Disclaimer and Legal Information

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting: http://www.intel.com/design/literature.htm

*Other names and brands may be claimed as the property of others.

Copyright © 2015, Intel Corporation. All rights reserved.
Revision History

<table>
<thead>
<tr>
<th>Revision Number</th>
<th>Description</th>
<th>Revision Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5</td>
<td>Revised legal disclaimer added</td>
<td>February 2015</td>
</tr>
<tr>
<td>3.3</td>
<td>Revised Revision History table</td>
<td>May 2014</td>
</tr>
<tr>
<td>3.3</td>
<td>Revised Section 8. Corrected Figure 1 in Section 3.</td>
<td>April 2014</td>
</tr>
</tbody>
</table>
Interacting with the Intel® Xeon Phi™ Coprocessor

CIFS Share on Windows

Host File System Share

NFS Share on Windows*

CIFS Share on Windows

13

14

14.1

14.2

14.3

14.4

14.5

14.6

14.7

14.8

User Groups

Deleting Users from MICUSERS Group

Transferring Files to and from the Intel® Xeon Phi™ Coprocessor

Using PowerShell* Cmdlets to Create NFS Share (optional)

Specify the NFS Share Permissions

Add the NFS Server

Specify the NFS Share Permissions

14.3.1 Transferring Files to and from the Intel® Xeon Phi™ Coprocessor

14.1 External Tools

14.2 Special User Account 'micuser'

14.3 Generating Public and Private SSH Keys

14.4 User Groups

14.4.1 Deleting Users from MICUSERS Group

14.5 Running the COI Tutorials (optional)

14.6 Running the MYO Tutorials (optional)

14.7 Running the SCIF Tutorials (optional)

14.8 Running the MicMgmt Tutorials (optional)

6.1.2 The file Filelist Directive

6.1.3 The slink Filelist Directive

6.1.4 The nod Filelist Directive

6.1.5 The pipe Filelist Directive

6.1.6 The sock Filelist Directive

6.2 Adding Persistent Files to Intel® Xeon Phi™ Coprocessor Filesystem

6.3 Creating the Download Image File

6.4 Adding Files to the Root File System

6.4.1 Adding Files by Copying

7

Intel® Xeon Phi™ Coprocessor Information Tool: Micinfo

7.1 Simple Method

7.2 For Advanced Users

8

Intel® Xeon Phi™ Coprocessor Platform Status Panel: Micsmc

9

Intel® Xeon Phi™ Coprocessor Verification Tool: Miccheck

10

Intel® Xeon Phi™ Coprocessor RAS Tool: Micras

10.1 Simple Method

10.1.1 Configure Micras Service to Start when Windows* Boots (optional)

10.2 For Advanced Users

11

Intel® Xeon Phi™ Coprocessor Utility: Micnativeloadex

12

Host File System Share

12.1 NFS Share on Windows* Server 2008 R2 SP1

12.1.1 Add the NFS Server

12.1.2 Provision a Folder for the NFS Share

12.1.3 Specify the NFS Share Permissions

12.2 NFS Share on Windows* Server 2012 and Windows* Server 2012 R2

12.2.1 Add the NFS Server

12.2.2 Provision a folder for the NFS Share

12.2.3 Specify the NFS Share Permissions

12.2.4 Use PowerShell* Cmdlets to Create NFS Share (optional)

12.3 Mount the NFS Share

13

CIFS Share on Windows

13.1 Mounting CIFS share with micctrl

13.2 Configuring a CIFS share with a bridge

13.3Unmounting CIFS share with micctrl

14

Interacting with the Intel® Xeon Phi™ Coprocessor

14.1 External Tools

14.2 Special User Account 'micuser'

14.3 Generating Public and Private SSH Keys

14.4 User Groups

14.4.1 Deleting Users from MICUSERS Group

14.5 Running the COI Tutorials (optional)

14.6 Running the MYO Tutorials (optional)

14.7 Running the SCIF Tutorials (optional)

14.8 Running the MicMgmt Tutorials (optional)
14.9 Troubleshooting ........................................................................................................... 58
15 Installing Intel® Xeon Phi™ Coprocessor Performance Workloads (optional) ......................................................................................................................... 60
   15.1 Requirements ........................................................................................................... 60
   15.2 Distributed Files .................................................................................................... 61
   15.3 Micp Python Installation ....................................................................................... 61
16 Important Considerations ............................................................................................. 63
   16.1 Disabling and Enabling the Memory Control Group (cgroup) ......................... 63
   16.2 Enabling Windows MIC GDB Debugging for Offload Processes ..................... 64
   16.3 Enabling Windows MIC Debugging for MYO Applications ............................... 64
   16.4 Copy RPMs to the Card Using PSCP ................................................................... 65
17 Related Documentation ................................................................................................. 66
   17.1 MYO Documentation ............................................................................................ 66
   17.1.1 MYO Tutorials & Other Document Location on Windows* ........................... 66
   17.2 COI Documentation ............................................................................................. 66
   17.2.1 COI Documentation for Windows* .................................................................. 66
   17.3 Intel® Xeon Phi™ Coprocessor Collateral ......................................................... 66

List of Tables

Table 1 Conventions and Symbols ................................................................................. 9
Table 2 Severity Levels ................................................................................................. 29

List of Figures

Figure 1 Boot process for Intel MPSS ........................................................................... 10
Figure 2 Internal Bridge Network ................................................................................. 15
Figure 3 External Bridge Network ................................................................................. 16
Figure 4 File Sharing screen ....................................................................................... 48
Figure 5 Advanced Sharing screen ............................................................................. 48
Figure 6 Permissions for Winshare window ................................................................. 49
1 Introduction

The Intel® Xeon Phi™ coprocessors are PCIe based add-in cards that run a version of Linux* tailored for these coprocessors. The Linux* OS for Intel® Xeon Phi™ coprocessors, as well as a range of drivers and utilities, are included in the Intel® Manycore Platform Software Stack (Intel® MPSS). The responsibilities of these drivers and utilities include:

- Placing the Linux* boot image and root file system into coprocessor memory.
- Controlling coprocessor booting, shutdown and reset.
- Providing an IP (over PCIe) networking connection to each coprocessor.
- Directing power management of each coprocessor.
- Supporting high speed data transfer to and from the coprocessor.

The PCIe bus is the only communication channel available to the Intel® Xeon Phi™ coprocessors. Therefore configuration and provisioning of the OS to be executed on each Intel® Xeon Phi™ coprocessor is performed by the host system in which the coprocessor is installed.

The Linux* kernel and file system image for the Intel® Xeon Phi™ coprocessors are installed into the host file system as part of Intel MPSS installation. The coprocessor file system image can be configured through the use of the micctrl utility described below and/or directly by the host root.

The micx64.sys driver is the component of Intel MPSS that provides PCIe bus access and implements the coprocessor boot process. To boot a coprocessor, micx64.sys injects the Linux* kernel image and a kernel command line into coprocessor memory and signals it to begin execution. A virtual network driver is installed as micvethx64.sys. Finally, micx64.sys directs power management of the installed coprocessors and provides a high speed data transfer over PCIe through its Intel® Symmetric Communications Interface (SCIF) driver.

Micctrl is a utility through which the user can control (boot, shutdown, reset) each of the installed Intel® Xeon Phi™ coprocessors. Micctrl also offers numerous options to simplify the process of configuring each coprocessor. Chapter 5 of this document, “The micctrl Utility”, describes the micctrl utility in detail.

---

1 Intel, Xeon, and Intel Xeon Phi are trademarks of Intel Corporation in the U.S. and/or other countries.
2 Post Installation Quick Configuration

After the installation of the EXE (consult the readme-windows.pdf file for installation instructions), the system administrator must complete the Intel® Xeon Phi™ coprocessor configuration before starting the Intel® Xeon Phi™ coprocessors.

2.1 Step 1: Ensure Admin Access

Users must be added to the MICUSERS group in order to log into the Intel® Xeon Phi™ coprocessor (refer to Section 14.4 for steps to create the MICUSERS group and add users to the filesystem).

User access to the Intel® Xeon Phi™ coprocessor node is provided through the secure shell utilities. Ensure the admin user has ssh keys. If no SSH keys exist, generate a set of SSH keys with an external utility (refer to Section 14.3 for key generation instructions).

2.2 Step 2: Change Configuration

Examine the global.xml and micN.xml files in the C:\Program Files\Intel\MPSS\ directory. If during installation, the base install directory was changed from the default, this path will differ. If the default configuration meets the requirements of the system, continue to Step 3. Otherwise, edit the configuration XML files (refer to Chapter 4, Configuration). If the micN.xml files do not exist, generate them by executing the following command as an Administrator:

Admin> micctrl -g

2.3 Step 3: Start the Intel® Xeon Phi™ Coprocessors

*Note:* In this document, lines preceded by User> are used to represent a Windows* host command prompt and lines preceded by Admin> are used to represent commands that must be performed by someone with Administrator privileges; text following this string on the same line represents commands to be executed in a Windows* command window.

The default configuration specifies that each Intel® Xeon Phi™ coprocessor card is booted when the host driver is loaded. This means that the Intel® Xeon Phi™ coprocessors will boot when the host system restarts. To start the Intel® Xeon Phi™ coprocessors, execute the following command as an Administrator:

Admin> micctrl --start
The call to micctrl will exit when it determines the Intel® Xeon Phi™ coprocessor cards have either booted successfully or failed to boot.

2.4 Conventions and Symbols

Table 1 Conventions and Symbols lists conventions used in this document.

<table>
<thead>
<tr>
<th><strong>Table 1 Conventions and Symbols</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>This type style</strong></td>
</tr>
<tr>
<td><strong>This type style</strong></td>
</tr>
<tr>
<td><strong>This type style</strong></td>
</tr>
<tr>
<td>[ items ]</td>
</tr>
<tr>
<td>{ item</td>
</tr>
<tr>
<td>... (ellipses)</td>
</tr>
<tr>
<td>\ (backslash)</td>
</tr>
<tr>
<td>micN</td>
</tr>
<tr>
<td>User&gt;</td>
</tr>
<tr>
<td>Admin&gt;</td>
</tr>
<tr>
<td>[micN] $</td>
</tr>
<tr>
<td>[micN] #</td>
</tr>
<tr>
<td>*</td>
</tr>
</tbody>
</table>
3 Intel MPSS Boot Process

Booting the Linux* kernel on the Intel® Xeon Phi™ coprocessor card requires a number of steps. Figure 1 shows the sequence of steps that are performed during the Intel MPSS boot process.

**Figure 1 Boot process for Intel MPSS**
3.1 Booting the Intel® Xeon Phi™ Coprocessor Card

This chapter describes the key steps that are performed during the Intel MPSS boot process on the Intel® Xeon Phi™ coprocessor card.

3.1.1 Kernel Command Line

On most Intel® based systems, loading and executing the Linux* kernel image is controlled by the grub boot loader. In the grub configuration file, each possible kernel definition contains a number of parameters to be passed to Linux* through its kernel command line. In the Intel MPSS boot system, this is done by micctrl parsing its configuration files. The kernel command line is created based on values in the configuration files and placed in the WMI entry cmdline for the driver to retrieve it.

3.1.2 Instruct the Driver to Boot the Intel® Xeon Phi™ Coprocessor Card

The micctrl utility requests the Intel® Xeon Phi™ coprocessor card to start executing the Linux* image by executing the BootMIC WMI method. This method is a link into the MIC driver through WMI.

The method ResetMIC may also be executed and will be discussed later.

When the driver receives the boot request, it first checks to see whether the coprocessor is in the ready state. If the coprocessor is not ready to boot it will return an error from WMI and will not attempt to boot the coprocessor. Otherwise it sets the state of the Intel® Xeon Phi™ coprocessor to booting.

The driver then saves the image file name for later retrieval through the image WMI entry. It also sets the mode to indicate it is booting a Linux* image.

The driver will copy the kernel command line setting request by the micctrl utility, along with a number of addresses in host memory required by various drivers in the Linux* image. It then copies the requested Linux* image file into the Intel® Xeon Phi™ coprocessor's memory.

The last step is to write to the Intel® Xeon Phi™ coprocessor register instructing it to start executing the injected image.

3.1.3 Linux* Kernel Executes

Executing the Linux* kernel code functions as it does on any Intel® based machine. It initializes hardware, starts kernel services, and sets all the CPUs to the “online” state. When the kernel is ready, it initializes its attached initial ram disk image and starts executing the init script in the image.

As on any Intel® based Linux* system, the initial ram disk contains the loadable modules required for the real root file system. Many of the arguments passed in the
kernel command line are addresses required for the modules to access host memory. The init script parses the kernel command line for needed information and loads the driver modules.

The last step is for the init script to check the root= parameter in the kernel command line for the type of device containing the root file system, and take the appropriate actions.

3.1.4 Root is the Initial Ram Disk

Setting the root to be the initial ram disk is for debug purposes only. The initial ram disk contains only a minimal set of tools and utilities.

3.1.5 Root is a Ram Disk Image

If the root is set to be a ram file system, an initial file system is written into the coprocessor memory.

After the file is written and the coprocessor booted, the init script creates a tmpfs (Linux* ram disk file system type) in Intel® Xeon Phi™ coprocessor card memory and extracts the compressed file system information into it. This image must contain everything needed to start a fully functional Linux* system.

The ram disk image is activated as the root device by calling the Linux* switch_root utility. This special utility instructs the Linux* kernel to remount the root device on the tmpfs mount directory, release all file system memory references to the old initial ram disk and start executing the new /sbin/init function.

/sbin/init performs the normal Linux* user level initialization. All the information required must have already been in the compressed cpio file.

3.1.6 Notify the Host that the Intel® Xeon Phi™ Coprocessor System is Ready

The last step of any of these initializations is to notify the host that the coprocessor card is ready for access. It does this by writing to its /sys/class/micnotify/notify/host_notified entry. This causes an interrupt into the host driver which updates the card’s state to online.
4 Configuration

This chapter focuses on configuring Intel® Xeon Phi™ coprocessor cards, including configuration files, kernel command line parameters, and authentication.

4.1 Configurable Components

On a typical Linux* system, the installation and configuration process is performed as a series of questions posed by the system and answered by the installer/operator. Since the Intel® Xeon Phi™ coprocessor cards do not have a file system of their own, this process is replaced by editing the configuration files and using the micctrl utility.

The configuration parameters have three categories:

1) Parameters that control loading the Intel® Xeon Phi™ coprocessor Linux* kernel onto the card and initiating the boot process.

2) Parameters to define the root file system to be used on the card.

3) Parameters to configure the host end of the virtual Ethernet connection.

4.2 Configuration Files

This section briefly discusses configuration file formats and use of the Include parameter to micctrl.

4.2.1 File Location and Format

Configuration is controlled by per card configuration files located in the C:\Program Files\Intel\MPSS directory. Each card has an associated micN.xml configuration file, where N is the integer ID of that card (for example: mic0.xml, mic1.xml, etc.).

Each of the configuration files contains a list of configuration parameters and their arguments.

4.3 Configuring Boot Parameters

The host system boots the Intel® Xeon Phi™ coprocessor card by injecting the Linux* kernel image and kernel command line into coprocessor memory and then instructing the coprocessor to start. To perform this operation, the host system reads the configuration files and builds the kernel command line from relevant parameters. By default, the boot parameters are placed in the per-card micN.xml files, allowing each card to be configured independently of the other cards. If a boot parameter is placed in the global.xml file then it will apply to all cards unless overridden.
4.3.1 Power Management

The PowerManagement parameter specifies the Intel® Xeon Phi™ coprocessor Linux* power management settings. The system owner can specify different power management settings by editing these values. The changes take effect upon executing either micctrl --start or micctrl -b.

4.3.2 Command Line

The CommandLine parameter controls what options are passed to the Intel® Xeon Phi™ coprocessor when an Intel® Xeon Phi™ coprocessor starts.

4.3.3 Networking

The Networking parameters specify various settings such as IPAddress, HostIPAddress, Subnet, MACAddress, and HostMACAddress. These settings will not take effect until the Intel® Xeon Phi™ coprocessor is restarted.

4.4 Root File System

Every Linux* system needs a root file system with a minimal set of files. Other nonessential files may be on the root or they may be on secondary mounts. Most modern Linux* OS releases assume the root file system will be large enough to install the complete release into. The Intel® Xeon Phi™ coprocessor embedded file system currently follows the same rule.

Files on the root fall into three categories: the binaries installed with the system, the files in the /etc directory, which are used for configuring parameters uniquely to an individual system, and the set of files for the users of the system.

Intel MPSS provides a set of configuration parameters that are used in building the root file system image. Refer to Section 4.3, Configuring Boot Parameters and Section 6.1, The File System Creation Process for more information.

User> route add default gw <IP Address of the Host machine>

4.5 Routing

Windows* provides functionality to route network traffic between two or more networks. IP routing is disabled by default in Windows*.

Steps to enable routing in windows:
1) Go to Run; type cmd.
2) Enter the following commands:
   User> sc config RemoteAccess start= auto
   User> sc start RemoteAccess
There is no change needed in the configuration files for each coprocessor in order to support this. Since micctrl utility currently does not support setting up the default gateway in the Intel® Xeon Phi™ coprocessor, it is necessary to manually change it using the following command:

```
User> route add default gw <IP Address of the Host machine>
```

## 4.6 Bridging

A network bridge is a way to connect two Ethernet segments or collision domains in a protocol independent way. It is a Link Layer device which forwards traffic between networks based on MAC addresses and is therefore also referred to as a Layer 2 device.

Two types of bridged networks are directly supported by MPSS.

### 4.6.1 Internal Bridging

Some distributed applications running on Intel® Xeon Phi™ coprocessors on a single node need to communicate between coprocessors, and perhaps with the host. An internal bridge allows for the connection of one or more Intel® Xeon Phi™ coprocessors within a single host system as a subnetwork. In this configuration, each Intel® Xeon Phi™ coprocessor can communicate with the host and with the other Intel® Xeon Phi™ coprocessors in the platform.

**Figure 2 Internal Bridge Network.**

Such a network configuration could, for example, be used to support communication between the ranks of an MPI application that is distributed across the Intel® Xeon Phi™ coprocessors and host. (However, use of the IBSCIF virtual InfiniBand* HCA driver will likely provide better performance.)
4.6.2 **External Bridging**

The external bridge configuration bridges Intel® Xeon Phi™ coprocessors to an external network. This is the typical configuration required when Intel® Xeon Phi™ coprocessors are deployed in a cluster to support remote communication among Intel® Xeon Phi™ coprocessors and/or Xeon® processors across different compute nodes.

This depicts a cluster in which the Intel® Xeon Phi™ coprocessors on each host node are bridged to the external network. The IP addresses in such a configuration can be assigned statically by the system administrator or by a DHCP server on the network, but must generally be on the same subnet.

InfiniBand* based networking is not shown in this figure. InfiniBand* based networking will usually provide significantly higher bandwidth than the IP networking supported by the MPSS Virtual Ethernet driver. Many clusters use Ethernet* networking for low bandwidth communication such as command and control and use InfiniBand* networking for high bandwidth communication as application data transfer.

**Figure 3 External Bridge Network**

To prepare for configuring this network topology, you should ensure that you have provided a large enough IP address space to accommodate the nodes of the externally bridged networks.
The micctrl Utility

The *micctrl* utility is a multi-purpose toolbox for the system administrator. It provides these categories of functionality.

- Card state control – boot and reset control.
- Configuration file initialization and propagation of values.
- Helper functions for modifying configuration parameters.

The *micctrl* utility requires a first argument specifying the action to perform, followed by option-specific arguments. The arguments may be followed by a list of Intel® Xeon Phi™ coprocessor card names, which is shown in the syntax statements as [mic card list]. The card list will be a list of the card names. **For example:** the list may be *mic1 mic3*, if these are the cards to control.

5.1 Card State Control

The *micctrl* utility provides mechanisms for individual card control. *Micctrl* controls Intel® Xeon Phi™ coprocessor card state and queries card state via the BootMIC WMI method. The *micname* value is literally the name of the mic card and will be in the format *mic0, mic1*, etc.

**Note:** Changing the state requires full administrative rights.

5.1.1 Booting Intel® Xeon Phi™ Coprocessor Cards

**Command syntax:**

```
micctrl -b [mic card list]
```

The Intel® Xeon Phi™ coprocessor card(s) must be in the *ready* state. The driver will inject the indicated Linux* image into the cards memory and start it booting.

5.1.2 Resetting Intel® Xeon Phi™ Coprocessor Cards

**Command syntax:**

```
micctrl -r [mic card list]
```

The Intel® Xeon Phi™ coprocessor card can be reset in any state. This command uses the ResetMIC WMI method. The driver will perform a soft reset on the card by setting the correct card PCI mapped register.

**NOTE:** Performing a reset may result in the loss of file data that has not been flushed to a remote file.
5.1.3 Waiting for Intel® Xeon Phi™ Coprocessor Card State Change

Command syntax:

    micctrl -w [mic card list]

The `wait` option waits for the status of the Intel® Xeon Phi™ coprocessor card to be either `online` or `ready`. It also allows for a brief pause to the `ready` state during mpss startup. It is intended for users to verify the mpss startup, shutdown, or reset procedure is complete. It has a built-in timeout value of 300 seconds.

5.1.4 Intel® Xeon Phi™ Coprocessor Card Status

Command syntax:

    micctrl -s [mic card list]

The `status` option displays the status of the Intel® Xeon Phi™ coprocessor cards in the system.

5.2 Helper Functions for Configuration Parameters

This section discusses command options for adding and removing users and groups.

5.2.1 Adding Users to the Intel® Xeon Phi™ Coprocessor File System

Adding a user to the Intel® Xeon Phi™ coprocessor file system is accomplished through the addition of a user to the MICUSERS user group. See Section 14.4 for detailed instructions on creating this group and adding users to the filesystem.

5.2.2 Removing Users from the Intel® Xeon Phi™ Coprocessor File System

Removing a user from the Intel® Xeon Phi™ coprocessor file system is accomplished through the deletion of a user from the MICUSERS user group. See Section 14.4.1 for detailed instructions on using this group to remove users from the filesystem.

5.2.3 Specifying the Host Secure Shell Keys

Command syntax:

    micctrl --addssh <user> -f <path to public key file>
    micctrl --addssh <username> <public-key>
The **--addssh** option adds the specified public key to the *authorized_keys* file in the coprocessor filesystem. The user must belong to the MICUSERS group for this to be effective on next boot. This command must be executed from a command prompt with full administrator privileges.

The public key file format can be OpenSSH public key format, PuTTY public key format, or PuTTY private key format (usually saved with a file extension .ppk).

Without specifying the `-f` option, the contents of an OpenSSH public key can be pasted onto the command line directly and will be added directly to the user’s *authorized_keys* file.

The secure shell keys will not be updated until the coprocessor is rebooted using micctrl. It is not sufficient to power cycle the host machine.

### 5.3 Other File System Helper Functions

#### 5.3.1 Updating the Compressed CPIO Image

**Command syntax:**

```
micctrl -g [mic card list]
```

The `-g` option updates the image from the parameters specified in configuration files and file lists. The new image will be used the next time the card boots.
6 Adding Software

Typical installations are not static, and usually require the system administrator to add additional files or directories to the Intel® Xeon Phi™ root file system that is downloaded to the card.

6.1 The File System Creation Process

In this section we describe the process of building a root file system.

The base component of a root filesystem is a filelist. These can be found in the installation directory under the filesystem subdirectory. There is one filelist for each coprocessor as well as a common filelist shared between all coprocessors.

Files can be added to the root file system by editing an existing filelist (for example: mic0.filelist) and adding the appropriate directives to the filelist.

There are six filelist directive types:

- `dir <name> <perms> <uid> <gid>`
- `file <name> <source> <perms> <uid> <gid>`
- `slink <name> <to> <perms> <uid> <gid>`
- `nod <name> <perms> <uid> <gid> <type> <major> <minor>`
- `pipe <name> <perms> <uid> <gid>`
- `sock <name> <perms> <uid> <gid>`

Each directive type is specific to one of six types of files available on a Linux* file system.

6.1.1 The dir Filelist Directive

The dir directive specifies a directory with name is to be created in the card file system. The perms, uid, and gid arguments specify the file’s permissions, user ID, and group ID, respectively. A typical entry is:

```
  dir /tmp 0777 0 0
```

The example defines the directory /tmp to be owned by user root and group root, and with global permissions for everybody.
### 6.1.2 The file Filelist Directive

The file directive specifies the file with name is to be created in the card file system image. The perms, uid, and gid arguments specify the file’s permissions, user ID, and group ID, respectively.

The source argument to file is relative to the location of the filelist itself.

**For example:** The following filelist directive in `C:\Program Files\Intel\MPSS\filesystem\mic0.filelist`:

```plaintext
file /etc/passwd etc/passwd 644 0 0
```

The file `/etc/passwd` will be added to the card file image and populated with the contents of the file `C:\Program Files\Intel\MPSS\filesystem\mic0\etc\passwd`. It will be owned by user root and group root, and with global read permission and root modification permission.

### 6.1.3 The slink Filelist Directive

The `slink` directive specifies that a symbolic link with name is to be created in the card file system image, and linked to source. The perms, uid, and gid arguments specify the symbolic link’s permissions, user ID, and group ID, respectively.

A typical use of symbolic links is found in the Linux* startup scripts. The filelist associated with the configuration parameter includes the following:

```plaintext
slink/etc/rc3.d/S80sshd ../init.d/sshd 0755 0 0
```

This directs the creation of a symbolic link on the cards file system accessing the `/etc/init.d/sshd` file when `/etc/rc.d/S80sshd` is accessed.

### 6.1.4 The nod Filelist Directive

The `nod` directive specifies that a device node with name and of type is to be created in the card file system image. `type` must be either the character `b` for block device or `c` for character device. The arguments major and minor must be integer values defining the correct values of the node. The perms, uid, and gid arguments specify the device node’s permissions, user ID, and group ID, respectively.

Most device nodes are created dynamically by a device driver. However, some legacy devices still require a hard coded entry.

**For example:** The filelist for BaseDir includes the following entry, which specifies the creation of a device node for the console:

```plaintext
nod/dev/console 0600 0 0 c 5 1
```
6.1.5 The pipe Filelist Directive

The pipe directive specifies that a named pipe device file with name is to be created in the card file system image. The perms, uid, and gid arguments specify the pipe’s permissions, user ID, and group ID, respectively.

6.1.6 The sock Filelist Directive

The sock directive specifies that a socket device file with name is to be created in the card file system image. The perms, uid, and gid arguments specify the socket’s permissions, user ID, and group ID, respectively.

6.2 Adding Persistent Files to Intel® Xeon Phi™ Coprocessor Filesystem

Files added to the Intel® Xeon Phi™ coprocessor filesystem during normal use are not stored persistently by default. There are several lists of files that are used to generate persistent file systems. They are located in C:\Program Files\Intel\MPSS\filesystem.

The \common directory and corresponding common.filelist can be used by the micctrl utility to add files and directories to the filesystem that are common to all coprocessors. Create \common directory and common.filelist to use this feature.

Finally, there is a directory mic0, mic1, etc., for each coprocessor installed in the system. These (and corresponding micN.filelist) are used to add files and folders unique to an individual coprocessor filesystem.

To add a file to the Intel® Xeon Phi™ coprocessor filesystem, the format is:

```
file /path/on/filesystem path/on/host PERM UID GROUP
```

where PERM indicates an octal Linux* file permission for the file. UID denotes the numeric user ID where 0 is root and 400 is micuser. GROUP denotes the numeric group ID where 0 is root and 400 is micuser.

To add a directory to the filesystem, the format is:

```
dir /path/on/filesystem PERM UID GROUP
```

with PERM, UID, and GROUP defined similarly as above.

Once the modifications have been made to the filelist, the necessary files must be placed into the corresponding directories. The following example adds the directory permanent to the root of the filesystem that contains a file foo.txt.

1) Append the following lines to mic0.filelist:

```
dir /permanent 755 400 400
```
2) Now, create the directory 
   \C:\Program Files\Intel\MPSS\filesystem\mic0\permanent and add the file "foo.txt" to the newly created directory.

3) Execute the following sequence to restart the coprocessor:
   
   Admin> micctrl -stop
   Admin> micctrl --start

   After restarting the coprocessor, the directory and file will be in the coprocessor filesystem persistently.

6.3 Creating the Download Image File

The download image file is created by processing the filelist for common and then the filelist for a coprocessor, in that order.

When the filelists are completely processed, micctrl -g will create a cpio entry for the file and append it to the file micN.image, where N is the numeral indicator for that device.

6.4 Adding Files to the Root File System

Adding a file to the root file system can be done by adding an entry to some existing filelist, indicating the location of the file.

6.4.1 Adding Files by Copying

When adding a file to an existing filelist, the first decision is whether the file should be accessible by all the cards or only a particular one. If it is required for all cards to have access, then copy the file to a location under the directory common\ and amend its filelist. Otherwise, copy the file to the directory mic0\ for coprocessor 0, mic1\ for coprocessor 1, etc. Then update the corresponding filelist.

If a directory had to be created for the added file, do not forget to insert the appropriate dir entry prior to the new file entry.
The Micinfo command displays information about the Intel® Xeon Phi™ coprocessor cards installed on the system along with relevant details about the host system, micro-OS and the drivers. The default installation location for the micinfo.exe tool is the C:\Program Files\Intel\MPSS\bin directory.

### 7.1 Simple Method

The following is the simplest way to execute Micinfo:

User> micinfo

### 7.2 For Advanced Users

**Command syntax:**

User> micinfo [OPTIONS]

**OPTIONS:**

- **-help:** Display command help.
- **-listDevices:** List all Intel® Xeon Phi™ coprocessor devices detected.
- **-deviceInfo <deviceNum> [ -group <groupName> ]** : Displays information about the user-specified Intel® Xeon Phi™ coprocessor card (determined by <deviceNum>). User may additionally specify the type of information with -group <groupName> option.

Valid values for <groupName> are:

- **Versions:** Show Flash and uOS versions.
- **Board:** Show Intel® Xeon Phi™ coprocessor card related information.
- **Core:** Show number of cores, voltage and frequency.
- **Thermal:** Show fan and thermal related data.
- **GDDR:** Show device memory related information.
- **-version:** Display the tool version.
8 Intel® Xeon Phi™ Coprocessor Platform Status Panel: Micsmc

_Micsmc_ is the Intel® Xeon Phi™ coprocessor Platform Status Panel. The _micsmc_ tool can function in two modes: graphical user interface (GUI) mode and command-line interface (CLI) mode. GUI mode provides real-time monitoring of all detected Intel® Xeon Phi™ coprocessors installed in the system. The CLI mode produces a snap-shot view of the status, which allows CLI mode to be used in cluster scripting applications. The _micsmc_ tool monitors core utilization, temperature, memory usage, power usage statistics, and error logs, among other features.

The default installation location for _MicSMC_ is `C:\Program Files\Intel\MPSS\bin`. If the default installation location for the MPSS package is changed at installation, _MicSMC_ can be found in `<location referenced by environment variable INTEL_MPSS_HOME>\bin`.

The _micsmc_ tool is based on the work of the Qwt project (http://qwt.sf.net).

The Status Panel User Guide is available in all supported languages, in PDF and HTML formats, at:

<location referenced by INTEL_MPSS_HOME environment variable>\docs\micmgmt
9 Intel® Xeon Phi™ Coprocessor Verification Tool: Miccheck

The **miccheck** utility is used to verify the configuration and current status of the Intel® Xeon Phi™ coprocessor software stack. It performs sanity checks on a host system with Intel® Xeon Phi™ coprocessor(s) installed, by running a suite of diagnostic tests. The default behavior is to run all enabled tests on the host system first, and then on each Intel® Xeon Phi™ coprocessor in turn.

For detailed information about **miccheck**, refer to the help option of the program:

```
User> miccheck --help
```
10 **Intel® Xeon Phi™ Coprocessor RAS Tool: Micras**

*Micras* is the application running on the Host system to collect and log RAS events generated by Intel® Xeon Phi™ coprocessor cards. This tool is also responsible for handling test and repair by kicking the card into Maintenance mode upon the detection of an uncorrectable or fatal RAS event. It runs as a Windows* service. The default installation location is \C:\Program Files\Intel\MPSS\service.

Micras logs messages into file micras.log located under \C:\Program Files\Intel\MPSS\service. The log messages include but are not limited to:

- MCA events including both correctable and uncorrectable events.
- Card reset, Maintenance mode test or repair messages.
- RAS daemon software operation messages.

An example of the RAS log entry appears below:

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Severity Level</th>
<th>Message Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tue Mar 5 16:24:29 2013</td>
<td>MICRAS ERROR</td>
<td>Card 2: failed getting card mode</td>
</tr>
</tbody>
</table>

10.1 **Simple Method**

Micras is installed as a Windows* service.

The following is the simplest way to start micras service:

Admin> micctrl -start
Admin> net start micras

To stop micras service, execute the following command:

Admin> net stop micras

10.1.1 **Configure Micras Service to Start when Windows* Boots (optional)**

To configure the micras service (RAS) to start when the Windows* OS boots, do the following:

1) In a command window, type services.msc, and press Enter.
2) In Services, right-click Intel(R) Xeon Phi(TM) coprocessor Reliable Availability Service, and click Properties.
3) On the General tab, in Startup type, select Automatic, and click OK.
10.2 For Advanced Users

The *micras* tool can be used in these various ways. See the detailed usage below:

Use `net start micras` to start micras service and `net stop micras` to stop micras service from a Windows command prompt.

**Command syntax:**

```
User> micras [OPTIONS]
```

**OPTIONS:**

- `help` : Display command help information.
- `maint` : Enable maintenance mode for error test and repair.
- `loglevel [loggingLevel]` : set the level of detail that gets logged with the *micras* tool. The accepted levels are from 1 to 15.

It is a 4-bit representation, where bits 0 - 3 denote the following:

- **Bit 0** – Enables Informational Messages
- **Bit 1** – Enables Warning Messages
- **Bit 2** – Enables Error Messages
- **Bit 3** – Enables Critical Messages

**default:** all messages on.

The severity level of *micras* log messages is mostly aligned with the standard RFC 5424 syslog severity level. Currently, there are four severity levels available (see Table 2, Severity Levels).

If *micras* is executed with no arguments, it runs at the console prompt, connects to devices, and waits for errors.

Use **Ctrl-C** to exit micras and return to the console prompt.

Use `~maint` option to enable maintenance mode for error test and repair.

Use `~help` to show the help info.

The severity level of micras log messages is mostly aligned with the standard RFC 5424 syslog severity level. Currently, there are four severity levels available (see Table 2, Severity Levels).
### Table 2  Severity Levels

<table>
<thead>
<tr>
<th>Severity</th>
<th>Description</th>
<th>General Description</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFO</td>
<td>Informational messages</td>
<td>Normal operational messages, for information purposes (such as reporting).</td>
<td>No action required.</td>
</tr>
<tr>
<td>WARNING</td>
<td>Warning conditions</td>
<td>Warning messages. Not an error, but an indication that an error might occur if action is not taken.</td>
<td>No immediate action required.</td>
</tr>
<tr>
<td>ERROR</td>
<td>Error conditions</td>
<td>Non-urgent failures. The failure might be recovered by software itself.</td>
<td>Inform your system administrator.</td>
</tr>
<tr>
<td>CRITICAL</td>
<td>Critical conditions</td>
<td>Critical conditions that should be corrected immediately. Micras software has some test and repair capability built-in. But some critical condition will require admin’s involvement like remove/replace components.</td>
<td>Immediate action required for some error conditions</td>
</tr>
</tbody>
</table>
11 Intel® Xeon Phi™ Coprocessor Utility: Micnativeloadex

The micnativeloadex utility will copy an Intel® Xeon Phi™ coprocessor native binary to a specified Intel® Xeon Phi™ coprocessor and execute it. The utility automatically checks library dependencies for the application and if they can be found in the default search path (set using the SINK_LD_LIBRARY_PATH environment variable), the libraries will also be copied to the device prior to execution. This simplifies running Intel® Xeon Phi™ coprocessor native applications since the utility automatically copies the required dependencies.

In addition, the utility can also redirect output from the application running remotely on the Intel® Xeon Phi™ coprocessor back to the local console. This feature is enabled by default, but can be disabled with a command line option. For further details on command line options, see the help section below.

If the application has any library dependencies, then the SINK_LD_LIBRARY_PATH environment variable must be set to include those directories. This environment variable works just like LD_LIBRARY_PATH for normal applications. To help determine the required libraries, execute micnativeloadex with the -l command line option. This will display the list of dependencies and indicate which ones have been found. Any dependencies not found will likely need to be included in the SINK_LD_LIBRARY_PATH.

**Note:** For more information about micnativeloadex, refer to:

```
User> micnativeloadex -h
```
12 Host File System Share

Sections 12.1 and 12.2 demonstrate how to set up an NFS share on Windows* Server.

- For Windows* Server 2008 R2 SP1, continue to Section 12.1.
- For Windows* Server 2012 and Windows* Server 2012 R2, skip to Section 12.2.

**NOTE:** NFS server is supported on Windows* Server 2008 R2 SP1, Windows* Server 2012, and Windows* Server 2012 R2 only. It is NOT supported on Windows* 7 or Windows* 8/8.x.

12.1 NFS Share on Windows* Server 2008 R2 SP1

12.1.1 Add the NFS Server

1) Start **Server Manager** and navigate to the **Roles** section. Right-click **Roles** and select **Add Roles**.

![server_manager_roles.png](image)

2) In the **Add Roles Wizard**, select the **File Services** Role:

![add_roles_wizard.png](image)
3) Continuing in the wizard, under File Service -> Role Services, select Services for Network File System.

**NOTE:** Clear the File Server check box, if necessary.

4) Click Next and then click Install. In the Results pane, confirm that the installation was successful.

![Image showing file services configuration]

### 12.1.2 Provision a Folder for the NFS Share

1) In Server Manager, select (highlight) the Share and Storage Management item that was just added under Roles -> File Services. Click Action -> Edit NFS Configuration...
2) Select **Use Services for NFS to Share Folders** and click **Provision a Shared Folder Wizard**.

3) On the **Shared Folder Location** page of the wizard, enter a **Location** value for the NFS share.

4) For this example, accept the default NTFS permissions.
5) Select the **NFS** check box and fill in the **Share name** field.

6) Under **Enable unmapped user access**, select **Allow unmapped user Unix access (by UID/GID)**.

### 12.1.3 Specify the NFS Share Permissions

1) In **Groups and host permissions**, change the Access field for **ALL MACHINES** group to **No Access**. This prevents unspecified hosts from accessing the NFS share. Click the **Add** button.
2) Now, specify a host and set its permissions to **Read-Write**. Select **Allow root access**. Click **OK** to return to the previous dialog box.

3) Verify the Host has **Read-Write** permissions and **Root Access** is **Allowed**. Click **Next**, then click **Create**.
4) Close the Provision a Shared Folder Wizard, Services for NFS Configuration Guide dialog box, and Server Manager.

5) Proceed to Section 12.3, "Mount the NFS Share".

12.2 NFS Share on Windows* Server 2012 and Windows* Server 2012 R2

Sections 12.2.1 through 12.2.3 use the Server Manager user interface to set up an NFS share on Windows* Server 2012 and Windows* Server 2012 R2. For an alternative method using PowerShell* cmdlets, skip to Section 12.2.4, Use PowerShell* Cmdlets to Create NFS Share (optional).

12.2.1 Add the NFS Server

1) Start Server Manager. On the dashboard, click Add Roles and Features on the Manage menu.

2) In the Add Roles and Features Wizard, click Installation type in the left column. Select Role-based or feature-based installation.
3) Click **Server Selection** and **Select a server from the server pool**. Select the server.

4) Click **Server Roles**. Under **Roles**, expand **File and Storage Services**, then expand **File and iSCSI Services**. Select **Server for NFS**.
5) A confirmation dialog box will appear. Click the **Add Features** button.

6) Returning to the **Add Roles and Features Wizard**, select **Confirmation** in the left column. Click the **Install** button.
7) In the **Results** pane, confirm that the installation was successful.

### 12.2.2 Provision a folder for the NFS Share

1) In Server Manager, click **File and Storage Services** in the left column of the Dashboard.
2) Select the desired server from the list. Click **Shares** in the left column.

3) Click **To create a file share, start the New Share Wizard.**
4) In the **New Share Wizard**, click **Select Profile** and select **NFS Share - Quick**.

5) Click **Share Location** and select the desired server from the list. Specify a path for the share.
6) Click **Share Name** and specify a name for the share (click OK if prompted to create the share location directory).

   ![New Share Wizard](image1.png)

7) Click **Authentication**. Select **No server authentication**, **Enable unmapped user access**, select **Allow unmapped user access by UID/GID**.

   ![New Share Wizard](image2.png)

### 12.2.3 Specify the NFS Share Permissions

1) Click **Share Permissions** and click **Add**. In the **Add Permissions** dialog box, change the **Share Permissions** field for the **All Machines** group to **No Access**. This prevents unspecified hosts from accessing the NFS share. Click **Add**.
2) Return to the **New Share Wizard** and click **Add**. Now, select **Host** in the **Add Permissions** dialog box. Specify a host and set its **Share Permissions** field to **Read/Write**. Select **Allow root access**. Click **Add**.

3) Return to the **New Share Wizard**. Review the **Share Permissions** settings. Verify the Host has **Read/Write** permissions and **Root Access** is **Allowed**.
4) In the left column, click **Permissions** then click **Confirmation**. Confirm your settings and click **Create**.

5) Confirm your results and close **New Share Wizard** and **Server Manager**.

6) Proceed to **Section 12.3, Mount the NFS Share**.
12.2.4 Use PowerShell* Cmdlets to Create NFS Share (optional)

Alternatively, the tasks in Sections 12.2.1 through 12.2.3 can be accomplished using the built-in PowerShell* NFS cmdlets in Windows* Server 2012 and Windows* Server 2012 R2, as shown below:

**Note:** The backslash \\ character, placed at the end of a line, is used to indicate continuation of a command on the next line. It is not part of the command itself.

1) Add the NFS server:

```
User> Add-WindowsFeature FS-NFS-Service
```

2) Provision the directory to be shared:

```
User> New-Item C:\MY_NFS_TEST -type directory
User> New-NfsShare -Name nfs-tests \\
-Path C:\MY_NFS_TEST -EnableUnmappedAccess $True \\
-Permission no-access -Authentication sys
```

3) Grant read/write and root access permissions for Host 192.168.1.100:

```
User> Grant-NfsSharePermission - Name nfs-tests \\
-Permission readwrite -ClientName 192.168.1.100 \\
-ClientType Host -AllowRootAccess $True
```

4) Confirm share permission settings:

```
User> Get-NfsSharePermission -Name nfs-tests
```

5) Proceed to Section 12.3, Mount the NFS Share.

12.3 Mount the NFS Share

**NOTE:** This section applies to Windows* Server 2008 R2 SP1, Windows* Server 2012, and Windows* Server 2012 R2.

**NOTE:** Refer to Section 14.1 for information on downloading and installing PuTTY tools.

1) Change to the directory where PuTTY tools are installed:

```
User> cd C:\Program Files\Intel\MPSS\bin
```

2) Login to the Intel® Xeon Phi™ coprocessor card as root:

```
User> putty.exe -ssh -i \n
<PATH_TO_PRIVATE_KEY> root@192.168.1.100
```
where `<PATH_TO_PRIVATE_KEY>` corresponds to the location and name of the private key file associated with this coprocessor.

**NOTE:** If prompted for a password, see Section 14.3 on setting up user SSH keys for the root user.

3) On the Intel® Xeon Phi™ coprocessor card, add a directory with the same name as the NFS share:

```bash
[root@knightscorner ~]# mkdir /tmp/nfs-tests
```

where N is the integer ID of that card (for example: mic0, mic1, etc.).

4) Mount the NFS share from the Intel® Xeon Phi™ coprocessor card. The nolock option is **required** on this step:

```bash
[root@knightscorner ~]# mount -t nfs -o nolock \n192.168.1.99:/nfs-tests /tmp/nfs-tests
```

5) Navigate to the directory that you created on the Intel® Xeon Phi™ coprocessor card. Verify that the contents of the NFS share are visible.

```bash
[root@knightscorner ~]# cd /tmp/nfs-tests
```
13 CIFS Share on Windows

13.1 Mounting CIFS share with micctrl

Command Syntax:

```
micctrl --addcifs <server> <sharename> <card mountpoint>
```

Note: You must restart the cards to access the mounted share.

1. Enable sharing on host machine

To mount CIFS share, you must give share permissions to Everyone and enable Turn off password protected sharing.

- Navigate to Control Panel > Network and Sharing Center > Change advanced sharing settings > Public profile > Enable Turn off password protected sharing and save changes.
- Navigate to the properties of the directory you want to share Sharing > Share. Add Everyone to the share group, assign appropriate permissions and select Share. (See: Figure 4 File Sharing screen)
Figure 4 File Sharing screen

Figure 5 Advanced Sharing screen

- Navigate to **Advance Sharing** from **Properties** window and select **Share this folder**. (See: Figure 5 Advanced Sharing screen)
• Navigate to Permissions and assign appropriate permissions to Everyone as above. (See: Figure 5 Advanced Sharing screen)

Figure 6 Permissions for Winshare window

Note: You have to enable share permissions in both Share and Advanced Sharing. The current mount utils on coprocessor card do not support nested host share paths.

1. Run micctrl --addcifs

You can use this option to enable mount on required cards by specifying [mic card list]. If not specified, micctrl --addcifs enables mount on all cards in the system. Running this command should add a <CIFS> entry in micN.xml file.

Example:

micctrl --addcifs winshare /mnt/wincifsshare mic0 mic2
micctrl --addcifs winshare /mnt/wincifsshare

3. Restart the cards
You can restart the cards using `micctrl --stop` and `--start` or `micctrl --shutdown/-r` and `-b`. This step updates the `micN.filelist`, corresponding `fstab` and mounts the share on boot. You can access the mounted share through `Putty`, which would need prior setup of ssh using the `micctrl --addssh` option.

You can manually edit the configuration file `micN.xml` if you want to modify a mounted share, it is the same as adding additional mount options. `<card mount point>` is used as unique identifier when there are multiple CIFS shares for example.; there should be one `<CIFS>` entry per mount point in configuration file. You may want to delete the old settings from `fstab` to avoid conflict. You have to restart the card to access the new mounted share.

You have the option to manually mount and unmount `password protected shares` as the root user on the card.

When accessing the share files on the card, you could encounter `cannot allocate memory` error. This error can be avoided by maximizing the throughout for file sharing and/or network applications in the registry.

Set the following value to 2 or 3 (for large file transfers)

```
HKLM\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters\size
```

You can also optionally set the following value to 1

```
HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\LargeSystemCache
```

Reboot your host or restart the `Server` service in `services.msc` after making the changes.

References:

- [http://support.microsoft.com/kb/232271](http://support.microsoft.com/kb/232271)

### 13.2 Configuring a CIFS share with a bridge

**Note:** The `IP addr` should be specified instead of `host`, when bridging is setup.

1. Configure a bridge,
2. Change the IP address in the xml files
3. Add a CIFS share with micctrl
4. See the xml file.
13.3 Umounting CIFS share with micctrl

Command Syntax:

```
micctrl --removecifs <card mount point> [mic card list]
```

`micctrl --removecifs` option prepares the setup required for umount. It doesn’t do the actual umount but instead cleans up the configuration, `filelist` and `fstab` settings updated in `--addcifs` step. You have to restart the cards for removing the mounted share.

1. Run `micctrl --removecifs`
   You can use this option to clean up the CIFS settings on required cards by specifying `[mic card list]`. If not specified, `micctrl --removecifs` attempts to clean up the settings on all available cards in the system.

Example:

```
micctrl --removecifs /mnt/wincifsshare mic0 mic2
micctrl --removecifs /mnt/wincifsshare
```

2. Restart the cards
   You can restart the cards using `micctrl --stop` and `--start` or `micctrl --shutdown/-r` and `-b`. The mounted share will be removed on boot.
14 Interacting with the Intel® Xeon Phi™ Coprocessor

It is possible to interact with the Intel® Xeon Phi™ coprocessor using standard SSH tools. For the purposes of this section it is assumed that the PuTTY tools will be used. Refer to Section 14.1, External Tools for information on downloading PuTTY tools.

14.1 External Tools

Some Intel tools require that PuTTY and PuTTYgen must be installed. Download PuTTY and PuTTYgen from the link: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html. Install PuTTY and PuTTYgen to the same location as the external tools (C:\Program Files\Intel\MPSS\bin).

14.2 Special User Account ‘micuser’

The default users for the Intel® Xeon Phi™ coprocessor are root and micuser. micuser is a special user account, reserved exclusively for COI offload applications. Users are not permitted to login as micuser.

The special user account micuser should not be confused with the user group MICUSERS. Users must be listed in the MICUSERS group in order to log into the coprocessor. For more information on adding and deleting users from the MICUSERS group, see Sections 14.4 and 14.4.1.

14.3 Generating Public and Private SSH Keys

Follow the chosen SSH key tool documentation to generate public and private SSH keys. For PuTTY the program is PuTTY Key Generator (puttygen.exe).

1) Click Generate. After following the on-screen instructions the following screen is displayed.
2) Click **Save public key** to save the public key to disk in SSH-2 format. **For example:** save the key with the name `publicKeys.pub`. Click **Save private key** to save the private key in a secure location not readable by other users. This key will be used later to connect to the Intel® Xeon Phi™ coprocessor. **For example:** save the key with the name `privateKeys.ppk` onto your Desktop. Optionally, save it with a passphrase; otherwise, click **Yes** to ignore the warning. Close the PuTTYgen tool.

3) Open a command prompt as an Administrator. Run the command:

```
Admin> micctrl --addssh <username> -f <path-to-key-file>
```

where `<username>` is the actual username to link with this SSH key and `<path-to-key-file>` is the full path to the public key file (saved during key generation).

**Note:** Alternative Procedure for Steps 2 & 3

Copy the entire text in **Public key for pasting into OpenSSH authorized_keys file** into the clipboard. Save the private key as described in Step 2 above. Close the PuTTYgen tool.

Open a command prompt as an Administrator. Run the command:

```
Admin> micctrl --addssh <username> <public-key>
```

where `<username>` is the actual username to link with this SSH key and `<public-key>` is the key that was copied to the clipboard from the PuTTYgen tool. Paste `<public-key>` directly into the command.
4) Restart the Intel® Xeon Phi™ coprocessor using the \texttt{micctrl} utility. A host reboot is not sufficient to generate a new filesystem. Use standard SSH tools to interact with the coprocessor once its state changes to \textit{online}. Note the user which now has SSH privileges – that is the username that you will use to interact with the coprocessor. The default accounts on the Intel® Xeon Phi™ coprocessor are \textit{root} and \textit{micuser}. The special user account \textit{micuser} is used exclusively for COI offload applications. Users are not permitted to login as \textit{micuser}.

5) Once MPSS has started, start PuTTY.exe. In the \textbf{Category} box, expand \textbf{Connection} $\rightarrow$ \textbf{SSH} and select \textbf{Auth}. In the \textbf{Private key file for authentication} field, browse to the Desktop and select \textit{privateKeys.ppk} (saved during key generation).

6) In the \textbf{Category} box, select \textbf{Session} at the top and click \textbf{Save} (you can choose default or create a new session name). This allows easy reconnection to the coprocessor without specifying private keys each time.

7) In the \textbf{Host Name (or IP address)} field, enter \texttt{192.168.1.100} and then click \textbf{Open}.

8) When the putty session opens to the coprocessor, enter the username specified in correctly set up it will not ask for a password. You should now have a terminal session set up to the file system on the coprocessor.

After completing these steps, a program such as WinSCP can be used to transfer files to and from the coprocessor.

\textbf{14.3.1 Transferring Files to and from the Intel® Xeon Phi™ Coprocessor}

This section describes using the WinSCP tool (version 5.5 shown) to transfer files to and from the Intel® Xeon Phi™ coprocessor. WinSCP is open source and can be downloaded from \url{http://winscp.net/}.

1) Launch WinSCP and select \textbf{File Protocol}: \textit{SCP}. Enter \texttt{192.168.1.100} in the \textbf{Host Name} field and \texttt{root} in the \textbf{User Name} field. Do not enter a password. Click \textbf{Advanced...} to display the \textbf{Advanced Site Settings} dialog.
2) In the **Advanced Site Settings** dialog, select **SSH -> Authentication**. In the **Private key file** field, enter the full path to the private key file generated above (or click ..., to browse to the file). Click **OK** to return to the previous dialog.

3) Optionally, click **Save As...** to save the new site to the tree in the left pane. Click **Login** to use the tool.
14.4 User Groups

The default users for the Intel® Xeon Phi™ coprocessor are root and micuser. The special user account micuser is used exclusively for COI offload applications. Users are not permitted to login as micuser. The administrator can add additional users. Any new user for the coprocessor must have a corresponding local or domain user account.

1) Click the Windows* Start button. Enter Run in the Search Programs and Files search box and press Enter. The Run window appears.

2) Enter lusrmgr.msc in the Open drop down box of the Run window and press Enter.

3) Right click Users from the left pane and select New User from the pop up menu to add a new user.

4) Right click Groups from the left pane, and select New Group. Type MICUSERS in the box titled Group name. Click Add under the Members box, and type in the name of the User you created in Step 3. After adding the user, select Create to create the MICUSERS group.

**NOTE:** Only users listed in the MICUSERS group are permitted to log into the coprocessor. After adding or deleting users in the MICUSERS group, the coprocessor must be stopped and started again.
14.4.1 Deleting Users from MICUSERS Group

The administrator can remove users from the MICUSERS group, as follows:

1) Click the Windows® Start button. Enter Run in the Search Programs and Files textbox and press Enter. The Run window will appear.

2) Enter lusrmgr.msc in the Open drop down box and press Enter.

3) Double click the Groups selection in the center panel.

4) Right click the MICUSERS group in the middle pane and select Properties. In the Member pane select the user name to delete from this group. Click Remove. Click OK.

5) Once all users have been removed from the group, click OK. The newly-removed users will no longer be available in the group once the Intel® Xeon Phi™ coprocessor has been restarted.

Note: Only users listed in the MICUSERS group are permitted to log into the coprocessor. After adding or deleting users in the MICUSERS group, the coprocessor must be stopped and started again.

14.5 Running the COI Tutorials (optional)

After installation is complete, it is possible to run the COI tutorials.

See C:\Program Files\Intel\MPSS\sdk\tutorials\coi\README_Windows_1.txt for instructions.

Intel® Coprocessor Offload Infrastructure (Intel® COI) provides a set of APIs to simplify development of tools and other applications using offload and reverse accelerator models.

14.6 Running the MYO Tutorials (optional)

After installation is complete, it is possible to run the MYO tutorials.

See C:\Program Files\Intel\MPSS\sdk\tutorials\myo\README.txt for instructions.

MYO is a library and API to support virtual shared memory between processes on a host process and Intel® Xeon Phi™ coprocessor cards. MYO is supplementary to other Intel® Xeon Phi™ hardware and software, and is intended for researchers and advanced users.

14.7 Running the SCIF Tutorials (optional)

After installation is complete, it is possible to run the SCIF tutorials.
SCIF provides a mechanism for communication between the components of a distributed application. It is intended for tools developers and application developers.

14.8 Running the MicMgmt Tutorials (optional)

After installation is complete, it is possible to run the MicMgmt tutorials.

The MicMgmt API is an aid to developing custom cluster functionality to access and control hardware registers and parameters of Intel® Xeon Phi™ coprocessors.

14.9 Troubleshooting

This section describes several methods of troubleshooting and recovery during the following events:

1) The coprocessor times out during the boot process

2) The coprocessor will not start

The micctrl utility logs information related to events in the Windows* Event Viewer. To open the event viewer, click the Start button and type Event Viewer. In Event Viewer, in the left column, expand Applications and Services and double-click MPSSLog.

This will display the following window:

This event log should be checked in case of any observable failure starting or stopping coprocessors. Failures include timeouts and errors running micctrl --start,
micctrl --stop, micctrl -r, and micctrl -b.

The event log for the coprocessors can be saved to a file from the command line with the following command (run as administrator):

```
Admin> wevtutil qe /f:text mpsslog > output.txt
```

Currently, it is recommended to use DebugView to obtain troubleshooting information related to the kernel mode drivers. DebugView can be obtained from the following website:


Once the zip file has been extracted, run Dbgview.exe as Administrator. From the Capture menu, make sure to select Capture Global Win32, Capture Kernel, and Enable Verbose Kernel Output. The result should look like the following image:

![DebugView](image)

Certain host kernel mode driver events will print a message to this window. To capture the output, click File and Save As, and enter the desired filename. DebugView is useful in cases where the micctrl utility gives commands to boot the coprocessor but the host driver fails the boot request.

Some errors can be caused by incorrect filelist specifications. See Section 6, Adding Software, for more details on the filelist syntax. If the filelist is incorrect, the micctrl utility cannot boot the coprocessors. To verify that the filelist syntax is correct before attempting to boot the coprocessors, run micctrl -g to see errors listed on the command line. Once the micctrl utility has attempted to boot the coprocessors, the same error messages will be listed in the Event Viewer log.
15 Installing Intel® Xeon Phi™ Coprocessor Performance Workloads (optional)

15.1 Requirements

1) Intel® Composer XE Requirements

There are two options to installing the Intel® Composer XE requirements. The first option is to install the full Intel® Composer XE MSI package.

If the full composer installation is not available, then two packages can be used instead. The required shared object libraries can be installed via the Intel® Composer XE redistributable package, freely distributed on the web at:


This download has an MSI package which will set up the environment and install the required dependencies.

Besides the shared object libraries, the MKL Linpack benchmark is also a requirement. This is also freely distributed on the web at:


This download is a zip file that can be unpacked anywhere, but the environment variable MKLROOT must point to the top level directory of the untarred package. For instance, if the user extracted the zip file into their home directory they should set MKLROOT as follows:

User> set MKLROOT=%HOMEPATH%\linpack_11.1.0

If MKLROOT is set in the user’s shell environment at run time then micprun will be able to locate the linpack binaries. Note that the version of linpack linked above may be newer than 11.1.0, and MKLROOT variable should reflect this.

2) MATPLOTLIB Requirements

The micpplot and micprun applications use the MATPLOTLIB Python module to plot performance statistics. The micprun application only creates plots when verbosity is set to two or higher, and only requires MATPLOTLIB for this use case. MATPLOTLIB must be installed in order to create plots. Download it from:

matplotlib.sourceforge.net
3) PuTTY Requirements

The PuTTY command line utilities pscp.exe and plink.exe must be installed to a location referenced by the user's PATH environment variable. Refer to Section 14.1, External Tools for information on downloading PuTTY tools.

**Note:** The user ID and SSH key required to log into the card can be set with the environment variables INTEL_MPSS_USER and INTEL_MPSS_SSH_KEY respectively.

**Note:** INTEL_MPSS_SSH_KEY should be the path to the PuTTY SSH key file.

4) User Access Requirements

Since micperf transfers files to the Intel® Xeon Phi™ coprocessor, additional steps must be performed for new users to gain access to the card:

a) By default, micperf uses the host user name to access the card. This user must be added to the MICUSERS group (refer to Section 14.4 for steps to create the MICUSERS group and add users to the filesystem).

b) After the user has been added to the MICUSERS group, refer to Section 14.3.1 to grant the user access to the card.

15.2 Distributed Files

The micperf software is part of the Intel MPSS EXE package and will be installed along with the rest of MPSS by default. The Intel MPSS EXE installs the micperf files to the directories:

- Benchmark source code: \Program Files\Intel\MPSS\sdk\tutorials\micperf
- Python source distribution: \Program Files\Intel\MPSS\sdk\micperf\micp
- Benchmark binaries: \Program Files\Intel\MPSS\sdk\micperf\libexec
- Reference data: \Program Files\Intel\MPSS\sdk\micperf\data

15.3 Micp Python Installation

**Note:** This section requires that Python be installed first. The script below requires Python version 2.7.6. Download Python version 2.7.6 from: [http://www.python.org/download/releases/2.7.6/](http://www.python.org/download/releases/2.7.6/)

Once the Intel MPSS EXE package has been installed, an additional step is required to access the micp Python package: either install it to your global Python site packages, or set up your environment to use the micp package from the installed location.

To install into the Python site packages, open a Windows* command prompt as administrator and run the following two commands:

Admin> cd \Program Files\Intel\MPSS\sdk\micperf\micp
Admin> setup.py install
This method provides access to the micp package and executable scripts by all non-admin users who use the same Python version as the administrator. If Python is in the default location and uses a standard configuration, setup.py installs the micp package to the directories:

- C:\PythonXY\Scripts
- C:\PythonXY\Lib\site-packages\micp

An intermediate product of running setup.py install is the creation of the directory:

- C:\Program Files\Intel\MPSS\sdk\micperf\micp\build

None of the products of running setup.py discussed above will be removed by uninstalling the Intel MPSS EXE. The installation with setup.py uses Python's distutils module, and this module does not support uninstall.
16 Important Considerations

16.1 Disabling and Enabling the Memory Control Group (cgroup)

The memory Control Group is disabled by default in this release, but it can be enabled in the global.xml file. Enabling the memory cgroup decreases the amount of memory available to applications on the coprocessor by about 120MB.

PREREQUISITES:

The following settings can only be modified by the admin user.

- To enable the memory cgroup, remove the `<cgroup_disable>memory</cgroup_disable>` element from the `<CommandLine>` element in `C:\Program Files\Intel\MPSS\global.xml`, and then restart the Intel® Xeon Phi™ coprocessors.

  1) In `C:\Program Files\Intel\MPSS\global.xml`:

     ```xml
     <CommandLine>
     <clocksource>tsc</clocksource>
     <highres>off</highres>
     <mce>on</mce>
     <!--Remove the quiet element for more verbose boot logging-->  
     <quiet />
     </CommandLine>
     ```

  2) Restart the Intel® Xeon Phi™ coprocessors:

     ```
     Admin> micctrl -stop
     Admin> micctrl --start
     ```

- To disable the memory cgroup, nest the `<cgroup_disable>memory</cgroup_disable>` element within the `<CommandLine>` element in `C:\Program Files\Intel\MPSS\global.xml`, and restart the Intel® Xeon Phi™ coprocessors:

  1) In `C:\Program Files\Intel\MPSS\global.xml`:

     ```xml
     <CommandLine>
     <clocksource>tsc</clocksource>
     <highres>off</highres>
     <mce>on</mce>
     <!--Remove the quiet element for more verbose boot logging-->  
     <quiet />
     ```
2) Restart the Intel® Xeon Phi™ coprocessors:
   Admin> micctrl -stop
   Admin> micctrl --start

16.2 Enabling Windows MIC GDB Debugging for Offload Processes

For Supporting the use of the Windows MIC GDB debugger through Microsoft* Visual Studio*, environment variables must be set in order to allow the debugger to have the files needed to attach to the card side offload processes. To do this, execute the following steps:

1) Close Visual Studio*, if you have not already done so.
3) In the Advanced tab, click Environment Variables...
4) Click New. Set Variable name to AMPLXE_COI_DEBUG_SUPPORT and Variable value to TRUE.
5) Click New. Set Variable name to MYO_WATCHDOG_MONITOR and Variable value to -1.
6) Reopen Visual Studio*, so that the new environment will be picked up.

Upon executing Steps 1-6, the necessary debug files will be created for the Windows MIC GDB debugger to find and attach to the card side offload processes for debugging issues in offload programs.

16.3 Enabling Windows MIC Debugging for MYO Applications

Background:

The MYO WATCHDOG MONITOR is a feature of the MYO runtime library, which monitors whether the host-side and card-side are still operating and have not suffered a fatal error. If a fatal error occurs on one peer (host or card), the other peer (card or host respectively) will abort with an error message. The MYO WATCHDOG MONITOR performs this task by making sure messages are being sent and received at a minimum rate (default 1 hertz), in order to communicate that the peers are still operating properly.

Caution: When a MYO application is stopped at a breakpoint in a debugger, the MYO WATCHDOG MONITOR assumes that the host or card side of the MYO application has suffered a fatal error and will abort, thus preventing debug of the MYO application.

To debug MYO applications for this release, the MYO WATCHDOG MONITOR must be turned off prior to starting the MYO application on the host. To do so, set the environment variable: MYO_WATCHDOG_MONITOR to the value -1.
16.4 **Copy RPMs to the Card Using PSCP**

In the following steps, `coreutils` and `libgmp` rpm files will be copied to the card and then installed. These steps can be used as a guide to copy and install any rpm files to a card.

1. Go to the [Intel® Developer Zone](https://www.developer.intel.com) website (Intel® DZ).

2. Download the `mpss-<version number>-k1om.tar` file from the **Software for Coprocessor OS** link associated with your MPSS release.

3. Extract the `mpss-<version number>-k1om.tar` file using a tool such as 7-Zip or WinZip.

4. In a Windows* command window, navigate to the extracted folder.
   
   User> cd mpss-<version number>

5. **Copy** the `coreutils` and `libgmp` RPMs to the card with a secure copy tool such as `pscp.exe`.
   
   User> `pscp -scp -i <PATH_TO_PRIVATE_KEY> coreutils*.rpm root@192.168.1.100:/tmp`
   
   User> `pscp -scp -i <PATH_TO_PRIVATE_KEY> libgmp*.rpm root@192.168.1.100:/tmp`

6. **SSH** to the card.
   
   User> `putty.exe -ssh -i <PATH_TO_PRIVATE_KEY> root@192.168.1.100`

7. **Install** the `coreutils` and `libgmp` RPMs.
   
   ```
   [root@<host_machine>-MicN ~]# rpm -ihv /tmp/coreutils*.rpm /tmp/libgmp*.rpm
   ```

   As an alternative mechanism for installing rpm files, once the `coreutils` are installed, `zypper` can be used to install rpm files as follows:

   ```
   [root@<host_machine>-MicN ~]# zypper --no-gpg-checks \ install /tmp/<rpm_package_name>
   ```

8. **Repeat Steps 4-6** for the remaining card(s).

   If the RPMs must persist in the coprocessor filesystem, see **Section 6.2, Adding Persistent Files to Intel® Xeon Phi™ Coprocessor Filesystem**.
17  Related Documentation

This section contains a listing of MYO and COI documentation, as well as links to various Intel® Xeon Phi™ coprocessor collateral documents.

17.1  MYO Documentation

17.1.1  MYO Tutorials & Other Document Location on Windows*

C:\Program Files\Intel\MPSS\sdk\tutorials\myo

17.2  COI Documentation

17.2.1  COI Documentation for Windows*

(MPSS EXE Installation Base)\docs\coi\ -release_notes.txt

(Future release)\coi\-coi_getting_started.pdf

(MPSS EXE Installation Base)\docs\coi\MIC_COI_API_Reference_Manual_0_65.pdf

(INTEL_MPSS_HOST_SDK)\include\intel-coi\-header files contain full API
descriptions

(INTEL_MPSS_HOST_SDK)\tutorials\coi\-README_Windows_1.txt

(for how to get card side binaries and
general building and use of tutorials)

17.3  Intel® Xeon Phi™ Coprocessor Collateral

Intel® Xeon Phi™ coprocessor Product Brief:

Intel® Xeon Phi™ coprocessor Specification Update:

Intel® Xeon Phi™ coprocessor Safety and Compliance Guide:
Intel® Xeon Phi™ coprocessor Datasheet:

Intel® Xeon Phi™ coprocessor Software Users Guide:

Intel® Xeon Phi™ coprocessor System Software Developers Guide:

Intel® Xeon Phi™ coprocessor Developers Quick Start Guide:

Intel® Xeon Phi™ coprocessor System Administration Guide:

Intel® Xeon Phi™ coprocessor Instruction Set Architecture Reference Manual: