Nokia ADF Spec

FN BBD

Exported on 09/23/2021

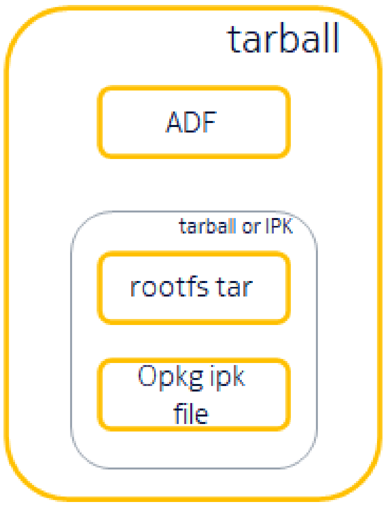
Table of Contents

1 Introduction 3
2 Notational Conventions 4
3 Configuration 5
3.1 annotations 5
3.2 hostname 6
3.3 config 6
3.4 linux 6
3.5 mounts 7
3.6 network 8
3.7 hooks 8
4 Example to make the APP package with ADF 10

1 Introduction

Nokia ADF(Application Description File) is combined with 3rd-party APP package. Application Description File is a json file with the file name of "ADF". It’s used to describe the content in the tarball and 3rd party APP runtime environment requirements.

Note: Tarball format is tar.gz.



2 Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119[1].

1 http://tools.ietf.org/html/rfc2119

3 Configuration

ADF contains metadata necessary to configure the container for 3rd-party APP. The configuration includes privilege or non-privilege container, container network type, container hooks, etc. Each 3rd-party APP can customize its runtime environment via ADF configuration.

Below is a detailed description of each field defined in ADF and valid values are specified.

3.1 annotations

annotations(object, REQUIRED) specifies the basic info and configuration of 3rd-party APP package.

- com.nokia.app_name(string, REQUIRED) 3rd-party APP name which will be displayed in data model "SoftwareModules.DeploymentUnit.{i}.Name" and "SoftwareModules.ExecutionUnit.{i}.Name".
- com.nokia.embedded_rootfs(bool, REQUIRED) If true then 3rd-party APP tarball must contain its rootfs tar.
- com.nokia.app_version(string, REQUIRED) 3rd-party APP version which will be displayed in data model "SoftwareModules.DeploymentUnit.{i}.Version".
- com.nokia.unprivileged(bool, OPTIONAL) If true then 3rd-party APP will be run in unprivileged container, defaults to true.
    - Privileged container configuration:
        - root right
        - can share the network with host
    - Unprivileged container configuration:
        - non-root right
        - can't share the network with host(network type must be "loopback" or "bridge")
        - must have embedded rootfs (com.nokia.embedded_rootfs must be "true")
- com.nokia.embedded_package(object, REQUIRED) Specify the information of the embedded package in tarball
    - name(string, REQUIRED) 3rd-party APP embedded tarball name or ipk name which will be used during installation and upgrade.
    - type(string, REQUIRED) 3rd-party APP package type: "tar" or "ipk".

Example

```
"annotations": {
“com.nokia.app_name”:”Speed test APP”,
"com.nokia.embedded_rootfs ": true,
"com.nokia.app_version": “1.0”,
“com.nokia.unprivileged”:false,
“com.nokia.embedded_package”: {
   “name”:”speed_test.tar.gz",
    “type”: “tar”
 }
},
```

3.2 hostname

hostname (string, OPTIONAL) specifies the container's hostname as seen by processes running inside the container. Defaults to empty string.

Example

"hostname": "NokiaRG"

3.3 config

config(object, REQUIRED) The basic initialization configuration of 3rd-party APP

- Env(array of strings, OPTIONAL) Entries are in the format of `VARNAME=VARVALUE`. These values act as defaults and are merged with any specified when creating a container. Defaults to empty array.
- Cmd(array of strings, REQUIRED) The executables to run in the container. Currently limit to only one string in array.

Example

```
“config”: {
   “Env”: [
      “PATH=/usr/local/sbin:/usr/local/bin[2] “,
      “LD_LIBRARY_PATH=/usr/lib“
    ],
    “Cmd”: [
       “/usr/local/sbin/speedtest_init”
     ]
},
```

3.4 linux

linux(object, OPTIONAL) specifies linux container configuration

- resources(object, OPTIONAL) specifies resource limitation of the container
  - memory(object, OPTIONAL) represents the cgroup subsystem `memory` and it's used to set limits on the container's memory usage. For more information, see the kernel cgroups documentation about memory[3].
    - limit(string, OPTIONAL) - sets limit of memory usage. Defaults to `"10M"`.
  - cpu(object, OPTIONAL) represents the cgroup subsystems `cpu` and `cpusets`. For more information, see the kernel cgroups documentation about cpusets[4].

2 http://sbin/usr/local/bin
3 https://www.kernel.org/doc/Documentation/cgroup-v1/memory.txt
4 https://www.kernel.org/doc/Documentation/cgroup-v1/cpusets.txt

- quota (string, OPTIONAL) - specifies the total amount of time in microseconds for which all tasks in a cgroup can run during one period (as defined by **`period`** below). Defaults to "5".
- period (string, OPTIONAL) - specifies a period of time in microseconds for how regularly a cgroup's access to CPU resources should be reallocated (CFS scheduler only). Defaults to "100".

Example

```
“linux”: {
  “resources”: {
     “memory”: {
       “limit”:"20M"
      },
     “cpu”: {
         “quota”:"5",
         “period”:"100"
      }
   }
},
```

3.5 mounts

mounts (array of objects, OPTIONAL) specifies additional mounts beyond `root`[5]. The runtime MUST mount entries in the listed order.

- destination (string, REQUIRED) Destination of mount point: path inside container. This value MUST be an absolute path.
- source (string, REQUIRED) A device name, but can also be a file or directory name for bind mounts or a dummy. Path values for bind mounts are either absolute or relative to the container.
- options (array of strings, OPTIONAL) Mount options of the filesystem to be used. For linux, the supported options are listed in the mount(8)[6] man page. Note both filesystem-independent[7] and filesystem-specific[8] options are listed. Defaults to empty array.
- automatically mount Temporary Storage as /temporary-data and Permanent Storage as /persistent-data for all containers.

Example

```
“mounts”: [
   {
    “destination”:”/data”,
    “source”:”/testing”,
```

5 https://github.com/opencontainers/runtime-spec/blob/master/config.md#root
6 http://man7.org/linux/man-pages/man8/mount.8.html
7 http://man7.org/linux/man-pages/man8/mount.8.html#FILESYSTEM-INDEPENDENT_MOUNT_OPTIONS
8 http://man7.org/linux/man-pages/man8/mount.8.html#FILESYSTEM-SPECIFIC_MOUNT_OPTIONS

```
     "options": ["rbind","rw"]
    }
],
```

3.6 network

network (object, OPTIONAL) specifies the container's network settings.

- type(string, OPTIONAL) specifies what network access the container is allow. There are 3 types of network access to be selected. Defaults to "loopback".
    - share: The container shares host's network and host's network is visible and accessible for the container
    - bridge: The container is connected with host's bridge and works as one LAN side device of host.
    - loopback: The container will only see a loopback device `lo`, which will only loopback inside the container. Host's network is invisible and inaccessible for the container.

Example

```
“network”: {
      “type”:”share”
} ,
```

3.7 hooks

hooks (object, OPTIONAL) specifies the container's hooks which will be invoked by host

- prestart(array of objects, OPTIONAL) specifies the hooks which will be invoked by host before starting the container.
    - path(string, OPTIONAL) specifies the path of hook. Defaults to empty string.
- poststop(array of objects, OPTIONAL) specifies the hooks which will be invoked by host after stopping the container
    - path(string, OPTIONAL) specifies the path of hook. Defaults to empty string.

Example

```
"hooks":{
  "prestart":[
   {
       "path": "/usr/bin/fix-mounts"
    }
],
"poststop":[
   {
        "path": "/usr/sbin/cleanup.sh"
    }
]
}
```

4 Example to make the APP package with ADF

for example: calibration-router.tar.gz which includes ADF and calibration-router_99.9.99-9_ipq.ipk

1. create temporary directory: mkdir tmp
2. copy ADF to tmp: cp ADF tmp/
3. copy APP image to tmp: cp calibration-router_99.9.99-9_ipq.ipk tmp/
4. Make APP tarball: cd tmp; tar cvfz ../calibration-router.tar.gz ADF calibration-router_99.9.99-9_ipq.ipk