LXC Application Development on Cisco IR829 IOx

Since the version of IOx SDK 1.0.1.0, it supports to generate the Container Style (LXC) applications. And meanwhile, the image of IR829 should be upgrade as well with the IOx bundle image of "ir800-universalk9-bundle.SPA.156-2.T" and GOS image of "ir800-ioxvm-1.1.0.4-T.bin". LXC on IR809 is exactly same with the LXC on IR829. You may download the related installation packages from CCO below.

- IOx Bundle Image 15.6.2T for IR829: https://software.cisco.com/download/release.html?mdfid=286287074&flowid=75322&softwareid=280805680&release=15.6.2T&relind=AVAILABLE&rellifecycle=ED&reltype=latest
- IOx VM (GOS) Image 1.1.0 for IR829: https://software.cisco.com/download/release.html?i=!y&mdfid=286287094&softwareid=286306224&release=1.1.0&os
- IOx SDK 1.1.0: https://software.cisco.com/download/release.html?i=!y&mdfid=286306005&softwareid=286306230&release=1.1.0&os=

1. PREREQUISITE

1.1. IOX SDK INSTALLATION

Since the version of IOx SDK 1.0.1.0, it supports to generate the Container Style (LXC) applications.

Download and install the latest SDK 1.1.0, please refer to https://developer.cisco.com/site/iox/documents/developer-guide/?ref=quickstart for how to install IOx SDK (browse "IOx SDK" → "Installation" to get it.)

There will be several examples in the folder of "/opt/ioxsdk-1.1.0/demo-apps". fp01-03 are the examples for PaaS style applications, while nt01-nt07 are the examples for Container style (LXC) applications. Here we will use nc02 for C program as example.

```
cisco@cisco-iox-sdk-1:/opt/ioxsdk-1.1.0/demo-apps $ ls -al
total 72
drwxr-xr-x 15 fogdir fogdir 4096 Apr 11 12:47 .
drwxr-xr-x 11 root   root   4096 Apr  6 08:14 ..
drwxr-xr-x  4 fogdir fogdir 4096 Apr  6 08:10 fp01-java
drwxr-xr-x  4 fogdir fogdir 4096 Apr  6 08:11 fp02-python
drwxr-xr-x  5 fogdir fogdir 4096 Apr  6 08:13 fp03-python_mods
```

```
-rw-r--r--  1 fogdir fogdir 4206 Apr  5 12:44 Makefile
drwxr-xr-x  3 fogdir fogdir 4096 Apr 11 14:54 nt01-python_simple
drwxr-xr-x  3 fogdir fogdir 4096 Apr 11 12:16 nt02-c_simple
drwxr-xr-x  3 fogdir fogdir 4096 Apr 11 14:57 nt03-python_mods
drwxr-xr-x  3 fogdir fogdir 4096 Apr  5 12:44 nt04-cpp
drwxr-xr-x  3 fogdir fogdir 4096 Apr  6 14:47 nt05-ssh
drwxr-xr-x  3 fogdir fogdir 4096 Apr  5 12:44 nt07-connected_machine
-rw-r--r--  1 fogdir fogdir 3492 Apr  5 12:44 README
drwxr-xr-x  2 fogdir fogdir 4096 Apr  6 17:34 shared
```

1.2. IR829 IMAGE UPGRADE

To support LXC applications, the image of IR829 should be upgrade as well with the IOx bundle image of “ir800-universalk9-bundle.SPA.156-2.T” and GOS image of “ir800-ioxvm-1.1.0.4-T.bin”.

Download and upgrade the IR829 image with Bundle Image: ir800-universalk9-bundle.SPA.156-2T in CCO (https://software.cisco.com/download/release.html?mdfid=286287074&flowid=75322&softwareid=280805680&release=15.6.2T&relind=AVAILABLE&rellifecycle=ED&reltype=latest) and GOS Image: ir800-ioxvm-1.1.0.4-T.bin Please refer to https://developer.cisco.com/site/iox/documents/developer-guide/?ref=quickstart for how to upgrade IR829 image (browse “IOx Enabled Devices” → “IR8xx Platforms” to get it.)

2. PRECEDURE

Here we will use nc02 for C program as example. Here are the detailed steps to generate and install the LXC application packages for IR829 IOx.

STEP 1: go to /opt/ioxsdk-1.1.0/demo-apps/ and reuse "nt02-c_simple” for example. You may copy it and rename it to “example” as an example with the command of “cp -r /opt/ioxsdk-1.1.0/demo-apps/nt02-c_simple /opt/ioxsdk-1.1.0/demo-apps/example”. There is one C file of “display-date-time.c” in the project. You may modify your program if necessary.

STEP 2: Check the files of app-lxc.yaml and iox-project-lxc.conf. Here are the example in ./src. You may modify the related description if necessary.

app-lxc.yaml:

```
descriptor-schema-version: "2.0"

info:
  name: ${APP_NAME}
  description: "Simple C Application"
  version: "0.1"
  author-link: "http://www.cisco.com"
  author-name: "Cisco Systems"

app:
```

```
type: lxc
cpuarch: ${CPU_ARCH}
kernel-version: "${KERNEL_VERSION}"

resources:
  profile: custom
  cpu: 500
  memory: 64
  disk: 50

  network:
    -
      interface-name: eth0
# Specify runtime and startup
startup:
  rootfs: app.ext2
  target: /sbin/init
```

iox-project-lxc.conf

```
{
    "//": "This is a sample project for a Native LXC C Application",
    "application": {
            "//1:": "Application descriptor location",
            "descriptor": "app-lxc.yaml",
            "//2:": "Type of application packaging",
            "type": "lxc"
    },
    "images": [
            {
                    "//0": "Identifier to build this image",
                    "id": "isr800-lxc",
                    "//1": "Final package name for this application",
                    "pkg-name": "application.tar",
                    "//2": "Machine this image is for",
                    "machine": "isr800-lxc",
                    "//3": "Linux distro to use",
                    "distro": "yocto-1.7",
                    "//4": "Location of the application data to be included in the artifacts archive",
                    "payload": "../out/isr800-lxc/pkg-staging/app.ext2"
            },
            {
                    "//0": "Identifier to build this image",
                    "id": "ir800-lxc",
                    "//1": "Final package name for this application",
                    "pkg-name": "application.tar",
                    "//2": "Machine this image is for",
                    "machine": "ir800-lxc",
                    "//3": "Linux distro to use",
                    "distro": "yocto-1.7",
```

```
                "//4": "Location of the application data to be included in the artifacts archive",
                "payload": "../out/ir800-lxc/pkg-staging/app.ext2"
            }
    ]
}
```

STEP 3: vi /opt/ioxsdk-1.1.0/demo-apps/example/Makefile, and add the related binaries and libraries for copying the related files to LXC. And set “YOCTO_MACHINE ?= ir800-lxc” and “YOCTO_MIRROR” as the specific folder to save package downloading time if you have the mirror files for Yocto.

Makefile:

```
# Select the target machine to build this application for
YOCTO_MACHINE ?= ir800-lxc

# Yocto mirror location (optional)
YOCTO_MIRROR ?= /your/mirror/path/if/have

# Set of source files to compile
SOURCES := \
        display-date-time.c

# Command set to use to generate a new LXC rootfs
ROOTFS_REPACK_SCRIPT = \
    $(IMG_BLDR) -c extract -d $(ROOTFS_STAGING_DIR) -i $(abspath $<) && \
  cp $(PROGRAM_BIN) $(ROOTFS_STAGING_DIR)/usr/bin && \
     $(IMG_BLDR) -c pack -p $(YOCTO_PROJECT_DIR) -d $(ROOTFS_STAGING_DIR) -r 0.5 -a 1 -i $@
```

If you are trying to start the applications automatically when the LXC starts, please create the script and put them the project folder. Here is the example script for the example.sh. Here it supposes you have your own application named as “examplebin”. And it runs “examplebin start/stop” to start/stop the service/process of binary of “examplebin”. For example, run “./examplebin start” to start the program, and “./examplebin stop” to stop the program.

```
#! /bin/sh
# /etc/init.d/example
case "$1" in
start)
  echo "Starting examplebin"
  # run application you want to start
  /usr/bin/examplebin start
  ;;
stop)
  echo "Stopping examplebin"
  # kill application you want to stop
  /usr/bin/examplebin stop
  ;;
*)
```

```
    echo "Usage: /usr/bin/examplebin {start|stop}"
    exit 1
    ;;
esac
```

Makefile (for automatic start of “examplebin”):

```
# Command set to use to generate a new LXC rootfs
ROOTFS_REPACK_SCRIPT = \
        $(IMG_BLDR) -c extract -d $(ROOTFS_STAGING_DIR) -i $(abspath $<) && \
    cp $(PROGRAM_BIN) $(ROOTFS_STAGING_DIR)/usr/bin && \
    cp examplebin $(ROOTFS_STAGING_DIR)/usr/bin && \
    cp example.sh $(ROOTFS_STAGING_DIR)/etc/init.d && \
    ln -s ../init.d/example.sh $(ROOTFS_STAGING_DIR)/etc/rc2.d/S100example && \
    ln -s ../init.d/example.sh $(ROOTFS_STAGING_DIR)/etc/rc3.d/S100example && \
    ln -s ../init.d/example.sh $(ROOTFS_STAGING_DIR)/etc/rc4.d/S100example && \
    ln -s ../init.d/example.sh $(ROOTFS_STAGING_DIR)/etc/rc5.d/S100example && \
        $(IMG_BLDR) -c pack -p $(YOCTO_PROJECT_DIR) -d $(ROOTFS_STAGING_DIR) -r 0.5 -a 1 -i $@
```

STEP 4: run “. /opt/ioxsdk-1.1.0/SOURCEME”, go to “/opt/ioxsdk-1.1.0/demo-apps/example", and run “make”. And then the application package of “example_ir800-lxc.tar” will be generated in ./out; It may take much time to generate the application package as it needs to download many packages from Internet. Please set the related proxy for HTTP, HTTPS, FTP in your intranet if necessary. Here is an example.

```
export http_proxy=http://proxy1.cisco.com:80
export https_proxy=http://proxy1.cisco.com:80
export ftp_proxy=http://proxy1.cisco.com:80
export no_proxy=localhost,127.0.0.1,.cisco.com
```

STEP 5: install the LXC application package with IOx Local Manager or Fog Director, activate it, and start it. For details of using Local Manger or Fog Director, please refer to http://www.cisco.com/c/en/us/td/docs/routers/access/800/software/guides/iox/lm/reference-guide/1-0/iox_local_manager_ref_guide.html or http://www.cisco.com/c/en/us/td/docs/routers/access/800/software/guides/iox/fog-director/reference-guide/1-0/fog_director_ref_guide.html.

STEP 6: Go the App-Info page of the application, put the RSA Private Key in the example.pem file, SSH the LXC with “ssh -p {SSH_PORT} -i example.pem appconsole@10.10.69.120” as example, you may run “display-date-time.bin” (or /usr/bin/display-date-time.bin) to run the program.

If you are using the program of “examplebin” which will be started automatically in LXC, you will see the process is started successfully with the command of “ps”.

STEP 7: go to IOS of IR829 to configure the PAT port for listening if necessary.