Manifest Specification draft for FAIR Containerized Computational Software (FAIR-CCS)

Mylene Simon
Software Engineer & WIPP Lead Developer
Prometheus Computing LLC
National Institute of Standards and Technology (NIST)

2nd International Workshop on FAIR Containerized Computational Software - December 5, 2023
Presentation outline

- Context: the Web Image Processing Pipelines (WIPP) project and plugin manifest
  
- FAIR-CCS manifest
  - General overview
  - Metadata/General information section
  - Inputs and outputs section
  - User Interface (UI) section
  - Resource requirements section
Lowering barriers for accessing, processing and exploring large-scale digital image collections - Key challenges

Big Data Management

- Data sharing and dissemination
  - In the past, hard drives exchanges between users and teams, desktop solutions
  - Need for scalable and interactive visualizations
- Computational scalability
  - Memory-consuming complex algorithms applied to Tera-Byte-sized datasets
  - Limitation of desktop machines, Use of cluster or cloud resources for advanced users

Heterogeneity in Image Processing software and algorithms

- Algorithms come from many libraries and programming languages
- Hardware and OS requirements for installation and execution
- Learning curve for end-user
- How to chain algorithms into complex workflows?

Traceability of data and software-based measurements

- How to link a value to specific dataset, software, algorithm, formula, parameters?
- Managing versions of software/algorithm

FAIR software, DevOps and productivity

- How to make computational tools/software Findable, Accessible, Interoperable and Reusable?
- How to decouple computational tools from execution platforms and facilitate their development, maintenance, integration and deployment?
Current solutions using FAIR-CCS at NIST: WIPP Platform, Plugins, and Plugin Registry

**WIPP Platform**
Open-source web-based algorithmic plugin platform for distributed computations, online data exploration and trusted image-based measurements from terabyte-sized images

**WIPP Plugins**
Interoperable containerized algorithmic plugins associated with a JSON plugin manifest.

Compatible with Docker, Singularity, Kubernetes, Slurm and Common Workflow Language (CWL).

**WIPP Plugin Registry**
Registry for storing, sharing and searching interoperable containerized plugins and computational workflows

Source code and deployment instructions: [https://github.com/usnistgov/WIPP](https://github.com/usnistgov/WIPP)

The WIPP Registry is powered by the NIST Configurable Data Curation System (CDCS) [https://cdcs.nist.gov/](https://cdcs.nist.gov/)
FAIR-CCS manifest: manifest describing inputs, outputs, requirements, UI and general information about a containerized computational software

- Documentation and links to examples:
  - [https://github.com/usnistgov/fair-chain-compute-container](https://github.com/usnistgov/fair-chain-compute-container)

- Online plugin manifest generation and validation tool:
  - [https://usnistgov.github.io/WIPP-Plugin-Manifest-generator/](https://usnistgov.github.io/WIPP-Plugin-Manifest-generator/)

- FAIR-CCS Manifest JSON schema

```json
{  
  "name": "wipp/wipp-simple-python-thresh",  
  "version": "0.0.1",  
  "title": "Simple Python Thresholding",  
  "author": "Mylene Siemens",  
  "instituition": "National Institute of Standards and Technology",  
  "repository": "https://github.com/usnistgov/WIPP-simple",  
  "website": null,  
  "citation": null,  
  "description": "Simple manual thresholding",  
  "containerID": "wipp/wipp-simple-python-thresh:0.0.1",  
  "inputs": [  
    {  
      "name": "inputImages",  
      "type": "collection",  
      "required": true,  
      "description": "Input images"  
    },  
    {  
      "name": "threshold",  
      "type": "number",  
      "required": true,  
      "description": "Threshold value"  
    }  
  ],  
  "outputs": [  
    {  
      "name": "output",  
      "type": "collection",  
      "description": "Output images"  
    }  
  ],  
  "ui": [  
    {  
      "key": "inputs.inputImages",  
      "title": "Images collection",  
      "description": "Pick a collection..."  
    },  
    {  
      "key": "inputs.threshold",  
      "title": "Threshold value: "  
    }  
  ]
}
```

Information about the software:
- name, version, description
- Docker image to use

Inputs description
- name, type, description

Outputs description
- name, type, description

UI description
- additional information to display on the form

Resource requirements
- optional resource/hardware requirements
General information about the containerized software

- **Required:** name, version, title, description, containerId
- **Optional:** institution, repository, website, citation, baseCommand

```json
{
    "name": "wipp/plugin-name",
    "version": "0.0.1",
    "containerId": "docker.io/wipp/plugin-name:0.1.0",
    "baseCommand": ["python3", "/opt/executable/main.py"],
    "title": "Example plugin",
    "description": "Example plugin description,
}
```

```json
"name": "Simple Python Thresholding Plugin",
"version": "1.0.0",
"title": "Simple Python Thresholding",
"author": "Mylene Simon",
"institution": "National Institute of Standards and Technology",
"repository": "https://github.com/usnistgov/WIPP-simple-python-thresh-plugin",
"website": null,
"citation": null,
"description": "Simple manual thresholding",
"containerId": "wipp/wipp-simple-python-thresh:0.0.1",
```
Description of inputs

Required properties:
- "name", "type", "description"

Optional properties:
- "required", "options"

Supported data types:
- "collection" (collection of Images),
- "stitchingVector" (MIST format),
- "pyramid" (DeepZoom format),
- "tensorflowModel" (AI model),
- "csvCollection" (collection of CSV files),
- "genericData" (other data),
- "notebook" (Jupyter notebook),
- "string", "number", "integer", "enum", "array", "boolean"
Description of outputs

Required properties:

“name”, “type”, “description”

Supported data types:

"collection" (collection of Images),
"stitchingVector" (MIST format),
"pyramid” (DeepZoomWeb format),
"tensorflowModel” (AI model),
“tensorboardLogs” (logs for Tensorboard visualization),
"csvCollection” (collection of CSV images),
“genericData” (other data)
Description of UI

One UI description per input

“key”: inputs.inputName (required)
“title”: input field label in form (required)
“description”: placeholder for string input field (optional)

Advanced options not shown here (conditional visibility, binds between fields, etc.)

```
"ui": [
  {
    "key": "inputs.input",
    "title": "Images collection: ",
    "description": "Pick a collection..."
  },
  {
    "key": "inputs.threshold",
    "title": "Threshold value: "
  }
]
```
Description of UI
Example of more advanced UI options
(conditional visibility and fieldsets)

Conditional visibility of fields

```json
{
  "key": "inputs.startTile",
  "title": "Start Tile: ",
  "description": "Specify the index of the first tile (0 or 1)",
  "condition": "model.inputs.filenamePatternType=='SEQUENTIAL'"
},
{
  "key": "inputs.startTileRow",
  "title": "Start Tile Row: ",
  "description": "Specify the index of the first tile row (0 or 1)",
  "condition": "model.inputs.filenamePatternType=='ROWCOL'"
}
```

Fieldsets

```json
{
  "key": "fieldsets",
  "fieldsets": [
    {
      "title": "Input images",
      "fields": [
        "assembleNoOverlap",
        "imageDir",
        "filenamePatternType",
        "filenamePattern"
      ]
    },
    {
      "title": "Acquisition setup",
      "fields": [
        "gridOrigin",
        "numberingPattern",
        "gridWidth",
        "gridHeight",
        "startTile",
        "startTileRow",
        "isTimeSlices",
        "timeSlices"
      ]
    }
  ]
}
```
FAIR-CCS manifest: Resource requirements section

Resource/hardware requirements

Optional

“ramMin”: minimum RAM in Mebibytes (Mi)
“coresMin”: minimum number of CPU cores
“cpuAVX”: Advanced Vector Extensions (AVX) CPU capability required
“cpuAVX2”: Advanced Vector Extensions (AVX) CPU capability required
“gpu”: GPU/accelerator required
“cudaRequirements”: GPU Cuda-related requirements
    “deviceMemoryMin”: minimum device memory
    “cudaComputeCapabilities”: single minimum value or array of valid values for required compute capability

```
"resourceRequirements": {
    "ram": 2048,
    "coresMin": 1,
    "cpuAVX": true,
    "cpuAVX2": false,
    "gpu": true,
    "cudaRequirements": {
        "deviceMemoryMin": 100,
        "cudaComputeCapability": "8.0"
    }
}
```
FAIR-CCS Github repository:
https://github.com/usnistgov/fair-chain-compute-container

Questions?
wipp-team@nist.gov