Bx-RO-Verifier: ByStar Remote-Operations Invocations And Verifications Framework

Tools And Strategies For Generalized OpenAPI/Swagger Based Verification Of Web-Services

Document #PLPC-180057
Version 0.4
February 12, 2019

This Document is Available on-line at:
http://www.by-star.net/PLPC/180057

Mohsen BANAN
Email: http://mohsen.1.banan.byname.net/contact

Contents

List of Figures

Part I

Overview Of mmwsIcm Web Services Testing Framework

1 Obtaining Related Software

Software (Open-Source):

- pip-pkg at PyPi: `https://pypi.org/project/unisos.mmwsIcm`
- GitHub: `https://github.com/bisos-pip/mmwsIcm`

Notes:

2 Related Documents

Interactive Command Modules (ICM) and Players A Framework For Cohesive Generalized Scripting http://www.by-star.net/PLPC/180050 — [4]

Remote Operations Interactive Command Modules (RO-ICM) Best Current (2019) Practices For Web Services Development http://www.by-star.net/PLPC/180056 — [3]

A Generalized Swagger (OpenAPI) Centered Web Services Invocations And Testing Framework http://www.by-star.net/PLPC/180057 — [1]

Extending SON To Clouds And Things GOSSONoT: A Generalized Open-Source Self Organizing Network of Things Platform http://www.by-star.net/PLPC/180052 — [2]

Notes:

3 Part Of A Much Bigger Picture – ByStar and BISOS

This Software is Part Of A Much Bigger Picture.

This Software Is Part Of: The Libre-Halaal ByStar Digital Ecosystem And Part Of:

BISOS: ByStar Internet Services OS

This software is primarily being used and developed in that context.

Notes:

4 Our Web Services And Remote Operations Model

4.1 Structure Of Web Services Implementation (Remote Operations)

Implemenation Of Remote Operations Can Typically Be Structured As:

1. Remote Performer Implementation – http://www.by-star.net/PLPC/180056
2. Remote Invoker Implementation – http://www.by-star.net/PLPC/180057
3. Direct Operations Implementation – http://www.by-star.net/PLPC/180050

This document focuses on Remote Invoker Implementation.

You should read this document alongside the mentioned documents.

Interactive Command Modules (ICM) allow for consistent Direct and Remote Operations.

Notes:

4.2 Interactive Command Modules (ICM) Direct And Remote Operations

---

Our implementation model for remote operations is based on the model of Interactive Command Modules (ICM).

The Interactive Command Modules Framework allows for a Direct Operation to be split into a Performer Remote Operation module and an Invoker Remote Operation module.

The Interactive Command Modules Framework allows for a Remote Operation to also be used as Direct Remote.

The Interactive Command Modules Framework allows for operations to be mapped to command-line invocations.

Notes:

5 Invokers Development Model

5.1 A Graphical Overview Of Invoker ICMs Development Model

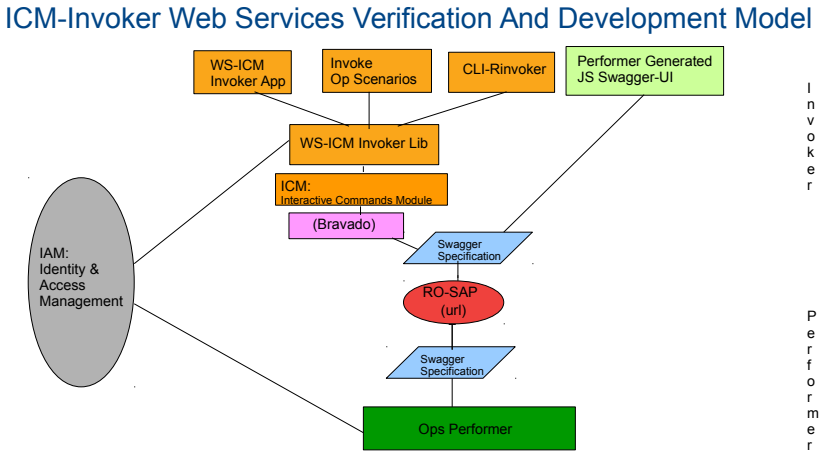

Figure 1: Swagger Based ICM Web Services Invoker Model

Notes:

5.2 Invoker ICMs Development Model

Given a Service Definition (a swagger file) and a Performer Server, you should be able to conveniently Invoke any of the offered Operations through:

1. swagger.ui interface – usually offered by the Performer Server
2. unisos.mmwsIcm :: rinvoker – command line and batch oriented equivalent of swagger.ui
3. unisos.mmwsIcm :: opScn – invoke-specification – invoke-verification – invoke-reporting
4. unisos.mmwsIcm :: Library – wsInvoker.py, ro.py – for building invoker Apps

---

Notes:

5.3 Invoker Framework: Ingredients

Main software packages that implement the framework include:

1. Python Bravado – Equivalent of Invoker Codegenartor But Better
2. unisos.icm – Interactive Command Module
   - Makes icm.Cmnd classes invokable at command-line
   - do-icm :: Direct Operation ICMs (Used by performers)
3. unisos.mmwsIcm
   - unisos.mmwsIcm.wsInvoker.py – Maps invocations to http requests
   - unisos.mmwsIcm.ro.py – Abstracts invoke-specifications
   - unisos.mmwsIcm.rinvoker.py – Maps command-line args to invokations

---

Notes:

5.4 Try It Out – Install The Software And Run The Examples

Install The Software:

- pip install unisos.mmwsIcm

Run The PetStore Example:

- rinvokerPetstore.py
- opScnPetstore.py

Notes:

Part II

Command Line Remote Invocation (rinvoker) – rinvokerPetstore.py Example

6 rinvoker Seed Features – Commands – Paramters – Arguments

6.1 rinvoker.py Seed Features – Commands

- Cmnd: -i svcOpsList

  ```
  svcOpsList command digests the Service Specification (swagger-file)
  specified on command line as --svcSpec= parameter and produces a
  complete list of ALL remotely invokable commands with their corresponding
  --resource, --opName and url or body arguments.

  Applicable options, parameters and arguments are:

    * Parameter (Mandatory) : --svcSpec=
    * Parameter (Optional)  : --perfSap=  --headers=
  ```

- Cmnd: -i rinvoker

  ```
  rinvoker command invokes the "opName" operation at "resource" with
  specified arguments.

  Applicable options, parameters and arguments are:

    * Parameter (Mandatory) : --svcSpec=  --resource=  --opName=
    * Parameter (Optional)  : --perfSap=  --headers=

    * Arguments             : name=value  bodyStr=jsonStr
  ```

Notes:

6.2 rinvoker.py Seed Features – Parameters

- Parameter: –svcSpec= (url, or swagger-file)

  The swagger file as a url or as a json/yaml file is specified with the –svcSpec= parameter.

- Parameter: –perfSap= (url)

  The Performer Service Access Point Address (perfSap) is specified as a URL with the –perfSap= parameter.

- Parameter: –header= (file)

  Additional headers (e.g., a token) can be included with the –svcSpec= parameter.

- Parameter: –resource= (string, corresponding to SvcSpec)

  The resource to be invoked should be specified with the –resource= parameter

- Parameter: –opName= (string, corresponding to SvcSpec)

  The operation name to be invoked should be specified with the –opName= parameter

---

Notes:

6.3 rinvoker.py Seed Features – Arguments

- Argument: name=value (string=string corresponding to SvcSpec's URL Params)
- Argument: bodyStr=jsonStr (bodyStr=string corresponding to SvcSpec's Body)

Notes:

7 rinvokerPetstore.py Example

Allows you to list all possible invocations based on a service specification (swagger file).

```
rinvoker.py --svcSpec="http://petstore.swagger.io/v2/swagger.json" -i svcOpsList
```

Allows you to fully specify an invocation on command line. Example:

```
rinvoker.py  --svcSpec="http://petstore.swagger.io/v2/swagger.json"
      --resource="user" --opName="createUser"  -i rinvoke
      bodyStr="{...}"
```

Notes:

Part III

Operation Scenarios – opScnPetstore.py Example

8 Invoke-Specifications, Invoke-Verification, Invoke-Reporting

8.1 Model Of Invoke – Specification, Verification And Reporting – Scenarios

---

- Invoke Scenarios Are pure python specification of sequence of invocations.
- Invoke-Expect Scenarios Are pure python specification of sequence of invocations subject to preparations and post-invoke verification and reporting.
- opInvoke class allows for complete invoke specification and complete results to be fully captured.

Notes:

8.2 Scenario Specification For Sequences Of Invocations

In pure python specify invocation of each operation, for example:

```
thisRo = ro.Ro_Op(
    svcSpec=petstoreSvcSpec,
    perfSap=petstoreSvcPerfSap,
    resource="pet",
    opName="getPetById",
    roParams=ro.Ro_Params(
        headerParams=None,
        urlParams={ "petId": 1},
        bodyParams=None,
        ),
    roResults=None,
    )
rosList.opAppend(thisRo)
```

Notes:

Building on the previously mentioned Operation Specification, in pure python you can the specify Operation Expectations, for example:

```
thisExpectation = ro.Ro_OpExpectation(
    roOp=thisRo,
    preInvokeCallables=[sleep1Sec],
    postInvokeCallables=[ verify_petstoreSvcCommonRo, ],
    expectedResults=None,
    )
roExpectationsList.opExpectationAppend(thisExpectation)
```

preInvokeCallables(ro.Ro_OpExpectation) can include a function that initializes the DB or sleepFor1Sec.

postInvokeCallables(ro.Ro_OpExpectation) can include a function that verifies the result was as expected and then reports success or failure.

Notes:

9 opScn-Seed (Remote Operation Scenarios) – Commands – Paramters – Arguments

9.1 opScn Seed Features – Commands

---

opScn-seed provides the following commands and parameters:

- Cmnd: -i roListInv

  ```
  roListInv command serially invokes the list of ro.Ro_Op() opersations
  specified in the loaded scenario files.

  roListInv displays the invokation and its results. But does not do any verifications.

  Applicable options, parameters and arguments are:

    * Parameter (Mandatory) : --load=
  ```

- Cmnd: -i roListExpectations

  ```
  roListExpectations command serially invokes the list of ro.Ro_OpExpectation()
  specified in the loaded scenario files.

  roListExpectations displays the invokation and its results and
  additionally runs the list of preInvokeCallables and
  postInvokeCallables.

  postInvokeCallables can include functions that verify the results of the invocation
  were as expected.

  Applicable options, parameters and arguments are:

    * Parameter (Mandatory) : --load=
  ```

Notes:

9.2 OpScn Outputs And Reportings

The output format is:

```
 * ->:: Invoke Request
 * <-:: Invoke Response
 * ==:: Invoke Validation (SUCCESS or FAILURE)
```

Additional information for each is include with "**" tags.

This output format can then be used in outline or org-mode.

Notes:

Part IV

Complete Invoker-Applications Development

10 Invoker-Apps Development Model

10.1 Invoker-Apps Can Easily Build On unisos.mmwsIcm Capabilities

---

- Bravado does invoker code-generation on the fly.
- unisos.mmwsIcm.opInvoke – Abstracts invoke-specifications
- unisos.mmwsIcm.wsInvoker – Allows for invokation and verification of opInvoke

With these in place, building Invoke-Apps becomes very simple.

Notes:

Part V

Security Strategies For Web Services Verification

11 IAM and AAA

11.1 Incorporation Of Authentication And Tokens In Swagger Specifications

---

- Support for formal specification of authentication methods in Swagger-2 is ad-hoc.
- Support for formal specification of authentication methods in OpenApi-3 is new.
- Proper, full specification of placement of JWT (Jason Web Tokens) in the Swagger spec is incomplete.
- Withing a bounded digital ecosystem, full support for AAA in the Swagger specification can be accomplished based on conventions.

Notes:

11.2 IAM Interactions

- Within a given specific digital ecosystem, it is practical to marry IAM with swagger specifications based on conventions and best practices.

Notes:

12 Identification Of Common API Vulnerabilities

12.1 Identification Of Some Common API Vulnerabilities

- A combination of machine and human based review of the swagger file can lead to identification of potential security vulnerabilities.
- Based on these OpScn tools, specific Penetration Tests can be devised.

Notes:

Part VI

Benefits Of Adopting This Generalized Swagger Centered Invocation Model

13 Benefits And Advantages Of The Generalized Swagger Centered Invocation Model

13.1 Taking Full Advantage Of Service Specification For Testing And Development

---

- The Generalized Model and Capabilities Presented Here Apply To Any Service That Exposes Its Swagger Specifications
- A great deal of automation capabilities have become possible based on swagger specifications.
- The Testing Framework Of (invoke-specification, invoke-verification and invoke-reporting) permits for disciplined compelte external testing of web-services based on swagger specifications.

Very Often, These Best Current Practices Are Not Being Followed.

Notes:

References

[1] " Mohsen BANAN ". " a generalized swagger (openapi) centered web services testing and invocations framework ". Permanent Libre Published Content "180057", Autonomously Self-Published, "December" 2018. http://www.by-star.net/PLPC/180057.

[2] " Mohsen BANAN ". " extending son to clouds and things gossonot: A generalized open-source self organizing network of things platform ". Permanent Libre Published Content "180052", Autonomously Self-Published, "December" 2018. http://www.by-star.net/PLPC/180052.

[3] " Mohsen BANAN ". " remote operations interactive command modules (ro-icm) best current (2018) practices for web services development ". Permanent Libre

Published Content ”180056”, Autonomously Self-Published, ”September” 2018. http://www.by-star.net/PLPC/180056.

[4] ” Neda Communications Inc”. ” interactive command modules (icm) and players a framework for cohesive generalized scripting a model for gui-line user experience ”. Permanent Libre Published Content ”180050”, Autonomously Self-Published, ”July” 2017. http://www.by-star.net/PLPC/180050.