Open Recon JSON Schema

(Version Mar 22nd 2024)

An Open Recon app is configured using a JSON-formatted descriptor stored in the Docker image's metadata. This configuration specifies how the app interfaces with the product reconstruction and the end user interface presented on the scanner. It is organized in 3 sections:

1. General configuration

The `general` section defines general information about the app including name, version, and description.

An example of the minimum parameters required for this section is:

```
"general": {
  "name": { "en": "PythonMRD" },
  "version": "1.0.1",
  "vendor": "Siemens Healthineers AG",
  "information": { "en": "Demo Python MRD server." },
  "id": "PythonMRD",
  "regulatory_information":{
    "device_trade_name":"PythonMRD",
    "production_identifier":"1.0.1",
    "manufacturer_address":"Erlangen, Germany",
    "made_in":"USA",
    "manufacture_date":"2023/02/14",
    "material_number":"PythonMRD_2.0.0",
    "gtin":"00860000171212",
    "udi":"(01)00860000171212(21)1.3.0",
    "safety_advices":"",
    "special_operating_instructions":"Demo Python MRD server supporting raw and
image workflows",
    "additional_relevant_information":""
  }
}
```

The fields are specified as:

- `name`: A multilingual user-facing string that is the name for the app. An English (`en`) string property is required and other languages are optional.
- `version`: Version string following the pattern `major.minor.patch`, e.g. `1.0.1` (semantic versioning). Type: string
- `vendor`: String identifing the vendor name responsible for the app. Type: string

- `information`: User-facing description of the app. An English (`en`) string property is required and other languages are optional.
- `id`: Internal identifier string for the app (invariant to language). Type: string
- `regulatory_information`: An object containing vendor-specific regulatory information with the following sub-properties:
  - `device_trade_name`: Trade name of the app. Type: string
  - `production_identifier`: Internal production identifier (e.g. version string). Type: string
  - `made_in`: Country of manufacture. Type: string
  - `manufacture_date`: Date of manufacture. Type: string
  - `material_number`: Vendor-defined material number. Type: string
  - `gtin`: Global Trade Item Number. Type: string
  - `udi`: Unique Device Identifier. Type: string
  - `safety_advices`: Safety relevant information for usage of the app. Type: string
  - `special_operating_instructions`: Operating instructions for the app. Type: string
  - `additional_relevant_information`: Additional information about the app. Type: string

2. Reconstruction configuration

The `reconstruction` section defines how the app interfaces with the product reconstruction, such as defining the emitter/injector types.

An example of the minimum parameters required for this section is:

```
"reconstruction": {
  "transfer_protocol": {
    "protocol": "ISMRMRD",
    "version": "1.4.1"
  },
  "port": 9002,
  "emitter": "image",
  "injector": "image",
  "can_process_adjustment_data": false,
  "can_use_gpu": false,
  "min_count_required_gpus": 0,
  "min_required_gpu_memory": 0,
  "min_required_memory": 512,
  "min_count_required_cpu_cores": 1,
  "content_qualification_type": "PRODUCT"
}
```

The fields are specified as:

- `transfer_protocol`: An object containing the following sub-properties describing the communications protocol:
  - `protocol`: The network transfer protocol. Currently limited to `ISMRMRD`, i.e. the MRD Session Protocol. Type: enum string
  - `version`: Version number of the transfer protocol. Currently limited to version `1.4.1`. Type: enum string

- `port`: TCP port that the app will listen on. Currently limited to port `9002`. Type: integer
- `emitter`: Type of data that is sent to the app from the Siemens ICE pipeline. Can be `raw` for "raw to complex image" workflows or `image` for "image to image" workflows. Type: enum string
- `injector`: Type of data that the app will send back to the Siemens ICE pipeline. Can be `complexiimage` for "raw to complex image" workflows or `image` for "image to image" workflows. Type: enum string
- `can_process_adjustment_data`: Optional parameter to send adjustment data (e.g. noise) to the app in a separate MRD session. Default: false. Type: optional boolean
- `can_use_gpu`: Set `true` if the app can make use of NVIDIA GPUs on the Siemens reconstruction computer (MARS). Type: boolean
- `min_count_required_gpus`: The minimum number of GPUs required to be present in the system. Default: 0. Type: integer
- `min_required_gpu_memory`: Optional parameter for the minimum amount of GPU memory in MB. Default: 0. Type: integer
- `min_count_required_cpu_cores`: Minimum number of CPU cores on the Siemens reconstruction computer. Minimum: 1. Type: integer
- `min_required_memory`: Minimum amount of system memory on the Siemens reconstruction computer in megabytes. Type: integer
- `content_qualification_type`: Specifies the qualification of data returned by the app, corresponding to the DICOM tag Content Qualification, (0018,9004). Values may be `PRODUCT`, `SERVICE`, or `RESEARCH`. A value of `PRODUCT` indicates that the resulting data is a released medical product and is subject to additional checks. Default: `RESEARCH`. Type: enum string

3. User interface configuration

The `parameters` section is used to define the user interface on the Open Recon card when running a sequence on the scanner. It is an array of up to 14 UI parameters that are editable by the scan operator. This section can also be blank if no UI parameters are to be presented to the user. There are 5 basic types of UI parameters -- integers, doubles, strings, booleans, and enumerations.

Each UI parameters has 3 required properties:

- `type`: An enum string describing its type. Must be one of `int`, `double`, `string`, `boolean`, `choice`. Type: enum string
- `id`: An alphanumeric identifier for the parameter to be used by the app. Must contain only upper case or lower case letters or numbers (no spaces or special characters). Type: string
- `label`: A multilingual user-facing string for the UI parameter. An English (`en`) string property is required and other languages are optional.

Booleans

The UI parameter type `boolean` is a boolean. It has the required properties:

- `type`: An enum string describing its type. Must be `boolean`. Type: enum string
- `id`: An alphanumeric identifier for the parameter to be used by the app. Must contain only upper case or lower case letters or numbers (no spaces or special characters). Type: string
- `label`: A multilingual user-facing string for the UI parameter. An English (`en`) string property is required and other languages are optional.
- `default`: The default value without user input. Values may be either `true` or `false`. Type: boolean

- `information`: An optional multilingual user-facing string displayed as a tooltip when the mouse hovers over the parameter. An English (`en`) string property is required and other languages are optional.

An example of a boolean parameter is:

```
{
  "id": "checkbox",
  "label": { "en": "My Checkbox" },
  "type": "boolean",
  "information": { "en": "Tooltip for a boolean parameter" },
  "default": true
}
```

Strings

The UI parameter type `string` is a character string. It has the required properties:

- `type`: An enum string describing its type. Must be `string`. Type: enum string
- `id`: An alphanumeric identifier for the parameter to be used by the app. Must contain only upper case or lower case letters or numbers (no spaces or special characters). Type: string
- `label`: A multilingual user-facing string for the UI parameter. An English (`en`) string property is required and other languages are optional.
- `default`: The optional default value without user input. If not specified, the default value is a blank string. Type: string
- `information`: An optional multilingual user-facing string displayed as a tooltip when the mouse hovers over the parameter. An English (`en`) string property is required and other languages are optional.

An example of a string parameter is:

```
{
  "id": "text",
  "label": { "en": "My Text" },
  "type": "string",
  "information": { "en": "Tooltip for a string parameter" },
  "default": ""
}
```

Enumerated Lists

The UI parameter type `choice` is an enumerated list of strings presented as a drop-down menu. It has the required properties:

- `type`: An enum string describing its type. Must be `choice`. Type: enum string
- `id`: An alphanumeric identifier for the parameter to be used by the app. Must contain only upper case or lower case letters or numbers (no spaces or special characters). Type: string
- `label`: A multilingual user-facing string for the UI parameter. An English (`en`) string property is required and other languages are optional.

- `values`: An array of the possible values for the parameter. Each value must have two subproperties:
  - `id`: An alphanumeric identifier for the value. Must contain only upper case or lower case letters or numbers (no spaces or special characters). Type: string
  - `name`: A multilingual user-facing string for the UI parameter. An English (`en`) string property is required and other languages are optional.
- `default`: The optional default value without user input -- must match one of the `values`. Type: string
- `information`: An optional multilingual user-facing string displayed as a tooltip when the mouse hovers over the parameter. An English (`en`) string property is required and other languages are optional.

An example of a choice parameter is:

```
{
  "id": "enum",
  "label": { "en": "My Enum" },
  "type": "choice",
  "values": [
    {
      "id": "choice1",
      "name": { "en": "Choice 1" }
    },
    {
      "id": "choice2",
      "name": { "en": "Choice 2" }
    },
  ],
  "default": "choice1",
  "information": { "en": "Tooltip for a choice parameter" }
}
```

Integers

The UI parameter type `int` is a 32-bit integer. It has the required properties:

- `type`: An enum string describing its type. Must be `int`. Type: enum string
- `id`: An alphanumeric identifier for the parameter to be used by the app. Must contain only upper case or lower case letters or numbers (no spaces or special characters). Type: string
- `label`: A multilingual user-facing string for the UI parameter. An English (`en`) string property is required and other languages are optional.
- `minimum`: The minimum allowed value -- must be larger than -2147483648 (the minimum for int32). Type: integer
- `maximum`: The maximum allowed value -- must be smaller than 2147483647 (the maximum for int32). Type: integer
- `default`: The default value without user input. Type: integer
- `unit`: An optional multilingual user-facing string for the parameter's units, e.g. "ms". An English (`en`) string property is required and other languages are optional. If absent, no units are displayed on the UI. If specified, must be between 1-3 characters.
- `information`: An optional multilingual user-facing string displayed as a tooltip when the mouse hovers over the parameter. An English (`en`) string property is required and other languages are optional.

An example of an integer parameter is:

```
{
  "id": "integer",
  "label": { "en": "My Integer" },
  "type": "int",
  "information": { "en": "Tooltip for an int parameter" },
  "minimum": 0,
  "maximum": 100,
  "default": 2
}
```

Doubles

The UI parameter type `double` is a 64-bit floating point value. It has the required properties:

- `type`: An enum string describing its type. Must be `double`. Type: enum string
- `id`: An alphanumeric identifier for the parameter to be used by the app. Must contain only upper case or lower case letters or numbers (no spaces or special characters). Type: string
- `label`: A multilingual user-facing string for the UI parameter. An English (`en`) string property is required and other languages are optional.
- `minimum`: The minimum allowed value. It must be larger than -2.225e-308 (the minimum for float32) and be a multiple of 0.1. Type: double
- `maximum`: The maximum allowed value -- must be larger than 1.798e308 (the minimum for float32) and be a multiple of 0.1. Type: double
- `default`: The default value without user input -- must be a multiple of 0.1. Type: double
- `unit`: An optional multilingual user-facing string for the parameter's units, e.g. "ms". An English (`en`) string property is required and other languages are optional. If absent, no units are displayed on the UI. If specified, must be between 1-3 characters.
- `information`: An optional multilingual user-facing string displayed as a tooltip when the mouse hovers over the parameter. An English (`en`) string property is required and other languages are optional.

An example of a double parameter is:

```
{
  "id": "double",
  "label": { "en": "My Double" },
  "type": "double",
  "information": { "en": "Tooltip for an double parameter" },
  "minimum": 0,
  "maximum": 100,
  "default": 0.2
}
```

Specifing the config for a Docker image

The Open Recon JSON configuration for an Open Recon app is specified in its Docker metadata. The JSON config text is base64-encoded and stored in the Docker image metadata label `com.siemens-healthineers.magneticresonance.OpenRecon.metadata:1.1.0`. This can be done by adding a `LABEL` command to the Dockerfile:

```
LABEL com.siemens-
healthineers.magneticresonance.OpenRecon.metadata:1.1.0=base64encodedjson
```

where `base64encodedjson` is the base64-encoded json text.

The validity of the JSON config should be tested prior to deployment on the scanner. The `OpenReconSchema_1.1.0.json` file can be used for validation using standard JSON validation tools. The `CreateORDockerImage.ipynb` Python Notebook can also be used to validate a JSON config, create a Docker image with the appropriate label, and package it in an OpenRecon .zip format.

Disclaimer

Open Recon is to add clinical reconstructions to the system, if signed and released for clinical use by Siemens Healthineers. Any other recon used e.g., by researchers is automatically labelled not for diagnostic use, which may require observation of national regulations.