"Stimulus already running"}	A request with "state":"on" was issued but the player is already running
400 (BadRequest)	{"Message":"Stimulus not running"}	A request with "state":"off" was issued but the player is not running
500	exception details	Indicates an exception processing the request

Example in CURL script

```
EXEC: curl -s -w "\nResponse: %{http_code}\n" -d '{"state":"on","radioframes":1}' -H
Content-Type:application/json -X PUT "http://localhost:9000/Modules/0/Player"

{"state":"on","radioframes":1}
```

.NET Client

StartPlayerAsync(), StopPlayerAsync()

Example in .NET REST Client

```
using OpenRANStudio.RestClient;

using OpenRANStudio.RestClient.Models;

RestClient client = new RestClient("127.0.0.1", 9000);

int moduleId = 0;

int radioFrames = 1;

bool result = await client.StartPlayerAsync(moduleId , radioFrames);

.

.

.

result = await client.StopPlayerAsync(moduleId);
```

Load PCAP file into Player

HTTP POST /Modules/<id>/Player/Pcap

Description

Loads a PCAP file into the player. The file should be in PCAPNG format and should be created by Open RAN Studio.

POST with content-type multipart/form-data uploads a PCAP file into the player.

Table 10 HTTP Response Codes

HTTP Response Code	Message	Description
200 (OK)	–	Indicates that the PCAP file was successfully uploaded into the player
400 (BadRequest)	{"Message":"Module Not Found"}	The specified module ID was not found
400 (BadRequest)	{"Message":"Failed to Load the PCAP file: <filename>"}	The PCAP file was uploaded successfully but failed to load. Most likely, this PCAP file is not a valid stimulus file
415 (UnsupportedMediaType)	{"Message":"expected a multipart form"}	An unexpected media type was encountered
500	{"Message":"Failed to upload pcap to server"}	The file upload failed
500	exception details	Could be returned if an exception processing the request

Example in CURL script

```
curl -s -w "\nResponse: %{http_code}\n" -F data=@Demo2.pcap "http://localhost:9000/modules/1/player/pcap"

Response: 200
```

.NET Client

LoadPcapAsync()

Example in .NET REST Client

```
using OpenRANStudio.RestClient;

using OpenRANStudio.RestClient.Models;

RestClient client = new RestClient("127.0.0.1", 9000);

int moduleId = 0;
```

```
string filename = "stimulus.pcap";   // this should be a file created by the Open RAN Studio Stimulus
builder

bool result = await client.LoadPcapAsync(moduleId, filename);
```

Recorder Functions

The following API functions are covered in this section:

- Get Recorder State
- Start/Stop Recorder
- Get Recorded PCAP

Get Recorder State

HTTP GET /Modules/<id>/Recorder

Description

Gets the state of the recorder.

Table 11 Returned parameters

Parameter	Value	Description
state	▪ on ▪ off ▪ error ▪ not present	▪ Recording is currently active ▪ Recording is currently stopped ▪ Indicates an error occurred during capturing ▪ No capture is present and recorder has not been started
radioframes	0...n	The number of radio frames that was specified.

Table 12 HTTP Response Codes

HTTP Response Code	Message	Description
200 (OK)		The PCAP file was successfully uploaded into the player
400 (BadRequest)	{"Message":"Module Not Found"}	The specified module ID was not found
500	exception details	Could be returned if there is an exception processing the request

Example in CURL script

```
EXEC: curl -s -w "\nResponse: %{http_code}\n" -X GET "http://localhost:9000/Modules/0/Recorder"

{"state":"on","radioframes":0}

Response: 200
```

.NET Client

GetRecorderStateAsync()

Example in .NET REST Client

```
using OpenRANStudio.RestClient;

using OpenRANStudio.RestClient.Models;

RestClient client = new RestClient("127.0.0.1", 9000);

int moduleId = 0;

RecorderInfo recorderInfo = await client.GetRecorderStateAsync(0);

// recorderInfo .state contains the state: "on", "off", "error", "not present"
```

Start/Stop Recorder

HTTP PUT /Modules/<id>/Recorder

Description

Issuing a HTTP Put with a JSON document to the Recorder starts and stops the recorder. The parameters in the JSON document are:

Table 13 Parameters in the JSON Document

Parameter	Value	Description
state	▪ on ▪ off	▪ Start Recording ▪ Stop Recording
radioframes	0...n	The number of radio frames to record. Note that the Hardware does not support sensing radio frames. The underlying implementation currently captures at least the number of radio frames specified.

Table 14 Returned parameters

Parameter	Value	Description
state	▪ on ▪ off ▪ error ▪ not present	▪ Recording is currently active ▪ Recording is currently stopped ▪ Indicates an error occurred during capturing ▪ No capture is present and recorder has not been started
radioframes	0...n	The number of radio frames that was specified to play.

Table 15 HTTP Response Codes

HTTP Response Code	Message	Description
200 (OK)	–	The PCAP file was successfully uploaded into the player
400 (BadRequest)	{"Message":"Module Not Found"}	The specified module ID was not found
400 (BadRequest)	{"Message":"capture already running"}	A request with "state":"on" was issued, but the recorder is already running
400 (BadRequest)	{"Message":"capture not running"}	A request with "state":"off" was issued, but the recorder is not running
500	exception details	Could be returned if there is an exception processing the request

Example in CURL script

```
EXEC: curl -s -w "\nResponse: %{http_code}\n" -d {"state":"on","radioframes":1} -H
Content-Type:application/json -X PUT "http://localhost:9000/Modules/0/Player"

{"state":"on","radioframes":1}
```

.NET Client

StartRecorderAsync(), StopRecorderAsync()

Example in .NET REST Client

```
using OpenRANStudio.RestClient;

using OpenRANStudio.RestClient.Models;

RestClient client = new RestClient("127.0.0.1", 9000);

int moduleId = 0;

int radioFrames = 1;
```

```
bool result = await client.StartRecorderAsync(moduleId , radioFrames);
.
.
.
result = await client.StopRecorderAsync(moduleId);
```

Get Recorded PCAP

HTTP GET /Modules/<id>/Recorder/Pcap

Description

Retrieves the last recorded PCAP. The following restrictions apply:

- The recorder must have been started to capture a PCAP
- The recorder must currently be stopped

Table 16 HTTP Response Codes

HTTP Response Code	Message	Description
200 (OK)	–	Indicates that the PCAP file was successfully uploaded into the player
400 (BadRequest)	{"Message":"Module Not Found"}	The specified module ID was not found
400 (BadRequest)	{"Message":"Capture has not been stopped"}	The capture is currently running. Capture must be stopped before downloading the recording.
400 (BadRequest)	{"Message":"Saving data to file <path> failed"}	A failure occurred downloading the capture from the PCIe card into the server
400 (BadRequest)	{"Message":"Could not find <path> after saving file"}	The capture was downloaded from the PCIe device, but the file could not be read
415 (UnsupportedMediaType)	{"Message":"expected a multipart form"}	An unexpected media type was encountered
500	exception details	Could be returned if an exception occurs while processing the request

Example in CURL script

```
curl -s -w "\nResponse: %{http_code}\n" -X GET "http://localhost:9000/Modules/0/Recorder/Pcap" -o
capture.pcap

Response: 200
```

.NET Client

GetRecordedPcap()

Example in .NET REST Client

```
using OpenRANStudio.RestClient;

using OpenRANStudio.RestClient.Models;

string catpureFile = "test.pcap";

RestClient client = new RestClient("127.0.0.1", 9000);

int moduleId = 0;

bool result = await client.GetRecordedPcap(moduleId, captureFile);

Assert.IsTrue(File.Exists(captureFile));
```

BER/BLER function

BER/BLER Computation

The APIs for BER/BLER measurements compute BIT Error Ratio (BER) or Block Error Ratio (BLER) measurements for Uplink data captured in a PCAP file.

Refer to the Open RAN Studio Online Help for details on how to generate the SCP, ORSTX, XML and PCAP files. It also includes specific details on how the PathWave Signal Generator Uplink waveform must be constructed.

HTTP POST /ComputeBLER

HTTP POST /ComputeBER

Description

The HTTP POST must include multi-part/formdata with File Data for each of the following files:

- The SCP file, where filename must end with .scp
- The ORSTX file, where filename must end with .orstx
- The XML file, where filename must end with .xml
- the PCAP file, where filename must end with .pcap

Table 17 HTTP Response Codes

HTTP Response Code	Message	Description
200 (OK)	–	Indicates success
415 (Unsupported Media Type)	{"Message":"unexpected file content: <filename>"}	An unexpected media type was encountered
	{"Message":"expected a multipart form"}	
500	exception details	Could be returned if an exception occurs while processing the request

Example in CURL script

```
# COMPUTE BLER

curl -s -w \nResponse: %{http_code}\n -F scp=@BLERTest.scp -F orstx=@BLER\BLERTest.orstx -F
xml=@BLERTest.xml -F pcap=@BLERTest.pcap http://localhost:9000/ComputeBLER

{"DataPoints":[{"FrameNumber":1,"ErrorRatioPercent":0.0}]}

Response: 200

# COMPUTE BER

curl -s -w \nResponse: %{http_code}\n -F scp=@BERTest.scp -F orstx=@BERTest.orstx -F xml=@BERTest.xml -F
pcap=@BERTest.pcap http://localhost:9000/ComputeBER

{"DataPoints":[{"FrameNumber":1,"ErrorRatioPercent":0.0}]}

Response: 200
```

.NET Client: ComputeBLER and ComputeBER

public async Task<BLERResults> ComputeBLER(string scpFilename, string orstxFilename, string xmlFilename, string pcapFilename, bool LocalPcap)

public async Task<BLERResults> ComputeBER(string scpFilename, string orstxFilename, string xmlFilename, string pcapFilename, bool LocalPcap)

- scpFilename - the local scp filename, which is used for obtain the IQ data.
- orstxFilename - the local orstx filename containing the Open RAN Studio Configuration.
- xmlFilename - the local BEBOP xml filename</param>
- pcapFilename - the name of pcap file on the (local/remote) system to perform BLER on.
- LocalPcap - one of the two values:
  - true - indicates the PCAP file is on the local system. In this case, the PCAP file will be transferred to the remote system, which may be time consuming.
  - false - indicates the PCAP file is on the remote system. In this case, the PCAP file must exist on the remote system.

Example in .NET REST Client

```
using OpenRANStudio.RestClient;

using OpenRANStudio.RestAPI.Models;

RestClient client = new RestClient(TestParams.Hostname, TestParams.Port);

string scpFilename   = @"BlerTest.scp";

string orstxFilename = @"BLERTest.orstx");
string xmlFilename   = @"BLERTest.xml");

string pcapFilename  = @"BLERTest.pcap");

// For BER, just change to "ComputeBER"

BLERResults results = await client.ComputeBLER(scpFilename, orstxFilename, xmlFilename, pcapFilename,
true);

Console.WriteLine("BLER RESULTS:");

foreach (var data in results.DataPoints)

Console.WriteLine($"  Frame: {data.FrameNumber}, Error Ratio %: {data.ErrorRatioPercent.ToString("F3")}");
```

Logging

Log Files are currently stored under '*C:\Windows\System32\Keysight\Open RAN Studio\OpenRANStudio.log*'.

Some errors, such as errors while starting the service or the errors due to the service exiting unexpectedly, may also be written to the "Windows Event Log".

Instrument Configuration File Format

This topic describes how to create the configuration file 'InstrumentConfig.xml' using Open RAN Studio REST APIs.

Overview

You use the InstrumentConfig.xml file to store the settings that are used to configure the S5040A appliance hardware module such as the stimulus and capture settings for O-RU port(s), external trigger out signals, loopback messages, and eCPRI one-way delay messages.

This file is used in the "“Configure Module”" API function of the Open RAN Studio REST APIs to configure the S5040A system. The same file format is also used in the Open RAN Studio application GUI to configure these settings.

Location of the InstrumentConfig.xml File

This file is saved at the following location:

Documents\Keysight\Open RAN Studio\InstrumentConfig.xml

XML Elements Hierarchy in the InstrumentConfig.xml File

The InstrumentConfig.XML file has the following high-level structure of XML elements:

```
<ModuleConfigSchema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<Modules>

    <FpgaModule>

        <Name/>

        <PortConfig>

             <EthernetProtocol/>

             <StimulusCapturePort/>

             <StimulusCaptureMode/>

             <PortSettings/>

             <PortSettings50G/>

             <PortAddressList/>
```

```
            </PortConfig>
            <TimeSyncConfig/>
            <TriggerOutConfig/>
            <StimulusConfig/>
            <ConnectivityConfig/>
        </FpgaModule>
      </Modules>
</ModuleConfigSchema>
```

All XML elements of the InstrumentConfig.XML file and their attributes are described in detail in tables below.

Table 18 ModuleConfigSchema / Modules / FpgaModule

XML Element	Description	Valid Value(s)
Name	Name of the module	TestModule
PortConfig	Contains XML elements and attributes used to configure stimulus and capture setting for O-RU port(s).	See Table 19 below for details.
TimeSyncConfig	Contains attributes that are used to configure time synchronization settings including reference clock and PTP settings.	See Table 21 below for details.
TriggerOutConfig	Contains attributes to configure settings that are used to generate external Trigger out signals from the S5040A appliance.	See Table 22 below for details.
StimulusConfig	This XML element is obsolete and no longer used (starting with U5040B Open RAN Studio release 2.2). Priori to release 2.2, this element was used to add/delete O-RU MAC address to which the stimulus will be sent.	See Table 23 below for details.
ConnectivityConfig *(New for U5040B release 2.2)*	Contains attributes to configure settings for testing connectivity with O-RU. This includes loopback messages and eCPRI one-way delay messages.	See Table 24 below for details.

Table 19 ModuleConfigSchema / Modules / FpgaModule / PortConfig

Field	Description	Valid Value(s)
EthernetProtocol (New for U5040B release 2.2)	To set the Ethernet protocol mode for the S5040A appliance to send stimulus and capture data. The following two modes are supported. ▪ 10G/25G - In this mode, you can configure the S5040A to run in 1 Port Stimulus Capture, 2 Port Stimulus Capture, or 2 Port Capture modes with the options of setting port(s) speed to 10G or 25G. ▪ 50G - In this mode, you can configure the S5040A to run in 1 Port Stimulus Capture mode at 50G.	10G/25G 50G
StimulusCapturePort	This field is historical and should be set to “1/1”.	1/1
StimulusCaptureMode (Introduced in U5040B release 2.1)	To set the stimulus and capture mode for the S5040A appliance. The following three modes are supported. ▪ 1 Port Stimulus and Capture – Requires U5040BSCB license (for U5040B GUI). Requires U5040MDLB or U5040MULB license (for REST API) ▪ 2 Port Stimulus and Capture – Requires the N7046A-001 license in addition to one of the above licenses. ▪ 2 Port Capture – Two O-RU ports are available only to capture data. Requires the U5040ORAB license.	1 Port Stimulus and Capture 2 Port Stimulus and Capture 2 Port Capture
PortSettings	Contains one or more PortInfo elements (one for each port) to define port settings when EthernetProtocol is set to “10G/25G”. NOTE: Depending on the EthernetProtocol mode used, you specify port settings using either the PortSettings element or the PortSettings50G element.	See Table 20 for details.
PortSettings50G (New for U5040B release 2.2)	Contains one or more PortInfo elements (one for each port) to define port settings when EthernetProtocol is set to “50G”. NOTE: Depending on the EthernetProtocol mode used, you specify port settings using either the PortSettings element or the PortSettings50G element.	See Table 20 for details.
PortAddressList	To define a space-delimited pool of Ethernet MAC addresses that can be specified as source (O-DU) or destination (O-RU) addresses for port(s). Ensure that all Ethernet MAC addresses that you specify in the Address and OruAddress fields of the PortSettings/PortSettings50G elements are added to this list. The Ethernet MAC addresses in the Instrument Configuration > Ports tab of the U5040B GUI are populated based on this list.	Ethernet MAC addresses with hexadecimal digits (0-9 and A-F) and addresses separated by space

Table 20 ModuleConfigSchema / Modules / FpgaModule / PortConfig / PortSettings (and Port-Setings50G) / PortInfo

Field	Description	Valid Value(s)
NetworkPortName	To specify the name to be used for the port in terms of QSFP Slot and Lane. You can choose from the following options based on the specified StimulusCaptureMode ▪ QSFP1_1 : QSFP Slot 1, Lane 1 (can be used in one and two port modes) ▪ QSFP1_2 : QSFP Slot 1, Lane 2 – (can be used in two port modes at 10G/25G only) ▪ QSFP3_1 : QSFP Slot 1, Lane 1 – (can be used for M-Plane Pass-through)	QSFP1_1 QSFP1_2 QSFP3_1
QSFP	To specify the QSFP Slot number to be used for the port.	1 3
Port	To specify the QSFP Lane number to be used for the port.	1 2
Speed	If you set the EthernetProtocol to 10G/25G, then the following speed options are available. ▪ 10G – for 10Gbps Ethernet ▪ 25G No FEC – 25G Ethernet with no Forward Error Correction ▪ 25G RS-FEC – 25G Ethernet with Reed Solomon Forward Error Correction If you set the EthernetProtocol to 50G, then the following speed option is available. ▪ 50G RS-FEC – 50G Ethernet with Reed Solomon Forward Error Correction For M-Plane Passthrough Port, you set the following speed. ▪ 1G – for 1 Gbps Ethernet	10G 25G No FEC 25G RS-FEC 50G RS-FEC 1G
Address	To specify the Ethernet MAC address that will be used as the source address for all stimulus and PTP played through the port. NOTE: This field is normally left empty for QSFP3_1 (M-Plane pass-through).	Hexadecimal digits (0-9 and A-F) nn:nn:nn:nn:nn:nn
OruAddress	To specify the Ethernet MAC address that will be used as the destination address for all stimulus and PTP played through the port. NOTE: This field is normally left empty for QSFP3_1 (M-Plane pass-through).	Hexadecimal digits (0-9 and A-F) nn:nn:nn:nn:nn:nn

Table 21 ModuleConfigSchema / Modules / FpgaModule / TimeSyncConfig

Field	Description	Valid Value(s)
RefClock	To set the clock frequency source for the S5040A system. ▪ Internal – This setting uses the internal oscillator on the S5040A Appliance for its clock. ▪ External 10 Mhz – This setting uses an external 10 MHz reference clock on the "Ref Clock In" Port. ▪ External 100 Mhz – This setting uses an external 100 MHz reference clock on the "Ref Clock In" Port.	Internal External 10 Mhz External 100 Mhz
PtpMode	To set the PTP mode. ▪ Disabled – PTP is not used. ▪ Master – PTP is in the Master mode. ▪ Slave – PTP is in the Slave mode.	Disabled Master Slave
PtpMulticast	To specify if the PTP multicast address is forwardable through switches or not.	Forwardable Not Forwardable
PtpDomain	To specify the PTP Domain number.	0-255
PtpPriority2	To specify the PTP Priority2 value. This value should be lower than 128 when running a PTP Master, and 128 or above when running a PTP Slave.	0-255
PtpUtcOffset	To specify the offset between PTP time and UTC time. This is normally 37, but may change in the future if leap seconds are added or removed.	37
Alpha	This field is NOT used. Either omit this field or set it to 0.	0
Beta	0 (integer) – determines the System Frame Number (SFN) beta offset	0
PPSInDelay	This field is NOT used. Either omit this field or set it to 0.	0

Table 22 ModuleConfigSchema / Modules / FpgaModule / TriggerOutConfig

Field	Description	Valid Value(s)
TriggerEnabled	To enable/disable the Trigger Out signal. ▪ false – disables the Trigger Out Signal ▪ true – enable the Trigger Out Signal	true false (lower case)

Field	Description	Valid Value(s)
TriggerDelay	To specify the number of nanoseconds to delay the trigger pulse out rising edge.	A positive integer
PulseDuration	To specify the pulse duration in microseconds.	A positive integer
TriggerModeSelectedIndex	To specify the trigger mode if Trigger Out signal is enabled. You can choose from the following values for the trigger mode: ▪ 0 – Pulse Per Second (pulses once per second) ▪ 1 – Pulse Per 10 milliseconds (pulses every 10 ms radio frame interval) ▪ 2 – Trigger on Stimulus Start (pulses when the stimulus starts) ▪ 3 – Trigger on Stimulus Start, repeat every 10 milliseconds (pulses when the stimulus starts and repeats every 10ms until stimulus stops) ▪ 4 – Trigger on Capture Start (pulses when capture starts) ▪ 5 – Trigger on Capture Start, repeat every 10 milliseconds (pulses when capture starts and repeats every 10 ms until capture stops)	0 1 2 3 4 5

Table 23 ModuleConfigSchema / Modules / FpgaModule / StimulusConfig

Field	Description	Valid Value(s)
NOTE: The StimulusConfig element is no longer used (starting with U5040B release 2.2) and therefore can be omitted from the InstrumentConfig.xml with U5040B release 2.2. If an older version (prior to release 2.2) of this file is read, this element will be read and the O-RU MAC Address specified for this element will be used as the O-RU MAC address for all ports.		
Delay	This field is not used / Ignored. Either omit this field or set it to 0.	0
ORUMACAddrs	To specify the Ethernet MAC address that will be used as the destination address for all stimulus and PTP played through the port.	Hexadecimal digits (0-9 and A-F) nn:nn:nn:nn:nn:nn
ORUMACAddrSelectedIdx	Zero based integer index of the selected O-RU MAC address to be used.	

Table 24 ModuleConfigSchema / Modules / FpgaModule / ConnectivityConfig (New for U5040B release 2.2)

Field	Description	Valid Value(s)
EnableLbm	To enable/disable loopback messages.	true false (lower case)
LbmMessageCount	To specify the number of loopback messages that you want the O-DU to send to O-RU.	A positive integer
LbmMel	To specify the MEL value (as per the ITU Y.1731) for the O-DU. This value will be used for the MEL parameter in the loopback request messages sent to O-RU.	0 - 7

Field	Description	Valid Value(s)
EnableEcpriDelay	To enable/disable eCPRI One-Way Delay messages.	true false (lower case)
EcpriDelayInitiator	To indicate the mode set for S5040A in the eCPRI Delay measurements. ▪ true: To configure the S5040A as the Initiator. In the Initiator mode, it can send eCPRI delay requests as well as receive eCPRI delay responses from O-RU. ▪ false: To configure the S5040A as the Responder. In the Responder mode, it can only respond to eCPRI delay requests.	true false (lower case)
EcpriDelayUseVlan	To enable/disable Virtual LAN (VLAN) for transmitted eCPRI delay packets	true false (lower case)
EcpriDelayVlanId	To specify the VLAN ID to use for eCPRI Delay messages when VLAN is enabled using EcpriDelayUseVlan.	0 - 4095

InstrumentConfig.xml File Examples

10/25G Configuration Example (2 port stimulus and capture)

```
<ModuleConfigSchema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <Modules>

    <FpgaModule>

      <Name>TestModule</Name>

      <PortConfig>

        <EthernetProtocol>10G/25G</EthernetProtocol>

        <StimulusCapturePort>1/1</StimulusCapturePort>

        <StimulusCaptureMode>2 Port Stimulus and
Capture</StimulusCaptureMode>

        <PortSettings>

          <PortInfo>

            <NetworkPortName>QSFP1_1</NetworkPortName>

            <QSFP>1</QSFP>

            <Port>1</Port>

            <Speed>25G No FEC</Speed>

            <Address>08:08:08:08:08:01</Address>

            <OruAddress>80:09:02:0F:08:01</OruAddress>
          </PortInfo>
          <PortInfo>
            <NetworkPortName>QSFP1_2</NetworkPortName>
            <QSFP>1</QSFP>
            <Port>2</Port>
            <Speed>25G No FEC</Speed>
            <Address>08:08:08:08:08:02</Address>
            <OruAddress>80:09:02:0F:08:02</OruAddress>
          </PortInfo>
          <PortInfo>
            <NetworkPortName>QSFP3_1</NetworkPortName>
            <QSFP>3</QSFP>
            <Port>1</Port>
            <Speed>Disabled</Speed>
            <Address />
            <OruAddress />
          </PortInfo>
        </PortSettings>
        <PortAddressList>80:09:02:0F:08:01 80:09:02:0F:08:02
08:08:08:08:08:01 08:08:08:08:08:02</PortAddressList>
      </PortConfig>
      <TimeSyncConfig>
        <RefClock>External 10 Mhz</RefClock>
        <PtpMode>Master</PtpMode>
        <PtpMulticast>Forwardable</PtpMulticast>
        <PtpDomain>24</PtpDomain>
        <PtpPriority2>128</PtpPriority2>
        <PtpUtcOffset>37</PtpUtcOffset>
        <Alpha>0</Alpha>
        <Beta>0</Beta>
        <PPSInDelay>0</PPSInDelay>
      </TimeSyncConfig>
      <TriggerOutConfig>
        <TriggerEnabled>false</TriggerEnabled>
        <TriggerDelay>0</TriggerDelay>
        <PulseDuration>30</PulseDuration>
        <TriggerModeSelectedIndex>2</TriggerModeSelectedIndex>
      </TriggerOutConfig>
      <StimulusConfig>
        <Delay>0</Delay>
        <ORUMACAddrs>02:04:06:08:0A:0C</ORUMACAddrs>
        <ORUMACAddrSelectedIdx>0</ORUMACAddrSelectedIdx>
      </StimulusConfig>
      <ConnectivityConfig>
        <EnableLbm>true</EnableLbm>
        <LbmInterval>1</LbmInterval>
        <LbmMessageCount>0</LbmMessageCount>
        <LbmMel>0</LbmMel>
        <EnableEcpriDelay>true</EnableEcpriDelay>
        <EcpriDelayInitiator>true</EcpriDelayInitiator>
        <EcpriDelayUseVlan>true</EcpriDelayUseVlan>
        <EcpriDelayVlanId>4095</EcpriDelayVlanId>
      </ConnectivityConfig>
    </FpgaModule>
  </Modules>
</ModuleConfigSchema>
```

50G Configuration Example (1 port stimulus and capture)

```
<ModuleConfigSchema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Modules>
    <FpgaModule>
      <Name>TestModule</Name>
      <PortConfig>
        <EthernetProtocol>50G</EthernetProtocol>
        <StimulusCapturePort>1/1</StimulusCapturePort>
        <StimulusCaptureMode>1 Port Stimulus and
Capture</StimulusCaptureMode>
        <PortSettings50G>
          <PortInfo>
            <NetworkPortName>QSFP1_1</NetworkPortName>
            <QSFP>1</QSFP>
            <Port>1</Port>
            <Speed>50G RS-FEC</Speed>
            <Address>08:09:02:0F:08:18</Address>
            <OruAddress>08:09:02:0F:08:19</OruAddress>
          </PortInfo>
          <PortInfo>
            <NetworkPortName>QSFP3_1</NetworkPortName>
            <QSFP>3</QSFP>
            <Port>1</Port>
            <Speed>Disabled</Speed>
            <Address />
            <OruAddress />
          </PortInfo>
        </PortSettings50G>
        <PortAddressList>08:09:02:0F:08:19
08:09:02:0F:08:18</PortAddressList>
      </PortConfig>
      <TimeSyncConfig>
        <RefClock>External 100 Mhz</RefClock
        <PtpMode>Master</PtpMode>
        <PtpMulticast>Forwardable</PtpMulticast>
        <PtpDomain>24</PtpDomain>
        <PtpPriority2>129</PtpPriority2>
        <PtpUtcOffset>37</PtpUtcOffset>
        <Alpha>0</Alpha>
        <Beta>0</Beta>
        <PPSInDelay>0</PPSInDelay>
      </TimeSyncConfig>
      <TriggerOutConfig>
        <TriggerEnabled>false</TriggerEnabled>
        <TriggerDelay>0</TriggerDelay>
        <PulseDuration>30</PulseDuration>
        <TriggerModeSelectedIndex>2</TriggerModeSelectedIndex>
      </TriggerOutConfig>
      <StimulusConfig>
        <Delay>0</Delay>
        <ORUMACAddrs>02:04:06:08:0A:0C</ORUMACAddrs>
        <ORUMACAddrSelectedIdx>0</ORUMACAddrSelectedIdx>
      </StimulusConfig>
      <ConnectivityConfig>
        <EnableLbm>false</EnableLbm>
        <LbmInterval>1</LbmInterval>
        <LbmMessageCount>0</LbmMessageCount>
        <LbmMel>0</LbmMel>
        <EnableEcpriDelay>true</EnableEcpriDelay>
        <EcpriDelayInitiator>true</EcpriDelayInitiator>
        <EcpriDelayUseVlan>false</EcpriDelayUseVlan>
        <EcpriDelayVlanId>0</EcpriDelayVlanId>
      </ConnectivityConfig>
    </FpgaModule>
  </Modules>
</ModuleConfigSchema>
```

Using the .NET Client

The .NET Client is a Visual Studio .NET Framework assembly that implements functions that call the REST interface and handle JSON serialization / deserialization details. This is implemented in the following assemblies below. These two assemblies must be copied to the same directory as the .NET application using them. A reference must be added for the Open RAN Studio REST API Client.dll Assembly.

C:\Program Files\Keysight\Open RAN Studio\Open RAN Studio REST API Client.dll

C:\Program Files\Keysight\Open RAN Studio\Open RAN Studio REST API Common.dll

Some general guidelines for using the APIs are:

- All of the APIs result in HTTP transactions and thus are asynchronized, so calling each API must use “await”.
- The APIs may throw exceptions in various error conditions:
  - The Service is not running on the specified host
  - The REST API call has an error
  - The specified module does not exist or is not in the correct state. For example, trying to start the player if it is already running
- A program using the .NET Client Assembly must add a reference to the assembly. Also, it must have a copy of the “Open RAN Studio REST API Client.dll” either in the PATH or in the same directory as the application at runtime.

The following namespaces are used for the API:

- OpenRANStudio.RestClient – This namespace is used for the RestClient class, which contains the complete interface for the .NET REST client.
- OpenRANStudio.RestAPI.Models – This namespace defines all of the objects used for inputs/outputs, which are then serialized to/from JSON when calling the REST APIs.

Example - Sample .NET test using REST API

Below is an example of a MSTest class that calls the .NET Client APIs:

```
using System;
using System.Threading.Tasks;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using OpenRANStudio.RestClient;
using OpenRANStudio.RestAPI.Models;
using System.IO;
using System.Collections.Generic;
namespace OpenRANStudioRESTAPITest
{
    [TestClass]
    public class RestAPITest
    {
        public string _hostname = "localhost";
        public int _port = 9000;
        public string _testFilePath = @"..\..\..\..\RestAPITest\";
        // Note: These tests can run in any order, however, only one test should run at a time
        [TestMethod]
        public async Task Test101Version()
        {
            System.Diagnostics.Trace.TraceInformation(">>> Test101Version");
            RestClient client = new RestClient(_hostname, _port);
            VersionInfo ver = await client.GetVersionAsync();
            Assert.AreEqual(ver.name, "OpenRanStudioAPI");
            Assert.AreEqual(ver.apiversion, "1.0");
            Assert.AreEqual(ver.version, "1.0.0.0");
            System.Diagnostics.Trace.TraceInformation("<<< Test101Version");
        }
        [TestMethod]
        public async Task Test102Modules()
        {
            System.Diagnostics.Trace.TraceInformation(">>> Test102Modules");
            RestClient client = new RestClient(_hostname, _port);

            List<ModuleInfo> moduleList = await client.GetModulesAsync();

            System.Diagnostics.Trace.TraceInformation($"GetModulesAsyncy returned
{moduleList.ToString()}");

            Assert.AreEqual(moduleList.Count, 1);

            Assert.AreEqual(moduleList[0].id, 0);

            Assert.AreEqual(moduleList[0].model, "BittWare PCIe FPGA");

            Assert.IsTrue((moduleList[0].address).Contains("PXI"));

            Assert.IsTrue((moduleList[0].address).Contains("INSTR"));

            System.Diagnostics.Trace.TraceInformation("<<< Test102Modules");

        }

        [TestMethod]

        public async Task Test130CaptureRecord()

        {

            System.Diagnostics.Trace.TraceInformation(">>> Test130CaptureRecord");

            bool result;

            RestClient client = new RestClient(_hostname, _port);

            result = await client.ConfigureAsync(0, Path.Combine(_testFilePath,
@"InstrumentConfig-LoopbackPtpDisabled.xml"));

            Assert.IsTrue(result);

            RecorderInfo recorderInfo = await client.GetRecorderStateAsync(0);

            Assert.AreEqual(recorderInfo.state, "not present");

            PlayerInfo playerInfo = await client.GetPlayerStateAsync(0);

            Assert.AreEqual(playerInfo.state, "not present");

            result = await client.LoadPcapAsync(0, Path.Combine(_testFilePath, @"Demo2.pcap"));

            Assert.IsTrue(result);

            Assert.IsTrue(await client.StartRecorderAsync(0, 1));

            recorderInfo = await client.GetRecorderStateAsync(0);

            Assert.AreEqual(recorderInfo.state, "on");

            //Assert.AreEqual(recorderInfo.radioframes, 1);

            Assert.IsTrue(await client.StartPlayerAsync(0, 1));

            playerInfo = await client.GetPlayerStateAsync(0);

            Assert.AreEqual(playerInfo.state, "on");

            //Assert.AreEqual(playerInfo.radioframes, 1);
            await Task.Delay(100);
            Assert.IsTrue(await client.StopPlayerAsync(0));
            Assert.IsTrue(await client.StopRecorderAsync(0));
            // upload capture
            string captureFile = Path.Combine(_testFilePath, @"capture130.pcap");
            if(File.Exists(captureFile))
                File.Delete(captureFile);
            Assert.IsTrue(await client.GetRecordedPcap(0, captureFile));
            Assert.IsTrue(File.Exists(captureFile));
            System.Diagnostics.Trace.TraceInformation("<<< Test130CaptureRecord");
        }
    }
}
```

3 REST API Sample Programs

S5040A Hardware settings

Perform the following steps on the S5040A Open RAN Studio Player and Capture Appliance Hardware.

1. Make sure that the U5040B Open RAN Studio software and the PathWave Signal Generation (PWSG) Desktop software are installed.
   - For the latest version of the U5040B software, navigate to the U5040B Open RAN Studio software download page.
   - For the latest version of the PWSG software, navigate to the PathWave Signal Generation (PWSG) Desktop Software download page.
2. Make sure that the necessary licenses installed. See Licensing for Open RAN Studio REST API on page 11.
3. Start the Open RAN Studio REST API services. See Starting the Open RAN Studio API service on page 12.

REST API Programming in .NET Client

Settings to run the REST API script (.NET)

Perform the following steps on the machine/hardware where you intend to run the REST API script using Visual Studio.

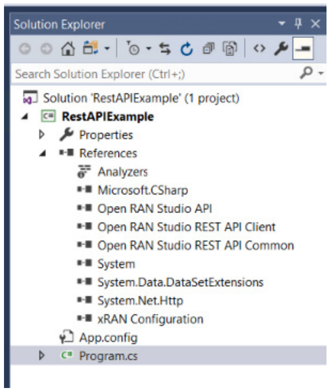
1 Make sure that the U5040B Open RAN Studio software and the PathWave Signal Generation Desktop 2022 software along with the respective licenses are installed. Note that the version of each software must match with that on the S5040A Appliance.

2 Make sure that Visual Studio 2017 or higher is installed on this machine.

3 Follow the steps below to create test for REST API:

NOTE

Open RAN Studio REST API Client and *Open RAN Studio REST API Common DLLs* are intended for REST API Services. If you are in offline mode, this “Option” is not displayed in Windows Services. This Service is of significance with the S5040A Hardware (Online) only.



Solution 'RestAPIExample' (1 project)
- **RestAPIExample**
  - Properties
  - References
    - Analyzers
    - Microsoft.CSharp
    - Open RAN Studio API
    - Open RAN Studio REST API Client
    - Open RAN Studio REST API Common
    - System
    - System.Data.DataSetExtensions
    - System.Net.Http
    - xRAN Configuration
  - App.config
  - Program.cs

Figure 4 Creating test for REST API in Visual Studio 2017

i Create a new Console App (.NET Framework)

ii Add the following references:

- Open RAN Studio API
- Open RAN Studio REST API Client

- Open RAN Studio REST API Common
- xRAN Configuration

iii Copy the sample script given in Sample script using .NET on page 55 and paste / replace with any existing script in the *Program.cs* file.

The structure appears as shown in Figure 4.

iv In the Program.cs file in Visual Studio, modify the following:

- Hostname (IP) of hardware setup
- stimulus file path (file path should be on the system with script)
- capture file path (file path should be on the system with script)
- configuration file path (file path should be on the system with script)

v Select “x64”, followed by generating build and running the script.

Sample script using .NET

```
//Add Open RAN Studio REST API Client & REST API Common dlls in reference.


using OpenRANStudio.RestAPI.Models;
using OpenRANStudio.RestClient;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace RestAPIExample
{
    class Program
    {
        static void Main(string[] args)
        {
            abc().Wait();
        }
        static async Task abc()
        {
          string _hostname = "10.113.65.84";//"141.121.236.170"; //"Enter Host IP Address Here";//
          int _port = 9000;
          bool result = false;

          string stimulusFile = @"C:\pcap\UL Bandwidth- FR1 5MHz.pcap";// "Enter Stimulus File Path";//
          string captureFile = @"C:\pcap\UL Bandwidth- FR1 5MHz_capture1.pcap";//"Enter capture file
path";//
          string configFile = @"C:\pcap\InstrumentConfig.xml";//"Enter Configuration file path";//

            RestClient client = new RestClient(_hostname, _port);// Create the link with client
            result = await client.ConfigureAsync(0, configFile);
            Console.WriteLine("Config Result" + result);
```

```
RecorderInfo recorderInfoMaster = await client.GetRecorderStateAsync(0);
Console.WriteLine("Recorder Info Master: " + recorderInfoMaster);

PlayerInfo playerInfoMaster = await client.GetPlayerStateAsync(0);
Console.WriteLine("playerInfo : " + playerInfoMaster);

List<ModuleInfo> moduleList = await client.GetModulesAsync();
int moduleId = moduleList[0].id;
Console.WriteLine("Module ID: " + moduleId.ToString());

result = await client.LoadPcapAsync(moduleId, stimulusFile); //Load the pcap file
Console.WriteLine("Result Load PCAP Info :" + result);

await Task.Delay(30000);

result = await client.StartRecorderAsync(moduleId, 3); //Start the recorder
Console.WriteLine("Recorder status for Master: " + result);

await Task.Delay(30000);
recorderInfoMaster = await client.GetRecorderStateAsync(moduleId);
Console.WriteLine("RecorderInfo" + recorderInfoMaster.state);

await client.StartPlayerAsync(moduleId, 0);

playerInfoMaster = await client.GetPlayerStateAsync(moduleId);
Console.WriteLine("PlayerInfo: " + playerInfoMaster.state);

await Task.Delay(10000);
await client.StopPlayerAsync(moduleId);
await client.StopRecorderAsync(moduleId);

recorderInfoMaster = await client.GetRecorderStateAsync(moduleId);
```

```
            Console.WriteLine("Recorder Info: " + recorderInfoMaster);

            playerInfoMaster = await client.GetPlayerStateAsync(moduleId);
            Console.WriteLine("Player Info: " + playerInfoMaster);

            // upload capture
            await client.GetRecordedPcap(0, captureFile);
            File.Exists(captureFile);
        }
    }
}
```

REST API Programming in Python

Settings to run the REST API script (Python)

Perform the following steps on the machine/hardware where you intend to run the REST API script.

1 Make sure that the U5040B Open RAN Studio software and the PathWave Signal Generation Desktop 2022 software along with the respective licenses are installed. Note that the version of each software must match with that on the S5040A Appliance.

2 Make sure that the latest version of Python is installed on this machine.

3 Make sure that Python “requests” package is available.

- If unavailable, run the command “*pip install requests*”

4 Follow the steps below to create test for REST API:

   i Copy the sample script given in Sample script using Python on page 59 and paste it in any Python Editor.

   ii Save the file and assign a name of your choice. For example, the script below is saved as “*REST_API_python_sample.py*”.

   iii In the <file-name>.py, modify the following:

   - Hostname (IP address) of the hardware setup

   By default, “url” is set to *localhost*. You can change the IP address according to your setup.

   - stimulus file path (from the machine where the script is being run)
   - configuration file path (from the machine where the script is being run)

   iv Launch the command prompt window or any Python IDE.

   v Run the python script using the command “*python <fine-name>.py*”.

   - In this example, run “python *REST_API_python_sample.py*”

   A captured response is added to the same location as the stimulus file with the suffix ‘_captured’ appended to the file name. For example, if the stimulus file name is *DLConfig.pcap*, the capture will have the name *DLConfig_captured.pcap*.

The python script returns an HTTP response code for each request. Refer to the Response Code for each function in Chapter 2, “Structure and Functions of the REST API” in case you receive any errors.

Sample script using Python

```
import requests
import time

url = "http://localhost:9000/Modules"

#Get Module List
payload = dict(id='0',model='S5040A',address='PXI0::2-0.0::INSTR')
resp = requests.get(url,data=payload)
print("Get Module List:"+ str(resp.content)[1:]+"\t Response Code:" +  str(resp.status_code))

url = "http://localhost:9000/Modules/0/"

#Configure Module
instrumentconfig = #Add InstrumentConfig.xml Path" #
files = {
    'data': (instrumentconfig, open(instrumentconfig, 'rb')),
       }

resp = requests.post(url+"Configure",files = files)
print("Load Configuration Status:" +  str(resp.content)[1:] +"\t Response Code:" +  str(resp.status_code))

#Player Recorder Functions
filename = #Add Stimulus File Path#
files = {
    'data': (filename, open(filename, 'rb')),
       }
payload = dict(state='on', radioframes='0')
headers = {
    'Content-Type': 'application/json',
       }

#load pcap
```

```
resp = requests.post(url+"Player/pcap",files = files)

print("Load Pcap Status:" +  str(resp.content)[1:] +"\t Response Code:" +  str(resp.status_code))

#start Recorder
data = '{"state":"on","radioframes":1}'
resp = requests.put(url+"Recorder",data=data,headers = headers)
print("Start Recorder Response:" + str(resp.content)[1:]+"\t Response Code:" +  str(resp.status_code))

#Get Recorder state
resp = requests.get(url+"Recorder",data=payload)
print("Recorder State:" + resp.text+"\t Response Code:" +  str(resp.status_code))

#start player
data = '{"state":"on","radioframes":1}'
resp = requests.put(url+"Player",data=data,headers = headers)
print("Start Player Response:" + str(resp.content)[1:]+"\t Response Code:" +  str(resp.status_code))

#Getplayer state
resp = requests.get(url+"Player",data=payload)
print("Player State:" + resp.text+"\t Response Code:" +  str(resp.status_code))

#stop player
data = '{"state":"off","radioframes":1}'
resp = requests.put(url+"Player",data=data,headers = headers)
print("Stop Player Response:" +  str(resp.content)[1:]+"\t Response Code:" +  str(resp.status_code))

#stop recorder
data = '{"state":"off","radioframes":1}'
resp = requests.put(url+"Recorder",data=data,headers = headers)
print("Stop Player Response:" +  str(resp.content)[1:]+"\t Response Code:" +  str(resp.status_code))

#getRecorder pcap
```

```
file = open(filename[:-5] + "_captured.pcap", "wb")
resp = requests.get(url+"Recorder/pcap")

file.write(resp.content)
file.close()
print("Action Complete")
```