Remote Operations Interactive Command Modules (RO-ICM)

Best Current (2019) Practices For Web Services Development

Article Format Of Presentation

Document #PLPC-180056
Version 0.4
February 12, 2019

This Document is Available on-line at:
http://www.by-star.net/PLPC/180056

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

Contents

List of Figures

Part I

Overview

1 Our Web Services And Remote Operations Model

1.1 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 Performer Implementation.

You should read this document alongside the mentioned documents.

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

Notes:

1.2 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:

2 Obtaining Related Software

Software (Open-Source):

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

Notes:

3 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:

4 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:

5 In 2019 What Are The Best Current Practices For Building Web Services

5.1 What Do We Mean By: Web Services

- machine-to-machine (application-to-application) oriented

- Modeled As Remote Operations
- Based on Web Protocols And Web Data Encoding Standards
  - http, xml, json

Notes: Frame Notes

5.2 What Do We Mean By: Best Current Practices

Technology, Processes, Procedures and Paractices That Reflect Industry Consensus Towards Being:

- Rich and Growing Tools Base And Environment For Development, Testing And Operation
- Easy to Code And Debug In Single Local Process Model
- Consistent Code Generation For Invokers And Performers
- Scalable and Reliable

Notes:

5.3 Technological Context And Contours

Technological Context

Contours Of Chosen Key Ingridients

- Web Services (Remote Operations Conetxt)
  - Fully FLOSS (Libre-Halaal) Tools and Environment
  - Service Specification – Open API 3 (fka Swagger)
  - Operations Protocol – http Web Services
  - Performer Language – Python 3
  - Invoker Language – Python 3
- Developemnt Framework: Interactive Commands Module (ICM)

Notes: Frame Notes

6 Web Services Development BCP – Tools Chain

6.1 Web Services Development BCP – Tools Chain – Service Specification

---

Web Services Development BCP – Tools Chain – Service Specification

Contours Of Chosen Key Tools

- Python Interactice Command Modules (ICM) Framework
- Swagger Tool Chain:
  - Swagger-Editor – Used For Looking At Operations Specification – Not For Editing Them
  - Swagger-Verifier – (validator-badge) to help verify that Operations-Specification is correct

---

Notes:

6.2 Web Services Development BCP – Tools Chain – Invokers And Performers

---

Web Services Development BCP – Tools Chain

Contours Of Chosen Key Tools

- Performer Tool Chain:
  - Swagger-codegen – Only Performer Side Python3 feature is used – Flask+Connexion+ServerStubs
- Invoker Tool Chain:
  - Swagger-UI – Auto included in every service
  - Python Bravado – Equivalent of Invoker Codegenartor But Better
  - unisos.mmwsIcm – rinvoker.py – Batch equivalent of Swagger-UI
  - unisos.mmwsIcm – ro.py – invoke-specification – invoke-verification – invoke-reporting

---

Notes:

Part II

Model And Terminology

7 Remote Operations Model And Terminology

7.1 Remote Operations Model And Terminology – Data Communications Vs Software Engineering

---

Model And Terminology – Data Communications Vs Software Engineering

ROSE: Remote Operations Services Element

- Remote Operations Services Model And Terminology Was First Introduced In 1988 By Data Comm Experts
- Models and Terminology Inferior To It Has Been Used By Software Engineers (Hacks) – And Have Become Mainstream

Best Practice is to use proper, complete and correct model and terminology with discipline.

We will be using ROSE's model and terminology of Operations (with some augmentations), Invokers and Performers – not clients and servers.

Notes:

7.2 Remote Operations Terminology Vs Web Services Terminology

---

Remote Operations Terminology Vs Web Services Terminology

See ROSE: Remote Operations Services Element – X.219 For Details

Remote Operations Terminology	Web Services Terminology
Service Specification	Swagger File – json swagger url or yaml swagger file url (paths)
Operation (OpId + Arguments + Results)	Client And Server
Invoker And Performer	Requests
Arguments	Responses
Results and Errors	

Notes:

7.3 Adding REST To ROSE

Adding REST to ROSE

Remote Operations Services Element and Representational State Transfer

The basic concepts of ROSE (Remote Operations Services Element) can easily be augmented by the basic concept of REST (Representational State Transfer).

In the Web Services context we can go from Remote Operations to REST's object, method model by introducing the notion of "RO-DestSap" (Remote Operation Destination Service Access Point).

Notes:

8 Outline:

- Universality Of Operations – Local/Direct and Remote
- Unified ICM Model – Direct Operations ICMs (DO-ICM) and Remote Operations ICMs (RO-ICMs)
- Direct ICMs Development Model
- Model Of Remote Operations ICMs
- Remote Invoker ICMs Development Model
- Remote Performer ICMs Development Model

Notes:

Part III

Universality Of Operations – Local/Direct and Remote

9 Parallels Between Command Line Invokations And Remote Operation Invokation

---

Parallels Between Command Line Invokations And Remote Operation Invokation

Options And Arguments Vs Parameters

Command Line:

- Options
- Args

Remote Operations:

- Parameters

Notes:

10 A Unified Model For Python Invokations, Command-Line Invokations And Remote-Op Invokations

---

Python Invokation Inputs: Complex Arguments Python Invokation Outputs: Complex Return Values Command-Line Invokation Inputs: Options And Args Command-Line Invokation Outputs: stdout, stderr Remote-Operation Invokation Inputs: parameters Remote-Operation Outputs: Results, Errors

Python Remote ICM (Interactive Commands Module) Model Transparently Unifies The Three

You just write your python code, the CLI and Remote Operations are fully auto generated.

Notes:

Part IV

The Unified ICM Model – Direct ICMs, Remote ICM Invokers, Remote ICM Performers

11 Benefits And Powers Of The ICM Unified Model

Most of your development life-cycle is in a local and single process environment.

At will you map to command line.

At will you can split the functionality to remote-operations (Web Services).

You can switch between the three models by maintaining a single code base.

Notes:

12 Development Workflow

---

1. Develop Your ICM As An Ordinary Local Command Line Module – With ICM Parameters And Args
2. Use/Test Your ICM On Command Line Or With A ICM-Player
3. Augment Your Local ICM With Swagger Annotations – Similar To Java's @Api
4. Create A Swagger Operations-Specification And Validate It
5. Pass The Swagger Operations-Specification Through codegen which will use the ICM's Operations
6. Run The Performer As A Service
7. Point the ICM-Invoker To The Performer's Operations-Specification
8. Build Your Application Based On The ICM-Invoker Cmnds

Notes:

13 Overview Of Continuity Of Direct, Performer and Invoker ICM Models

---

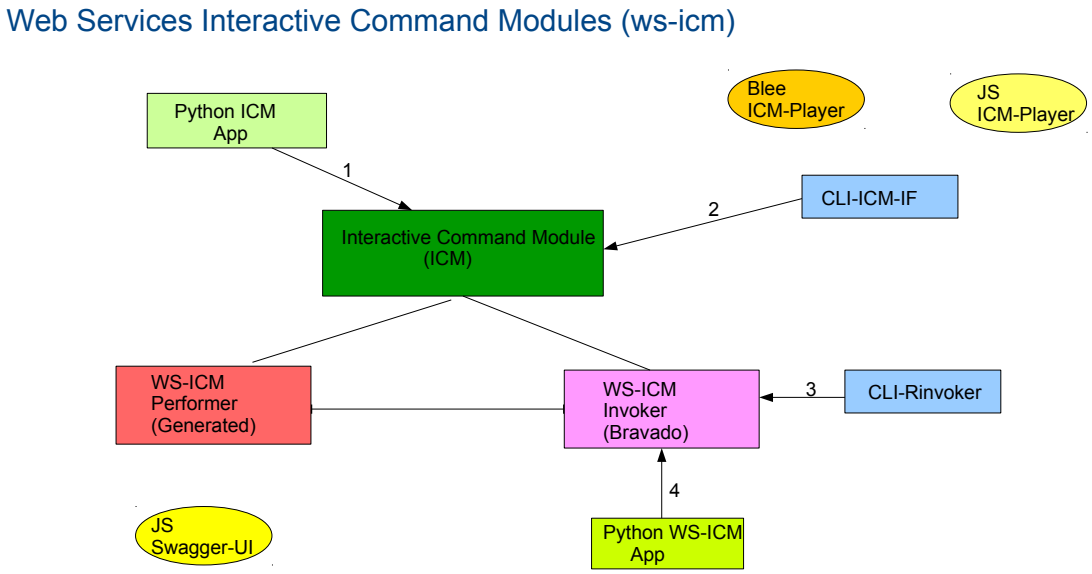

Figure 1: Web Services Interactive Command Module (ws-icm) Overview

---

Notes: Frame Notes

13.1 Direct Operations ICM (DO-ICM) Model

ICM-Commands are directly invoked.

In a single process model where parameters and arguments and results are through the command line and file system.

Notes:

13.2 ICM Performer Model

Based on the ICM's self contained info, ICM-Performers can be launched to respond to Remote Operation Invokations.

The ICM Performer is auto generated from the ICM code.

Notes:

13.3 ICM Invoker Model

Based on the Swagger file, The Remote-Invoker Maps the Swagger file onto command line.

Notes:

Part V

Direct ICMs Development Model

14 Direct ICMs Development Model

Basic Elements Of Direct ICMs

1. Cmnd Method
2. Cmnd Params
3. Cmnd Args
4. Interactivity or Not
5. Cmnd Outcome

Notes:

15 Python ICM-Command Concept Vs Python Functions

The Python ICM-Command Class (icm.Cmnd) includes:

Cmnd.cmnd :

Cmnd.Args: Cmnd.Params:

Cmnd.WebSvcApi:

Full description of ICM is provided in PLPC-180050

Notes:

16 About ICM Players

Based on the ICM's self-contained info, ICM modules can be used at cmnd-line or through auto-generated User-Interfaces.

Notes:

17 About ICM Libraries (Collections Of Reusable ICMs)

ICM “Commands” can be included in ICM-Libraries which can then be combined.

Notes:

Part VI

The Concept Of Remote Operation ICMs

17.1 Overview

Remote Operation ICMs (RO-ICMS) are governed by RO-Specifications (Swagger Spec, OpenApi Spec)

RO-Specifications are the axis around which everything revolves.

RO-Specifications are the "service contract" that permit language bindings for invokers and stable performer evolution.

Notes:

18 Three Ways Of Specifying The RO-Specification (Swagger-file)

---

1. Design and write the Service Specification in full in one place – e.g., with swagger-editor. Then use code-generators for both performer and invoker.
2. Design and write the Service Specification in pieces along with implementration (a la dropwizard). Then publish the aggregated swagger for code-generation of invokers. Performer is framework driven – no generated code.
3. Design and write the Service Specification in pieces along with implementration for single process usage. Then generate the aggregated Service Specification for use with code-generators for both performer and invoker.

Method (3) is that of Web Services ICM.

Notes:

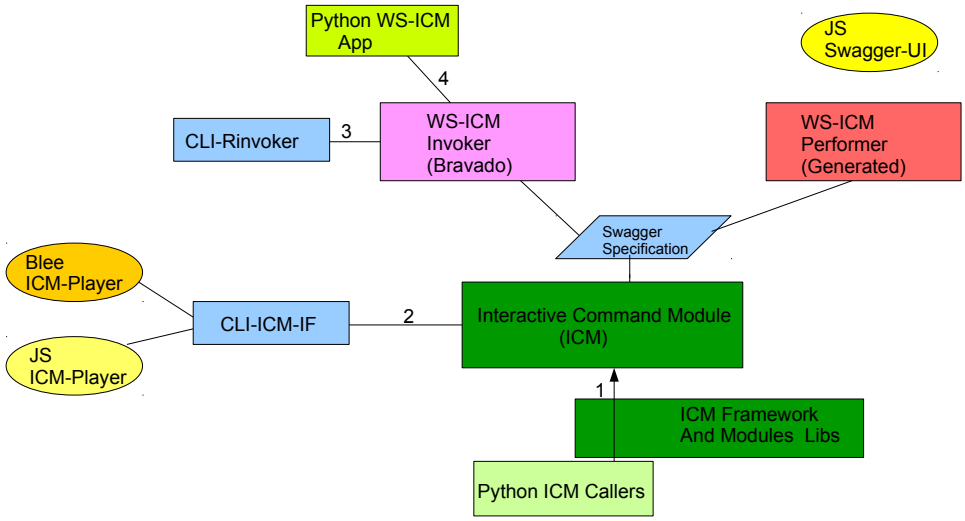

Figure 2: Web Services Interactive Command Module (WS-ICM) Using Swagger Code Generators

---

Notes: Frame Notes

19 RO Authentication And Authorization

---

1. Bearer Tokens
2. Auth Library

Notes:

Part VII

Invokers Development Model

20 A Generalized Swagger (OpenAPI) Centered Web Services And Invokations Testing Framework

---

A Generalized Swagger (OpenAPI) Centered Web Services And Invokations Testing Framework

PLPC-180057

Invokers Development Model is described in:

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

Notes:

Part VIII

Remote Performer ICMs Development Model

21 Model And Process Of Building Performers With Swagger Code Generators

---

Model And Process Of Building Performers With Swagger Code Generators

The general process and model of building a RO-Performer service involves:

1. Obatin Or Create A Swagger File (svcSpec)
2. With The Swagger File, Generate Python-Flask Code (svcSpec -> PerformerCodeGen)
3. Add controllers For Each Operation (PerformerCodeGen + controllerCode)
4. Wrap and run the performer in a web server (Performers+Flask+Connexion+Apache+wsgi) – Other choices for Apache-wsgi include:
    - nginx
    - uWSGI

Notes:

22 Common Ways Of Building ICM Based Performers

Common Ways Of Building ICM Based Performers

1. Custom Performers With DO-ICM Controllers
2. ICM Derived Performers
3. Database Oriented ICM Performers

Notes:

23 Custom Performers With DO-ICM Controllers

Custom Performers With DO-ICM Controllers

For Direct-Operations ICM Apps, The Stack Is:

1. Existing Swagger File (svcSpec)
2. Direct-Ops ICM (doIcm-ops)
3. Swagger Python-Flask Code Generation (svcSpec -> PerformerCodeGen -> doIcm-ops)
4. Flask
5. Connexion
6. Apache-2 wsgi

Notes:

24 ICM Derived Performers

For Direct-Operations ICM Apps, The Stack Is:

1. Direct-Ops ICM (doIcm-ops)
2. DO-ICM Auto Swagger File Generation (doIcm -> svcSpec)
3. Swagger Python-Flask Code Generation (svcSpec -> PerformerCodeGen -> doIcm-ops)
4. Flask
5. Connexion
6. Apache-2 wsgi

Notes:

24.1 Database Oriented ICM Performers

For Database Oriented Apps, The Stack Is:

1. Direct-Ops ICM (DO-ICM)
2. DO-ICM Auto Swagger File Generation (doIcm -> svcSpec)
3. Swagger Python-Flask Code Generation (svcSpec -> PerformerCodeGen -> doIcm-ops)
4. Sqlalchemy
5. Mysql
6. Flask
7. Connexion
8. Apache-2 wsgi

Notes:

25 With The Right Tools, It Is Very Easy To Build RO-ICMs Based On DO-ICMs

---

With The Right Tools, It Is Very Easy To Build RO-ICMs Based On DO-ICMs

1. Build And Test Your DO-ICMs
2. Generate Swagger Files For Your DO-ICMs
3. Convert Your DO-ICMs to Performer-RO-ICMs
4. Use The Invoker-RO-ICMs Framework To Test The Remote Performer
5. Use The Invoker-RO-ICMs Framework Build Your Invoker Apps

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.