# Holoscan Application Package Specification (HAP)

# Table of contents

# Introduction

The Holoscan Application Package specification extends the MONAI Deploy Application Package specification to provide the streaming capabilities, multi-fragment and other features of the Holoscan SDK.

# Overview

This document includes the specification of the Holoscan Application Package (HAP). A HAP is a containerized application or service which is self-descriptive, as defined by this document.

## Goal

This document aims to define the structure and purpose of a HAP, including which parts are optional and which are required so that developers can easily create conformant HAPs.

## Assumptions, Constraints, Dependencies

The following assumptions relate to HAP execution, inspection and general usage:

- Containerized applications will be based on Linux x64 (AMD64) and/or ARM64 (aarch64).
- Containerized applications' host environment will be based on Linux x64 (AMD64) and/or ARM64 (aarch64) with container support.
- Developers expect the local execution of their applications to behave identically to the execution of the containerized version.
- Developers expect the local execution of their containerized applications to behave identically to the execution in deployment.
- Developers and operations engineers want the application packages to be self-describing.
- Applications may be created using tool other than that provided in the Holoscan SDK or the MONAI Deploy App SDK.

- Holoscan Application Package may be created using a tool other than that provided in the Holoscan SDK or the MONAI Deploy App SDK.
- Pre-existing, containerized applications must be “converted” into Holoscan Application Packages.
- A Holoscan Application Package may contain a classical application (non-fragment based), a single-fragment application, or a multi-fragment application. (Please see the definition of fragment in Definitions, Acronyms, Abbreviations)
- The scalability of a multi-fragment application based on Holoscan SDK is outside the scope of this document.
- Application packages are expected to be deployed in one of the supported environments. For additional information, see Holoscan Operating Environments.

# Definitions, Acronyms, Abbreviations

| Term | Definition |
|---|---|
| ARM64 | Or, AARCH64. See Wikipedia for details. |
| Container | See What’s a container? |
| Fragment | A fragment is a building block of the Application. It is a directed graph of operators. For details, please refer to the MONAI Deploy App SDK or Holoscan App SDK. |
| Gigibytes (GiB) | A gibibyte (GiB) is a unit of measurement used in computer data storage that equals to 1,073,741,824 bytes. |
| HAP | Holoscan Application Package. A containerized application or service which is self-descriptive. |
| Hosting Service | A service that hosts and orchestrates HAP containers. |
| MAP | MONAI Application Package. A containerized application or service which is self-descriptive. |
| Mebibytes (MiB) | A mebibyte (MiB) is a unit of measurement used in computer data storage that equals to 1,048,576 bytes. |
| MONAI | Medical Open Network for Artificial Intelligence. |
| SDK | Software Development Kit. |

| | |
|---|---|
| Semantic Version | See Semantic Versioning 2.0. |
| x64 | Or, x86-64 or AMD64. See Wikipedia for details. |

# Requirements

The following requirements MUST be met by the HAP specification to be considered complete and approved. All requirements marked as `MUST` or `SHALL` MUST be implemented in order to be supported by a HAP-ready hosting service.

## Single Artifact

- A HAP SHALL comprise a single container, meeting the minimum requirements set forth by this document.
- A HAP SHALL be a containerized application to maximize the portability of its application.

## Self-Describing

- A HAP MUST be self-describing and provide a mechanism for extracting its description.
    - A HAP SHALL provide a method to print the metadata files to the console.
    - A HAP SHALL provide a method to copy the metadata files to a user-specified directory.
- The method of description SHALL be in a machine-readable and writable format.
- The method of description SHALL be in a human-readable format.
- The method of description SHOULD be a human writable format.
- The method of description SHALL be declarative and immutable.
- The method of description SHALL provide the following information about the HAP:
    - Execution requirements such as dependencies and restrictions.

    - Resource requirements include CPU cores, system memory, shared memory, GPU, and GPU memory.

## Runtime Characteristics of the HAP

- A HAP SHALL start the packaged Application when it is executed by the users when arguments are specified.
- A HAP SHALL describe the packaged Application as a long-running service or an application so an external agent can manage its lifecycle.

## IO Specification

- A HAP SHALL provide information about its expected inputs such that an external agent can determine if the HAP can receive a workload.
- A HAP SHALL provide sufficient information about its outputs so that an external agent knows how to handle the results.

## Local Execution

A HAP MUST be in a format that supports local execution in a development environment.

[Note] See Holoscan Operating Environments for additional information about supported environments.

## Compatible with Kubernetes

- A HAP SHALL support deployment using Kubernetes.

## OCI Compliance

The containerized portion of a HAP SHALL comply with Open Container Initiative format standards.

### Image Annotations

All annotations for the containerized portion of a HAP MUST adhere to the specifications laid out by The OpenContainers Annotations Spec

- `org.opencontainers.image.title`: A HAP container image SHALL provide a human-readable title (string).
- `org.opencontainers.image.version`: A HAP container image SHALL provide a version of the packaged application using the semantic versioning format. This value is the same as the value defined in `/etc/holoscan/app.json#version` in the Table of Application Manifest Fields.
- All other OpenContainers predefined keys SHOULD be provided when available.

## Hosting Environment

The HAP Hosting Environment executes the HAP and provides the application with a customized set of environment variables and command line options as part of the invocation.

- The Hosting Service MUST, by default, execute the application as defined by `/etc/holoscan/app.json#command` and then exit when the application or the service completes.
- The Hosting Service MUST provide any environment variables specified by `/etc/holoscan/app.json#environment`.
- The Hosting Service SHOULD monitor the Application process and record its CPU, system memory, and GPU utilization metrics.
- The Hosting Service SHOULD monitor the Application process and enforce any timeout value specified in `/etc/holoscan/app.json#timeout`.

### Table of Environment Variables

A HAP SHALL contain the following environment variables and their default values, if not specified by the user, in the Application Manifest `/etc/holoscan/app.json#environment`.

| Variable | Default | Format | Description |
|---|---|---|---|
| `HOLOSCAN_INPUT_PATH` | `/var/holoscan/input/` | Folder Path | Path to the input folder for the Application. |
| `HOLOSCAN_OUTPUT_PATH` | `/var/holoscan/output/` | Folder Path | Path to the output folder for the Application. |

| | | | |
|---|---|---|---|
| `HOLOSCAN_WORKDIR` | `/var/holoscan/` | Folder Path | Path to the Application's working directory. |
| `HOLOSCAN_MODEL_PATH` | `/opt/holoscan/models/` | Folder Path | Path to the Application's models directory. |
| `HOLOSCAN_CONFIG_PATH` | `/var/holoscan/app.yaml` | File Path | Path to the Application's configuration file. |
| `HOLOSCAN_APP_MANIFEST_PATH` | `/etc/holoscan/app.config` | File Path | Path to the Application's configuration file. |
| `HOLOSCAN_PKG_MANIFEST_PATH` | `/etc/holoscan/pkg.config` | File Path | Path to the Application's configuration file. |
| `HOLOSCAN_DOCS` | `/opt/holoscan/docs` | Folder Path | Path to the folder containing application documentation and licenses. |
| `HOLOSCAN_LOGS` | `/var/holoscan/logs` | Folder Path | Path to the Application's logs. |

# Architecture & Design

## Description

The Holoscan Application Package (HAP) is a functional package designed to act on datasets of a prescribed format. A HAP is a container image that adheres to the specification provided in this document.

## Application

The primary component of a HAP is the application. The application is provided by an application developer and incorporated into the HAP using the Holoscan Application Packager.

All application code and binaries SHALL be in the `/opt/holoscan/app/` folder, except for any dependencies installed by the Holoscan Application Packager during the creation of the HAP.

All AI models (PyTorch, TensorFlow, TensorRT, etc.) SHOULD be in separate sub-folders of the `/opt/holoscan/models/` folder. In specific use cases where the app package

developer is prevented from enclosing the model files in the package/container due to intellectual property concerns, the models can be supplied from the host system when the app package is run, e.g., via the volume mount mappings and the use of container env variables.

## Manifests

A HAP SHALL contain two manifests: the Application Manifest and the Package Manifest. The Package Manifest shall be stored in `/etc/holoscan/pkg.json`, and the Application Manifest shall be stored in `/etc/holoscan/app.json`. Once a HAP is created, its manifests are expected to be immutable.

### Application Manifest

#### Table of Application Manifest Fields

| Name | Required | Default | Type | Format | Description |
|---|---|---|---|---|---|
| `apiVersion` | No | 0.0.0 | string | semantic version | Version of the manifest file schema. |
| `command` | **Yes** | N/A | string | shell command | Shell command used to run the Application. |
| `environment` | **No** | N/A | object | object w/ name-value pairs | An object of name-value pairs that will be passed to the application during execution. |
| `input` | **Yes** | N/A | object | object | Data structure which provides information about Application inputs. |
| `input.formats` | **Yes** | N/A | array | array of objects | List of input data formats accepted by the Application. |
| `input.path` | No | input/ | string | relative file-system path | Folder path relative to the working directory from which the application will read inputs. |

| | | | | | |
|---|---|---|---|---|---|
| `readiness` | No | N/A | object | object | An object of name-value pairs that defines the readiness probe. |
| `readiness.type` | **Yes** | N/A | string | string | Type of the probe: `tcp`, `grpc`, `http-get` or `command`. |
| `readiness.command` | **Yes** (when type is `command`) | N/A | array | shell command | Shell command and arguments in string array form. |
| `readiness.port` | **Yes** (when type is `tcp`, `grpc`, or `http-get`) | N/A | integer | number | The port number of readiness probe. |
| `readiness.path` | **Yes** (when type is `http-get`) | N/A | string | string | HTTP path and query to access the readiness probe. |
| `readiness.initialDelaySeconds` | No | 1 | integer | number | Number of seconds after the container has started before the readiness probe is initialized and performed. |
| `readiness.periodSeconds` | No | 10 | integer | number | Number of seconds between performing the readiness probe. |
| `readiness.timeoutSeconds` | No | 1 | integer | number | Number of seconds after which the probe times out. |
| `readiness.failureThreshold` | No | 3 | integer | number | Number of retries to be performed before considering the application is unhealthy. |
| `liveness` | No | N/A | object | object | An object of name-value pairs that defines the liveness probe. Recommended for service applications. |

| | | | | | |
|---|---|---|---|---|---|
| `liveness.type` | **Yes** | N/A | string | string | Type of the probe: `tcp`, `grpc`, `http-get` or `command`. |
| `liveness.command` | **Yes** (when type is `command`) | N/A | array | shell command | Shell command and arguments in string array form. |
| `liveness.port` | **Yes** (when type is `tcp`, `grpc`, or `http-get`) | N/A | integer | number | The port number of the liveness probe. |
| `liveness.path` | **Yes** (when type is `http-get`) | N/A | string | string | HTTP path and query to access the liveness probe. |
| `liveness.initialDelaySeconds` | No | 1 | integer | number | Number of seconds after the container has started before the liveness probe is initialized and performed. |
| `liveness.periodSeconds` | No | 10 | integer | number | Number of seconds between performing the liveness probe. |
| `liveness.timeoutSeconds` | No | 1 | integer | number | Number of seconds after which the probe times out. |
| `liveness.failureThreshold` | No | 3 | integer | number | Number of retries to be performed before considering the application is unhealthy. |
| `output` | **Yes** | N/A | object | object | Data structure which provides information about Application output. |
| `output.format` | **Yes** | N/A | object | object | Details about the format of the outputs produced by the application. |

| | | | | | |
|---|---|---|---|---|---|
| `output.path` | No | output/ | string | relative file-system path | Folder path relative to the working directory to which the application will write outputs. |
| `sdk` | No | N/A | string | string | SDK used for the Application. |
| `sdkVersion` | No | 0.0.0 | string | semantic version | Version of the SDK used the Application. |
| `timeout` | No | 0 | integer | number | The maximum number of seconds the application should execute before being timed out and terminated. Recommended for a single batch/execution type of applications. |
| `version` | No | 0.0.0 | string | semantic version | Version of the Application. |
| `workingDirectory` | No | /var/holoscan/ | string | absolute file-system path | Folder, or directory, in which the application will be executed. |

The Application Manifest file provides information about the HAP's Application.

- The Application Manifest MUST define the type of the containerized application ( `/etc/holoscan/app.json#type` ).
    - Type SHALL have the value of either `service` or `application.`
- The Application Manifest MUST define the command used to run the Application ( `/etc/holoscan/app.json#command` ).
- The Application Manifest SHOULD define the version of the manifest file schema ( `/etc/holoscan/app.json#apiVersion` ).
    - The Manifest schema version SHALL be provided as a semantic version string.
    - When not provided, the default value `0.0.0` SHALL be assumed.

- The Application Manifest SHOULD define the SDK used to create the Application ( `/etc/holoscan/app.json#sdk` ).
- The Application Manifest SHOULD define the version of the SDK used to create the Application ( `/etc/holoscan/app.json#sdkVersion` ).
    - SDK version SHALL be provided as a semantic version string.
    - When not provided, the default value `0.0.0` SHALL be assumed.
- The Application Manifest SHOULD define the version of the application itself ( `/etc/holoscan/app.json#version` ).
    - The Application version SHALL be provided as a semantic version string.
    - When not provided, the default value `0.0.0` SHALL be assumed.
- The Application Manifest SHOULD define the application's working directory ( `/etc/holoscan/app.json#workingDirectory` ).
    - The Application will execute with its current directory set to this value.
    - The value provided must be an absolute path (the first character is `/` ).
    - The default path `/var/holoscan/` SHALL be assumed when not provided.
- The Application Manifest SHOULD define the data input path, relative to the working directory, used by the Application ( `/etc/holoscan/app.json#input.path` ).
    - The input path SHOULD be a relative to the working directory or an absolute file-system path to a directory.
        - When the value is a relative file-system path (the first character is not `/` ), it is relative to the application's working directory.
        - When the value is an absolute file-system path (the first character is `/` ), the file-system path is used as-is.
    - When not provided, the default value `input/` SHALL be assumed.

- The Application Manifest SHOULD define input data formats supported by the Application ( `/etc/holoscan/app.json#input.formats` ).
    - Possible values include, but are not limited to, `none` , `network` , `file` .
- The Application Manifest SHOULD define the output path relative to the working directory used by the Application ( `/etc/holoscan/app.json#output.path` ).
    - The output path SHOULD be relative to the working directory or an absolute file-system path to a directory.
        - When the value is a relative file-system path (the first character is not `/` ), it is relative to the application's working directory.
        - When the value is an absolute file-system path (the first character is `/` ), the file-system path is used as-is.
    - When not provided, the default value `output/` SHALL be assumed.
- The Application Manifest SHOULD define the output data format produced by the Application ( `/etc/holoscan/app.json#output.format` ).
    - Possible values include, but are not limited to, `none` , `screen` , `file` , `network` .
- The Application Manifest SHOULD configure a check to determine whether or not the application is "ready."
    - The Application Manifest SHALL define the probe type to be performed ( `/etc/holoscan/app.json#readiness.type` ).
        - Possible values include `tcp` , `grpc` , `http-get` , and `command` .
    - The Application Manifest SHALL define the probe commands to execute when the type is `command` ( `/etc/holoscan/app.json#readiness.command` ).
        - The data structure is expected to be an array of strings.
    - The Application Manifest SHALL define the port to perform the readiness probe when the type is `grpc` , `tcp` , or `http-get` . ( `/etc/holoscan/app.json#readiness.port` )

        - The value provided must be a valid port number ranging from 1 through 65535. (Please note that port numbers below 1024 are root-only privileged ports.)
    - The Application Manifest SHALL define the path to perform the readiness probe when the type is `http-get` (`/etc/holoscan/app.json#readiness.path`).
        - The value provided must be an absolute path (the first character is `/`).
    - The Application Manifest SHALL define the number of seconds after the container has started before the readiness probe is initiated. (`/etc/holoscan/app.json#readiness.initialDelaySeconds`).
        - The default value `0` SHALL be assumed when not provided.
    - The Application Manifest SHALL define how often to perform the readiness probe (`/etc/holoscan/app.json#readiness.periodSeconds`).
        - When not provided, the default value `10` SHALL be assumed.
    - The Application Manifest SHALL define the number of seconds after which the probe times out (`/etc/holoscan/app.json#readiness.timeoutSeconds`)
        - When not provided, the default value `1` SHALL be assumed.
    - The Application Manifest SHALL define the number of times to perform the probe before considering the service is not ready (`/etc/holoscan/app.json#readiness.failureThreshold`)
        - The default value `3` SHALL be assumed when not provided.
- The Application Manifest SHOULD configure a check to determine whether or not the application is “live” or not.
    - The Application Manifest SHALL define the type of probe to be performed (`/etc/holoscan/app.json#liveness.type`).
        - Possible values include `tcp`, `grpc`, `http-get`, and `command`.

- The Application Manifest SHALL define the probe commands to execute when the type is `command` ( `/etc/holoscan/app.json#liveness.command` ).
    - The data structure is expected to be an array of strings.
- The Application Manifest SHALL define the port to perform the liveness probe when the type is `grpc` , `tcp` , or `http-get` . ( `/etc/holoscan/app.json#liveness.port` )
    - The value provided must be a valid port number ranging from 1 through 65535. (Please note that port numbers below 1024 are root-only privileged ports.)
- The Application Manifest SHALL define the path to perform the liveness probe when the type is `http-get` ( `/etc/holoscan/app.json#liveness.path` ).
    - The value provided must be an absolute path (the first character is `/` ).
- The Application Manifest SHALL define the number of seconds after the container has started before the liveness probe is initiated. ( `/etc/holoscan/app.json#liveness.initialDelaySeconds` ).
    - The default value `0` SHALL be assumed when not provided.
- The Application Manifest SHALL define how often to perform the liveness probe ( `/etc/holoscan/app.json#liveness.periodSeconds` ).
    - When not provided, the default value `10` SHALL be assumed.
- The Application Manifest SHALL define the number of seconds after which the probe times out ( `/etc/holoscan/app.json#liveness.timeoutSeconds` )
    - The default value `1` SHALL be assumed when not provided.
- The Application Manifest SHALL define the number of times to perform the probe before considering the service is not alive ( `/etc/holoscan/app.json#liveness.failureThreshold` )
    - When not provided, the default value `3` SHALL be assumed.

- The Application Manifest SHOULD define any timeout applied to the Application ( `/etc/holoscan/app.json#timeout` ).
    - When the value is `0` , timeout SHALL be disabled.
    - When not provided, the default value `0` SHALL be assumed.
- The Application Manifest MUST enable the specification of environment variables for the Application ( `/etc/holoscan/app.json#environment` )
    - The data structure is expected to be in `"name": "value"` members of the object.
    - The field's name will be the name of the environment variable verbatim and must conform to all requirements for environment variables and JSON field names.
    - The field's value will be the value of the environment variable and must conform to all requirements for environment variables.

### Package Manifest

#### Table of Package Manifest Fields

| Name | Required | Default | Type | Format | Description |
|---|---|---|---|---|---|
| `apiVersion` | No | `0.0.0` | string | semantic version | Version of the manifest file schema. |
| `applicationRoot` | **Yes** | `/opt/holoscan/app/` | string | absolute file-system path | Absolute file-system path to the folder which contains the Application |
| `modelRoot` | No | `/opt/holoscan/models/` | string | absolute file-system path | Absolute file-system path to the folder which contains the model(s). |
| `models` | No | N/A | array | array of objects | Array of objects which describe models in the package. |

| | | | | | |
|---|---|---|---|---|---|
| `models[*].name` | **Yes** | N/A | string | string | Name of the model. |
| `models[*].path` | No | N/A | string | Relative file-system path | File-system path to the folder which contains the model that is relative to the value defined in `modelRoot`. |
| `resources` | No | N/A | object | object | Object describing resource requirements for the Application. |
| `resources.cpu` | No | `1` | decimal (2) | number | Number of CPU cores required by the Application or the Fragment. |
| `resources.cpuLimit` | No | N/A | decimal (2) | number | The CPU core limit for the Application or the Fragment. (1) |
| `resources.gpu` | No | `0` | decimal (2) | number | Number of GPU devices required by the Application or the Fragment. |
| `resources.gpuLimit` | No | N/A | decimal (2) | number | The GPU device limit for the Application or the Fragment. (1) |
| `resources.memory` | No | `1Gi` | string | memory size | The memory required by the Application or the Fragment. |
| `resources.memoryLimit` | No | N/A | string | memory size | The memory limit for the Application or the Fragment. (1) |
| `resources.gpuMemory` | No | N/A | string | memory size | The GPU memory required by the Application or the Fragment. |
| `resources.gpuMemoryLimit` | No | N/A | string | memory size | The GPU memory limit for the Application or the Fragment. (1) |
| `resources.sharedMemory` | No | `64Mi` | string | memory size | The shared memory required by the Application or the |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | Fragment. |
| resources.fragments | No | N/A | object | objects | Nested objects which describe resources for a Multi-Fragment Application. |
| resources.fragments.<fragment-name> | **Yes** | N/A | string | string | Name of the fragment. |
| resources.fragments.<fragment-name>.cpu | No | 1 | decimal (2) | number | Number of CPU cores required by the Fragment. |
| resources.fragments.<fragment-name>.cpuLimit | No | N/A | decimal (2) | number | The CPU core limit for the Fragment. (1) |
| resources.fragments.<fragment-name>.gpu | No | 0 | decimal (2) | number | Number of GPU devices required by the Fragment. |
| resources.fragments.<fragment-name>.gpuLimit | No | N/A | decimal (2) | number | The GPU device limit for the Fragment. (1) |
| resources.fragments.<fragment-name>.memory | No | 1Gi | string | memory size | The memory required by the Fragment. |

| | | | | | |
|---|---|---|---|---|---|
| resources.fragments.<fragment-name>.memoryLimit | No | N/A | string | memory size | The memory limit for the Fragment. (1) |
| resources.fragments.<fragment-name>.gpuMemory | No | N/A | string | memory size | The GPU memory required by the Fragment. |
| resources.fragments.<fragment-name>.gpuMemoryLimit | No | N/A | string | memory size | The GPU memory limit for the Fragment. (1) |
| resources.fragments.<fragment-name>.sharedMemory | No | 64Mi | string | memory size | The shared memory required by the Fragment. |
| version | No | 0.0.0 | string | semantic version | Version of the package. |

[Notes] (1) Use of resource limits depend on the orchestration service or the hosting environement's configuration and implementation. (2) Consider rounding up to a whole number as decimal values may not be supported by all orchestration/hosting services.

The Package Manifest file provides information about the HAP's package layout. It is not intended as a mechanism for controlling how the HAP is used or how the HAP's Application is executed.

- The Package Manifest MUST be UTF-8 encoded and use the JavaScript Object Notation (JSON) format.
- The Package Manifest SHOULD support either CRLF or LF style line endings.

- The Package Manifest SHOULD specify the folder which contains the application ( `/etc/holoscan/pkg.json#applicationRoot` ).
    - When not provided, the default path `/opt/holoscan/app/` will be assumed.
- The Package Manifest SHOULD provide the version of the package file manifest schema ( `/etc/holoscan/pkg.json#apiVersion` ).
    - The Manifest schema version SHALL be provided as a semantic version string.
- The Package Manifest SHOULD provide the package version of itself ( `/etc/holoscan/pkg.json#version` ).
    - The Package version SHALL be provided as a semantic version string.
- The Package Manifest SHOULD provide the directory path to the user-provided models. ( `/etc/holoscan/pkg.json#modelRoot` ).
    - The value provided must be an absolute path (the first character is `/` ).
    - When not provided, the default path `/opt/holoscan/models/` SHALL be assumed.
- The Package Manifest SHOULD list the models used by the application ( `/etc/holoscan/pkg.json#models` ).
    - Models SHALL be defined by name ( `/etc/holoscan/pkg.json#models[*].name` ).
        - Model names SHALL NOT contain any Unicode whitespace or control characters.
        - Model names SHALL NOT exceed 128 bytes in length.
    - Models SHOULD provide a file-system path if they're included in the HAP itself ( `/etc/holoscan/pkg.json#models[*].path` ).
        - When the value is a relative file-system path (the first character is not `/` ), it is relative to the model root directory defined in `/etc/holoscan/pkg.json#modelRoot` .

        - When the value is an absolute file-system path (the first character is `/`), the file-system path is used as-is.
        - When no value is provided, the name is assumed as the name of the directory relative to the model root directory defined in `/etc/holoscan/pkg.json#modelRoot`.

- The Package Manifest SHOULD specify the resources required to execute the Application and the fragments for a Multi-Fragment Application.

  This information is used to provision resources when running the containerized application using a compatible application deployment service.

- A classic Application or a single Fragment Application SHALL define its resources in the `/etc/holoscan/pkg.json#resource` object.
    - The `/etc/holoscan/pkg.json#resource` object is for the whole application. It CAN also be used as a catchall for all fragments in a multi-fragment application where applicable.
    - CPU requirements SHALL be denoted using the decimal count of CPU cores (`/etc/holoscan/pkg.json#resources.cpu`).
    - Optional CPU limits SHALL be denoted using the decimal count of CPU cores (`/etc/holoscan/pkg.json#resources.cpuLimit`)
    - GPU requirements SHALL be denoted using the decimal count of GPUs (`/etc/holoscan/pkg.json#resources.gpu`).
    - Optional GPU limits SHALL be denoted using the decimal count of GPUs (`/etc/holoscan/pkg.json#resources.gpuLimit`)
    - Memory requirements SHALL be denoted using decimal values followed by units (`/etc/holoscan/pkg.json#resources.memory`).
        - Supported units SHALL be mebibytes (`MiB`) and gibibytes (`GiB`).
            - Example: `1.5Gi`, `2048Mi`

- Optional memory limits SHALL be denoted using decimal values followed by units (`/etc/holoscan/pkg.json#resources.memoryLimit`).
  - Supported units SHALL be mebibytes (`MiB`) and gibibytes (`GiB`).
    - Example: `1.5Gi`, `2048Mi`
- GPU memory requirements SHALL be denoted using decimal values followed by units (`/etc/holoscan/pkg.json#resources.gpuMemory`).
  - Supported units SHALL be mebibytes (`MiB`) and gibibytes (`GiB`).
    - Example: `1.5Gi`, `2048Mi`
- Optional GPU memory limits SHALL be denoted using decimal values followed by units (`/etc/holoscan/pkg.json#resources.gpuMemoryLimit`).
  - Supported units SHALL be mebibytes (`MiB`) and gibibytes (`GiB`).
    - Example: `1.5Gi`, `2048Mi`
- Shared memory requirements SHALL be denoted using decimal values followed by units (`/etc/holoscan/pkg.json#resources.sharedMemory`).
  - Supported units SHALL be mebibytes (`MiB`) and gibibytes (`GiB`).
    - Example: `1.5Gi`, `2048Mi`
- Optional shared memory limits SHALL be denoted using decimal values followed by units (`/etc/holoscan/pkg.json#resources.sharedMemoryLimit`).
  - Supported units SHALL be mebibytes (`MiB`) and gibibytes (`GiB`).
    - Example: `1.5Gi`, `2048Mi`
- Integer values MUST be positive and not contain any position separators.
  - Example legal values: `1`, `42`, `2048`

        - Example illegal values: `-1`, `1.5`, `2,048`
    - Decimal values MUST be positive, rounded to the nearest tenth, use the dot (`.`) character to separate whole and fractional values, and not contain any positional separators.
        - Example legal values: `1`, `1.0`, `0.5`, `2.5`, `1024`
        - Example illegal values: `1,024`, `-1.0`, `3.14`
    - When not provided, the default values of `cpu=1`, `gpu=0`, `memory="1Gi"`, and `sharedMemory="64Mi"` will be assumed.
- A Multi-Fragment Application SHOULD define its resources in the `/etc/holoscan/pkg.json#resource.fragments.<fragment-name>` object.
    - When a matching `fragment-name` cannot be found, the `/etc/holoscan/pkg.json#resource` definition is used.
    - Fragment names (`fragment-name`) SHALL NOT contain any Unicode whitespace or control characters.
    - Fragment names (`fragment-name`) SHALL NOT exceed 128 bytes in length.
    - CPU requirements for fragments SHALL be denoted using the decimal count of CPU cores (`/etc/holoscan/pkg.json#resources.fragments.<fragment-name>.cpu`).
    - Optional CPU limits for fragments SHALL be denoted using the decimal count of CPU cores (`/etc/holoscan/pkg.json#resources.fragments.<fragment-name>.cpuLimit`).
    - GPU requirements for fragments SHALL be denoted using the decimal count of GPUs (`/etc/holoscan/pkg.json#resources.fragments.<fragment-name>.gpu`).

- Optional GPU limits for fragments SHALL be denoted using the decimal count of GPUs (`/etc/holoscan/pkg.json#resources.fragments.<fragment-name>.gpuLimit`).
- Memory requirements for fragments SHALL be denoted using decimal values followed by units (`/etc/holoscan/pkg.json#resources.fragments.<fragment-name>.memory`).
  - Supported units SHALL be mebibytes (`MiB`) and gibibytes (`GiB`).
    - Example: `1.5Gi`, `2048Mi`
- Optional memory limits for fragments SHALL be denoted using decimal values followed by units (`/etc/holoscan/pkg.json#resources.fragments.<fragment-name>.memoryLimit`).
  - Supported units SHALL be mebibytes (`MiB`) and gibibytes (`GiB`).
    - Example: `1.5Gi`, `2048Mi`
- GPU memory requirements for fragments SHALL be denoted using decimal values followed by units (`/etc/holoscan/pkg.json#resources.fragments.<fragment-name>.gpuMemory`).
  - Supported units SHALL be mebibytes (`MiB`) and gibibytes (`GiB`).
    - Example: `1.5Gi`, `2048Mi`
- Optional GPU memory limits for fragments SHALL be denoted using decimal values followed by units (

`/etc/holoscan/pkg.json#resources.fragments.<fragment-name>.gpuMemoryLimit`
).

- Supported units SHALL be mebibytes ( `MiB` ) and gibibytes ( `GiB` ).
  - Example: `1.5Gi` , `2048Mi`

- Shared memory requirements for fragments SHALL be denoted using decimal values followed by units ( `/etc/holoscan/pkg.json#resources.fragments.<fragment-name>.sharedMemory` ).
  - Supported units SHALL be mebibytes ( `MiB` ) and gibibytes ( `GiB` ).
    - Example: `1.5Gi` , `2048Mi`
- Optional shared memory limits for fragments SHALL be denoted using decimal values followed by units ( `/etc/holoscan/pkg.json#resources.fragments.<fragment-name>.sharedMemoryLimit` ).
  - Supported units SHALL be mebibytes ( `MiB` ) and gibibytes ( `GiB` ).
    - Example: `1.5Gi` , `2048Mi`
- Integer values MUST be positive and not contain any position separators.
  - Example legal values: `1` , `42` , `2048`
  - Example illegal values: `-1` , `1.5` , `2,048`
- Decimal values MUST be positive, rounded to the nearest tenth, use the dot ( `.` ) character to separate whole and fractional values, and not contain any positional separators.
  - Example legal values: `1` , `1.0` , `0.5` , `2.5` , `1024`

        - Example illegal values: `1,024`, `-1.0`, `3.14`
    - When not provided, the default values of `cpu=1`, `gpu=0`, `memory="1Gi"`, and `sharedMemory="64Mi"` will be assumed.

# Supplemental Application Files

- A HAP SHOULD package supplemental application files provided by the user.
    - Supplemental files SHOULD be in sub-folders of the `/opt/holoscan/docs/` folder.
    - Supplemental files include, but are not limited to, the following:
        - README.md
        - License.txt
        - Changelog.txt
        - EULA
        - Documentation
        - Third-party licenses

## Container Behavior and Interaction

A HAP is a single container supporting the following defined behaviors when started.

#### Default Behavior

When a HAP is started from the CLI or other means without any parameters, the HAP shall execute the contained application. The HAP internally may use `Entrypoint`, `CMD`, or a combination of both.

#### Manifest Export

A HAP SHOULD provide at least one method to access the *embedded application*, *models*, *licenses*, *README*, or *manifest files*, namely, `app.json` and `package.json`.

- The Method SHOULD provide a container command, `show`, to print one or more manifest files to the console.
- The Method SHOULD provide a container command, `export`, to copy one or more manifest files to a mounted volume path, as described below
    - `/var/run/holoscan/export/app/`: when detected, the Method copies the contents of `/opt/holoscan/app/` to the folder.
    - `/var/run/holoscan/export/config/`: when detected, the Method copies `/var/holoscan/app.yaml`, `/etc/holoscan/app.json` and `/etc/holoscan/pkg.json` to the folder.
    - `/var/run/holoscan/export/models/`: when detected, the Method copies the contents of `/opt/holoscan/models/` to the folder.
    - `/var/run/holoscan/export/docs/`: when detected, the Method copies the contents of `/opt/holoscan/docs/` to the folder.
    - `/var/run/holoscan/export/`: when detected without any of the above being detected, the Method SHALL copy all of the above.

Since a HAP is an OCI compliant container, a user can also run a HAP and log in to an interactive shell, using a method supported by the container engine and its command line interface, e.g. Docker supports this by setting the entrypoint option. The files in the HAP can then be opened or copied to the mapped volumes with shell commands or scripts. A specific implementation of a HAP may choose to streamline such a process with scripts and applicable user documentation.

## Table of Important Paths

| Path | Purpose |
|---|---|
| `/etc/holoscan/` | HAP manifests and immutable configuration files. |
| `/etc/holoscan/app.json` | Application Manifest file. |
| `/etc/holoscan/pkg.json` | Package Manifest file. |

| | |
|---|---|
| `/opt/holoscan/app/` | Application code, scripts, and other files. |
| `/opt/holoscan/models/` | AI models. Each model should be in a separate sub-folder. |
| `/opt/holoscan/docs/` | Documentation, licenses, EULA, changelog, etc... |
| `/var/holoscan/` | Default working directory. |
| `/var/holoscan/input/` | Default input directory. |
| `/var/holoscan/output/` | Default output directory. |
| `/var/run/holoscan/export/` | Special case folder, causes the Script to export contents related to the app. (see: Manifest Export) |
| `/var/run/holoscan/export/app/` | Special case folder, causes the Script to export the contents of `/opt/holoscan/app/` to the folder. |
| `/var/run/holoscan/export/config/` | Special case folder, causes the Script to export `/etc/holoscan/app.json` and `/etc/holoscan/pkg.json` to the folder. |
| `/var/run/holoscan/export/models/` | Special case folder, causes the Script to export the contents of `/opt/holoscan/models/` to the folder. |

# Operating Environments

Holoscan SDK supports the following operating environments.

| Operating Environment Name | Characteristics |
|---|---|
| AGX Devkit | Clara AGX devkit with RTX 6000 dGPU only |
| IGX Orin Devkit | Clara Holoscan devkit with A6000 dGPU only |
| IGX Orin Devkit - integrated GPU only | IGX Orin Devkit, iGPU only |
| IGX Orin Devkit with discrete GPU | IGX Orin Devkit, with RTX A6000 dGPU |
| Jetson AGX Orin Devkit | Jetson Orin Devkit, iGPU only |
| Jetson Orin Nano Devkit | Jetson Orin Nano Devkit, iGPU only |

| | |
|---|---|
| X86_64 | dGPU only on Ubuntu 18.04 and 20.04 |