Seed Standard Definition

Jonathan Meyer, Mike Holt, Derick Faller, John Tobe

Version 1.0.0-snapshot

Table of Contents

1. Introduction

Seed is a general standard to aid in the discovery and consumption of a discrete unit of work contained within a Docker image. While initially developed to support the Scale data processing system with job discovery, it is designed to be readily applied to other systems as well.

Seed compliant images must be named in a specific fashion due to the lack of label search capability on Docker Hub and Registry services. The suffix `-seed` must be used when naming images to enable discovery, prior to Hub or Registry push. This requirement will be deprecated as label search support is standardized across Docker registry services. Use of the CLI developed by the Seed team is highly recommended to speed the development and packaging of jobs according to the Seed specification.

1.1. Format

The Docker image created must adhere to a specific naming convention. The standard requires specification of both an job and a packaging version, tracking changes individually between the job logic and the packaging of it. The following image naming template maps to members defined under the Job Objects:

```
<name>-<jobVersion>-seed:<packageVersion>
```

Dockerfile snippet containing required label for Seed compliance:

```
FROM alpine

ENTRYPOINT /app/job.sh

LABEL com.ngageoint.seed.manifest="{\"seedVersion\":\"1.0.0-snapshot\",\"job\":{ ...
}}"
```

The `com.ngageoint.seed.manifest` label contents must be serialized as a string-escaped JSON object. The following is a complete example including required and optional keys:

```
{
  "seedVersion": "1.0.0-snapshot",
  "job": {
    "name": "my-job",
    "jobVersion": "1.0.0",
    "packageVersion": "1.0.0",
    "title": "My first job",
    "description": "Reads an HDF5 file and outputs two png images, a CSV and manifest
containing cell_count",
    "tags": [
      "hdf5",
      "png",
      "csv",
      "image processing"
    ],
    "maintainer": {
```

```
    "name": "John Doe",
    "organization": "E-corp",
    "email": "jdoe@example.com",
    "url": "http://www.example.com",
    "phone": "666-555-4321"
  },
  "timeout": 3600,
  "interface": {
    "command": "${INPUT_FILE} ${OUTPUT_DIR} ${VERSION}",
    "inputs": {
      "files": [
        {
          "name": "INPUT_FILE",
          "required": true,
          "mediaTypes": [
            "image/x-hdf5-image"
          ],
          "partial": true
        }
      ],
      "json": [
        {
          "name": "INPUT_JSON",
          "type": "string",
          "required": true
        }
      ]
    },
    "outputs": {
      "files": [
        {
          "name": "output_file_pngs",
          "mediaType": "image/png",
          "multiple": true,
          "pattern": "outfile*.png"
        },
        {
          "name": "output_file_csv",
          "mediaType": "text/csv",
          "pattern": "outfile*.csv",
          "required": false
        }
      ],
      "json": [
        {
          "name": "cell_count",
          "key": "cellCount",
          "type": "integer"
        },
        {
          "name": "dummy",
          "type": "integer",
          "required": false
```

```
        }
      ]
    },
    "mounts": [
      {
        "name": "MOUNT_PATH",
        "path": "/the/container/path",
        "mode": "ro"
      },
      {
        "name": "WRITE_PATH",
        "path": "/write",
        "mode": "rw"
      }
    ],
    "settings": [
      {
        "name": "VERSION",
        "secret": false
      },
      {
        "name": "DB_HOST",
        "secret": false
      },
      {
        "name": "DB_PASS",
        "secret": true
      }
    ]
  },
  "resources": {
    "scalar": [
      {
        "name": "cpus",
        "value": 1
      },
      {
        "name": "mem",
        "value": 1024
      },
      {
        "name": "sharedMem",
        "value": 1024
      },
      {
        "name": "disk",
        "value": 1000,
        "inputMultiplier": 4
      }
    ]
  },
  "errors": [
    {
```

```
        "code": 1,
        "name": "error-name-one",
        "title": "Error Name",
        "description": "Error Description",
        "category": "data"
      },
      {
        "code": 2,
        "name": "error-name-two",
        "title": "Error Name",
        "description": "Error Description",
        "category": "job"
      }
    ]
  }
}
```

1.2. Definitions

- Seed specific terms defined in the Glossary supersede all following definitions. These terms can be found *italicized* throughout the specification.
- GeoJSON, and the terms Geometry and Polygon are defined in RFC 7946 GeoJSON
- Internet Assigned Numbers Authority (IANA), and the terms Media Types and MIME Types are defined in IETF RFC 6838
- ISO 8601 and the specifics of the date format are defined in IETF RFC 3339
- JavaScript Object Notation (JSON), and the terms object, name, value, array, integer, and number, are defined in JSON Schema.
- Semantic Versioning (SemVer), and the terms major, minor, and patch version are defined at http://semver.org/spec/v2.0.0.html
- The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in IETF RFC 2119.

2. Standard

The Seed standard is intended to provide a complete definition of the runtime processing, memory and storage requirements of a discrete unit of work, in addition to the inputs, outputs and potential errors produced. Completeness is fundamental but the standard accommodates both simple and complex jobs by defining a minimal subset of REQUIRED properties. The following sections detail every possible REQUIRED and OPTIONAL manifest property in both root and child objects.

A complete Seed object contained within a `com.ngageoint.seed.manifest` label is always a string-escaped serialized object. In Seed, an object consists of a collection of name/value pairs – also called members. For each member, the name is always a string. Member values are either a string, number, object, array or one of the literals: true, false, and null. An array consists of elements where each element is a value as described above.

2.1. Seed Object

The Seed object is the root JSON object that MUST be placed within a `com.ngageoint.seed.manifest` Docker image label. At a minimum this object MUST define the `seedVersion` and `job` names.

Name	Requirement	Value
`seedVersion`	Required	MUST be a string that conforms to the SemVer standard.
`job`	Required	MUST be a Job Objects.

```
{
  "seedVersion": "1.0.0-snapshot",
  "job": { ... }
}
```

2.1.1. Job Object

The Job object is the core member for describing a single unit of work and the resources it requires.

Name	Requirement	Value
`name`	Required	MUST be a string of only alphanumeric or dash characters (defined by the regex† `^[a-zA-Z0-9-]+$`).
`jobVersion`	Required	MUST be a string that conforms to the SemVer standard.
`packageVersion`	Required	MUST be a string that conforms to the SemVer standard.
`title`	Required	MUST be a string and SHOULD contain a short descriptive title of the job.
`description`	Required	MUST be a string and SHOULD contain a full job abstract.
`tags`	Optional	MUST be an array of strings and MAY contain any number of elements.
`timeout`	Required	MUST be a integer indicating a timeout period measured in seconds. Consuming systems MUST honor this value as a hard limit on job execution time.
`maintainer`	Required	MUST be an object as defined in Maintainer Object.
`resources`	Recommended	MUST be an object as defined in Resources Object. It is highly advised that this member be specified, without it resources provided will be default for the implementing framework.
`interface`	Optional	MUST be an object as defined in Interface Object.
`errors`	Optional	MUST be an array containing elements defined in Errors Objects
The following annotated snippet provides quick reference to the use of Job object:		

Name	Requirement	Value

```
{
    "name": "my-job", ①
    "jobVersion": "1.0.0", ②
    "packageVersion": "1.0.0", ③
    "title": "My first job", ④
    "description": "Reads an HDF5 file and outputs two TIFF images, a CSV and
manifest containing cell_count", ⑤
    "timeout": 3600,⑥
    "maintainer": { ... }, ⑦
    "resources": { ... }, ⑧
    "interface": { ... }, ⑨
    "errors": [ ... ] ⑩
}
```

① Required string containing job identifier. `name` and `jobVersion` members combined should be unique system-wide.

② Required string containing version identifier of job in SemVer format. `name` and `jobVersion` members combined should be unique system-wide.

③ Required string containing packaging version identifier in SemVer format. `packageVersion` is used to indicate updates to the job interface, it should NEVER be used to indicate changes to the job.

④ Required string containing short job title.

⑤ Required string containing job abstract. Inline markup should be avoided, but not prohibited.

⑥ Required integer containing job timeout value in seconds.

⑦ Required Maintainer Object.

⑧ Optional Resources Object.

⑨ Optional Interface Object.

⑩ Optional array of Errors Objects.

2.1.1.1. Maintainer Object

The Maintainer object is the member that identifies the individual and organization (optional) acting as a point of contact for a Seed job.

Name	Requirement	Value
`name`	Required	MUST be a string and SHOULD contain the full name of maintaining individual.
`email`	Required	MUST be a string and SHOULD contain the best contact email for maintaining individual or organization.
`phone`	Optional	MUST be a string and SHOULD contain the best contact phone number for maintaining individual or organization.
`organization`	Optional	MUST be a string and SHOULD contain the organization responsible for maintaining or sponsoring Seed job.
`url`	Optional	MUST be a string and SHOULD contain a publicly accessible URL to complete job design or usage documentation.

Name	Requirement	Value
The following annotated snippet provides quick reference to the use of Maintainer object:		

```
{
    "name": "John Doe", ①
    "email": "jdoe@example.com", ②
    "phone": "666-555-4321", ③
    "organization": "E-corp", ④
    "url": "http://www.example.com" ⑤
}
```

① Required string containing full name of maintaining individual.

② Required string containing best contact email for maintaining individual or organization.

③ Optional string containing best contact phone number for maintaining individual or organization.

④ Optional string containing organization responsible for maintaining or sponsoring Seed job.

⑤ Optional string containing publicly accessible URL to complete job design or usage documentation.

2.1.1.2. Resources Object

The Resources object is the member that identifies all resource requirements for a job. This is most commonly CPU, memory and disk scalar resources, but MAY in the future accommodate more complex types such as ranges and sets. The final computed resources allocated for all `scalar` elements MUST be injected as environment variables to the job at run time. Reference Environment Variables and Resource Defaults for clarification on what the implementing framework MUST provide.

Name	Requirement	Value
`scalar`	Required	MUST be an array of `Scalar` objects and MAY contain any number of elements. There is no other standard restriction on the array size.

Scalar Object

The Scalar objects MAY include any arbitrary custom resource name, but there are reserved resources `cpus`, `disk`, `mem` and `sharedMem` that have special meaning. The reserved resource names `cpus`, `disk` and `mem` SHOULD be populated by all Seed compliant images, as the defaults provided at runtime will likely be inadequate to run all but the most minimal job. The `sharedMem` resource applies primarily to high-performance and scientific applications and will rarely be needed.

Name	Requirement	Value
`name`	Required	MUST be a string of only alphanumeric, dash or underscore characters (defined by the regex† `^[a-zA-Z0-9_-]+$`) indicating the resource required by the job. Refer to Injection for details on environment variable available at execution time.
`value`	Required	MUST be a number indicating the quantity of the resource required by the job. When dealing with storage resources such as `mem` or `disk` units of Mebibytes (MiB) MUST be used.
`inputMultiplier`	Optional	MUST be a number indicating the factor by which input file size is multiplied and added to the constant value for resource.

Use of `inputMultiplier` for `mem` or `disk` resource types is useful when memory or output disk requirements of a job are a function of input file size. The following basic formula computes the resource requirement when an `inputMultiplier` is defined.

```
resourceRequirement = inputVolume * inputMultiplier + constantValue
```

For example, when total input file size is 2.0MiB and an `inputMultiplier` of `4.0` and `value` of `0.1` is specified for `disk`, the following computes the resource requirement:

```
diskRequirement = 2.0MiB * 4.0 + 0.1MiB
```

The following annotated snippet provides quick reference to the use of Scalar object:

```
[
    { "name": "cpus", "value": 1.0 }, ①
    { "name": "disk", "value": 4.0, "inputMultiplier": 4.0 }, ②
    { "name": "mem", "value": 64.0, "inputMultiplier": 4.0 }, ③
    ... ④
]
```

① Recommended Scalar object demonstrating single constant scalar value for specifying CPU requirement of job.

② Optional Scalar object demonstrating single constant scalar value in addition to a multiplier based on total input file size for scaling disk requirement of job. This multiplier allows for scaling the output disk space required as a function of input file size.

③ Recommended Scalar object demonstrating single constant scalar value in addition to a multiplier based on total input file size for scaling memory requirement of job.

④ Optional additional Scalar objects for any custom resources needed by job.

2.1.1.3. Interface Object

The Interface object is the primary member that describes the command arguments, environment variables, mounts, settings, inputs and outputs defined for a job.

Name	Requirement	Value
`command`	Optional	MUST be a string specifying the complete string passed to the container at run time. Based on the Linux shell, shell escaping of special characters MAY be required. If a Docker ENTRYPOINT is defined that launches the executable, omission of the executable MAY be necessary in `command` string. The Seed `command` member can be treated as analogous to the Docker CMD statement.
`inputs`	Optional	MUST be an object as defined in Inputs Object.
`outputs`	Optional	MUST be an object as defined in Outputs Object.
`mounts`	Optional	MUST be an array of `Mounts` objects (see Mounts Object) and MAY contain any number of elements. There is no other standard restriction on the array size.

Name	Requirement	Value
`settings`	Optional	MUST be an array of `Settings` objects (see Settings Object) and MAY contain any number of elements. There is no other standard restriction on the array size.

2.1.1.3.1. Inputs Object

The Inputs object is the member responsible for indicating immutable input data available to the Seed image at runtime.

Name	Requirement	Value
`files`	Optional	MUST be an array of objects defined in the Files Object sub-section.
`json`	Optional	MUST be an array of objects defined in the JSON Object sub-section.

Files Object

Critical implementation details related to `multiple` member should be referenced in environment variables. The following table defines the `files` object members.

Name	Requirement	Value
`name`	Required	MUST be a string of only alphanumeric, dash or underscore characters (defined by the regex† `^[a-zA-Z0-9_-]+$`) indicating the unique name to use for referring to this input. All inputs will be passed as environment variables, with the environment variable names based upon the input names. Refer to Injection for details on environment variables available at execution time.
`mediaTypes`	Optional	MUST be an array of strings that MUST indicate the IANA Media types that the job accepts. An executor† MAY use any provided media types to report validation warnings to the user in the case of mismatched types.
`multiple`	Optional	MUST be a boolean indicating whether multiple physical files are processed by this `Files` element. If omitted, the default value MUST be treated as false. If true, the value injected into the `command` placeholder will be an absolute directory containing all files for this input. If false or omitted, the value injected into the `command` placeholder will be an absolute path to a single file.
`partial`	Optional	MUST be a boolean indicating whether input file is required in whole or in part. This allows an executor† to make intelligent choices when providing very large files to a job. This should only be set to true if the job is expected to use less than half of very large input files. If omitted, the default value MUST be treated as false.
`required`	Optional	MUST be a boolean indicating whether this input value SHOULD always be expected. If omitted, the default value MUST be treated as true.

JSON Object

The following table defines the `json` object members.

Name	Requirement	Value
`name`	Required	MUST be a string of only alphanumeric, dash or underscore characters (defined by the regex† `^[a-zA-Z0-9_-]+$`) indicating the unique name to use for referring to this input. All inputs will be passed as environment variables, with the environment variable names based upon the input names. Refer to Injection for details on environment variables available at execution time.
`type`	Required	MUST be a string and indicate a valid JSON schema type.
`required`	Optional	MUST be a boolean indicating whether this input value SHOULD always be expected. If omitted, the default value MUST be treated as true.

The following annotated snippet provides quick reference to the use of Inputs object:

```
{
    "files": [ ①
        {
            "name": "INPUT_FILE", ②
            "mediaTypes": [ "image/x-hdf5-image" ], ③
            "multiple": false, ④
            "partial": true, ⑤
            "required": true ⑥
        },
        ...
    ]
    "json": [ ⑦
        {
            "name": "INPUT_STRING",⑧
            "type": "string", ⑨
            "required": false ⑩
        }
    ]
}
```

① Optional array containing elements defined by Files Object sub-section.

② Required string containing unique name used to refer to this input.

③ Optional array containing a list of accepted media types.

④ Optional boolean indicating whether this element represents multiple files (flat directory) vs one file (false). Default is `false`.

⑤ Optional boolean indicating that a job consumes only a small portion of input file. Default is `false`.

⑥ Optional boolean indicating whether job requires this particular file. Default is `true`.

⑦ Optional array containing elements defined by JSON Object sub-section.

⑧ Required string containing unique name used to refer to this input.

⑨ Required string containing a valid JSON schema type for input validation.

⑩ Optional boolean indicating whether job requires this particular JSON input. Default is `true`.

2.1.1.3.2. Outputs Object

The Outputs object is the member responsible for indicating all output data and the means to capture that data following the execution of a Seed image. Data can be captured in two different forms: directly as a file or extracted JSON from a manifest. File type output is simply matched based on a standard glob pattern. Recursively scanning directories is NOT supported, but a known subdirectory structure will work (e.g. base/sub-**/**.ext). JSON objects are expected to be gathered from a JSON manifest that by Seed standard convention MUST be written at the root of the job output directory as `seed.outputs.json`. The absolute path to the job output directory is REQUIRED to be passed into the container at job execution time in the `OUTPUT_DIR` environment variable. Special attention should be given to output file permissions and support is provided for defining extended metadata.

Name	Requirement	Value
`files`	Optional	MUST be an array of objects defined in the Files Object sub-section.
`json`	Optional	MUST be an array of objects defined in the JSON Object sub-section.

Files Object

The following table defines the `files` object members.

Name	Requirement	Value
`name`	Required	MUST be a string of only alphanumeric, dash or underscore characters (defined by the regex† `^[a-zA-Z0-9_-]+$`) indicating the unique name to use for referring to this output.
`mediaType`	Optional	MUST indicate the IANA Media type for the file being captured by Outputs.
`pattern`	Required	MUST indicate a standard glob pattern for the capture of files.
`multiple`	Optional	MUST be a boolean indicating whether multiple output files may be captured by this `Files` element. If omitted, the default value MUST be treated as false.
`required`	Optional	MUST be a boolean indicating whether this input value SHOULD always be expected. If omitted, the default value is `true`.

JSON Object

The following table defines the `json` object members.

Name	Requirement	Value
`name`	Required	MUST be a string of only alphanumeric, dash or underscore characters (defined by the regex† `^[a-zA-Z0-9_-]+$`) indicating the unique name to use for referring to this output. When `key` member is omitted, it must be a case-sensitive match of the member key in `seed.outputs.json` file.
`type`	Required	MUST be a string indicating the JSON schema type of the member being captured from the `seed.outputs.json` file.
`key`	Optional	MUST be a string indicating the case-sensitive `seed.outputs.json` member to capture. If omitted, the member key is assumed to be a case-sensitive match for the above defined `name` member.
`required`	Optional	MUST be a boolean indicating whether this input value SHOULD always be expected. If omitted, the default value MUST be treated as true.

The following annotated snippets provides quick reference to the use of Outputs object:

Seed `outputs` object snippet:

```
{
    "files": [ ①
        {
            "name": "OUTPUT_TIFFS", ②
            "mediaType": "image/tiff", ③
            "pattern": "outfile*.tif", ④
            "multiple": false, ⑤
            "required": true ⑥
        },
        ...
    ],
    "json": [ ⑦
        {
            "name": "CELL_COUNT", ⑧
            "type": "integer", ⑨
            "key": "cellCount" ⑩
        },
        ...
    ]
}
```

seed.outputs.json:

```
{
    "cellCount": 256, ⑪
    ...
}
```

The following annotated snippets provides quick reference to the use of Outputs object:

① Optional array containing elements defined by Files Object sub-section.

② Required string containing unique output identifier.

③ Optional string containing IANA Media type of file.

④ Required string containing glob expression for file capture. Executor† is expected to capture output relative to OUTPUT_DIR.

⑤ Optional boolean indicating whether a single or multiple values are supported. Default value is false.

⑥ Optional boolean indicating whether executor† should assume failure if output data is missing. Default value is true.

⑦ Optional array containing elements defined by JSON Object sub-section.

⑧ Required string containing unique output identifier. MUST be used by executor† to match member for capture from `seed.outputs.json` in absence of `key` member.

⑨ Required string containing JSON schema type of member extracted from `seed.outputs.json` file.

⑩ Optional string containing key of member for extraction. This allows mapping from a `seed.outputs.json` file member key that differs from the value of `name` member.

⑪ Example output manifest containing key defined in (10).

2.1.1.3.3. Mounts Object

The Mounts object is the member responsible for indicating any additional directories that must be mounted into the container for the Job to run. A mount directory is typically a shared file system directory that contains some set of reference data that the Job requires.

Name	Requirement	Value
`name`	Required	MUST be a string of only alphanumeric, dash or underscore characters (defined by the regex† `^[a-zA-Z0-9_-]+$`) that correlates mount references elsewhere in the Interface to an external mount configuration that specifies how the mount is provided.
`path`	Required	MUST be an absolute file system path specifying where in the container the Job expects the shared directory to be mounted.
`mode`	Optional	MUST be a string that either specifies "ro" for read-only access to the directory or "rw" for read-write access. Default value is "ro".

The following annotated snippet provides quick reference to the use of Mounts object:

```
[
    {
        "name": "MOUNT1", ①
        "path": "/the/container/path", ②
        "mode": "ro" ③
    },
    ...
]
```

Name	Requirement	Value
① Required string containing the name to be used to lookup uses in the Interface. ② Required string indicating the absolute file system path where the directory should be mounted. ③ Optional string indicating whether the directory should be mounted in read-only ("ro") or read-write ("rw") mode.		

2.1.1.3.4. Settings Object

The Settings object is the member responsible for indicating all content not related to data that is needed for the Seed job to run. These will be exposed as environment variables at run time. Most commonly, Settings will be used for environment specific configuration or external credentials.

While it is *highly* advised that Seed jobs SHOULD limit input / output to the provided constructs (`inputs` / `outputs`), there are justified use cases for violating this encapsulation. If database ingestion or downstream messaging are necessary, this is a reasonable mechanism to accomplish that.

Name	Requirement	Value
`name`	Required	MUST be a string of only alphanumeric, dash or underscore characters (defined by the regex† `^[a-zA-Z0-9_-]+$`) indicating the environment variable to be injected at run time. Refer to Injection for details on environment variable available at execution time.
`secret`	Optional	MUST be a boolean that indicates whether the value associated with the named setting is secret and stored as a secure string.

The following annotated snippet provides quick reference to the use of Settings object:

```
[
   {
       "name": "SETTING1", ①
       "secret": true ②
   },
   ...
]
```

① Required string containing the environment variable name to be injected at run time.

② Optional boolean indicating whether the setting value is sensitive and stored as a secret.

2.1.1.4. Errors Object

The Errors object allows for developers† to map arbitrary exit codes to meaningful textual descriptions. This is useful in passing information to the executor† to differentiate between data and job errors.

Name	Requirement	Value
`code`	Required	MUST be an integer indicating the exit code of the executing job process.

Name	Requirement	Value
`name`	Required	MUST be a string of only alphanumeric, dash or underscore characters (defined by the regex† `^[a-zA-Z0-9_-]+$`) indicating the unique name to use for referring to this error. An executor† MAY use member for correlation of error codes across job versions.
`title`	Optional	MUST be a string indicating the short descriptive title of the error.
`description`	Optional	MUST be a string indicating the complete error description and possible causes.
`category`	Optional	MUST be a string containing one of the following values: `job` or `data`. If omitted, the default value is `job`.

The following annotated snippet provides quick reference to the use of Errors object:

```
[
    {
        "code": 1, ①
        "name": "error-name", ②
        "title": "Error Name", ③
        "description": "Error Description", ④
        "category": "job" ⑤
    },
    ...
]
```

① Required integer indicating job process exit code.

② Required string containing machine-friendly identifier of error.

③ Optional string containing human-friendly short name of error.

④ Optional string containing complete error code description.

⑤ Optional string containing the error type. This value MUST be either: `job` or `data`. The default value is `job`.

3. Usage

3.1. Implementing

A few requirements must be satisfied when an implementer† is building a system capable of executing Seed standardized images. The following sections detail behavior that is expected of the executor†, but these details are also important for developers† to understand what execution context they are provided.

3.1.1. Environment Variables

Environment variable injection MUST be performed as it is the primary means of providing the context required by the defined interface of a Seed job. These environment variables MAY be consumed by a job directly or shell variable expansion MAY be leveraged in the `interface.command` member. Implementing frameworks MAY perform variable expansion, but it MUST follow Bash expansion conventions.

3.1.1.1. Normalization

All environment variables injected MUST be normalized to remove any special characters. The majority of environment variable injection is dynamic and tied to `name` member values. These values MUST be sanitized based on the following rules:

- Only upper-case alphabetic, numeric and underscore characters are permitted unaltered.
- All lower-case alphabetic characters MUST be capitalized.
- All dash characters MUST be replaced with underscores.

3.1.1.2. Injection

Collisions between the reserved variables below and `interface.inputs.files`, `interface.inputs.json` and `interface.settings` `name` member values MUST NOT occur.

The following variables MUST be provided:

- OUTPUT_DIR: Absolute path where all output products MUST be placed by job for Seed executor† capture.
- All `name` member values of `resources.scalar` elements MUST map to environment variables. This SHALL be done by normalizing all `name` values and prefixing with `ALLOCATED_`. For a `my-demo-resourceNew` with a value of `5.0`, an environment variable `ALLOCATED_MY_DEMO_RESOURCENEW` SHALL be set to `5.0` at run time. For a resource with an `inputMultiplier` value the environment variable must include the final computed *output* resource allocation as defined in the `scalar` element under Resources object. It SHOULD be clarified that the allocated value computed using an `inputMultiplier` value MUST NOT include the size of input files - only the additional space needed for output files during execution.
- All `name` member values of `interface.inputs.files` elements MUST map to environment variables. Variable names must follow normalization rules. Executor's† MUST ensure data is mounted and provide a container resolvable absolute path. *Injection behavior is dependent on the boolean value of* `multiple` *member.* When `multiple` is `true` the injected environment variable MUST be an absolute path to a *directory* with all files immediately beneath it. When `multiple` is `false` it MUST be an absolute path to the single *file* provided. This environment variable MUST be left unset if member `required` is false and the input is not present.
- All `name` member values of `interface.inputs.json` elements MUST map to environment variables. Variable names must follow normalization rules and `array`, `object` and `string` JSON types MUST be injected string encoded. This environment variable MUST be left unset if `required` is false and the input is not present.
- All `name` member values of `interface.settings` elements MUST map to environment variables. Variable names must follow normalization rules.

Supporting complex variable expansion where jobs require parameters associated with a switch can be accomplished with use of Bash. Taking the example of an optional `inputs.files` element, the following `command` value will expand with a preceding switch only when the input is available:

```
${MY_INPUT/#/-d }
```

This will expand to `-d ${MY_INPUT}` only when `MY_INPUT` is set, otherwise the entire expression will be omitted. It is important to note this is specifically a Bash shell feature and is not present with competing

Linux shells.

3.1.2. Output Data Capture

All output data generated by a Seed compliant job MUST be placed within the location identified by OUTPUT_DIR environment variable. This location MUST be made available by the executor† so that the job is given full write access. Developers MUST ensure any files captured by `interface.outputs.files` member element `pattern` values, as well as the optional `seed.outputs.json` file, are marked with read permissions at minimum. This MAY be done with the following sample command: `chmod +r yourfile.txt`

Special attention should also be given to the number of files matched per `interface.outputs.files` element to ensure it is consistent in plurality to that defined by `multiple` member. If multiple files are matched using the `pattern` member value, while the `multiple` member is set to false, these jobs SHOULD be failed by the executor†. On the other hand, even if multiple files are not present when `multiple` is true it SHOULD NOT force a job failure by the executor†.

3.1.3. Extended File Metadata

There is often a need by the executor† to capture additional job extracted metadata on output files. The Seed standard allows for this through the use of side-car files. The side-car files must be named exactly as the file they describe, with the addition of the `.metadata.json` extension to the original file name (extension included). The file must be formatted according to the Seed Metadata Schema. This allows for both spatial and any custom associated metadata to be specified. The `properties` member is of the object type which allows for any JSON member type that satisfies your specific use-cases.

The following snippet is a notional example specifying a geometry and temporal bounded feature associated with a file:

Metadata JSON

```
{
    "type": "Feature", { ①
    "geometry": { ②
        "type": "Polygon",
        "coordinates": [
            [ [ 100.0, 0.0 ], [ 101.0, 0.0 ], [ 101.0, 1.0 ], [ 100.0, 1.0 ], [ 100.0,
0.0 ] ]
        ]
    },
    "properties": { ③
        "time": {
            "start": "2016-08-06T00:00:00.000Z",
            "end": "2016-08-06T00:00:00.000Z"
        }
    }
}
```

① Required string indicating the GeoJSON type being defined.

② Required geometry member defining spatial extent of file.

③ Required properties member containing example definition temporal extent of file.

3.1.4. Resource Defaults

At a minimum, the executor† MUST provide at least the resources indicated by a Seed manifest at run time. If the resource requirement specified by the manifest is below the minimum amount allowed by the executor† it MAY increase the requirement to that value. Developers† SHOULD specify the `cpus`, `mem` and `disk` requirements of their Seed job, but if these are not set, the executor† is free to allocate minimal defaults.

For the `sharedMem` reserved resource, the executor† SHOULD make use of the Docker run argument `shm-size` to provide the requested resource.

When resources are indicated by a Seed manifest that are not recognized by an executor†, the job SHOULD not be run.

3.2. Examples

The Seed standard is intended to support both simple and complex job packaging. To that end the standard allows for sensible defaults to take the place of a fully specified manifest. The following examples identify both a minimal Seed use and a more realistic, fully exercised standard.

3.2.1. Random Number Generator Job

Minimal manifest demonstrating the simplest possible Seed definition.

```
{
  "seedVersion": "1.0.0-snapshot",
  "job": {
    "name": "random-number-gen",
    "jobVersion": "0.1.0",
    "packageVersion": "0.1.0",
    "title": "Random Number Generator",
    "description": "Generates a random number and outputs on stdout",
    "maintainer": {
      "name": "John Doe",
      "email": "jdoe@example.com"
    },
    "timeout": 10
  }
}
```

Serialized as a label in a Dockerfile snippet:

```
FROM alpine

ENTRYPOINT /app/job.sh

LABEL com.ngageoint.seed.manifest="{\"seedVersion\":\"1.0.0-
snapshot\",\"job\":{\"name\":\"random-number-
gen\",\"jobVersion\":\"0.1.0\",\"packageVersion\":\"0.1.0\",\"title\":\"Random Number
Generator\",\"description\":\"Generates a random number and outputs on
stdout\",\"maintainer\":{\"name\":\"John
Doe\",\"email\":\"jdoe@example.com\"},\"timeout\":10}}"
```

3.2.2. Image Watermark Job

Image watermark job taking a single image and returning with watermark applied using Seed definition.

```
{
  "seedVersion": "1.0.0-snapshot",
  "job": {
    "name": "image-watermark",
    "jobVersion": "0.1.0",
    "packageVersion": "0.1.0",
    "title": "Image Watermarker",
    "description": "Processes an input PNG and outputs watermarked PNG.",
    "maintainer": {
      "name": "John Doe",
      "email": "jdoe@example.com"
    },
    "timeout": 30,
    "interface": {
      "command": "${INPUT_IMAGE} ${OUTPUT_DIR}",
      "inputs": {
        "files": [
          {
            "name": "INPUT_IMAGE"
          }
        ]
      },
      "outputs": {
        "files": [
          {
            "name": "OUTPUT_IMAGE",
            "pattern": "*_watermark.png"
          }
        ]
      }
    },
    "resources": {
      "scalar": [
        {
          "name": "cpus",
          "value": 1
```

```
        },
        {
          "name": "mem",
          "value": 64
        }
      ]
    },
    "errors": [
      {
        "code": 1,
        "name": "image-Corrupt-1",
        "description": "Image input is not recognized as a valid PNG.",
        "category": "data"
      },
      {
        "code": 2,
        "name": "algorithm-failure"
      }
    ]
  }
}
```

Serialized as a label in a Dockerfile snippet:

```
FROM alpine

ENTRYPOINT /app/watermark.py

LABEL com.ngageoint.seed.manifest="{\"seedVersion\":\"1.0.0-
snapshot\",\"job\":{\"name\":\"image-
watermark\",\"jobVersion\":\"0.1.0\",\"packageVersion\":\"0.1.0\",\"title\":\"Image
Watermarker\",\"description\":\"Processes an input PNG and outputs watermarked
PNG.\",\"maintainer\":{\"name\":\"John
Doe\",\"email\":\"jdoe@example.com\"},\"timeout\":30,\"interface\":{\"command\":\"\${I
NPUT_IMAGE}
\${OUTPUT_DIR}\",\"inputs\":{\"files\":[{\"name\":\"INPUT_IMAGE\"}]},\"outputs\":{\"fi
les\":[{\"name\":\"OUTPUT_IMAGE\",\"pattern\":\"*_watermark.png\"}]}},\"resources\":{\
"scalar\":[{\"name\":\"cpus\",\"value\":1},{\"name\":\"mem\",\"value\":64}]},\"errors\
":[{\"code\":1,\"name\":\"image-Corrupt-1\",\"description\":\"Image input is not
recognized as a valid PNG.\",\"category\":\"data\"},{\"code\":2,\"name\":\"algorithm-
failure\"}]}}"
```

4. Discovery

A primary intention of this standard is for simple job discovery from public images hosted within either Docker Hub, Docker Trusted Registry or Docker Registry. There is significant fragmentation of APIs between the various Docker offerings and the following sections describe the steps that may be taken to access the labels defined by Seed.

None of the Docker registry services support label search in any fashion. This incurs the requirement of

applying a secondary means to subset image results. The standard presently requires that all job images are named with the suffix `-seed`. This allows for quick filtering of results to a manageable set for discovery.

4.1. Docker Hub

Docker Hub stores Docker image manifest information in a readily accessible format only for Automated Builds. This enforces the need for all developers† wishing to support simple discovery from Docker Hub to support Hub builds, as opposed to local image builds followed by a docker push. Given this caveat, a service such as ImageLayers can be used to quickly identify manifest content after discovering available images.

The following two steps may be taken to find and identify labels within Docker Hub:

- Perform HTTP GET to find Docker images:
  - URL: https://hub.docker.com/v2/search/repositories/?query=-seed
- Perform HTTP POST to get label details for images found in previous request:
  - BODY: {"repos":[{"name":"myorg/myjob-seed","tag":"latest"}]}
  - URL: https://imagelayers.io/registry/analyze

The ImageLayers service is a 3rd-party service by CenturyLink Labs, but the source code is openly available at ImageLayers and can be used as a reference implementation.

4.2. Docker Registry

Docker Registry does not natively support any type of search, but does provide a catalog API that allows for listing the entire registry contents. Using this along with tag and manifest inspection will allow label inspection.

The following steps may be taken to find and identify labels of Seed compliant images within Docker Registry:

All references to `{registry}`, `{image-id}` and `{tag}` in the following URLs should be replaced with your environment specific values.

- Perform HTTP GET against catalog endpoint to find `-seed` suffixed images:
  - URL: http://{registry}/v2/_catalog
- Perform HTTP GET against tags endpoint for each image matched:
  - http://{registry}/v2/{image-id}/tags/list
- Perform HTTP GET against manifests endpoint to retrieve labels per tag (extract labels from history JSON member):
  - http://{registry}/v2/{image-id}/manifests/{tag}

4.3. Docker Trusted Registry

There is a ticket in with Docker Trusted Registry team to natively support label search. Presently there is no API support to inspect hosted images for label metadata. Images must be pulled locally for inspection.

4.4. Silo

The Seed team has developed a tool to overcome these limitations in Docker registries called Silo.

Silo offers a REST API, implemented by the Seed team, for discovering Seed images. The API allows users to scan one or more repositories for Seed images and then search the resulting images and their manifests by keywords. An executor† can leverage these endpoints to assist users in discovering Seed images and creating jobs out of them.

5. Glossary

The following terms are specific to Seed and are provided to clarify their meaning. They are marked with a dagger (†) throughout the document.

Term	Description
developer	creator and packager of Seed compliant image
executor	process responsible for injecting run-time context (inputs, mounts and environment settings) and capturing all resulting output (files, json).
implementer	developer of framework for discovering or running Seed compliant images - must provide executor
regex	short hand for "Regular Expression," a text string that defines a pattern to be applied to other strings. More info
glob	a string defining a pattern to be applied to files on *nix operating systems. More info

6. Schema

6.1. Seed Manifest

The following JSON Schema should be used to validate Seed manifests prior to label serialization into a Dockerfile for publish. It may be downloaded here: Seed Manifest Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "additionalProperties": false,
  "properties": {
    "seedVersion": {
      "type": "string",
      "pattern": "^1\\.0\\.0-snapshot$"
    },
    "job": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "name": {
          "type": "string",
          "pattern": "^[a-zA-Z0-9-]+$"
        },
```

```
        "jobVersion": {
          "type": "string",
          "pattern": "^(0|[1-9][0-9]*)\\.(0|[1-9][0-9]*)\\.(0|[1-9][0-9]*)(-(0|[1-
9][0-9]*|[0-9]*[a-zA-Z-][0-9a-zA-Z-]*)(\\.(0|[1-9][0-9]*|[0-9]*[a-zA-Z-][0-9a-zA-Z-
]*))*)?(\\+[0-9a-zA-Z-]+(\\.[0-9a-zA-Z-]+)*)?$"
        },
        "packageVersion": {
          "type": "string",
          "pattern": "^(0|[1-9][0-9]*)\\.(0|[1-9][0-9]*)\\.(0|[1-9][0-9]*)(-(0|[1-
9][0-9]*|[0-9]*[a-zA-Z-][0-9a-zA-Z-]*)(\\.(0|[1-9][0-9]*|[0-9]*[a-zA-Z-][0-9a-zA-Z-
]*))*)?(\\+[0-9a-zA-Z-]+(\\.[0-9a-zA-Z-]+)*)?$"
        },
        "title": {
          "type": "string"
        },
        "description": {
          "type": "string"
        },
        "tags": {
          "type": "array",
          "items": {
            "type": "string"
          }
        },
        "maintainer": {
          "type": "object",
          "additionalProperties": false,
          "properties": {
            "name": {
              "type": "string"
            },
            "organization": {
              "type": "string"
            },
            "email": {
              "type": "string"
            },
            "url": {
              "type": "string"
            },
            "phone": {
              "type": "string"
            }
          },
          "required": [
            "name",
            "email"
          ]
        },
        "timeout": {
          "type": "integer"
        },
        "resources": {
```

```
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "scalar": {
          "type": "array",
          "items": {
            "type": "object",
            "additionalProperties": false,
            "properties": {
              "name": {
                "type": "string",
                "pattern": "^[a-zA-Z0-9_-]+$"
              },
              "value": {
                "type": "number"
              },
              "inputMultiplier": {
                "type": "number"
              }
            },
            "required": [
              "name",
              "value"
            ]
          },
          "required": [
            "scalar"
          ]
        }
      }
    },
    "interface": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "command": {
          "type": "string"
        },
        "inputs": {
          "type": "object",
          "additionalProperties": false,
          "properties": {
            "files": {
              "type": "array",
              "items": {
                "type": "object",
                "additionalProperties": false,
                "properties": {
                  "name": {
                    "type": "string",
                    "pattern": "^[a-zA-Z0-9_-]+$"
                  },
                  "required": {
```

```
                "type": "boolean",
                "default": true
              },
              "mediaTypes": {
                "type": "array",
                "items": {
                  "type": "string"
                }
              },
              "multiple": {
                "type": "boolean",
                "default": false
              },
              "partial": {
                "type": "boolean",
                "default": false
              }
            },
            "required": [
              "name"
            ]
          }
        },
        "json": {
          "type": "array",
          "items": {
            "type": "object",
            "additionalProperties": false,
            "properties": {
              "name": {
                "type": "string",
                "pattern": "^[a-zA-Z0-9_-]+$"
              },
              "required": {
                "type": "boolean",
                "default": true
              },
              "type": {
                "type": "string",
                "enum": [
                  "array",
                  "boolean",
                  "integer",
                  "number",
                  "object",
                  "string"
                ]
              }
            },
            "required": [
              "name",
              "type"
            ]
```

```
        }
      }
    }
  },
  "outputs": {
    "type": "object",
    "additionalProperties": false,
    "properties": {
      "files": {
        "type": "array",
        "items": {
          "type": "object",
          "additionalProperties": false,
          "properties": {
            "name": {
              "type": "string",
              "pattern": "^[a-zA-Z0-9_-]+$"
            },
            "mediaType": {
              "type": "string"
            },
            "pattern": {
              "type": "string"
            },
            "multiple": {
              "type": "boolean",
              "default": false
            },
            "required": {
              "type": "boolean",
              "default": true
            }
          },
          "required": [
            "name",
            "pattern"
          ]
        }
      },
      "json": {
        "type": "array",
        "items": {
          "type": "object",
          "additionalProperties": false,
          "properties": {
            "name": {
              "type": "string",
              "pattern": "^[a-zA-Z0-9_-]+$"
            },
            "key": {
              "type": "string"
            },
            "type": {
```

```
                "type": "string",
                "enum": [
                  "array",
                  "boolean",
                  "integer",
                  "number",
                  "object",
                  "string"
                ]
              },
              "required": {
                "type": "boolean",
                "default": true
              }
            },
            "required": [
              "name",
              "type"
            ]
          }
        }
      }
    },
    "mounts": {
      "type": "array",
      "items": {
        "type": "object",
        "additionalProperties": false,
        "properties": {
          "name": {
            "type": "string",
            "pattern": "^[a-zA-Z0-9_-]+$"
          },
          "path": {
            "type": "string"
          },
          "mode": {
            "enum": [
              "ro",
              "rw"
            ],
            "default": "ro"
          }
        },
        "required": [
          "name",
          "path"
        ]
      }
    },
    "settings": {
      "type": "array",
      "items": {
```

```
            "type": "object",
            "additionalProperties": false,
            "properties": {
              "name": {
                "type": "string",
                "pattern": "^[a-zA-Z0-9_-]+$"
              },
              "secret": {
                "type": "boolean",
                "default": false
              }
            },
            "required": [
              "name"
            ]
          }
        }
      }
    },
    "errors": {
      "type": "array",
      "items": {
        "type": "object",
        "additionalProperties": false,
        "properties": {
          "code": {
            "type": "integer"
          },
          "name": {
            "type": "string",
            "pattern": "^[a-zA-Z0-9_-]+$"
          },
          "title": {
            "type": "string"
          },
          "description": {
            "type": "string"
          },
          "category": {
            "type": "string",
            "default": "job",
            "enum": [
              "job",
              "data"
            ]
          }
        },
        "required": [
          "code",
          "name"
        ]
      }
    }
```

```
      },
      "required": [
        "name",
        "jobVersion",
        "packageVersion",
        "title",
        "description",
        "maintainer",
        "timeout"
      ]
    }
  },
  "required": [
    "seedVersion",
    "job"
  ]
}
```

6.2. Seed Metadata

The following JSON Schema should be used to validate the side-car metadata files generated alongside Seed job output files. It may be downloaded here: Seed Metadata Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://tools.ietf.org/html/rfc7946#",
  "title": "Geo JSON object",
  "description": "Schema for a Geo JSON object",
  "type": "object",
  "required": [ "type" ],
  "properties": {
    "bbox": { "$ref": "http://json-schema.org/geojson/bbox.json#" }
  },
  "oneOf": [
    { "$ref": "#/definitions/geometry" },
    { "$ref": "#/definitions/geometryCollection" },
    { "$ref": "#/definitions/feature" },
    { "$ref": "#/definitions/featureCollection" }
  ],
  "definitions": {
    "geometry": {
      "type": "object",
      "required": [
        "type",
        "coordinates"
      ],
      "oneOf": [
        {
          "title": "Point",
          "additionalProperties": false,
          "properties": {
```

```
            "type": {
              "enum": [
                "Point"
              ]
            },
            "coordinates": {
              "$ref": "#/definitions/position"
            }
          }
        },
        {
          "title": "MultiPoint",
          "additionalProperties": false,
          "properties": {
            "type": {
              "enum": [
                "MultiPoint"
              ]
            },
            "coordinates": {
              "$ref": "#/definitions/positionArray"
            }
          }
        },
        {
          "title": "LineString",
          "additionalProperties": false,
          "properties": {
            "type": {
              "enum": [
                "LineString"
              ]
            },
            "coordinates": {
              "$ref": "#/definitions/lineString"
            }
          }
        },
        {
          "title": "MultiLineString",
          "additionalProperties": false,
          "properties": {
            "type": {
              "enum": [
                "MultiLineString"
              ]
            },
            "coordinates": {
              "type": "array",
              "items": {
                "$ref": "#/definitions/lineString"
              }
            }
```

```
        }
      },
      {
        "title": "Polygon",
        "additionalProperties": false,
        "properties": {
          "type": {
            "enum": [
              "Polygon"
            ]
          },
          "coordinates": {
            "$ref": "#/definitions/polygon"
          }
        }
      },
      {
        "title": "MultiPolygon",
        "additionalProperties": false,
        "properties": {
          "type": {
            "enum": [
              "MultiPolygon"
            ]
          },
          "coordinates": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/polygon"
            }
          }
        }
      }
    ]
  },
  "geometryCollection": {
    "title": "GeometryCollection",
    "description": "A collection of geometry objects",
    "required": [ "geometries" ],
    "properties": {
      "type": { "enum": [ "GeometryCollection" ] },
      "geometries": {
        "type": "array",
        "items": { "$ref": "#/definitions/geometry" }
      }
    }
  },
  "feature": {
    "title": "Feature",
    "description": "A Geo JSON feature object",
    "required": [ "geometry", "properties" ],
    "properties": {
      "type": { "enum": [ "Feature" ] },
```

```
      "geometry": {
        "oneOf": [
          { "type": "null" },
          { "$ref": "#/definitions/geometry" }
        ]
      },
        "properties": { "type": [ "object", "null" ] },
        "id": { "type": [ "string", "number" ] }
    }
  },
  "featureCollection": {
    "title": "FeatureCollection",
    "description": "A Geo JSON feature collection",
    "required": [ "features" ],
    "properties": {
      "type": { "enum": [ "FeatureCollection" ] },
      "features": {
        "type": "array",
        "items": { "$ref": "#/definitions/feature" }
      }
    }
  },
    "position": {
      "description": "A single position",
      "type": "array",
      "minItems": 2,
      "maxItems": 3,
      "items": [
        {
          "type": "number"
        },
        {
          "type": "number"
        },
        {
          "type": "number"
        }
      ],
      "additionalItems": false
    },
    "positionArray": {
      "description": "An array of positions",
      "type": "array",
      "items": {
        "$ref": "#/definitions/position"
      }
    },
    "lineString": {
      "description": "An array of two or more positions",
      "allOf": [
        {
          "$ref": "#/definitions/positionArray"
        },
```

```
          {
            "minItems": 2
          }
        ]
      },
      "linearRing": {
        "description": "An array of four positions where the first equals the last",
        "allOf": [
          {
            "$ref": "#/definitions/positionArray"
          },
          {
            "minItems": 4
          }
        ]
      },
      "polygon": {
        "description": "An array of linear rings",
        "type": "array",
        "items": {
          "$ref": "#/definitions/linearRing"
        }
      }
  }
}
```