Intel® VTune™ Amplifier Installation Guide - macOS* (2018 Beta)

Intel Corporation

www.intel.com

Legal Information

Contents

Legal Information

Introduction 1

This document explains how to install and configure Intel® VTune™ Amplifier on a macOS* system. Intel VTune Amplifier on macOS* can be used to view results collected using VTune Amplifier on a Linux* or Windows* system. Results cannot be collected on a macOS* system.

To install VTune Amplifier, you can use:

- Graphical user interface (GUI) installer: Presents installation options and allows you to choose product components.

Prerequisites 2

The following information is important to consider before beginning to install Intel® VTune™ Amplifier:

- Review the system requirements listed in the Release Notes document. The document can be found online and in your installation media.
- It is recommended that an administrator account is used to install, change, or uninstall the product. Users without administrator (or root user) access can install the product, but not all collectors will be available.
- If you will be using Xcode*, make sure that a supported version of Xcode is installed. If you install a new version of Xcode in the future, you need to reinstall VTune Amplifier.
- You do not need to uninstall previous versions or updates of VTune Amplifier before installing a newer version. However, if you do not remove older updates before installing a newer version, all product components that belong to the same major release will be replaced with the latest version for each major release update.

  For example, if you have VTune Amplifier 20xx Update 1 installed and are currently installing VTune Amplifier 20xx Update 2, the older version will be uninstalled and replaced with the new content for Update 2. If you are installing the next major release, VTune Amplifier 20xy, your installation of VTune Amplifier 20xx Update 1 will remain and the new release will be installed beside the old version, duplicating common files, documentation, samples, and product components.
- If you are installing in a cluster environment, you can install and use multiple versions of the product on the same system. However, kernel driver usage is limited to a single version of VTune Amplifier. This means you can have multiple copies of VTune Amplifier installed without the SEP drivers and a single version of the product with the drivers installed. The latter would be enabled with the advanced types of analysis using hardware event based sampling analysis data collection.
- The user installing the product should have read and write permissions for the `/tmp` and `/Users/Shared/Library/Application Support/Intel/Licenses` directories.

Product Activation and Licensing

The named-user license provisions in the Intel® end-user license agreement allow Intel VTune™ Amplifier to be installed on up to three systems. Product licensing checks for the number of systems when it checks for valid licenses and tracks systems on which the software is installed by the system host identifier. Starting with Intel VTune Amplifier 2017 Update 2, you can download additional host operating system versions, Windows*, Linux*, or macOS*, with a valid license for any operating system. For more information about cross-operating system support, see https://software.intel.com/en-us/articles/intel-vtune-amplifier-intel-advisor-and-intel-inspector-now-include-cross-os-support.

To install on another system after reaching the limit, an old system host identifier must be released from the registration system. For more information about releasing an existing system, contact Intel Technical Support through https://software.intel.com/en-us/support. You can also review the information provided by Intel Technical Support on the following page: https://software.intel.com/en-us/articles/required-license-upgrade-for-intel-system-studio-2016-and-intel-parallel-studio-xe-2016.

The end-user license agreement is available in the following locations:

- `<install-dir>/Contents/documentation/<language>/EULA.rtf`
- `<install-dir>/Contents/documentation/<language>/EULA.txt`

NOTE:

If you are updating from the Beta program to the production version of Intel VTune Amplifier, you will need an updated license.

There are several methods for product activation during installation:

- Activation using a serial number. Internet connection is required for this option.
- Remote activation using a serial number. Use this method when your computer is not connected to the internet. You must use another computer with internet access to complete activation.
- Activation using a license file.
- Activation using a license server.

Installation Steps

3

The Intel® VTune™ Amplifier installation package contains all components of the product in a downloadable file. The installer can be run as an administrator from a GUI.

The Intel Software Manager installs automatically with all Intel Software Development Products on Windows*, Linux*, and macOS* systems. The Intel Software Manager is a utility that allows users to:

- Download and install updates for your Intel Software Development Products.
- Manage subscription status of installed software.
- Activate serial numbers.
- Find out about the latest news for Intel Software Development Products.
- Intel Software Manager requires an internet connection to connect to a remote server for information and updates.

Refer to the following site for more information about Intel Software Manager: https://registrationcenter-ssl.intel.com/Docs/ism.htm

The following sections detail the steps required to install Intel VTune Amplifier.

- Installing with the Installer Graphical User Interface
- Installing for Use with a Virtual Machine
- Installation Folders

Installing with the Intel® VTune™ Amplifier Installer Graphical User Interface

Use the following steps to launch the installer GUI:

1. Double click the `vtune_amplifier_<version>.dmg` file to mount the disk image.
2. Double click the `Intel VTune Amplifier <version>.app` file to start the installation program.
3. Use the information in the installer panels to complete the installation.

The installation process includes the following steps:

- Installation Location and Components

  Lists the default installation location and options. Select the components to install and change the default installation location (optional).
- Prerequisites

  Lists all prerequisites that would prevent a fully successful installation. Prerequisites could include additional requirements, information about setting up drivers, a reminder to restart your system after installation completes, and so on.
- Activation and Options

  - Provides activation options. For more information, see Product Activation and Licensing.
  - Install all options or click **Customize** to select a subset of options.

  Click the **Install** button to begin installation.
- Complete

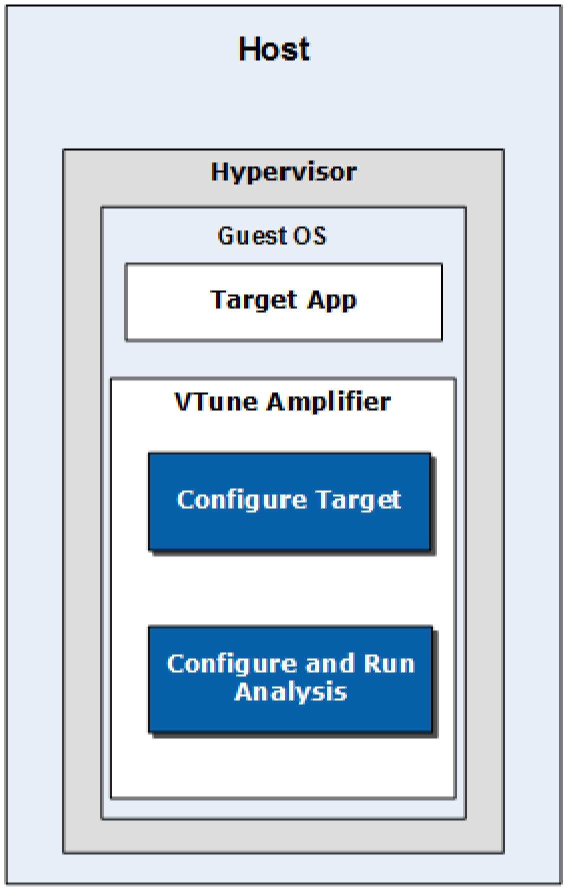
  The Intel VTune Amplifier Getting Started page displays after installation succeeds.

Installing Intel® VTune™ Amplifier for Use with a Virtual Machine

It is possible to collect data on a virtual machine for analysis with Intel VTune Amplifier. In most cases, VTune Amplifier is installed on the guest system using the installer user interface . The data collection and analysis with VTune Amplifier is run on the virtual machine. For KVM and XEN Project*, the analysis is run using Perf*-based event-based sampling collection. For all other supported VMMs, it is run using the installed VTune Amplifier drivers. For more information about supported virtual machines, see Using VTune Amplifier with a Virtual Machine.

TIP:

Intel VTune Amplifier may not run if it is not installed on a non-privileged guest OS. Refer to the documentation for your VMM to learn how to set up a privileged system.



Enabling Analysis on the Virtual Machine

Additional steps are required to enable performance analysis on the virtual machine.

- Enable Parallels* Desktop Analysis

Not all analysis types and data can be collected on a virtual machine. Availability of data depends on which collectors are virtualized by the virtual machine manager. Additional information is available from Using VTune Amplifier with a Virtual Machine and from the documentation for your virtual machine manager.

Intel® VTune™ Amplifier Installation Folders

By default, Intel® VTune™ Amplifier is installed with the following directory structure: `/Applications/Intel VTune Amplifier <version>.app`

```
Frameworks
MacOS
Resources
backend
config
contrib
documentation
frontend
include
message
sdk
storage_snapshot
```

Where *<version>* is the release level of your installed product. Having a release-specific directory allows for the installation of multiple versions of Intel VTune Amplifier on the same system.

Post-Installation Steps 4

The following sections detail the steps required to configure your Intel® VTune™ Amplifier installation.

- Preparing a Target Linux* System for Remote Analysis
- Preparing a Target Embedded Linux* System for Remote Analysis
- Preparing a Target FreeBSD System for Remote Analysis
- Preparing a Target Android System for Remote Analysis
- Preparing an Intel Xeon Phi Coprocessor System for Analysis
- Using Intel VTune Amplifier with a Virtual Machine
- Configuring SSH Access for Remote Collection

The table below details the suggested reading paths based on your analysis needs:

Analyze performance on a remote Linux* system	Install the remote collectors and configure SSH access on a remote Linux system. If the remote collector installation fails to install the appropriate sampling drivers, the drivers can be installed manually. **1.** Preparing a Target Linux System for Remote Analysis **2.** Build the Sampling Driver (optional) **3.** Configuring SSH Access for Remote Collection
Analyze performance on a remote Android* system	Configure the Android device for analysis and connect via ADB. If the appropriate sampling drivers are not available, the drivers can be installed manually. Compile the Android application for analysis. Specify the project search directories. **1.** Preparing a Target Android System for Remote Analysis **2.** Managing the Sampling Driver for Android Targets (optional) **3.** Preparing an Android Application for Analysis **4.** Specifying Search Directories for Android
Analyze performance on a FreeBSD* system	Install VTune Amplifier collectors and drivers on a FreeBSD system. A FreeBSD license for Intel® System Studio is required. For more information, see Preparing a Target FreeBSD System for Remote Analysis.
Analyze performance on embedded Linux systems	Review the list of supported embedded Linux systems. Configure the embedded Linux environment for performance analysis. Begin with Preparing a Target Embedded Linux* System for Remote Analysis.
Monitor a virtual machine	Review the list of supported virtual machine managers and analysis type limitations. Configure the virtual machine for performance analysis. Begin with Using Intel VTune Amplifier with a Virtual Machine.

Linux* System Setup for Remote Analysis

NOTE:

The installation package for macOS* does not contain collectors for a Linux* target. The collectors can be obtained through the Linux* installation package. Refer to the *Intel VTune Amplifier Installation Guide - Linux* OS* for additional configuration details, including information about the sampling and power drivers.

You can collect data remotely on a remote Linux* system by specifying the system as the analysis target in Intel® VTune™ Amplifier. VTune Amplifier will automatically install the appropriate collectors on the target system when you switch away from the **Analysis Target** tab. Specify a location for the install using the **VTune Amplifier installation directory on the remote system** field under the **Advanced** settings.

If the collectors are not automatically installed or you get an error message, use the following steps to manually prepare for data collection on a remote Linux system:

1. Install the VTune Amplifier collector on the target system.
2. Set up SSH access to the target system.
3. Set up the analysis target in VTune Amplifier.

NOTE:

The automatic installation on the remote Linux system does not build and install the sampling drivers. Driverless sampling data collection is based on the Linux Perf* tool functionality, which has a limited scope of analysis options. To collect advanced hardware event-based sampling data, manually install the sampling driver.

Installing the VTune Amplifier Collectors on the Target Device Manually

Use the following steps to set up analysis on a target regular or embedded Linux target system.

1. Download and extract the Linux installation package.
2. Copy the required target package archive to the target device using ftp, sftp, or scp. The following target packages are available:
   - `<install_dir>/target/linux/vtune_amplifier_target_sep_x86.tgz` - provides hardware event-based sampling collector only (SEP) for x86 systems
   - `<install_dir>/target/linux/vtune_amplifier_target_sep_x86_64.tgz` - provides hardware event-based sampling collector only (SEP) for 64-bit systems
   - `<install_dir>/target/linux/vtune_amplifier_target_x86.tgz` - provides all VTune Amplifier collectors for x86 systems
   - `<install_dir>/target/linux/vtune_amplifier_target_x86_64.tgz` - provides all VTune Amplifier collectors for 64-bit systems

   NOTE:

   - Use both `*_x86` and `*_x86_64` packages if you plan to run and analyze 32-bit processes on 64-bit systems.
   - Remote collectors are also available in the .

3. On the target device, unpack the product package to the `/opt/intel` directory:

   ```
   target> tar -zxvf <target_package>.tgz
   ```

VTune Amplifier target package is located in the newly created directory `/opt/intel/vtune_amplifier_<version>_for_systems.<package_num>`.

When collecting data remotely, the VTune Amplifier looks for the collectors on the target device in its default location: `/opt/intel/vtune_amplifier_<version>.<package_num>`. It also temporary stores performance results on the target system in the `/tmp` directory. If you installed the target package to a different location and need to specify another temporary directory, make sure to configure your target properties in the **Analysis Target** tab as follows:

- Use the **VTune Amplifier installation directory on the remote system** option to specify the path to the VTune Amplifier on the remote system. If default location is used, the path is provided automatically.
- Use the **Temporary directory on the remote system** option to specify a non-default temporary directory.

Alternatively, use the`-target-install-dir` and `-target-tmp-dir` options from the `amplxe-cl` command line.

Building and Installing the Drivers Manually

NOTE:

Building the sampling drivers is only required if the drivers were not built as part of the collector installation. The installation output should inform you if building the sampling driver is required.

To enable hardware event-based sampling analysis on your target device:

1. Build the sampling driver on the target system.

   NOTE:

   - Make sure kernel headers correspond to the kernel version running on the device. For details, see the `README.txt` file in the `sepdk/src`directory.
   - Make sure compiler version corresponds to the architecture (x86 or x86_64) of the kernel running on the target system.
   - For Advanced Hotspots, General Exploration and Custom event-based sampling analysis types, you may not need root credentials and installing the sampling driver for systems with kernel 2.6.32 or higher, which exports CPU PMU programming details over `/sys/bus/event_source/devices/cpu/format` file system. Your operating system limits on the maximum amount of files opened by a process as well as maximum memory mapped to a process address space still apply and may affect profiling capabilities. These capabilities are based on Linux Perf* functionality and all its limitations fully apply to the VTune Amplifier as well. For more information, see the *Tutorial: Troubleshooting and Tips* topic at https://perf.wiki.kernel.org/index.php/Main_Page.

2. Copy the `<install_dir>/sepdk` folder to the `/opt/intel/vtune_amplifier_<version>` directory on the target device.

   You may choose another directory if this location is not accessible/does not exist or you can mount those folders via NFS.
3. On the target device, install the drivers.

   If the insmod-sep script does not work on the target system due to absence of standard Linux commands, you may install drivers manually using the Linux OS `insmod` command directly.

NOTE:

To build the sampling driver as RPM using build services as Open Build Service (OBS), use the `sepdk.spec` file located at the `<install_dir>/sepdk/src` the directory.

Set up SSH Access

After installing the collectors and ensuring that the appropriate drivers are installed, set up SSH access to the target system.

Set up Analysis Target

After completing all other configuration steps for the remote Linux system, you can run an analysis using VTune Amplifier. Before running the first analysis, you must set up the analysis target. For more information, see "Analysis Target Setup" in the VTune Amplifier online help, available from `<install-dir>/documentation/<language>/help`.

Embedded Linux* System Setup for Remote Analysis

Use the Intel® VTune™ Amplifier for performance analysis on Embedded Linux systems, Wind River*, Yocto Project*, Tizen*, FreeBSD* and others.*

Embedded device performance data can be collected remotely by installing the collectors on the embedded device and running the analysis from an instance of VTune Amplifier installed on the host system. This is useful when you want to collect performance information using the VTune Amplifier GUI and the pre-configured analysis types. Alternatively, hardware-based sampling data can be collected using a command-line collection tool that can be copied to a host system and viewed in the VTune Amplifier GUI. Use this type of data collection when space on the embedded device is limited or when only hardware event-based sampling data is required.

NOTE:

Root access to the operating system kernel is required to install the collectors and drivers required for performance analysis using VTune Amplifier.

There are two ways to enable performance analysis on an embedded device:

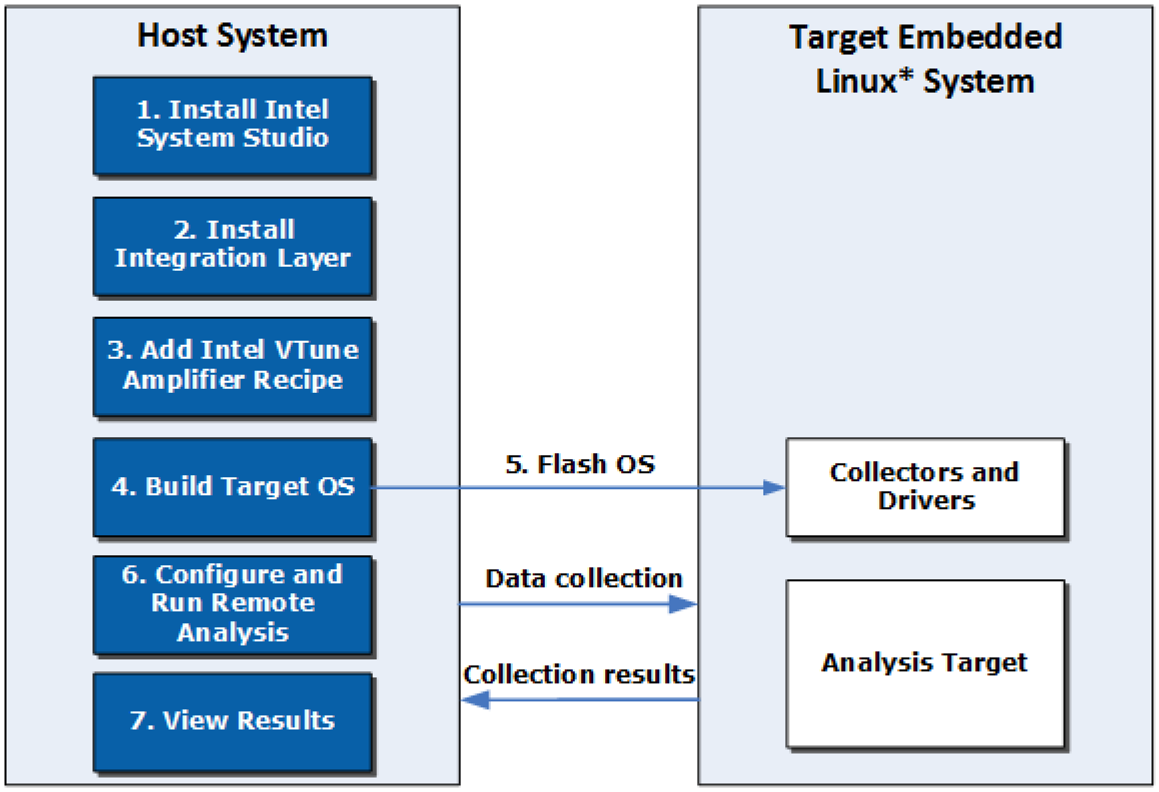
- Using the Intel System Studio integration layer (Wind River* Linux and Yocto Project* only)
- Using the bundled VTune Amplifier installation packages

Using the Intel System Studio Integration Layer

NOTE:

The Intel System Studio integration layer works for embedded systems with Wind River Linux or Yocto Project installed.

The Intel System Studio integration layer allows the Intel System Studio products to be fully integrated with a target operating system by building the drivers and corresponding target packages into the operating system image automatically. Use this option in the case where a platform build engineer has control over the kernel sources and signature files, but the application engineer does not. The platform build engineer can integrate the product drivers with the target package and include them in the embedded device image that is delivered to the application engineer.


Host System
1. Install Intel System Studio
2. Install Integration Layer
3. Add Intel VTune Amplifier Recipe
4. Build Target OS
6. Configure and Run Remote Analysis
7. View Results
5. Flash OS
Data collection
Collection results
Target Embedded Linux* System
Collectors and Drivers
Analysis Target

1. Install Intel System Studio.	Install Intel System Studio using the installer GUI. **TIP:** If you already have Wind River Linux and Workbench* installed on your host system, select the **Integrate to Wind River* Linux and Wind River* Workbench** checkbox when installing Intel System Studio. If you install Intel System Studio before installing Wind River Linux and Workbench, ignore the automatic integration step during installation and manually integrate after installing Wind River Linux and Workbench.
2. Install the Intel System Studio integration layer.	**1.** Copy the integration layer from the Intel System Studio installation folder to the target operating system development folder. **2.** Run the post-installation script: `wr-iss-<version>/<OS>-setup/postinst_<OS>_iss.sh <ISS_BASE_dir>` **NOTE:** This step can be completed as part of the Intel System Studio installation for Wind River Linux system integration. If you already have Wind River Linux and Workbench installed on your host system, select the **Integrate to Wind River* Linux and Wind River* Workbench** checkbox when installing Intel System Studio. If you install Intel System Studio before installing Wind River Linux and Workbench, ignore the automatic integration step during installation and manually integrate after installing Wind River Linux and Workbench.

3. Build the recipe that includes the appropriate VTune Amplifier package.	**1.** Add the path to the `wr-iss-<version>` to the `bblayers.conf` file:
	`BBLAYERS= "\`
	`...`
	`<OS_INSTALL_DIR>/wr-iss-<version>\`
	`...`
	`"`
	2. Add the VTune Amplifier recipes to `conf/local.conf`. Possible recipes include `intel-iss-vtune-target` for remote command line and GUI usage mode or `intel-iss-vtune-sep-target` for command line hardware event-based sampling analysis. For more information about these collection methods, see Remote Analysis Workflow for Linux Systems in the VTune Amplifier help.
	NOTE:
	This step can be completed when creating a new Wind River Linux platform project using Wind River Workbench.

4. Build the target operating system.	Build the target operating system, which will complete the integration of the VTune Amplifier collectors and drivers.
5. Flash the operating system to the target embedded device.	After flashing the operating system to the target embedded device, ensure that the appropriate VTune Amplifier drivers are present. For more information, see Building and Managing the Sampling Driver.
6. Run the analysis on the target embedded device.	Depending on the recipe chosen, the analysis is either run from the host system using an SSH connection or using the SEP commands. Use the following steps to run the analysis from the host system: **1.** Set up a password-less SSH access to the target using RSA keys. **2.** Specify your target application and remote system. Specify your target application and remote system. **3.** Choose an analysis type. **4.** Configure the analysis type. **5.** Run the analysis from the host. Use the information available in the *Sampling Enabling Product User's Guide* to run the SEP commands.
7. View results in the VTune Amplifier GUI	View the collected data on the host.

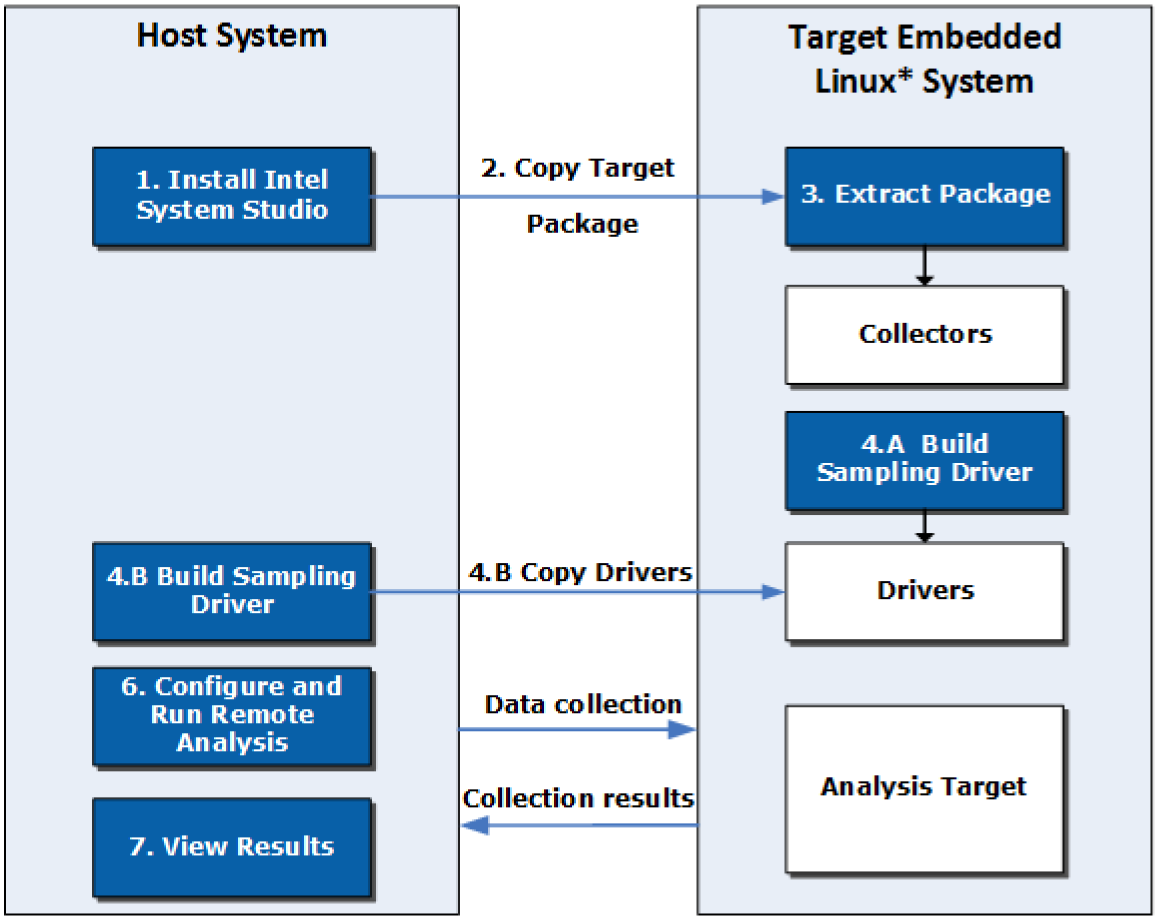
Examples

Configuring Wind River Linux with the Intel System Studio Integration Layer

Configuring Yocto Project* with the Intel System Studio Integration Layer

Using the Bundled Intel VTune Amplifier Installation Packages

You can build the appropriate drivers and install the VTune Amplifier collectors on your kernel image manually with a command line. This option requires root access to the configured kernel source.



1. Install Intel System Studio.	Install Intel System Studio using the installer GUI.
2. Copy the target package to the target embedded Linux device.	Copy the appropriate target package archive to the target device using ftp, sftp, or scp. The following target packages are available from `<install-dir>/target/linux32[64]`: • `linux32/vtune_amplifier_target_sep_x86.tgz` - provides hardware event-based sampling collector only (SEP) for x86 systems • `linux32/vtune_amplifier_target_x86.tgz` - provides all VTune Amplifier collectors for x86 systems • `linux64/vtune_amplifier_target_sep_x86_64.tgz` - provides hardware event-based sampling collector only (SEP) for 64-bit systems • `linux64/vtune_amplifier_target_x86_64.tgz` - provides all VTune Amplifier collectors for 64-bit systems

NOTE:

- Use both `*_x86` and `*_x86_64` packages if you plan to run and analyze 32-bit processes on 64-bit systems.
- The `vtune_amplifier_target_sep_x86.tgz` and `vtune_amplifier_target_sep_x86_64.tgz` packages do not support remote data collection using the VTune Amplifier GUI. Use the `linux32/vtune_amplifier_target_x86.tgz` or `linux64/vtune_amplifier_target_x86_64.tgz` package for remote data collection support.
- Remote collectors are also available in the <Intel_System_studio_install_dir>/targets/system_studio_target.tgz.

3. Extract the package on the target embedded Linux device.	Extract the package to the `/opt/intel` directory.
4. [Optional] Build the sampling driver on the target (A) or host (B) system.	This option is only required if the drivers were not built as part of the collector installation. The installation output should inform you if building the sampling driver is required. The drivers are built either on the target system or on the host system, depending on compiler toolchain availability: A. If the compiler toolchain is available on the target system: **1.** On the target embedded device, build the driver from the `<install-dir>/sepdk/src` directory using the `./build-driver` command. **2.** Load the driver into the kernel using the `./insmod-sep` command. B. If the compiler toolchain is not available on the target system: **1.** On the host system, cross-build the driver using the driver source from the target package `sepdk/src` directory with the `./build-driver` command. Provide the cross-compiler (if necessary) and the target kernel source tree for the build. **2.** Copy the `sepdk/src` folder to the target system. **3.** Load the driver into the kernel using the `./insmod-sep` command. For more information, see Building and Managing the Sampling Driver.
6. Run the analysis on the target embedded device.	Depending on the target installation package chosen, the analysis is either run from the host system using an SSH connection or using the SEP commands. Use the following steps to run the analysis from the host system: **1.** Set up a password-less SSH access to the target using RSA keys. **2.** Specify your target application and remote system. Specify your target application and remote system. **3.** Choose an analysis type. **4.** Configure the analysis type. **5.** Run the analysis from the host.

	Use the information available in the *Sampling Enabling Product User's Guide* to run the SEP commands.
7. View results in the VTune Amplifier GUI	View the collected data on the host.

Example

Configuring Yocto Project with Intel VTune Amplifier Target Packages

Configuring Wind River* Linux* and Intel® VTune™ Amplifier with the Intel System Studio Integration Layer

Intel® VTune™ Amplifier can collect and analyze performance data on embedded Linux* devices running Wind River* Linux*. This topic provides an example of setting up Intel VTune Amplifier to collect performance data on an embedded device with Wind River Linux installed using the Intel System Studio integration layer provided with the product installation files. The process integrates the VTune Amplifier product drivers with the target package and includes them in the embedded device image. Root access to the kernel is required.

Install the Intel System Studio with the Integration Layer

Prerequisite: Wind River Linux is already installed on the same system on which Intel System Studio will be installed.

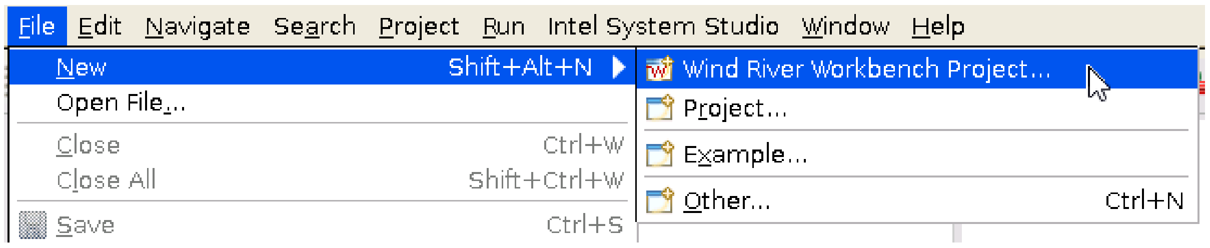
During the Intel System Studio installation, the installer displays an option for integration with Wind River Linux. Select the checkbox and browse to the location where Wind River Linux is installed. The integration will be done automatically with your Wind River Linux on your host system as Intel System Studio is installed.

NOTE:

If you do not have Wind River Linux installed on the host machine, you can ignore the automatic integration during the installation. You can run the integration manually after you have Wind River Linux installed. Refer to the instructions available from this article: https://software.intel.com/en-us/articles/using-intel-system-studio-with-wind-river-linux-build-environment.

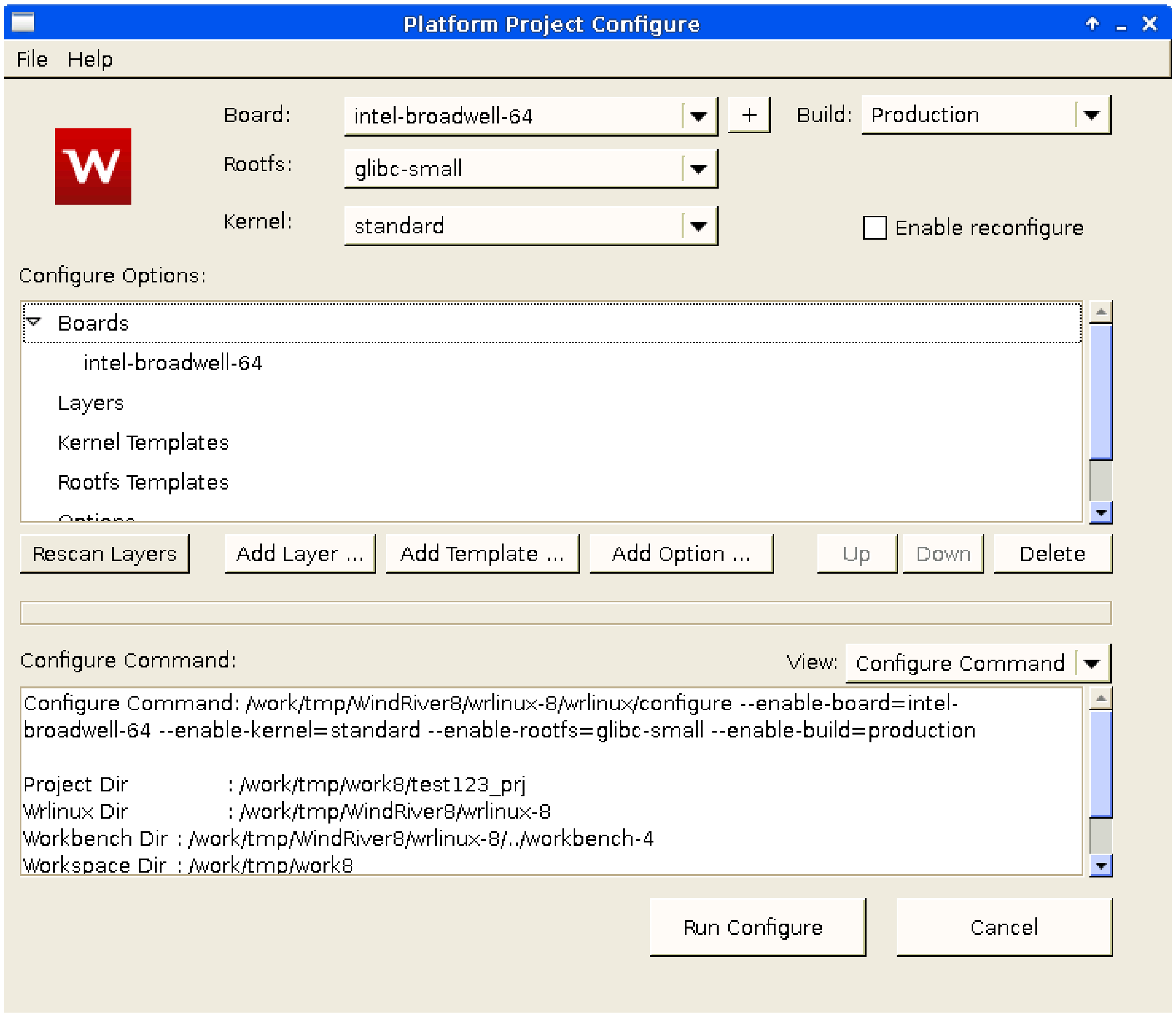
Configure the Intel VTune Amplifier Package for the Target System

1. Select **File** > **New** > **Wind River Linux Platform Project** from the Wind River Linux Workbench.



2. Select **Platform** from the **Build type** drop-down list and click **Next**.
3. Give your project a unique name and click **Finish**.

   The **Platform Project Configure** window opens.

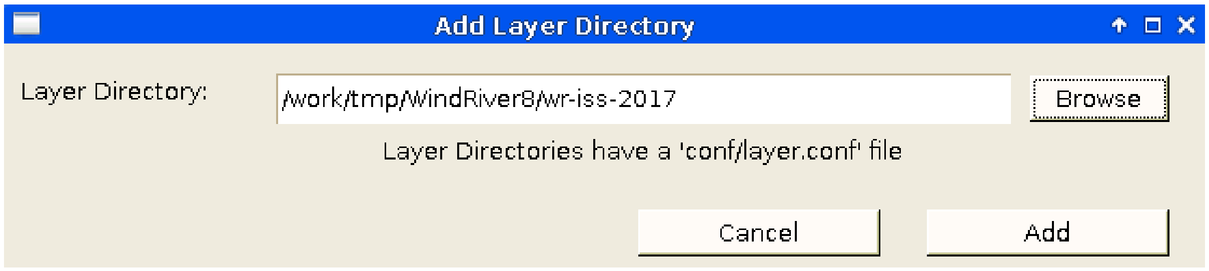


```
Configure Command: /work/tmp/WindRiver8/wrlinux-8/wrlinux/configure --enable-board=intel-
broadwell-64 --enable-kernel=standard --enable-rootfs=glibc-small --enable-build=production

Project Dir      : /work/tmp/work8/test123_prj
Wrlinux Dir      : /work/tmp/WindRiver8/wrlinux-8
Workbench Dir : /work/tmp/WindRiver8/wrlinux-8/../workbench-4
Workspace Dir  : /work/tmp/work8
```

4. Select the appropriate board type from the **Board** drop-down list.

5. Click **Add Layer...** and browse to the Intel System Studio integration layer, which is located in `<WIND_HOME>/wr-iss-<version>`. Click **Add** to close the window and add the layer to the configuration.

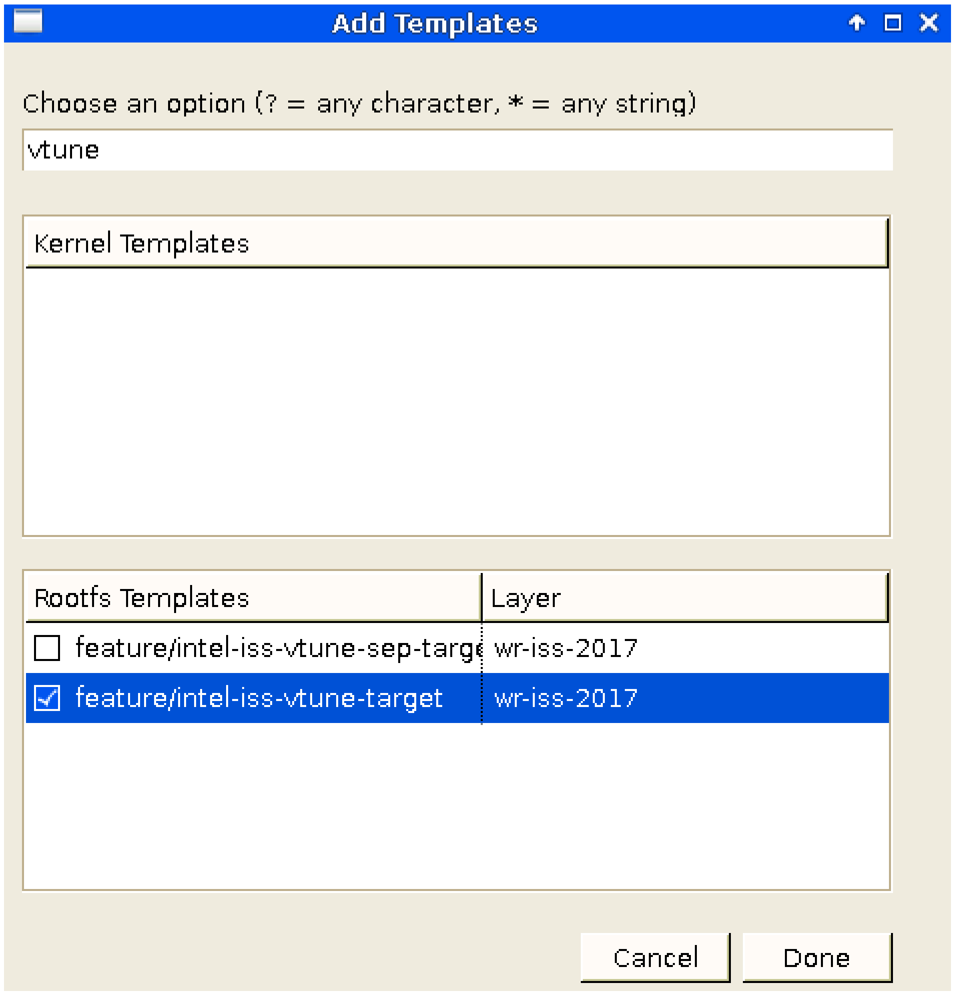


Layer Directory: /work/tmp/WindRiver8/wr-iss-2017

Layer Directories have a 'conf/layer.conf' file

6. Click **Rescan Layers** on the **Platform Project Configure** window.

7. When the scan is finished, click **Add Template...** . The **Add Templates** window appears.



Rootfs Templates	Layer
☐ feature/intel-iss-vtune-sep-targe	wr-iss-2017
✓ feature/intel-iss-vtune-target	wr-iss-2017

8. Select the appropriate Intel VTune Amplifier template and click **Done**. Only one template can be selected.
   - Select `intel-iss-vtune-target` to support remote data collection from the host system to the target system.
   - Select `intel-iss-vtune-sep-target` to support native data collection using the sampling enabling driver command line utility.

TIP:

Type "vtune" at the top of the **Add Templates** window to filter the list of templates.

9. Click **Run Configure** to execute the configure command.
10. Click **Finish** to complete the setup.

NOTE:

If you have an existing Wind River Linux Platform Project, you can add the Intel VTune Amplifier features manually. Refer to the instructions available from this article: https://software.intel.com/en-us/articles/using-intel-system-studio-with-wind-river-linux-build-environment.

Build and Boot the Target System

The VTune Amplifier drivers are built automatically and integrated with your target system image. The VTune Amplifier components are available from the `/opt/intel/` directory on the target system.

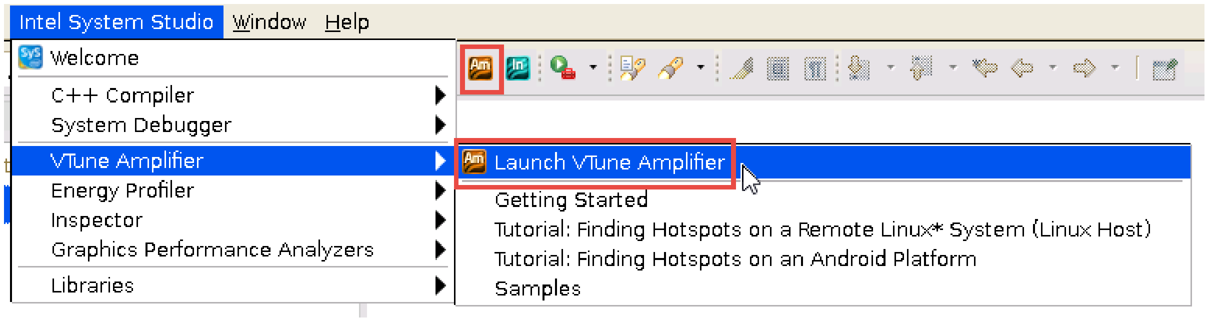
Configure and Run Remote Analysis

Use the following steps on the host system to set up and launch the analysis on the embedded device. These steps assume the `intel-iss-vtune-target` template was selected.

1. Set up a password-less SSH access to the target using RSA keys.
2. Create a new project.
3. Select the **remote Linux (SSH)** analysis target and specify the collection details.
4. Configure the analysis type.
5. Start the analysis.

TIP:

You can also launch the performance analysis directly from Wind River Linux Workbench. Select **Intel System Studio** > **Intel VTune Amplifier** > **Launch VTune Amplifier** to open the VTune Amplifier GUI. A product icon is also available from the Workbench toolbar.



Configuring Yocto Project* and Intel® VTune™ Amplifier with the Intel System Studio Integration Layer

Intel® VTune™ Amplifier can collect and analyze performance data on embedded Linux* devices running Yocto Project*. This topic provides an example of setting up Intel VTune Amplifier to collect performance data on an embedded device with Yocto Project 1.8 installed using the Intel System Studio integration layer provided with the product installation files. The process integrates the VTune Amplifier product drivers with the target package and includes them in the embedded device image. Root access to the kernel is required.

Install the Intel System Studio Integration Layer

Prerequisite: Install Intel System Studio on the host system.

1. Copy the integration layer from the Intel System Studio installation folder to the Yocto Project development folder. For example:

```
cp -r /opt/intel/system_studio_2017.0.0/wr-iss-2017 /path/to/poky-fido-10.0.0/
```

2. Run the post-installation script. For example:

```
cd /path/to/poky-fido-10.0.0
wr-iss-2017/yp-setup/postinst_yp_iss.sh /opt/intel/system_studio_2017.0.0
```

Add the Intel VTune Amplifier Recipe

1. Add the path to the `wr-iss-<version>` to the `bblayer.conf` file. For example:

   ```
   vi /path/to/poky-fido-10.0.0/build/conf/bblayers.conf
   BBLAYERS = "$HOME/source/poky/wr-iss-2017\"
   ```

   Your file should look similar to the following:

   ```
   BBLAYERS ?= " \
     $HOME/source/poky/meta \
     $HOME/source/poky/meta-poky \
     $HOME/source/poky/meta-yocto-bsp \
     $HOME/source/poky/meta-intel \
     $HOME/source/poky/wr-iss-2017 \
     "
   ```
2. Add the Intel VTune Amplifier recipe to `conf/local.conf`. Two recipes are available, `intel-iss-vtune-target` and `intel-iss-vtune-sep-target`. In this example, the `intel-iss-vtune-target` is used so the analysis can be run from the VTune Amplifier GUI on the host system.

   ```
   vi "conf/local.conf"
   IMAGE_INSTALL_append = " intel-iss-vtune-target"
   ```

   NOTE:

   You cannot add both `intel-iss-vtune-target` and `intel-iss-vtune-sep-target` at the same time.

Build and Flash the Target Operating System

1. Build the target operating system. For example:

   ```
   bitbake core-image-sato
   ```
2. Flash the operating system to the embedded device.

Configure and Run Remote Analysis

Use the following steps on the host system to set up and launch the analysis on the embedded device:

1. Set up a password-less SSH access to the target using RSA keys.
2. Create a new project.
3. Select the **remote Linux (SSH)** analysis target and specify the collection details.
4. Configure the analysis type.
5. Start the analysis.

Configuring Yocto Project* and Intel® VTune™ Amplifier with the Linux* Target Package

Intel® VTune™ Amplifier can collect and analyze performance data on embedded Linux* devices running Yocto Project*. This topic provides an example of setting up Intel VTune Amplifier to collect performance data on an embedded device using the Linux* target installation files provided in the VTune Amplifier installation directories. The process manually adds the VTune Amplifier collectors and drivers to the embedded device. Root access to the kernel is required if you need to build the drivers.

Copy and Extract Target Linux Package

Prerequisite: Install Intel VTune Amplifier on the host system.

1. Copy the target package archive to the target device. The following target packages are available:

- *<install-dir>*/target/vtune_amplifier_target_sep_x86.tgz - provides hardware event-based sampling collector only (SEP) for x86 systems
- *<install-dir>*/target/vtune_amplifier_target_x86.tgz - provides all VTune Amplifier collectors for x86 systems
- *<install-dir>*/target/vtune_amplifier_target_sep_x86_64.tgz - provides hardware event-based sampling collector only (SEP) for 64-bit systems
- *<install-dir>*/target/vtune_amplifier_target_x86_64.tgz - provides all VTune Amplifier collectors for 64-bit systems

For example, the following command copies the `vtune_amplifier_target_x86_64.tgz` package to the embedded device using SCP:

```
> scp -r vtune_amplifier_target_x86_64.tgz root@123.45.67.89:/opt/intel/
```

2. Extract the file on the target system. For example:

```
> tar -xvsf vtune_amplifier_target_x86_64.tgz
```

3. Make sure the sampling driver is available on the target system. The installation output should inform you if building the sampling driver is required. If it is not, you will need to build the sampling driver and install it on the target system.

If the compiler toolchain is available on the target embedded system, build the driver on the target device using the following steps:

a. Open a command prompt and navigate to the `<install-dir>/sepdk/src` directory. For example:

```
> cd /opt/intel/vtune_amplifier_2018.0.0.0/sepdk/src
```

b. Build the driver using the `./build-driver` command. For example:

```
> ./build-driver -ni \ --kernel-src-dir=/usr/src/kernel/ \ --kernel-version=4.4.3-
yocto-standard \ --make-args="PLATFORM=x64 ARITY=smp"
```

c. Load the driver into the kernel using the `./insmod-sep` command.

If the compiler toolchain is not available on the target embedded system, build the driver on the host system and install it on the target device using the following steps:

a. Open a command prompt and navigate to the `<install-dir>/sepdk/src` directory. For example:

```
> cd /opt/intel/vtune_amplifier_2018.0.0.0/sepdk/src
```

b. Cross-build the driver using the using the `./build-driver` command. Provide the cross-compiler (if necessary) and the target kernel source tree for the build. For example:

```
> mkdir drivers
>./build-driver -ni \
--c-compiler=i586-i586-xxx-linux-gcc \
--kernel-version=4.4.3-yocto-standard \
--kernel-src-dir=/usr/src/kernel/ \
--make-args="PLATFORM=x32 ARITY=smp" \
--install-dir=./drivers
```

c. Copy the `sepdk/src/drivers` folder to the target system.

d. Load the driver into the kernel using the `./insmod-sep` command.

For more information, see Building and Managing the Sampling Driver.

Configure and Run Remote Analysis

Use the following steps on the host system to set up and launch the analysis on the embedded device:

1. Set up a password-less SSH access to the target using RSA keys.

2. Create a new project.
3. Select the **remote Linux (SSH)** analysis target and specify the collection details.
4. Configure the analysis type.
5. Start the analysis.

FreeBSD* System Setup

Intel® VTune™ Amplifier allows you to collect performance data remotely on a FreeBSD target system.*

Intel® VTune™ Amplifier includes a target package for collecting event-based sampling data on a FreeBSD* target system either via the remote collection capability or by collecting the results locally on the FreeBSD system and copying them to a Linux*, Windows*, or macOS* host system. The collected data is then displayed on a host system that supports the graphical interface.

NOTE:

You need a FreeBSD license to use VTune Amplifier with a FreeBSD target system.

1. Install VTune Amplifier on your Linux, Windows, or macOS host. Refer to the Installation Guide for your host system for detailed instructions.
2. Install the appropriate sampling drivers on the FreeBSD target system.
3. Collect performance data using one of the following methods. For more information about each of these methods, see "Remote Analysis Workflow for Linux Systems" in the Intel VTune Amplifier online help..
   - Remote analysis from the host system using the VTune Amplifier command line or GUI.
   - Native analysis on the target system using the VTune Amplifier command line.
   - Native analysis on the target system using the sampling enabling product (SEP) collectors.
4. Review the results on the host system.

Install the Sampling Drivers on FreeBSD

Use the following steps to configure your FreeBSD target system for event-based sampling analysis. Root privileges are required on the target system to install the VTune Amplifier drivers.

1. Copy the `vtune_amplifier_<version>_freebsd.tar.gz` file to the target system using ftp, sftp, or scp.
2. Extract the archive to the `/opt/intel` directory on the target system.
3. Navigate to the following location: `/opt/intel/sepdk/modules`
4. Run the following commands to build the appropriate drivers:

```
> make
> make install
```

5. Run the following command to install the drivers:

```
> kldload sep pax
```

Allow non-root users to run an event-based sampling analysis by running the following commands after installing the drivers:

```
> chgrp -R <user_group> /dev/pax
> chgrp -R <user_group> /dev/sep
```

Remove the Sampling Drivers from FreeBSD

Run the following command to unload the sampling drivers:

```
> kldunload sep pax
```

Android* System Setup

When using the VTune Amplifier to collect data remotely on a target Android device, make sure to:

- Configure your Android device for analysis.
- Gain adb access to an Android device.
- For hardware event-based sampling, gain a root mode adb access to the Android device.
- Use the pre-installed drivers on the target Android system.

Optionally, do the following:

- Enable Java* analysis.
- To view functions within Android-supplied system libraries, device drivers, or the kernel, get access from the host development system to the exact version of these binaries with symbols not stripped.
- To view sources within Android-supplied system libraries, device drivers, or the kernel, get access from the host development system to the sources for these components.

Configuring an Android Device for Analysis

To configure your Android device, do the following:

1. Allow Debug connections to enable adb access:

   a. Select **Settings** > **About <*device*>**.

   b. Tap **Build number** seven times to enable the **Developer Options** tab.

   c. Select the **Settings** > **Developer Options** and enable the **USB debugging** option.

   NOTE:

   Path to the **Developer Options** may vary depending on the manufacture of your device and system version.

2. Enable **Unknown Sources** to install the VTune Amplifier Android package without Google* Play. To do this, select **Settings** > **Security** and enable the **Unknown Sources** option.

Gaining ADB Access to an Android Device

VTune Amplifier collector for Android requires connectivity to the Android device via adb. Typically Android devices are connected to the host via USB. If it is difficult or impossible to get adb access to a device over USB, you may get adb over Ethernet or WiFi. To connect ADB over Ethernet or WiFi, first connect to Ethernet or connect to a WiFi access point and then do the following:

1. Find the IP Address of the target. The IP address is available in Android for Ethernet via **Settings>Wireless&Networks>Ethernet>IP Address** or for Wi-Fi via **Settings>Wireless&Networks>Wi-Fi><*Connected Access Point*>>IP Address**.

2. Make sure adb is enabled on the target device. If not enabled, go to Terminal App (of your choice) on the device and type:

```
> su
> setprop service.adb.tcp.port 5555
> stop adbd
> start adbd
```

3. Connect adb on the host to the remote device. In the Command Prompt or the Terminal on the host, type:

```
> adb connect <IPAddres>:5555
```

Gaining a Root Mode ADB Access to the Android Device

For performance analysis on Android platforms, you typically need a root mode adb access to your device to:

- Install and load drivers needed for hardware event-based sampling.
- Enable the Android device to support Java* analysis.
- Run hardware event-based sampling analysis.

NOTE:

There are several analysis types on Android systems that do NOT require root privileges such as Basic Hotspots and Perf*-based driverless sampling event-based collection.

Depending on the build, you gain root mode adb access differently:

- **User/Production builds** : Gaining root mode adb access to a user build of the Android OS is difficult and different for various devices. Contact your manufacturer for how to do this.
- **Engineering builds** : Root-mode adb access is the default for engineering builds. Engineering builds of the Android OS are by their nature not "optimized". Using the VTune Amplifier against an engineering build is likely to result in VTune Amplifier identifying code to optimize which is already optimized in user and userdebug builds.
- **Userdebug builds** : Userdebug builds of the Android OS offer a compromise between good results and easy-to-run tools. By default, userdebug builds run adb in user mode. VTune Amplifier tools require root mode access to the device, which you can gain via typing `adb root` on the host. These instructions are based on userdebug builds.

Using the Pre-installed Drivers on the Target Android System

For hardware event-based sampling analysis, the VTune Amplifier needs sampling drivers to be installed. On some versions of Android systems, including most of the Intel supplied reference builds for SDVs, the following drivers are pre-installed in `/lib/modules` or `/system/lib/modules` :

- Hardware event-based analysis collectors:
  - `socperf2_x.ko`
  - `pax.ko`
  - `sep3_x.ko`
  - `sep4_x.ko`
  - `vtsspp.ko`

Typically having pre-installed drivers is more convenient. You can check for pre-installed drivers by typing:

```
adb shell ls [/lib/modules|/system/lib/modules]
```

If the drivers are not available or the version does not match requirements, consider:

- Building and Managing the Sampling Drivers

Enabling Java* Analysis

Enabling Java Analysis on Rooted Devices

By default, the VTune Amplifier installs the remote collector on the target rooted Android devices with the `--jitvtuneinfo=src` option. To change the Java profiling option for rooted devices, you need to re-install the remote collector on the target manually using the `--jitvtuneinfo=[jit|src|dex|none]` option on or . For example:

On Windows*:

On Linux*:

VTune Amplifier updates the `/data/local.prop` file as follows:

1. Basic information about the compiled trace: `root@android:/ # cat /data/local.prop dalvik.vm.extra-opts=-Xjitvtuneinfo:jit`
2. Mapping from JIT code to Java source code and basic information about the compiled trace: `root@android:/ # cat /data/local.prop dalvik.vm.extra-opts=-Xjitvtuneinfo:src`
3. Mapping from JIT code to DEX code and basic information about the compiled trace: `root@android:/ # cat /data/local.prop dalvik.vm.extra-opts=-Xjitvtuneinfo:dex`
4. JIT data collection. By default, JIT collection is disabled if you do not supply any options: `root@android:/ # cat /data/local.prop dalvik.vm.extra-opts=-Xjitvtuneinfo:none`

Additionally, if your Dalvik JVM supports instruction scheduling, disable it by adding `-Xnoscheduling` at the end of `dalvik.vm.extra-opts`. For example:

```
root@android:/ # cat /data/local.prop dalvik.vm.extra-opts=-Xjitvtuneinfo:src -
Xnoscheduling
```

NOTE:

Java analysis currently requires an instrumented Dalvik JVM. Android systems running on the 4th Generation Intel® Core™ processors or Android systems using ART vs. Dalvik for Java are not instrumented to support JIT profiling. You do not need to specify `--jitvtuneinfo=`*N*.

Enabling Java Analysis for Code Generated with ART* Compiler

To enable a source-level analysis, the VTune Amplifier requires debug information for the analyzed binary files. By default, the ART compiler does not generate the debug information for Java code. Depending on your usage scenario, you may choose how to enable generating the debug information with the ART compiler:

NOTE:

For releases prior to Android 6.0 Marshmallow*, the `--generate-debug-info` in the examples below should be replaced with `--include-debug-symbols`.

To Do This:	Do This:

Profile a 3rd party application or system application installed as an `.apk` file	**1.** Set the system property `dalvik.vm.dex2oat-flags` to `--generate-debug-info`:
	`adb shell setprop dalvik.vm.dex2oat-flags --generate-debug-info`
	2. If you use `--compiler-filter=interpret-only`, set the optimization level to `O2`:
	`adb shell setprop dalvik.vm.dex2oat-filter O2`
	3. (Re-)install the application.
	`adb shell install -r TheApp.apk`

Profile all applications installed as `.apk` or `.jar` files by re-building the	**1.** On your host system, open the `/build/core/dex_preopt_libart.mk` file, located in your Android OS directory structure.

To Do This:	Do This:
Android image when pre-optimization for private applications is enabled (`LOCAL_DEX_PREOPT:=true` property set in `device.mk`)	**2.** Modify the `--no-generate-debug-info` line to `--generate-debug-info` and save and close the file. **3.** Rebuild the Android image and flash it to your device. **4.** If you are using an Android image that is not PIC configured (`WITH_DEXPREOPT_PIC:=false` property set in `device.mk`), generate `classes.dex` from odex using the `patchoat` command. `classes.dex` should appear in `/data/dalvik-cache/x86/system@app@appname@appname.apk@classes.dex`
Profile all applications installed as `.apk` or `.jar` files by re-building the Android image when pre-optimization for private applications is disabled (`LOCAL_DEX_PREOPT:=false` property set in `device.mk`)	**1.** Set the system property `dalvik.vm.dex2oat-flags` to `--generate-debug-info`: `adb shell rm -rf /data/dalvik-cache/x86/system@app@webview@webview.apk@classes.dex` `adb shell setprop dalvik.vm.dex2oat-flags --generate-debug-info` **2.** Stop and start the adb shell: `adb shell stop` `adb shell start` **3.** Generate the dex file: `adb shell ls /data/dalvik-cache/x86/system@app@webview@webview.apk@classes.dex` `adb pull /data/dalvik-cache/x86/system@app@webview@webview.apk@classes.dex`
Profile an application executed by the `dalvikvm` executable	Add the compiler option `--generate-debug-info` followed by `-Xcompiler-option`. Make sure the application has not been compiled yet. `rm -f /data/dalvik-cache/*/*TheApp.jar*` `adb shell dalvikvm -Xcompiler-option --include-debug-symbols -cp TheApp.jar`
Profile system and core classes	Set the system property `dalvik.vm.image-dex2oat-flags` to `--generate-debug-info` and force recompilation: `adb shell stop` `adb shell rm -f /data/dalvik-cache/*/*` `adb shell setprop dalvik.vm.dex2oat-flags --generate-debug-info` `adb shell setprop dalvik.vm.image-dex2oat-flags --generate-debug-info` `adb shell start` If you run the application before the system classes are compiled, you should add another compiler option `-Ximage-compiler-option --generate-debug-info`: `adb shell rm -f /data/dalvik-cache/*/*` `adb shell dalvikvm -Xcompiler-option --generate-debug-info -Ximage-compiler-option --generate-debug-info -cp TheApp.jar`

To Do This:	Do This:
NOTE: This action is required if Java core classes get compiled to the `/data/dalvik-cache/` subdirectory. Manufacturers may place them in different directories. If manufactures supply the precompiled `boot.oat` file in `/system/framework/x86`, Java core classes will not be resolved because they cannot be re-compiled with debug information.	

TIP:

If you are able to see the `--generate-debug-info` option in the logcat output (`adb logcat *:S dex2oat:I`), the compiler uses this option.

Preparing an Android* Application for Analysis

Before starting an analysis with the VTune Amplifier, make sure your Android application is compiled with required settings:

Compilation Settings

Performance analysis is only useful on binaries that have been optimized and have symbols to attribute samples to source code. To achieve that:

- Compile your code with release level settings (for example, do not use the `/O0` setting on GCC*).
- Do not set `APP_OPTIM` to `debug` in your `Application.mk` as this setting disables optimization (it uses `/O0`) when the compiler builds your binary.
- To run performance analysis (Basic Hotspots) on non-rooted devices, make sure to compile your code setting the `debuggable` attribute to `true` in `AndroidManifest.xml`.

NOTE:

If your application is debuggable (`android:debuggable="true"`), the default setting will be `debug` instead of `release`. Make sure to override this by setting `APP_OPTIM` to `release`.

By default, the Android NDK build process for Android applications using JNI creates a version of your `.so` files with symbols.

The binaries with symbols included go to `[ApplicationProjectDir]/obj/local/x86`.

The stripped binaries installed on the target Android system via the `.apk` file go to `[ApplicationProjectDir]/libs/x86`. These versions of the binaries cannot be used to find source in the VTune Amplifier. However, you may collect data on the target system with these stripped binaries and then later use the binaries with symbols to do analysis (as long as it is an exact match).

When the VTune Amplifier finishes collecting the data, it copies `.so` files from the device (which have had their symbols stripped). This allows the very basic functionality of associating samples to assembly code.

Search Directories for Android* Targets

For accurate module resolution and source analysis of your Android* application, make sure to specify search paths for binary and source files when configuring performance analysis:

- from command line, use the `--search-dir`/`--source-search-dir` options; for example:

```
host>./amplxe-cl --collect advanced-hotspots -r system_wide_r@@@ --search-dir ~/
AndroidOS_repo/out/target/product/ctp_pr1/symbols/
```

- from GUI, use the Binary/Symbol Search and Source Search dialog boxes

If you have not set the project search directories at the time of collection or import, you will not be able to open the source code. Only Assembly view will be available for source analysis.

Consider the following when adding search paths:

- By default, the VTune Amplifier pulls many binaries from the target device.
- The Kernel `[vmlinux]` is one file that does not contain symbols on the target device. Typically it is located in `[AndroidOSBuildDir]/out/target/product/[your target]/linux/kernel/vmlinux`.
- Many operating system binaries with symbols are located in either `[AndroidOSBuildDir]/out/target/product/[`*`your target`*`]/symbols`, or `[AndroidOSBuildDir]/out/target/product/[`*`your target`*`]/obj`.
- Application binaries with symbols are located in `[AndroidAppBuildDir]/obj/local/x86`.
- Application source files for the C/C++ modules are usually located in `[AndroidAppBuildDir]/jni`, not in `[AndroidAppBuildDir]/src` (where the Java *source files are). Some third-party software in Android does not provide binaries with symbols. You must contact the third party to get a version of the binaries with symbols.
- You can see if a binary has symbols by using the `file` command in Linux and make sure that it says `not stripped`.

```
 file MyBinary.ext
MyBinary.ext: ELF 32-bit LSB shared object, Intel 80386, version 1
(SYSV), dynamically linked, not stripped
```

Intel® Xeon Phi™ Coprocessor Setup

- Build the target application on the host with full optimizations, which is recommended for performance analysis.
- **Windows* OS only**: When using an offload or cross compiler, make sure to manually install binary utilities (Binutils) included in the Intel Xeon Phi installation zip file package. For installation instructions, please refer to the Intel Compiler documentation.
- Create a user account on the coprocessor. See the Intel MPSS installation document for details.
- For native Intel Xeon Phi application analysis, configure SSH access for remote collection to the Intel Xeon Phi coprocessor card.
- For native Intel Xeon Phi application analysis, copy the application to the coprocessor card.

  For example, on Linux*:

```
scp matrix.mic mic0:/tmp
```

  On Windows:

```
pscp matrix.mic mic0:/tmp
```

You may add this command to a build script to automate a copy action after the binary recompilation. Or you can mount the host directory so that the binary is visible on the Intel Xeon Phi coprocessor. See the *NFS Mounting a Host Export* topic in the **Intel® Manycore Platform Software Stack (Intel® MPSS)** help for details.

NOTE:

Make sure you have copied any data files needed by your application to the card in a known location.

- **Windows* OS only**: Set the execution permissions, for example:

```
plink -l user -i C:\Users\jsmith\Documents\privatekey.ppk user@192.168.1.100  chmod "+x" /tmp/
matrix.mic
```

Using Intel® VTune™ Amplifier with a Virtual Machine

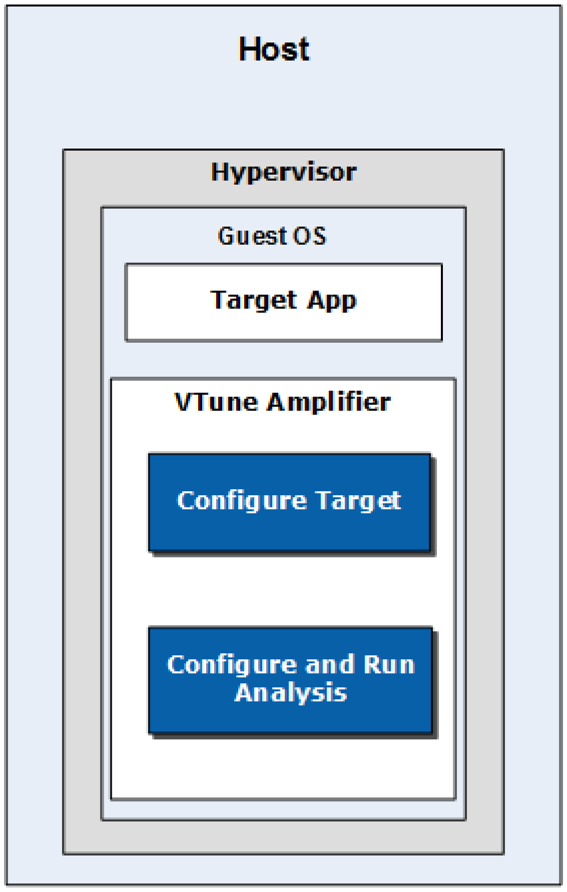
Virtual machines are made up of the following components:

- Host operating system: system from which the virtual machine is accessed. Supported host systems: Linux*, macOS*, Windows*
- Virtual machine manager (VMM): tool used to access and manage the virtual machine. Examples: VMware*, Parallels*
- Guest operating system: system accessed via the VMM and profiled using Intel VTune Amplifier. Supported guest systems: Linux*, Windows*

In most cases, Intel VTune Amplifier is installed on the guest operating system and analysis is run on the guest system. The guest system may not have full access to the system hardware to collect performance data. Analysis types that require access to system hardware, such as those that require precise event counters, will not work on a virtual machine.

NOTE:

Typically the host operating system has access to the system hardware to collect performance data, but there are cases in which the host system may also be virtualized. If this is the case and you want to collect performance data on the host system, treat the host system as you would a guest system and assume that it no longer has the same level of access to the system hardware.



Virtual Machine Host/Guest Support

A typical virtualized environment includes a host operating system, which boots first and from which the VMM is loaded, and virtual machines (VMs) running guest operating systems. There are multiple combinations of each and support varies based on each component. For example, the VMM Parallels can be used only with a macOS host system, but can analyze performance with VTune Amplifier installed on either a Linux or Windows guest system.

	Linux Host	**Windows Host**	**macOS Host**
Linux Guest	KVM XEN Project VMware	VMware	Parallels
Windows Guest	VMWare	VMware	Parallels
macOS Guest	--	--	--

Additional steps are required to enable performance analysis on the virtual machine after Intel VTune Amplifier is installed. For more information about installing VTune Amplifier on a virtual machine, see the VTune Amplifier installation guide for your host operating system. Refer to the following topics to enable the vPMU for your hypervisor:

- Enable Parallels* Desktop Analysis

Intel VTune Amplifier Analysis Type Support

Support for VTune Amplifier analysis types varies depending upon which counters have been virtualized by the VMM. If you run an analysis type that cannot be run in a virtualized environment, VTune Amplifier displays a warning message.

In general, the Basic Hotspots analysis type will work on every supported VMM because the analysis type does not require access to the system hardware. The Advanced Hotspots analysis type and other analysis types that use hardware event-based sampling collection have limited reporting functionality. For example, it will not include accurate results for stacks or call counts because this data relies on information provided by precise events (uncore). Running analysis types that rely on precise events will return results, but the collected data will be incomplete or distorted. That is, the result may not point to the actual instruction that caused the event, which can be difficult to differentiate from correct events. Refer to the documentation for your VMM to understand which counters have been virtualized.

- **Full Features**
  - Basic Hotspots
- **Limited Features**
  - Advanced Hotspots: Stacks, Call Counts, and Loop Trip Counts not supported
  - General Exploration: Stacks not supported

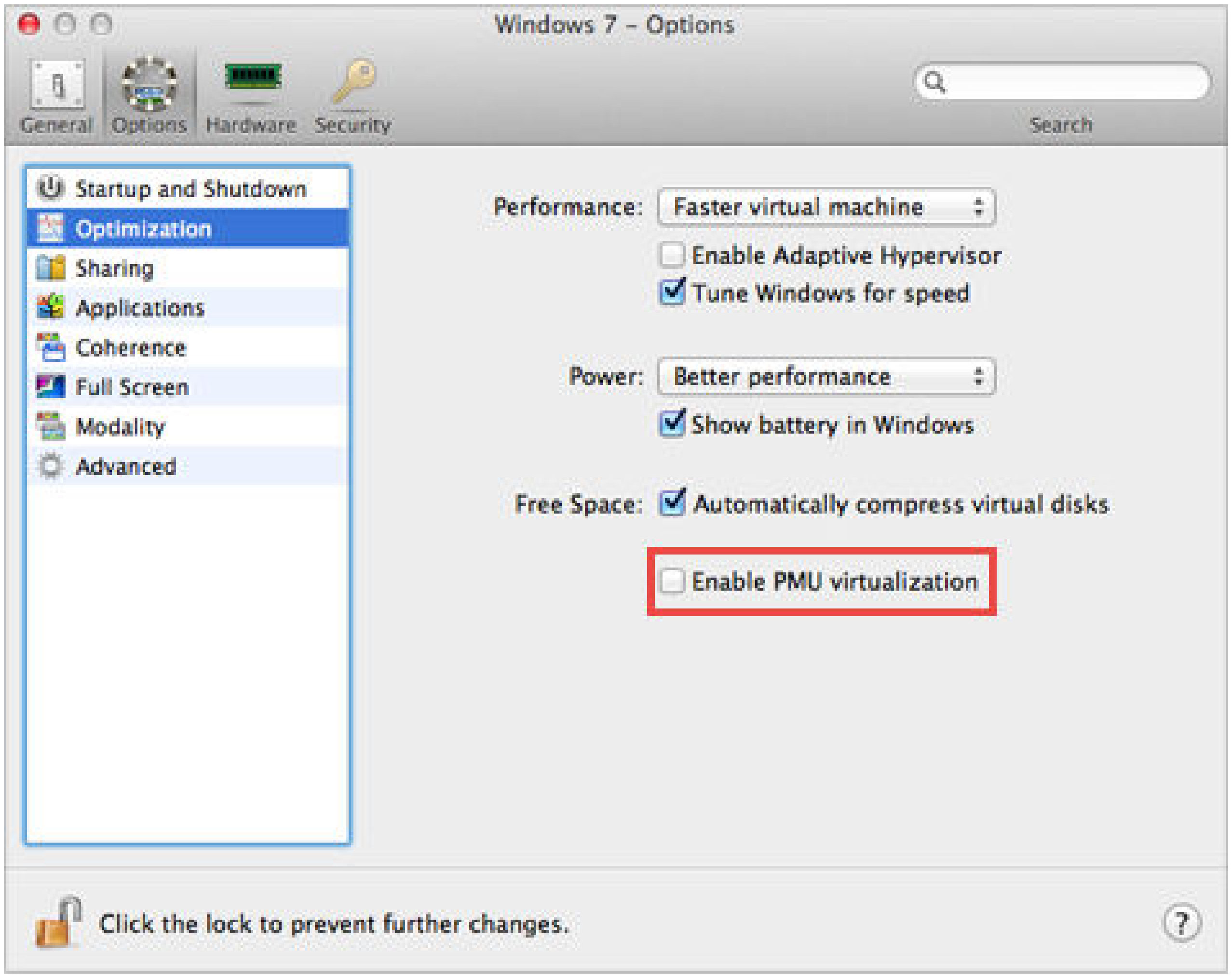
Enabling Intel® VTune™ Amplifier Analysis on a Parallels* Guest System

Intel VTune Amplifier can be installed and configured to analyze performance on a Parallels guest system.*

Parallels* Desktop users can use Intel® VTune™ Amplifier to analyze a Windows* or Linux* virtual guest system using a macOS* host. Intel VTune Amplifier is installed and run on the guest system. Additional information about installing VTune Amplifier is available from the installation guides. Refer to the installation guide for the guest system operating system (Windows or Linux).

Use the following steps to enable event based sampling analysis inside Parallels virtual machines. Refer to the Parallels documentation for the most up-to-date information.

1. Open the configuration options:
   - Click the Parallels icon in the menu bar, press and hold the Option (Alt) key, and choose **Configure**.
   - Choose **Virtual Machine** > **Configure** from the Parallels Desktop menu bar at the top of the screen.
2. Select the **Options** tab.
3. Select **Optimization**.
4. Select the **Enable PMU virtualization** checkbox.



Performance: Faster virtual machine

Enable Adaptive Hypervisor

Tune Windows for speed [✓]

Power: Better performance

Show battery in Windows [✓]

Free Space: Automatically compress virtual disks [✓]

Enable PMU virtualization

Click the lock to prevent further changes.

See Also

Using VTune Amplifier with a Virtual Machine

Configuring SSH Access for Remote Collection

For remote collection on a Linux host system, set up the password-less mode as follows:

1. Generate the key with an empty passphrase:

   ```
   host> ssh-keygen
   ```
2. Add the key to the ssh-agent:

   ```
   host> ssh-add
   ```

   If the ssh-agent has not started, launch it as follows:

   ```
   host> eval "$(ssh-agent -s)"
   ```
3. Copy the key to target system:

   ```
   host> ssh-copy-id user@target
   ```

   For the Intel Xeon Phi coprocessor: `ssh-copy-id mic0`

   Alternatively, if you do not have `ssh-copy-id` on your host system, do the following:

   ```
   host> cat ~/.ssh/id_rsa.pub | ssh user@target 'cat >> ~/.ssh/authorized_keys'
   ```

```
host> ssh user@target 'chmod 600 ~/.ssh/authorized_keys'
```

For the Intel Xeon Phi coprocessor:

```
host> cat ~/.ssh/id_rsa.pub | ssh mic0 'cat >> ~/.ssh/authorized_keys'

host> ssh mic0 'chmod 600 ~/.ssh/authorized_keys'
```

4. Verify that a password is not required anymore, for example:

```
host> ssh user@target ls
```

For the Intel Xeon Phi coprocessor: `ssh mic0 ls`

Sampling Drivers

Intel® VTune™ Amplifier uses kernel drivers to enable the hardware event-based sampling. VTune Amplifier installer automatically uses the Sampling Driver Kit to build drivers for your kernel with the default installation options. If the drivers were not built and set up during installation (for example, lack of privileges, missing kernel development RPM, and so on), VTune Amplifier provides an error message and, on Linux* and Android* systems, enables driverless sampling data collection based on the Linux Perf* tool functionality, which has a limited scope of analysis options. But you may still enable a full-scale sampling data collection by:

- Building and managing the sampling driver

NOTE:

- You may need kernel header sources and other additional software to build and load the kernel drivers on Linux. For details, see the `README.txt` file in the `sepdk/src`directory.
- A Linux kernel update can lead to incompatibility with VTune Amplifier drivers set up on the system for event-based sampling (EBS) analysis. If the system has installed VTune Amplifier boot scripts to load the drivers into the kernel each time the system is rebooted, the drivers will be automatically re-built by the boot scripts at system boot time. Kernel development sources required for driver rebuild should correspond to the Linux kernel update.
- If you loaded the drivers but do not use them and no collection is happening, there is no execution time overhead of having the drivers loaded. The memory overhead is also minimal. You can let the drivers be loaded at boot time (for example, via the `install-boot-script`, which is used by default) and not worry about it. Unless data is being collected by the VTune Amplifier, there will be no latency impact on system performance.

Drivers Installation Options on Linux* Systems

During product installation on Linux OS, you may control the drivers installation options via the **Change advanced options** menu item. VTune Amplifier provides the following options:

Use This Option	To Do This
Sampling driver install type [build driver (default) / driver kit files only]	Choose the driver installation option. By default, VTune Amplifier uses the Sampling Driver Kit to build the driver for your kernel. You may change the option to **driver kit files only** if you want to build the driver manually after installation.
Driver access group [vtune (default)]	Set the driver access group ownership to determine which set of users can perform the collection on the system. By default, the group is `vtune`. Access to this group is not restricted. To restrict access, see the **Driver permissions** option below. You may set your own group during installation in the Advanced options or change it manually after installation by executing: `./boot-script --group <your_group>` from the `<install_dir>/sepdk/src` directory.

Use This Option	To Do This
Driver permissions [660 (default)]	Change permissions for the driver. The default permissions allow any user to access the driver. Using this access the user can profile the system, an application, or attach to a process.
Load driver [yes (default)]	Load the driver into the kernel.
Install boot script [yes (default)]	Use a boot script that loads the driver into the kernel each time the system is rebooted. The boot script can be disabled later by executing: `./boot-script --uninstall` from the `<install_dir>/sepdk/src` directory.
Enable per-user collection mode [no (default) / yes]	Install the hardware event-based collector driver with the per-user filtering on. When the filtering is on, the collector gathers data only for the processes spawned by the user who started the collection. When it is off (default), samples from all processes on the system are collected. Consider using the filtering to isolate the collection from other users on a cluster for security reasons. The administrator/root can change the filtering mode by rebuilding/restarting the driver at any time. A regular user cannot change the mode after the product is installed. **NOTE:** For MPI application analysis on a Linux* cluster, you may enable the **Per-user Hardware Event-based Sampling** mode when installing the Intel Cluster Studio XE. This option ensures that during the collection the VTune Amplifier collects data only for the current user. Once enabled by the administrator during the installation, this mode cannot be turned off by a regular user, which is intentional to preclude individual users from observing the performance data over the whole node including activities of other users. After installation, you can use the respective `-vars.sh` files to set up the appropriate environment (PATH, MANPATH) in the current terminal session.
Driver build options ...	Specify the location of the kernel header files on this system, the path and name of the C compiler to use for building the driver, the path and name of the make command to use for building the driver.

Building and Managing the Sampling Drivers

Intel® VTune™ Amplifier uses a sampling driver to enable the hardware event-based sampling analysis. The sampling driver is installed by default during product installation. If the driver was not properly installed or unavailable for the current system, VTune Amplifier provides an error message and, on Linux* and Android* systems with support, enables driverless sampling data collection based on the Linux Perf* tool functionality, which has a limited scope of analysis options.

Depending on your target system, do the following to ensure a successful sampling collection:

- Linux* targets:
  - Make sure the driver is installed.
    - Build the driver, if required.

        - Install the driver, if required.
    - Verify the driver configuration.
- Android* targets: Verify the sampling driver is installed. If required, build and install the driver.
- Intel Xeon Phi™ coprocessor: Verify the sampling driver is installed correctly. If required, install the driver.

For some analysis types on Linux* and Android* target systems, the VTune Amplifier may automatically enable a driverless event-based sampling collection with a limited set of analysis options. But you still may build and load the sampling driver as a root user after product installation and enable a full-scale sampling collection.

Managing the Sampling Driver for Linux Targets

Prerequisites for remote target systems: You need root access to the target system.

To verify that the sampling driver is installed correctly (host only):

1. Check whether the sampling drivers are installed:

   ```
   > cd <install-dir>/sepdk/src
   ```

   ```
   > ./insmod-sep -q
   ```

   This provides information on whether the drivers are currently loaded and, if so, what the group ownership and file permissions are on the driver devices.
2. Check group permissions.

   If drivers are loaded, but you are not a member of the group listed in the `query` output, request your system administrator to add you to the group. By default, the driver access group is `vtune`. To check which groups you belong to, type `groups` at the command line. This is only required if the permissions are other than 660 or 666.

To verify kernel configuration:

1. Make sure that the kernel header sources are present on your host system. The kernel version should be 2.6.28 or later. To find the kernel version, explore `kernel-src-dir/include/linux/utsrelease.h`, or, depending on the kernel version: `kernel-src-dir/include/generated/utsrelease.h`. For more details, see the `README.txt` file in the `sepdk/src` directory.
2. Make sure the following options are enabled in the kernel configuration for hardware event-based sampling (EBS) collection:

   - `CONFIG_MODULES=y`
   - `CONFIG_MODULE_UNLOAD=y`
   - `CONFIG_PROFILING=y`
   - `CONFIG_SMP=y`
   - `CONFIG_TRACEPOINTS=y` (optional but recommended)
3. In addition to the options above, make sure the following options are enabled in the kernel configuration for EBS collection *with stacks*:

   - `CONFIG_KPROBES=y`
   - `CONFIG_RING_BUFFER=y`
   - `CONFIG_FRAME_POINTER=y` (optional but recommended for kernel stack analysis)
4. For remote target systems, determine if signed kernel modules are required (`CONFIG_MODULE_SIG_FORCE=y`). If they are, you must have the signed key that matches your target system.

   If you are building the sampling drivers from a fresh kernel source and want to use it for an existing target system, get the original key files and sign the sampling driver with the original key. Alternatively, build the new kernel and flash it to the target device so the target device uses your kernel build.

To build the driver if it is missing:

Prerequisites:

- You need kernel header sources and other additional software to build and load the kernel drivers on Linux. Refer to the Verify kernel configuration section.
- To cross-build drivers for a remote target Linux system, extract the package from the `<install_dir>/target` folder to `<extract_dir>`.

1. Change the directory to locate the build script:
   - To build drivers for a local system: `> cd <install_dir>/sepdk/src`
   - To cross-build drivers for a remote target system: `> cd <extract_dir>/sepdk/src`
2. Use the `build-driver` script to build the drivers for your kernel. For example:

- `> ./build-driver`

  The script prompts the build option default for your local system.
- `> ./build-driver -ni`

  The script builds the driver for your local system with default options without prompting for your input.
- `> ./build-driver -ni -pu`

  The script builds the driver with the per-user event-based sampling collection on without prompting for your input.

-
  ```
  > ./build-driver -ni \

  --c-compiler=i586-i586-xxx-linux-gcc \

  --kernel-version="<kernel-version>" \

  --kernel-src-dir=<kernel-source-dir> \

  --make-args="PLATFORM=x32 ARITY=smp"

  --install-dir=<path>
  ```

  The script builds the drivers with a specified cross-compiler for a specific kernel version. This is usually used for the cross-build for a remote target system on the current host. This example uses the following options:

  - `-ni` disables the interactive during the build.
  - `--c-compiler` specifies the cross build compiler. The compiler should be available from the PATH environment. If the option is not specified, the host GCC compiler is used for the build.
  - `--kernel-version` specifies the kernel version of the target system. It should match the `uname -r` output of your target system and the UTS_RELEASE in `kernel-src-dir/include/generated/utsrelease.h` or `kernel-src-dir/include/linux/utsrelease.h`, depending on your kernel version.
  - `--kernel-src-dir` specifies the kernel source directory.
  - `--make-args` specifies the build arguments. For a 32-bit target system, use `PLATFORM=x32`. For a 64-bit target system, use `PLATFORM=x32_64`
  - `--install-dir` specifies the path to a writable directory where the drivers and scripts are copied after the build succeeds.

Please use `./build-driver -h` to get the detailed help message on the script usage.

To install the drivers:

Prerequisites for remote target systems : Copy the `sepdk/src` folder or the folder specified by the `--install-dir` option when building the driver to the target system using ssh, ftp, adb, sdb, or other supported means.

1. If building the drivers succeeds, install them manually with the `insmod-sep` script:

```
> cd <install_dir>/sepdk/src
> ./insmod-sep -r -g <group>
```

where `<group>` is the group of users that have access to the driver. If the `-g` option is not specified, the group `vtune` will be used by default.

To install the driver that is built with the per-user event-based sampling collection on, use the `-pu (-per-user)` option as follows:

```
> ./insmod-sep -g <group> -pu
```

If you are running on a resource-restricted environment, add the `-re` option as follows:

```
> ./insmod-sep -re
```

2. Enable the Linux system to automatically load the drivers at boot time:

```
> cd <install_dir>/sepdk/src
> ./boot-script --install -g <group>
```

The `-g <group>` option is only required if you want to override the group specified when the driver was built.

To remove the driver on a Linux system, run:

```
./rmmod-sep -s
```

To build the sampling driver as RPM using build services such as Open Build Service (OBS):

Use the `sepdk.spec` file located at the `<install_dir>/sepdk/src` directory.

Managing the Sampling Driver for Android Targets

On some versions of Android systems, including most of the Intel supplied reference builds for SDVs, the required drivers are pre-installed in `/lib/modules` or `/system/lib/modules`. If the drivers are not pre-installed in any of these directories, you need to build them manually from the command line. Optionally, you can get the drivers integrated into the Android build so that they are built and installed when the operating system is built.

Android requires signed drivers. Every time the Android kernel is built, a random private/public key is generated. Drivers must be signed with the random private key to be loaded. The drivers (`socperf2_x.ko`, `pax.ko` , `sep4_x.ko` , and `vtsspp.ko`) must be signed with the same key and be compiled against the same kernel headers/sources as what is installed on the Android target system.

VTune Amplifier has options for building a new driver on the Linux host system and installing it on a target Android system. This is not the default and will only work if you provide the proper kernel headers/sources and a signing key. For example, the VTune Amplifier uses the `--with-drivers` option for building PMU drivers and `--kernel-src-dir` option for providing the configured kernel headers/sources tree path.

To build the sampling drivers on the host Linux system, enter:

```
<install-dir>/bin{32,64}/amplxe-androidreg.sh --package-command=build --with-drivers --
kernel-src-dir=/ path /to/configured/kernel/sources [--jitvtuneinfo=jit|src|dex|none]
```

To install the sampling drivers from the Linux host, enter:

```
<install-dir>/bin{32,64}/amplxe-androidreg.sh --package-command=install --with-drivers
--kernel-src-dir=/ path/to/configured/kernel/sources [--jitvtuneinfo=jit|src|dex|none]
```

To sign the drivers after the drivers are built:

Typically the VTune Amplifier automatically signs drivers if kernel sources with the keys are available when it builds the drivers. Otherwise, to manually sign the drivers, use the following command:

```
$KERNEL_SRC/source/scripts/sign-file CONFIG_MODULE_SIG_HASH $KERNEL_SRC/
signing_key.priv $KERNEL_SRC/signing_key.x509 driver.ko
```

where the `CONFIG_MODULE_SIG_HASH` value is extracted from the `$KERNEL_SRC/.config` file.

NOTE:

You need the "exact" signing key that was produced at the time and on the system where your kernel was built for your target.

Managing the Sampling Driver for the Intel Xeon Phi™ Coprocessor

For hardware event-based sampling analysis on an Intel Xeon Phi coprocessor based on Intel Many Integrated Core (Intel MIC) architecture, sampling drivers should be installed on the coprocessor cards to be sampled. Typically, the VTune Amplifier installs the drivers by default during product installation. If for some reasons the sampling driver was not properly installed on the Intel Xeon Phi coprocessor card(s) or needs to be reinstalled, follow the instructions below.

To install the sampling drivers manually, run the following command lines (on Linux):

For Intel® Xeon Phi™ coprocessor code named Knights Corner:

1. Copy the install files to a system location:

   `> <host_install_dir>/bin64/k1om/micboot_install.` (Linux) or `micboot_install` (Windows).
2. Start (or restart) the Intel Xeon Phi coprocessor service (this also restarts the sampling drivers once the files are copied in the previous step):

   Linux:

   Windows:

   `> micctrl -w`

For Intel® Xeon Phi™ coprocessor code named Knights Landing (Linux only): Copy the install files to a system location and load drivers:

```
> <host_install_dir>/bin64/knl-lb/mic_install.sh
```

To uninstall the sampling server and drivers, run the following commands (on Linux). For example, on Linux:

For Intel® Xeon Phi™ coprocessor code named Knights Corner:

```
>

> <host_install_dir>/bin64/k1om/micboot_uninstall.

>

> micctrl -w
```

For Intel® Xeon Phi™ coprocessor code named Knights Landing:

```
> <host_install_dir>/bin64/knl-lb/mic_uninstall.sh
```

NOTE:

You may receive error messages when restarting the service. Please refer to the Intel Manycore Platform Software Stack (Intel MPSS) documentation for details.

Getting Started with Intel® VTune™ Amplifier

5

Information about using Intel® VTune™ Amplifier after installing the product is available from the Getting Started page, which is available from the following location: `<install-dir>/documentation/<language>/welcomepage/get_started.htm`

Uninstalling Intel® VTune™ Amplifier 6

Use the following steps to uninstall Intel® VTune™ Amplifier from your system:

1. Navigate to the **Applications** folder.
2. Double-click the `Intel VTune Amplifier <version> Uninstaller.app` file to start the process of uninstalling the product.

Index