WNDRVR
SUPPORT NETWORK

WIND RIVER HELIX DEVICE CLOUD AGENT CONFIGURATION AND BUILD GUIDE, 2.3

Corporate Headquarters

Wind River
500 Wind River Way
Alameda, CA 94501-1153
U.S.A.
Toll free (U.S.A.): +1-800-545-WIND
Telephone: +1-510-748-4100
Facsimile: +1-510-749-2010

For additional contact information, see the Wind River website:

www.windriver.com

For information on how to contact Customer Support, see:

www.windriver.com/support

Wind River Helix Device Cloud Agent Configuration and Build Guide, 2.3

27 January 2019

1. WIND RIVER HELIX DEVICE CLOUD AGENT CONFIGURATION AND BUILD GUIDE, 2.3

2. CONFIGURATION AND BUILD OVERVIEW

Before you develop applications on either the host or the target for Wind River operating systems, you must build a target image that contains the Helix Device Cloud agent.

This guide contains information about building the agent for the following operating systems:

- Wind River Linux 7.0 and Wind River Linux 8.0
- Wind River Intelligent Device Platform XT
- Wind River VxWorks 7

Before using this guide, ensure that you installed the required software on your host computer. For more information, see *Wind River Helix Device Cloud Getting Started*.

Network Configuration

To enable your device to connect to the server without a proxy server, you must ensure that outbound port 443 is open on your network.

To connect to the server through a proxy server, you need your proxy information. The agent supports HTTP and SOCKS5 proxy protocols.

Certificates and Credentials

You receive the following credentials for Helix Device Cloud:

- your tenant administrator user name and password

You need this information to log in to the following:

the management console at https://www.helixdevicecloud.com
the administration utility at https://admin.helixdevicecloud.com

You need to sign in to the management console to confirm that your device connects to the server after it boots.

For more information, see *Wind River Helix Device Cloud Getting Started*.

3. WHERE TO FIND INFORMATION

Documentation is available through the Wind River Knowledge Library.

The following documentation is available:

Wind River Helix Device Cloud Documentation

Wind River Helix Device Cloud Getting Started

Provides instructions to install the HDC agent on a host computer and information about getting started with the platform.

Wind River Helix Device Cloud Release Notes

?Provides general product information about Helix Device Cloud, changes in this release, usage caveats, and known problems.

Wind River Helix Device Cloud Management Console User's Guide

Provides information about performing device and platform management tasks.

Wind River Helix Device Cloud Administration Utility User's Guide

Provides information about performing platform administration tasks.

Wind River Helix Device Cloud Platform Programmer's Guide

Provides information about writing Web-based applications for Helix Device Cloud.

Wind River Helix Device Cloud Interactive REST API Reference

Provides reference information for the platform REST API and an environment to explore the APIs.

Wind River Helix Device Cloud Agent Programmer's Guide

Provides instructions for configuring devices to run the agent and to write Helix Device Cloud applications.

Wind River Helix Device Cloud Agent Configuration and Build Guide

Provides instructions to build images that include the agent for target devices that run Wind River operating systems.

Wind River Helix Device Cloud Troubleshooting Guide

Provides information to help resolve issues with Helix Device Cloud.

Wind River Helix Device Cloud Agent API Reference

Provides reference information for the agent API.

Wind River Linux Documentation

The following documents are available for both Wind River Linux 7.0 and 8.0:

?Wind River Linux Getting Started Guide

?Provides instructions for creating, modifying, deploying, and debugging platform and application projects using the command-line and Workbench.

Wind River Linux Platform Developer's Guide

Provides information about ?command-line instructions for configuring, building, and developing platform projects as well as detailed information on the development environment and build system.

?Wind River Linux Getting Started Workbench Tutorials

?Provides procedures and examples for using Workbench to configure, build, and debug Wind River Linux application, platform, and kernel module projects.

Wind River Linux User Space Developer's Guide

Provides information about using the Wind River Linux SDK to develop Linux user space applications.

Wind River Intelligent Device Platform XT

The following documents are available for Wind River IDP XT:

Wind River Linux Intelligent Device Platform XT Programmer's Guide, 3.1

?Provides instructions for installing and configuring IDP XT and modifying it for your specific requirements.

?Wind River Intelligent Device Platform XT Security Guide, 3.1

Provides guidance on performing a security analysis and matching IDP XT capabilities with assessed needs.

Wind River Intelligent Device Platform XT Release Notes, 3.1

?Provides general product information, changes in this release, usage caveats, and known problems.

VxWorks 7

The following documents are available for VxWorks 7:

VxWorks 7 Getting Started Guide

Provides information about getting started with development on VxWorks 7.

VxWorks 7 Release Notes

Provides general product information, changes in the VxWorks 7 components, and known problems.

VxWorks 7 Configuration and Build Guide

Provides information about instructions for configuring, building, and developing platform projects as well as detailed information on the development environment and build system.

Accessing Documentation

To access the Helix Device Cloud documentation on the Knowledge Library (http://knowledge.windriver.com), select **Products > Internet of Things > Helix Device Cloud 2**.

To access the interactive REST API Reference, go to https://www.helixdevicecloud.com, click the **Help** button in the upper-right corner, and select **REST API Reference**.

4. BUILDING AND BOOTING FOR WIND RIVER LINUX AND IDP XT

1. Building and Booting for Wind River Linux and IDP XT

You build a platform project on your development host to create an image that you can use to boot your device.

Platform Project Configuration

The file system you build contains the supported operating system and the agent.

You need the following configuration settings:

- the **wr-iot-meta** and **wr-iot-dl** layers

Optionally, you can also enable the following:

- the **nmap** package to enable remote login through a proxy server
- the **wr-iot-apps** layer (for details about the examples provided in this layer, see the *projDir* **/layers/wr-iot-apps/README.md** file after you run the configure command with the layer included)
- the **feature/self-hosted** template and **iot-dev** package to enable development on the target and include the required files if you export the SDK for application development

For information about exporting the SDK, see the following:

Wind River Linux Platform Developer's Guide, 7.0: *Configuration and Build*
Wind River Linux Platform Developer's Guide, 8.0: *Configuration and Build*

For information about supported development hosts for Wind River Linux, see the following:

Wind River Linux System Requirements - Recommended Hosts List, 7.0
Wind River Linux System Requirements - Recommended Hosts List, 8.0

For information about supported targets for IDP XT, see the following:

Wind River Intelligent Device Platform XT Release Notes, 3.1

Agent Build Configuration

As part of the configuration process, run the iot-config.py script to specify configuration for the agent to include in the rootfs image. You can specify the agent authentication file, **startup.bin**, that contains your tenant-specific authentication information to enable the agent to connect to the server when the device first boots. For more information about the options available in the iot-config.py script, see iot-config.py on page 26.

Agent Run-time Configuration

After the device boots, you can run the iot-control command to configure the agent to connect to the server through a proxy server. If you include the **nmap** package in your platform project, the agent configuration automatically routes remote login connections through the proxy server. If you install the package after you boot the device, you must run the iot-control command to specify the proxy information.

You can also run the iot-control command to install the **startup.bin** file if you did not do so during the configuration process when you built your platform project. If you have an administrator account for Helix Device Cloud, you can provide your account information to download the file or you can sign in to the management console and download the file from your profile page (see Signing in to the HDC Management Console). Otherwise, contact your tenant administrator to get the file. If you do not provide your account information when you run iot-control, you must copy the file to the device first using a program such as scp.

Building and Booting Workflow

1. Configure the platform project.
2. Configure the agent.
3. Build the platform project.
4. Deploy the images to the boot media.
5. Boot the device.
6. (Optional) Specify proxy information.
7. (Optional) Install the **startup.bin** file on the device.
8. Confirm that the device successfully registered with the server.

2. About the wrenv.sh Script for Wind River Linux and IDP XT

The wrenv.sh script sets all the Wind River Linux-related environment variables, including the path to the configure command for platform projects.

The command to run the script is the following, where *releaseNumber* is the release of Wind River Linux:

```
$ installDir/wrenv.sh -p wrlinux-releaseNumber
```

If you choose not to run wrenv.sh, you must explicitly specify the full path to the configure command instead of just specifying **$WIND_LINUX_CONFIGURE_CLI** in your configure line. The full path is similar to *installDir***/wrlinux-**_releaseNumber_**/wrlinux/configure**.

All the examples in this guide for Wind River Linux and Wind River IDP XT assume that you have executed wrenv.sh.

3. About the deploy.sh Script

After you build your Wind River Linux platform project, the deploy.sh script is available on your host computer to deploy your images to your boot media with support for software factory reset.

The deploy.sh script is only supported for Wind River Linux and Wind River IDP XT.

The deploy.sh script contains example instructions to copy your images onto your boot media and create a factory default image. You can use the script as provided, modify it as needed, or use a different method to create your boot media.

If you choose not to use the deploy.sh script and you want to support the **Restore Factory Images** (also referred to as software factory reset) action from the server, you must implement it in your own solution using the same approach as the script.

NOTE: If you use the deploy.sh script, you cannot disable the creation of a factory default image.

For more information about the script, change to your *projDir* directory and execute ./deploy.sh -h.

The -b option of the deploy.sh script is required for ARM targets. For more information, see Deploying Your Wind River Linux Platform Project to an ARM Target Device on page 13.

The recommended options are -s 8G -f *rootfsFilename* -u -y -F *rootfsFilename*. For more details about the options, see deploy.sh Script Options for HDC on page 26.

4. Building Your Wind River Linux Platform Project for Intel Target Devices

?The platform project for your board creates a file system, kernel image, and boot loader.

The instructions in this section provide the steps to build an image for an Intel target device with the minimal configuration required to run the agent under Wind River Linux.

The steps in this section use Intel Bay Trail and Intel Quark boards as examples. For a list of the boards available for the --enable-board option for your specific installation of Wind River Linux, execute $WIND_LINUX_CONFIGURE_CLI -h.

For more information about the additional configuration options available in Wind River Linux, see the following:

Wind River Linux Platform Developer's Guide, 7.0
Wind River Linux Platform Developer's Guide, 8.0

Prerequisites

You must have installed Wind River Linux with the HDC agent on your host computer. For more information, see the following sections in *Wind River Helix Device Cloud Getting Started*:

- Installing the HDC Agent Using the GUI Installer
- Installing the HDC Agent on Using the Maintenance Installer GUI
- Installing the HDC Agent from the Command-Line

You must have the **startup.bin** file, which contains tenant-specific authentication information that your device uses to connect to the server and enables the server to provide your device with a unique identifier. If you have an administrator account, you can download it from your profile page on the management console. For more information, see Signing in to the HDC Management Console. Otherwise, contact your tenant administrator to get the file.

NOTE: You must ensure that the **startup.bin** file has read permissions for all users.

You need the following additional packages on your Linux host:

- bc
- bsdmainutils
- kpartx
- lvm2
- rpm2cpio

Procedure

1. Create a new directory *projDir* for your platform project and configure the platform project for your board.

   Add --with-layer=wr-iot-meta,wr-iot-dl,wr-iot-apps to the $WIND_LINUX_CONFIGURE_CLI command for your board.

   Optionally, add --with-template=feature/self-hosted and --with-package=iot-dev.

   Optionally, add --with-package=nmap to enable remote login through a proxy server.

   For example, for an Intel Bay Trail board, do the following:

   ```
   $ mkdir -p projDir
   $ cd projDir
   $ $WIND_LINUX_CONFIGURE_CLI --enable-board=intel-baytrail-64 \
   --enable-kernel=standard --enable-rootfs=glibc-std \
   --with-layer=wr-iot-meta,wr-iot-dl,wr-iot-apps \
   --enable-reconfig
   ```

   For example, for an Intel Quark board, do the following:

   ```
   $ mkdir -p projDir
   $ cd projDir
   $ $WIND_LINUX_CONFIGURE_CLI --enable-board=intel-quark \
   --enable-kernel=standard --enable-rootfs=glibc-std \
   --with-layer=wr-iot-meta,wr-iot-dl,wr-iot-apps \
   --enable-reconfig
   ```

   For example, for an Intel Bay Trail board to include the files for development on the target and in the SDK, do the following:

   ```
   $ mkdir -p projDir
   $ cd projDir
   $ $WIND_LINUX_CONFIGURE_CLI --enable-board=intel-baytrail-64 \
   --enable-kernel=standard --enable-rootfs=glibc-std \
   --with-layer=wr-iot-meta,wr-iot-dl,wr-iot-apps \
   --with-template=feature/self-hosted --with-package=iot-dev \
   --enable-reconfig
   ```

   The files for the agent are now available in the following directories:

   projDir **/layers/wr-iot-meta**
   projDir **/layers/wr-iot-dl**

   The source files for the example applications included in the wr-iot-apps layer are now available in the following directory:

projDir **/layers/wr-iot-apps**

2. To configure the agent, run the iot-config.py script.

   You must run the iot-config.py script from your project directory (*projDir*) after the $WIND_LINUX_CONFIGURE_CLI command and before the make command.

   This example ensures that the **startup.bin** file has read permissions for all users and configures the agent with the mandatory tenant-specific information.

   ```
   $ cd projDir
   $ chmod 644 /pathTo/startup.bin
   $ layers/wr-iot-meta/scripts/iot-config.py \
   -s 'SECURITY_BUNDLE:/pathTo/startup.bin'
   ```

   For information about additional configuration options, see iot-config.py on page 26.

3. Build the platform project for your board.

   ```
   $ make
   ```

   The files for the root file system, kernel image, and boot loader are now available in the following directory:

   projDir **/export/images**

4. (Optional) Export the SDK for application development.

   ```
   $ make export-sdk
   ```

   The shell script to extract the SDK is now available in the *projDir* **/export** directory. The file name is target specific and ends with **-sdk.sh**.

You are ready to your prepare your boot media and boot your device.

5. Deploying Your Wind River Linux Platform Project for Intel Target Devices

During the deployment process, you create your boot media and use it to boot your board.

You can deploy your platform project to any Intel target device supported by Wind River Linux.

The instructions in this section use a USB flash drive for deployment.

Prerequisites

Before you create the boot media and boot your device, you need the following:

- boot media with at least eight GB capacity
- the images generated from your platform project (see Building Your Wind River Linux Platform Project for Intel Target Devices on page 7)
- outbound port 443 open on the firewall of the network to which the device connects if you do not use a proxy server

Procedure

1. Copy the images from your host computer to a USB flash drive.

   The following example uses the deploy.sh script with the options for an Intel Bay Trail board, and ?*yourDevice* is the location of your USB flash drive in the file system on your host, for example, **sdb**.

   ```
   $ cd projDir /export
   $ sudo ../deploy.sh \
   -s 8G -f intel-baytrail-64-glibc-std-standard-dist.tar.bz2 \
   -u -y -F intel-baytrail-64-glibc-std-standard-dist.tar.bz2 \
   -d /dev/yourDevice
   ```

   The following example shows the options for an Intel Quark board:

   ```
   $ cd projDir /export
   $ sudo ../deploy.sh \
   -s 8G -f intel-quark-glibc-std-standard-dist.tar.bz2 \
   -u -y -F intel-quark-glibc-std-standard-dist.tar.bz2 \
   -d /dev/yourDevice
   ```

2. Connect your device to a display device compatible with its display port.
3. Connect your device to the network.
4. With the device powered off, insert the USB flash drive and then power on the device.

   The device boots and the login prompt appears. The last four digits of the prompt are the last four digits of the MAC address of the device.

   ```
   Wind River Linux 7.0.0.14 WR-LX-0605 ttyS0

   WR-LX-0605 login:
   ```

If you included the **startup.bin** file in your platform project and you do not need to configure a proxy server, the agent services start and the device connects to the server.

Postrequisites

If your platform project included a **startup.bin** file, you must confirm that the device connects successfully to the server. For more information, see Confirming Successful Device Registration on Wind River Linux and IDP XT on page 19. Otherwise, follow the steps to connect your device to the server. For more information, see Connecting Your Wind River Linux and IDP XT Device to the Server on page 18.

6. Building Your Wind River Linux Platform Project for ARM Target Devices

The platform project for your board includes a file system, kernel image, and boot loader.

The instructions in this section provide the steps to build an image for an ARM target device with the minimal configuration to run the HDC agent under Wind River Linux.

For more information about the additional configuration options available in Wind River Linux, see the following:

Wind River Linux Platform Developer's Guide, 7.0
Wind River Linux Platform Developer's Guide, 8.0

The steps in this section use Microzed and BeagleBone Black boards as examples. For a list of the ARM boards available for the --enable-board option, execute $WIND_LINUX_CONFIGURE_CLI -h.

Prerequisites

You must have installed Wind River Linux with the HDC agent on your host computer. For more information, see the following sections in *Wind River Helix Device Cloud Getting Started*:

- Installing the HDC Agent Using the GUI Installer
- Installing the HDC Agent on Using the Maintenance Installer GUI
- Installing the HDC Agent from the Command-Line

You must have the **startup.bin** file, which contains tenant-specific authentication information that your device uses to connect to the server and enables the server to provide your device with a unique identifier. If you have an administrator account, you can download it from your profile page on the management console. For more information, see Signing in to the HDC Management Console. Otherwise, contact your tenant administrator to get the file.

NOTE: You must ensure that the **startup.bin** file has read permissions for all users.

You need the following additional packages on your Linux host:

- bc
- bsdmainutils
- kpartx
- lvm2
- rpm2cpio

Procedure

1. Create a new directory *projDir* for your platform project and configure the platform project for your board.

   Add --with-layer=wr-iot-meta,wr-iot-dl,wr-iot-apps to the $WIND_LINUX_CONFIGURE_CLI command for your board.

   Optionally, add --with-template=feature/self-hosted and --with-package=iot-dev.

   Optionally, add --with-package=nmap to enable remote login through a proxy server.

   For example, for a MicroZed board, do the following:

   ```
   $ mkdir -p projDir
   $ cd projDir
   $ $WIND_LINUX_CONFIGURE_CLI --enable-board=xilinx-zynq \
   --enable-kernel=standard --enable-rootfs=glibc-std \
   --with-layer=wr-iot-meta,wr-iot-dl,wr-iot-apps \
   --enable-reconfig
   ```

   For example, for a BeagleBone Black board, do the following:

   ```
   $ mkdir -p projDir
   $ cd projDir
   ```

```
$ $WIND_LINUX_CONFIGURE_CLI --enable-board=ti-am335x \
--enable-kernel=standard --enable-rootfs=glibc-std \
--with-layer=wr-iot-meta,wr-iot-dl,wr-iot-apps \
--enable-reconfig
```

For example, for a BeagleBone Black board to include the files for development on the target and in the SDK, do the following:

```
$ mkdir -p projDir
$ cd projDir
$ $WIND_LINUX_CONFIGURE_CLI --enable-board=ti-am335x \
--enable-kernel=standard --enable-rootfs=glibc-std \
--with-layer=wr-iot-meta,wr-iot-dl,wr-iot-apps \
--with-template=feature/self-hosted --with-package=iot-dev \
--enable-reconfig
```

The files for the agent are now available in the following directories:

projDir **/layers/wr-iot-meta**
projDir **/layers/wr-iot-dl**

The source files for the example applications included in the wr-iot-apps layer are now available in the following directory:

projDir **/layers/wr-iot-apps**

2. To configure the agent, run the iot-config.py script.

   You must run the iot-config.py script from your project directory (*projDir*) after the $WIND_LINUX_CONFIGURE_CLI command and before the make command.

   This example ensures that the **startup.bin** file has read permissions for all users and configures the agent with the mandatory tenant-specific information.

   ```
   $ cd projDir
   $ chmod 644 /pathTo/startup.bin
   $ layers/wr-iot-meta/scripts/iot-config.py \
   -s 'SECURITY_BUNDLE:/pathTo/startup.bin'
   ```

   For information about additional configuration options, see iot-config.py on page 26.

3. Build the platform project for your board.

   ```
   $ make
   ```

   The files for the root file system, kernel image, and boot loader are now available in the following directory:

   projDir **/export/images**

4. (Optional) Export the SDK for application development.

   ```
   $ make export-sdk
   ```

   The shell script to extract the SDK is now available in the *projDir* **/export** directory. The file name is target specific and ends with **-sdk.sh**.

You are ready to your prepare your boot media and boot your device.

7. Deploying Your Wind River Linux Platform Project to an ARM Target Device

During the deployment process, you create your boot media and use it to boot your board.

You can deploy your platform project to any ARM target device supported by Wind River Linux.

If you deploy the image to your board using the deploy.sh script, you do not need to make any additional changes.

If you choose not to use the deploy.sh script and you want to support kernel bootonce and software factory reset, you need specific U-Boot environment variables. The file **uEnv.txt** is available in *projDir* **/layers/wr-iot-meta/templates/feature/no-idp/meta/arm/recipes-kernel/linux/files/***boardname* for the MicroZed and BeagleBone Black boards. You may need to customize this file for your board and you may need to modify your U-Boot environment to read this file from the FAT partition on the boot media.

The instructions in this section use an SD card for deployment.

Prerequisites

Before you create the boot media and boot your device, you need the following:

- the images generated from your platform project (see Building Your Wind River Linux Platform Project for ARM Target Devices on page 10)
- outbound port 443 open on the firewall of the network to which the device connects if you do not use a proxy server

Procedure

1. Deploy the images to your boot media and boot your board.
    1. Copy the images from your host computer to an SD card.

       The following example uses the deploy.sh script with the options for the MicroZed board, and *yourDevice* is the location of your SD card in the file system on your host, for example, **mmcblk0**.

       ```
       $ cd projDir /export
       $ sudo ../deploy.sh
       -s 8G -f ?xilinx-zynq-glibc-std-standard-dist.tar.bz2 \
       -u -y -F xilinx-zynq-glibc-std-standard-dist.tar.bz2 \
       -d /dev/yourDevice \
       ?-b zed
       ```

    2. Connect your device to a display device compatible with its display port.
    3. Connect your device to the network.
    4. With the device powered off, insert the SD card, and then power on the device.
2. The first time you boot your device, if necessary, configure the boot loader.

   For example, the MicroZed board requires changes to the boot loader configuration, while the BeagleBone Black board does not.

    1. During the boot sequence, press any key when you see the following on the serial console:

       ```
       Hit any key to stop autoboot: 3
       ```

    2. Set up the U-boot environment to boot from the SD card and continue the boot process.

       NOTE: Ensure that you execute the following commands only the first time you boot the device. Execute the commands one at a time on the command line. Do not use a script or batch file.

```
uboot >setenv bootcmd 'fatload mmc 0:1 0x03000000 uEnv.txt; env import -t 0x03000000 $filesi
ze; run uenvcmd'
uboot >setenv last_boot_flag
uboot >saveenv
uboot> boot
```

The device boots and the login prompt appears. The last four digits of the prompt are the last four digits of the MAC address of the device.

```
Wind River Linux 7.0.0.14 WR-LX-0605 ttyS0

WR-LX-0605 login:
```

If you included the **startup.bin** file in your platform project and you do not need to configure a proxy server, the agent services start and the device connects to the server.

Postrequisites

If your platform project included a **startup.bin** file, you must confirm that the device connects successfully to the server. For more information, see Confirming Successful Device Registration on Wind River Linux and IDP XT on page 19. Otherwise, follow the steps to connect your device to the server. For more information, see Connecting Your Wind River Linux and IDP XT Device to the Server on page 18.

8. Building Your Wind River IDP XT Platform Project

The platform project for your board includes a file system, kernel image, and boot loader.

The instructions in this section provide the steps to build an image for target devices supported by Wind River IDP XT with the minimal configuration required to run the HDC agent with the default security features (**wr-srm** and **wr-mcafee-essential**) enabled.

For more information about the additional configuration options available in Wind River IDP XT and available security features, see the following:

Wind River Intelligent Device Platform XT Programmer's Guide: Building and Booting
Wind River Intelligent Device Platform XT Programmer's Guide: IDP Security

For more information about the additional configuration options available in Wind River Linux, see the following:

Wind River Linux Platform Developer's Guide, 7.0
Wind River Linux Platform Developer's Guide, 8.0

Prerequisites

You must have installed Wind River IDP XT with the HDC agent on your host computer. For more information, see the following sections in *Wind River Helix Device Cloud Getting Started*:

- Installing the HDC Agent Using the GUI Installer
- Installing the HDC Agent on Using the Maintenance Installer GUI
- Installing the HDC Agent from the Command-Line

You must have the **startup.bin** file, which contains tenant-specific authentication information that your device uses to connect to the server and enables the server to provide your device with a unique identifier. If you have an administrator account, you can download it from your profile page on the management console. For more information, see Signing in to the HDC Management Console. Otherwise, contact your tenant administrator to get the file.

NOTE: You must ensure that the **startup.bin** file has read permissions for all users.

You need the following additional packages on your Linux host:

- bc
- bsdmainutils
- kpartx
- lvm2
- rpm2cpio

Procedure

1. Create a new directory *projDir* for your platform project and configure the platform project for your board.

   Add --enable-addons=wr-idp, --with-layer=wr-iot-meta,wr-iot-dl,wr-iot-apps to the $WIND_LINUX_CONFIGURE_CLI command for your board. Optionally, add --with-template=feature/self-hosted and --with-package=iot-dev.

   Optionally, add --with-package=nmap to enable remote login through a proxy server.

   For example, for an Intel Bay Trail board, do the following:

   ```
   $ mkdir -p projDir
   $ cd projDir
   $ $WIND_LINUX_CONFIGURE_CLI --enable-board=intel-baytrail-64 \
   --enable-rootfs=idp --enable-kernel=idp --enable-addons=wr-idp \
   --with-layer=wr-iot-meta,wr-iot-dl,wr-iot-apps \
   ?--enable-reconfig
   ```

   For example, for an Intel Quark board, do the following:

   ```
   $ mkdir -p projDir
   $ cd projDir
   $ $WIND_LINUX_CONFIGURE_CLI --enable-board=intel-quark \
   --enable-rootfs=idp --enable-kernel=idp --enable-addons=wr-idp \
   --with-layer=wr-iot-meta,wr-iot-dl,wr-iot-apps \
   ?--enable-reconfig
   ```

   For example, for an Intel Bay Trail board to include the files for development on the target and in the SDK, do the following:

   ```
   $ mkdir -p projDir
   $ cd projDir
   $ $WIND_LINUX_CONFIGURE_CLI --enable-board=intel-baytrail-64 \
   --enable-rootfs=idp --enable-kernel=idp --enable-addons=wr-idp \
   --with-layer=wr-iot-meta,wr-iot-dl,wr-iot-apps \
   --with-template=feature/self-hosted --with-package=iot-dev \
   --enable-reconfig
   ```

   The files for the agent are now available in the following directories:

projDir **/layers/wr-iot-meta**
projDir **/layers/wr-iot-dl**

The source files for the example applications included in the wr-iot-apps layer are now available in the following directory:

projDir **/layers/wr-iot-apps**

The resulting configuration creates an image with the default security features enabled.

2. To configure the agent, run the iot-config.py script.

   You must run the iot-config.py script from your project directory (*projDir*) after the $WIND_LINUX_CONFIGURE_CLI command and before the make command.

   This example ensures that the **startup.bin** file has read permissions for all users and configures the agent with the mandatory tenant-specific information.

   ```
   $ cd projDir
   $ chmod 644 /pathTo/startup.bin
   $ layers/wr-iot-meta/scripts/iot-config.py \
   -s 'SECURITY_BUNDLE:/pathTo/startup.bin'
   ```

   For information about additional configuration options, see iot-config.py on page 26.

3. Build the platform project for your board.

   ```
   $ make
   ```

   The files for the root file system, kernel image, and boot loader are now available in the following directory:

   projDir **/export/images**

4. (Optional) Export the SDK for application development.

   ```
   $ make export-sdk
   ```

   The shell script to extract the SDK is now available in the *projDir* **/export** directory. The file name is target specific and ends with **-sdk.sh**.

You are ready to your prepare your boot media and boot your device.

9. Deploying Your Wind River IDP XT Platform Project

During the deployment process, you create your boot media and use it to boot your board.

You can deploy your platform project to any device supported by Wind River IDP XT.

The instructions in this section use a USB flash drive for deployment.

Prerequisites

Before you create the boot media and boot your device, you need the following:

- the images generated from your platform project (see Building Your Wind River IDP XT Platform Project on page 14)
- boot media with at least eight GB capacity
- outbound port 443 open on the firewall of the network to which the device connects if you do not use a proxy server

Before you boot your device, ensure that your device BIOS is up to date. For information about how to update the BIOS image, see *Wind River Intelligent Device Platform XT Programmer's Guide: Building and Booting*.

Procedure

1. Copy the images from your host computer to a USB flash drive.

   In the following examples, *yourDevice* is the location of your USB flash drive in the file system on your host, for example, **sdb**.

   The following example shows the options for an Intel Bay Trail board:

   ```
   $ cd projDir/export
   $ sudo ../deploy.sh
   -s 8G -f intel-baytrail-64-idp-idp-dist.tar.bz2 \
   -u -y -F intel-baytrail-64-idp-idp-dist.tar.bz2 \
   -d /dev/yourDevice
   ```

   The following example shows the options for a Cross Hill board:

   ```
   $ cd projDir/export
   $ sudo ../deploy.sh
   -s 8G -f intel-quark-idp-idp-dist.tar.bz2 \
   -u -y -F intel-quark-idp-idp-dist.tar.bz2 \
   -d /dev/yourDevice
   ```

2. Connect your device to a display device compatible with its display port.
3. Connect your device to the network.
4. With the device powered off, insert the USB flash drive, and then power on the device.

   The device boots and the login prompt appears. The last four digits of the prompt are the last four digits of the MAC address of the device.

   ```
   Wind River Linux 7.0.0.14 WR-IDP-0604 ttyS1

   WR-IDP-0604 login:
   ```

If you included the **startup.bin** file in your platform project and you do not need to configure a proxy server, the agent services start and the device connects to the server.

Postrequisites

If your platform project included a **startup.bin** file, you must confirm that the device connects successfully to the server. For more information, see Confirming Successful Device Registration on Wind River Linux and IDP XT on page 19. Otherwise, follow the steps to connect your device to the server. For more information, see Connecting Your Wind River Linux and IDP XT Device to the Server on page 18.

10. Connecting Your Wind River Linux and IDP XT Device to the Server

If your device needs to connect through a proxy server or you did not add the agent authentication file, **startup.bin**, to your platform project, you must perform additional steps to enable your device to connect to the server.

If you need to install the **startup.bin** file and you have an administrator account for Helix Device Cloud, you can provide your account information as part of this procedure, or you can sign in to the management console and download the **startup.bin** file from your profile page (see Signing in to the HDC Management Console) before you start this procedure. Otherwise, contact your tenant administrator to get the file. If you do not provide your account information, you must copy the file to the device first using a program such as scp.

Prerequisites

You need the following if you did not include the **startup.bin** in your platform project:

- the **startup.bin** file copied to the device or the user name and password for an administrator account for Helix Device Cloud
- outbound port 443 open on the firewall of the network to which the device connects if you do not use a proxy server

You need the following to configure the device to connect through a proxy server:

- configuration information for your proxy server
- the **nmap** package installed on the device

Procedure

1. In a terminal window, run the iot-control program.

```
# iot-control
Looking for configuration file: /etc/iot/iot.cfg
Looking for configuration file: /var/lib/iot/iot.cfg
Looking for configuration file: ./iot.cfg
Looking for configuration file: /usr/bin/iot.cfg
Proxy's type (none/http/socks5/Enter to skip):
```

2. Specify the proxy configuration information or press **ENTER** to skip proxy configuration.

```
Proxy's type (none/http/socks5/Enter to skip): http
Proxy host: 192.180.132.21
Proxy's port: 3128
Proxy's username: (press ENTER to skip): proxyUser
Proxy's password:
Path to local startup.bin file (press ENTER for download):
```

3. If you did not include the **startup.bin** file in your platform project, do one of the following:

Option	Description
Use a local file	Type the full path to your **startup.bin** file, for example **/home/user/startup.bin**, and press **ENTER**.
Download the file	1. Press **ENTER** 2. Type the information to download the file.

Option	Description
	`` Cloud host: www.helixdevicecloud.com Username: hdcAdminUser Password: Downloading file... Success: File saved to "/var/lib/iot/startup.bin" Stopping service: Internet of Things Mux... Skipped (not found) Stopping service: Internet of Things Device Manager... Skipped (n ot found) Stopping service: Internet of Things Core Service... Skipped (no t found) Stopping service: Internet of Things Connection Gateway... Skippe d (not found) Stopping service: Mosquitto Broker... Skipped (not found) Starting service: Mosquitto Broker... Success Starting service: Internet of Things Connection Gateway... Succes s Starting service: Internet of Things Core Service... Success Starting service: Internet of Things Device Manager... Success Starting service: Internet of Things Mux... Success Press any key to continue... ```

The agent services start and the device connects to the server.

Postrequisites

You must confirm that the device connects successfully to the server. For more information, see Confirming Successful Device Registration on Wind River Linux and IDP XT on page 19.

11. Confirming Successful Device Registration on Wind River Linux and IDP XT

After you boot your device and if necessary, complete additional steps to connect to the server, you must ensure that it registers successfully with the server.

Prerequisites

You need your Helix Device Cloud user name and password.

Procedure

1. To help you identify your device on the management console, find the device ID.

```
# iot-control
Looking for configuration file: /etc/iot/iot.cfg
Looking for configuration file: /var/lib/iot/iot.cfg
Looking for configuration file: /home/jkeffer1/iot.cfg
```

```
Looking for configuration file: /usr/bin/iot.cfg
Current device id: 5C1FB6BB-4FE8-E8FA-0FBC-AC12ABE36C18
Proxy's type (none/http/socks5/Enter to skip):
```

In this example, the device identifier is 5C1FB6BB-4FE8-E8FA-0FBC-AC12ABE36C18.

2. To exit, press **ENTER** three times.
3. If you have not already done so, sign in to the management console with your user name and password at https://www.helixdevicecloud.com.
4. Click **DEVICES**.
5. In the **Device Name, ID, MAC Address** box, begin typing the device name, device ID, MAC address, or any other device attribute until your device appears in the list.
6. In the Device ID column, click your device.

   Your device is connected to the server if **Online** appears on the device details page under the device name in the upper-left corner of the page.

You are now ready to develop applications using the agent APIs.

Postrequisites

You can run one of the available sample applications to start telemetry and run actions from the management console. For more information, see Sample Applications.

If your device does not appear on the server, see Diagnosing Initial Connectivity Failures on Wind River Linux and IDP XT.

5. CONFIGURATION AND BUILD FOR VXWORKS 7

1. Configuration and Build for VxWorks 7

To include the Helix Device Cloud agent in your VxWorks 7 image, you need to include the agent source and components in your VSB and VIP.

You need persistent, writable storage on the device to store the following files:

startup.bin

Contains tenant-specific authentication information that your device uses to connect to the server and enables the server to provide your device with a unique identifier. If you have an administrator account, you can download it from your profile page on the management console. For more information, see Signing in to the HDC Management Console. Otherwise, contact your tenant administrator to get the file.

agentguid.bin

Contains the unique identifier of the agent generated at run time when the device registers with the server. It is stored in the same directory as the **startup.bin** file.

dxlpolicy.bin

Contains certificates and other connection information data generated at run time when the device registers with the server. It is stored in the same directory as the **startup.bin** file.

iot.cfg

Contains the agent properties that appear on the management console. For details about the file contents and its use, see iot.cfg and Agent Configuration. Some configuration options do not apply to VxWorks 7.

The sample file *installDir* **/vxworks-7/pkgs/app/hdc-***releaseNum* **/agent/sample_cfg/etc/iot.cfg** is provided and you can modify it as needed.

mosquitto.conf

Contains the configuration for the MQTT broker required for communication between the agent and the server.

The sample file *installDir* **/vxworks-7/pkgs/app/mosquitto-***releaseNum* **/sample_cfg/etc/iot.cfg** is provided and you can modify it as needed.

Before you boot your board, you must copy the following files to your persistent storage in the locations you specify in your VIP parameters:

- **startup.bin**
- **iot.cfg**
- **mosquitto.conf**

Networking

You must include DNS in your VIP configuration to resolve the Helix Device Cloud domain name.

The device clock time is used to add the timestamp to telemetry data samples sent from the device to the server. The server remembers the timestamp and only updates the management console with telemetry data that has a timestamp greater than the last sample transmitted. When the device reboots, it must get the current time from an external source, such as an NTP server, or store the time to preserve the timestamp sequence. After the device boots, telemetry data does not appear on the management console until after the system clock updates to the correct time.

You can include a VIP component to configure the time using NTP or SNTP, or your BSP must support a real-time clock. For more information about DNS, NTP, and SNTP, see VxWorks 7 Networking.

Additional Information Sources

For information about supported development hosts for VxWorks 7, go to VxWorks 7 Getting Started and see the latest version of the *VxWorks 7 Release Notes*.

For information about supported targets see VxWorks 7 Processor & Device Support. The agent is only supported on Intel and ARM processors.

This document describes only the specific components you need to include the Helix Device Cloud agent in your device image. For general information about configuration and build for VxWorks 7, see the documents and tutorials available on the Knowledge Library:

VxWorks 7 Configuration and Build
Getting Started with VxWorks 7 and the Intel Galileo Gen 2 Board
Getting Started with VxWorks 7 and the BeagleBone Black Board

Basic Workflow

1. Create your VSB with the HDC layer.
2. Create your VIP with the HDC components and network components.
3. Copy the configuration files to your persistent storage.
4. Boot your board.
5. Confirm that your device registered successfully. For more information, see Confirming Successful Device Registration for VxWorks on page 24.

2. Source Build Configuration (VSB)

You must add the Helix Device Cloud layers to include the agent in your VSB.

Adding the HDC layer to your source code in your build automatically includes the following layers:

Layer	Description
HDC_AGENT HDC_DEVICE _MANAGER HDC_CCG_BROKER	Helix Device Cloud agent components for connectivity and basic device management
MOSQUITTO	MQTT broker required for communication with the server

3. Kernel Configuration Options (VIP)

You must add the Helix Device Cloud components and network components to your VIP.

The **PROFILE_HDC** profile is available for you to specify when you create your VIP.

If you do not create your VIP with the **PROFILE_HDC**, add the following components:

Component	Description
INCLUDE_HDC Automatically adds the following components: **INCLUDE_HDC_AGENT** **INCLUDE_HDC_CCG_BROKER**	Helix Device Cloud agent components for connectivity and basic device management

The following parameters specify the location and names of the HDC configuration files in the file system on your persistent storage. For example, if you use a micro SD card, specify **/sd0:1** as the value of *storage*.

Parameter	Description	Required Value
HDC_AGENT_CONFIG_FILE	Specifies the name and location of the agent configuration file.	The location can be anywhere on the persistent file system. Use the file name **iot.cfg**. Example: "*storage*/iot.cfg"
HDC_DEVICE_START_CONFIG_FILE	Specifies the name of the file that contains your tenant-specific authentication information.	Default: "startup.bin" Do not change the default value.
HDC_DEVICE_POLICY_CONFIG_FILE	Specifies the name of the file that contains authentication information generated when the device first connects to the server.	Default: "dxlpolicy.bin" Do not change the default value.
HDC_DEVICE_GUID_DATA_FILE	Specifies the name of the file that contains a unique device identifier generated when the device first connects to the server.	Default: "agentguid.bin" Do not change the default value.
HDC_DEVICE_CONFIG_PATH	Specifies the location of the following agent files: **HDC_DEVICE_START_CONFIG_FILE** **HDC_DEVICE_POLICY_CONFIG_FILE** **HDC_DEVICE_GUID_DATA_FILE**	The location can be anywhere on the persistent file system. Example: "*storage*/"
MOSQUITTO_CONFIG_FILE	Specifies the name and location of the MQTT broker configuration file.	The location can be anywhere on the persistent file system, but you must specify the full path, including the location of the persistent storage. Use the file name **mosquitto.conf**.

Parameter	Description	Required Value
		Example: *"storage*/mosquitto.conf"

Network Components

You must enable DNS to enable the device to resolve the Helix Device Cloud domain name. You can use the following component and specify configuration parameters in your VIP or in a configuration file.

- **INCLUDE_IPDNSC**

The device must have the correct date and time. You can add one of the following components to your VIP to set the time during the boot sequence:

- **INCLUDE_IPNTP**
- **INCLUDE_IPSNTPC**

For more information about configuring networking, see VxWorks 7 Networking.

Additional Configuration

If your VIP contains services that require file descriptors, such as networking services, you may exceed the default number of descriptors. You may need to configure the following to increase the number of file descriptors:

Parameter	Value
NUM_FILES	100

4. Confirming Successful Device Registration for VxWorks

After you boot your VxWorks device, you must ensure that it registers successfully with the server.

To identify your device on the management console, you need the device identifier from the device, which you can find by viewing the contents of the **agentguid.bin** file in the location you specified for the **HDC_DEVICE_CONFIG_PATH** parameter in your VIP.

Alternatively, you can identify your device by its MAC address, which may be referred to as the hardware address or physical address. If you have multiple network interfaces, the device may appear with the MAC address of any of the interfaces.

Prerequisites

You need your Helix Device Cloud user name and password.

Procedure

1. If you have not already done so, sign in to the management console with your user name and password at https://www.helixdevicecloud.com.
2. Click **DEVICES**.
3. In the **Device Name, ID, MAC Address** box, begin typing the device name, device ID, MAC address, or any other device attribute until your device appears in the list.

4. In the Device ID column, click your device.

   Your device is connected to the server if **Online** appears on the device details page under the device name in the upper-left corner of the page.

You are now ready to develop applications using the agent APIs.

Postrequisites

You can run one of the available sample applications to start telemetry and run actions from the management console. For more information, see the following:

Sample Applications

If your device does not appear on the server, see the following:

Diagnosing Initial Connectivity Failures on VxWorks 7

6. REFERENCES

1. iot-config.py

The iot-config.py script for Wind River Linux and IDP XT enables you to specify HDC-specific configuration information before building the target image.

The script requires Python version 2.7*x*.

Execute the iot-config.py script after the configure command and before the make command. You must run the script from your platform project directory (*projDir*).

-h, --help

Displays a help message and exits.

-s '*key*:*value*', --string='*key*:*value*'

Specifies a comma-separated string of *key*:*value* pairs, as follows:

- A *key* can be the following (all upper case):

  SECURITY_BUNDLE

  agent authentication file for your tenant, including the path

  ENABLE_MIGRATION

  enables migration from the 2.0 to 2.1 release of the agent

  Valid values: 1

Validation is not performed on the *key*:*value* pairs when you execute the iot-config.py script.

-c, --clean

Removes all *key*:*value* pairs previously specified. You can also specify a *key*:*value* pair to remove individual configuration entries.

Example

```
$ cd projDir
$ ?layers/wr-iot-meta/scripts/iot-config.py -s
'SECURITY_BUNDLE:/home/user/startup.bin'
```

2. deploy.sh Script Options for HDC

The deploy.sh script is provided for use with devices that run the HDC agent on Wind River Linux and IDP XT.

For complete information about the deploy.sh script options, change to your platform project directory (*projDir*) and execute ./deploy.sh -h.

The following table describes the recommended options for HDC and the interaction with other options.

Option	Purpose
-s 8G	Creates an eight GB file if the device specified by the -d option is a file. Otherwise, it is ignored.
-f *rootfsFilename*	Specifies the file name of the rootfs image to deploy. If the -F option is not specified, the same image file is used as the factory default image.
-y	Formats the device specified by the -d option and creates a VFAT and ext3 partition on the device.
-u	Deploys UEFI files in the first VFAT partition.
-F *rootfsFilename*	Specifies the rootfs image to use as the factory default image. If no other options are specified, it updates only the factory default image on the boot media; the contents of the other partitions are not affected.