BEA WebLogic Operations Control

LiquidVM User Guide

Version 1.0
Document Revised: March 2008

Contents

CHAPTER 1

LiquidVM Overview

LiquidVM is a single-process, single-user VirtualMachine designed to run Java processes efficiently on a virtual environment. Details about LiquidVM are provided in the following topics:

- What is LiquidVM?
- Understanding the LiquidVM File System
- Using the Virtual Local Disk
- Using the LiquidVM SSH Service
- Using LiquidVM in Passive Mode

What is LiquidVM?

LiquidVM is a virtualization enabled version of the JRockit JVM that can run on a hypervisor without a standard OS, allowing Java applications to run directly on the virtualized hardware.

The LiquidVM Tools component in WLOC provides tools to create LVM-based instances that can run in a VMWare ESX environment, as well as the run-time components necessary to manage such LVM-based instances.

LiquidVM provides the following features:

- JRockit JVM as the Java runtime component.

- A scaled-down OS kernel that replaces the OS for LiquidVM. It differs from a normal OS in that it is a single-user, single-process environment that is designed to only run a single JVM. No other processes can be started. It implements the following services that the JRockit JVM needs to run Java:
  - Low-level memory management
  - Thread-scheduling
  - File system
  - Networking
  - Interaction with the hypervisor
- Java-based services, started after the JRockit JVM has started, that run in threads that are separate from your application. The LiquidVM services are:
  - LiquidVM SSH-service
  - LiquidVM heap-resizer
  - LiquidVM syslog publisher
- Tools that are used to create and control LiquidVM instances. The LiquidVM tools run on a standard OS; they do not run inside LiquidVM. LiquidVM tools include:
  - LiquidVM launcher—used to start and stop LiquidVM instances
  - LiquidVM Configuration Wizard—used to configure the connection between LiquidVM and the VMware Virtual Center
- A virtual local disk for each virtual machine. The local disk removes the dependence on NFS and provides faster and more secure file transfers.
- The ISO Builder component allows you to create an ISO-9660 file from a template ISO-9660 file, typically a WLS-VE-based or LVM-based ISO file. The tool also allows you to add data files (e.g., application files) along with the content of the source template file in the resulting ISO-9660 file. For more information, see Using the WLOC ISO Builder.

Understanding the LiquidVM File System

The LiquidVM file system is very similar to most Unix-like OSes. There is a single root (`/`) directory; disks and remote NFS file-shares can be mounted in sub-directories. By default, the virtual disk, if configured, is mounted in the root directory (`/`).

The directory structure for WLOC is described in Table 1-1.

Table 1-1 WLOC Directory Structure

Directory	Contents	Notes
`/appliance`	The contents of the LiquidVM ISO image	This directory is read-only. The LVM ISO image file to be used is specified in the `bea.lvm.info` file using the `vmwareDiskPath` parameter. For example: `vmwareDiskPath=[storage1] wloc/wloc10.iso`
`/bea`	• `patch_weblogic`*`nnn`*	The default location for the BEA Home directory on the virtual local disk. You can change the location of the BEA Home directory by specifying the `-Dbea.home` option in the WLOC start script. The BEA Home directory contains any WLS patches.
`/domain`	• server-root directory • log files • application classes	The default location of the WLS domain and the current working directory for WLOC.
`/tmp`		Directory for temporary files created when WLS boots. The temporary files are erased each time the LiquidVM instance reboots.

Using the Virtual Local Disk

LiquidVM provides a virtual local disk for each virtual machine. The local disk can be mapped to a SAN disk attached to the ESX server. You specify the size of the disk by passing the `diskSize` parameter as a startup option to the LiquidVM launcher. To do so, specify the disk size in the WLOC start script using the `LVM_DISKSIZE` option. The default is 1GB (1024). For more information, see Using the LiquidVM Launcher Commands.

If no disk is specified, the local disk is not created.

The first time that you boot LiquidVM, it detects that a virtual hard disk is attached and that it is empty. LiquidVM formats the disk and mounts it in the root directory (`/`).

To the VMware ESXserver, the virtual hard disk is a `vmdk` file. The `vmdk` file is placed in the same directory as the Virtual Machine's configuration file on the ESX server/SAN. You can specify which VMware datastore the VM configuration files and the virtual hard disk should be placed in by specifying the `vmwareVmDatastore` option to the launcher. To do so, add a `vmwareVmDatastore=` entry to the `bea.lvm.info` file.

You can transfer files to and from the local disk by using the LiquidVM SSH service. For more information, see Using the LiquidVM SSH Service.

Determining the Amount of Free Disk Space

To determine the amount of free disk space on your virtual machine while the WLOC instance is running, press **F1** in the VMware VM console. Details about the running system, including the amount of free disk space, are displayed in the console.

Increasing the Size of the Local Disk

If your disk is full, you can shut-down WLOC and specify a larger disk by increasing the value of the `LVM_DISKSIZE` property in the WLOC start script.

VMware does not provide a way to increase the size of the disk. Instead, when you restart WLOC, the LiquidVM launcher creates a new larger disk and copies the files from the old disk to the new disk. When you increase the size of the disk, the initial restart of WLOC will take longer depending on the size of the disk and the amount of files to be copied.

Note: LiquidVM does not provide a mechanism to reduce the size of a disk.

Using the LiquidVM SSH Service

LiquidVM provides a SSH2-compatible service for transporting files to and from LiquidVM. The SSH service does not provide shell services; that is, LiquidVM does not support scripts or editing files from the SSH shell.

The SSH service in LiquidVM provides an encrypted communication channel between the server and the client. The encryption protocol used is AES-128. Unencrypted communication is not supported.

You can transfer files using the `scp` and `sftp` extensions to SSH. The Linux OS includes `sftp` and `scp` clients. On Windows systems, several free SSH2 clients are available for download (e.g., `PuTTy`, `WinSCP`, `FSecure`, and `FileZilla`.)

Note: LiquidVM does not support SSH1 clients.

You authenticate with the SSH service using either password-based or public/private key authentication. For more information, see Authenticating with the SSH Service.

LiquidVM is a single-process, single-user environment, therefore only the `liquidvm` user is supported. Multiple users cannot login into LiquidVM.

SSH Listen Port

The SSH2 server normally listens on the standard SSH port (port 22), but you can change the SSH listen port by setting the Java property `-Dlvm.ssh.port`. You may prefer to change the SSH port to something other than the default (port 22), since most SSH attacks try to attack the default port.

Authenticating with the SSH Service

When you use the LiquidVM launcher to create a new instance, you can specify the type of authentication to be used. To use public/private key authentication, you need to provide your public key as a startup option to the LiquidVM launcher. To do so, specify the location of the public key in the WLOC start script using the `LVM_SSH_PUBLIC_KEY` option. For more information, see Using the LiquidVM Launcher Commands. When you attempt to authenticate with the SSH service, you will be prompted to provide your private key.

In a development environment, you may not want to bother with keys and secure passwords. In that case, LiquidVM provides a simpler, but unsafe method, of specifying an SSH password in clear-text from the launcher. You can specify the password in the WLOC start script using the `LVM_SSH_UNSAFE_PASSWORD` option.

Caution: The password is stored in clear-text. This option should not be used in a production environment. BEA recommends using public/private key authentication in development environments also. Once you have specified a real password or set up an SSH public-key, the unsafe password is no longer valid.

Installing a Real Password In Addition to a Public Key

If you should happen to lose your private key, it is good idea to have a secure *real* password as well, so that you can still log in and access the files on the local disk. You can do password-based authentication that is not clear-text, follows:

1. Using either the public/private-key authentication or the unsafe clear-text password, log into the virtual machine using SSH.
2. Type `passwd`. You will be asked for the existing password. If you are using a public key, leave it blank and press `Enter`.

3. When prompted for new password, enter a secure real password.
4. Confirm the new password.

This installs a real password; therefore, the unsecure clear-text password will no longer work. The public key authentication will continue to work if you prefer to use that method.

Auditing SSH Actions

The SSH service will send audit messages to the syslog for the following actions:

- Successful or failed log-in attempts. These audit messages include information about where the client tried to login from and the authentication method used.
- Logouts

You can use remote logging facilities to send these message to a remote log-collector for compliancy verification.

Using LiquidVM in Passive Mode

To copy files to or from the LiquidVM instance before starting WLS, you can use LiquidVM in passive mode. In passive mode, only the LiquidVM services, including the SSH service, are started. WLS is not started.

To start LiquidVM in passive mode, add the `startMode=passive` option to the launcher start arguments. Once LiquidVM is started in passive mode, you can log in over SSH and transfer your files. When you have finished transferring your files, you can either restart the server or login over SSH and run the start command to resume execution.

If you want the LiquidVM launcher to wait for the SSH service to be started before the launcher exits, you can specify `waitForSSH=true`. This can be useful in a scripting environment where you first start the instance and then you want to copy files from or to the server as soon as SSH is running on the newly started server.

Configuring LiquidVM Connection Parameters

After installing WLOC, you need to run the LiquidVM Configuration Wizard to configure the connection between LiquidVM and the VMware Virtual Center server. You can run the wizard in either a GUI-driven, graphical mode or a command-line-based console mode. Configuring LiquidVM is a critical step because this is where you identify the file system locations of all WLOC components. An improperly configured LiquidVM will not run. This procedure is described in the following topics:

- Before You Begin
- Configuring LiquidVM in Graphical Mode
- Configuring LiquidVM in Console Mode
- Understanding the bea.lvm.info File

Before You Begin

Before you run the LiquidVM Configuration Wizard, be sure that you have access to the configuration data listed in Table 2-1. You will be prompted to provide this information as you step through the configuration wizard.

Table 2-1 LiquidVM Configuration Data

Configuration Data	Description
Virtual Center server	The fully-qualified host name of the server hosting the Virtual Center to which you need to connect.
VirtualCenter login credentials: • VC user name • VC password	The user name and password required to access the Virtual Center server. These should be provided to you by your VMware administrator.
ESX datacenter name	The VMware datacenter in which your WLOC instance is created. The datacenter is the top-level structure in the Virtual Center server.
ESX compute resource	The IP address or fully-qualified name of your ESX host, or cluster of hosts, in which the WLOC instances will run.
ESX resource pool (Optional)	The default VMware resource pool into which LiquidVM should place new VMs. You can override this parameter using the `VI_RESOURCE_POOL` parameter in your WLOC startup scripts. A VMware resource pool is a mechanism provided by VMware that allows you to allocate resources dynamically across a large set of servers. See the VMware documentation for more information on resource pools.
VMware network (Optional)	The VMware virtual network to use. If set to <any>, LiquidVM uses the first available VMware network. Use the drop-down menu to see a list of the available VMware networks.
Location of LiquidVM disk on ESX server	The location of the WLOC ISO image on the ESX server. The location consists of two components: the name of the datastore on the ESX server and the pathname to the ISO image on the disk. The datastore name is always enclosed in square brackets; for example, `[Storage1]`. The storage location and path can be specified as: `[Storage1] myLocalStore/myISO.iso` The ISO image is installed during installation of your BEA software and is copied to the ESX server as described in "Copying the WLOC ISO Image" in the *WLOC Installation Guide*.

Configuring LiquidVM in Graphical Mode

Note: If you are a Linux user and don't have access to a GUI, then use the procedure described in "Configuring LiquidVM in Console Mode" on page 2-5.

To run the LiquidVM Configuration Wizard in graphical mode, use the following procedure.

1. Start the LiquidVM Configuration Wizard in graphical mode as shown in the following table.

To start the LiquidVM Configuration Wizard on this platform . . .	Perform the following steps . . .
Windows	From the command-line: 1. Open a Command Prompt window and navigate to `BEA_HOME\WLOC_HOME\lvm-1.2\bin\` 2. Enter the following command: `lvm_configwizard.cmd`
Linux	1. Set the `DISPLAY` environment variable 2. Open a command shell and navigate to `BEA_HOME/WLOC_HOME/lvm-1.2/bin` 3. Enter the following command: `lvm_configwizard.sh`

Note: In these command-lines, *`BEA_HOME`* represents the BEA Home directory in which you installed WLOC. For more information about the BEA Home directory, see "The BEA Home Directory" in the *WLOC Installation Guide*.

The Configuration Wizard starts and the Virtual Center server window appears.

2. Log into the Virtual Center server by performing the following steps:

   a. In the Virtual Center server field, enter the IP address or fully-qualified name of the server on which VirtualCenter is running.

   b. Select **Secure connection to VC** to use SSL for communication with Virtual Center (recommended). This is selected by default.

   c. Enter the username and password for the Virtual Center server in the VC user name and VC password fields.

Note: The first time that you run the LiquidVM Configuration Wizard, the username and password are saved in the `bea.lvm.info` file. If you want to log in as a different user, select **Change VC login credentials** and enter the username and password for the new user**.**

d. Click **Next.**

A status window is displayed as you are connected to the Virtual Center. Once you are connected, the datacenter information window of the Configuration Wizard is displayed. It may take a few moments for the wizard to obtain the available configuration from the Virtual Center server.

Note: To connect to the Virtual Center and retrieve configuration information using an unsecure connection, the Virtual Center must be configured to support access to the SDK using HTTP. Otherwise, a connection failure message will be displayed when an unsecure connection is attempted:

```
Failed to connect to VMware. It appears that the webservices stack
is not running on the specified port.
```

For more information, refer to the Virtual Center documentation.

3. Specify the Datacenter, Host, and Resources by completing the following steps:

   a. Select the ESX datacenter name that contains your WLOC instance. You can obtain this name from the datacenter administrator.

   b. Select the ESX compute resource from the list of available hosts. The ESX compute resource is the name of the ESX host as displayed from within Virtual Center.

   c. Optionally, select an ESX resource pool in which to place your WLOC instance.

   If you accept the default <root>, the VM is placed in the compute resource directly, and not in any resource pool within that compute resource. For more information about resource pools, see the VMware documentation at `http://www.vmware.com/support/pubs/vi_pubs.html`.

   d. Click **Next**.

   The LiquidVM disk location window is displayed.

4. Specify the VMware network and location of the disk on the ESX server as follows:

   a. From the drop-down menu, select the VMware network for LiquidVM to use. If you accept the default, <any>, LiquidVM uses the first available VMware network.

   b. In the Location of LiquidVM disk on ESX server field, do one of the following:

- Enter the name of the datastore and the path to the LiquidVM iso image on the disk, using the format shown; that is:
  `[`*`storage name`*`]` *`path/filename`*`.iso`

  You must include the square brackets around the name of the datastore.
- Click Browse and navigate to the location of the LiquidVM ISO image on the disk. Select the file and click **Select**.

  The field is populated with the datastore name and path to the LiquidVM image in the proper format.

c. Click **Finish**.

The successful configuration confirmation window appears.

5. Click **Close** to close the Configuration Wizard.

When you have successfully configured LiquidVM, the LiquidVM Configuration Wizard creates a file named `bea.lvm.info` in your home directory on your system (e.g., `c:\Document and Settings\`*`username`*), which contains all of the information you provided while running the wizard. LiquidVM reads this file when it launches to determine the location of critical files. For more information on the `bea.lvm.info` file, see "Understanding the bea.lvm.info File" on page 2-9.

Configuring LiquidVM in Console Mode

Console-mode installation is an interactive, text-based method for configuring your software from the command-line. This mode is useful for Linux users who don't have a GUI display or don't want to otherwise use the graphical configuration mode described in "Configuring LiquidVM in Graphical Mode" on page 2-3. You can also use console mode on a Windows platform.

Be sure you have access to the configuration data provided in Table 2-1, since you will be prompted to supply this information as you step through the wizard.

To complete the console-mode configuration process, respond to the prompts by entering the text representing your choice (filepath, server name, etc.) or by pressing Enter to accept the default. To exit the configuration process, press Ctrl-C in response to any prompt.

To configure LiquidVM using the LiquidVM Configuration Wizard in console mode, follow these steps:

1. Start the LiquidVM Configuration Wizard as shown in the following table.

To start the LiquidVM Configuration Wizard on this platform . . .	Perform the following steps . . .
Windows	1. Open a Command Prompt window and navigate to `BEA_HOME\WLOC_HOME\lvm-1.2\bin\` 2. Enter the following command: `lvm_configwizard.cmd -mode=console`
Linux	1. Open a command shell and navigate to `BEA_HOME/WLOC_HOME/lvm-1.2/bin` 2. Enter the following command: `lvm_configwizard.sh -mode=console`

Note: In these command-lines, *BEA_HOME* represents the BEA Home directory in which you installed WLOC. For more information about the BEA Home directory, see "The BEA Home Directory" in the *WLOC Installation Guide*.

The system responds:

```
LiquidVM Configuration Wizard for VMware ESX (text-mode)
-------------------------------------------------
Collecting information VMware Virtual Infrastructure environment...

Virtual center server
```

2. Enter either the IP address or the fully-qualified name (that is, you must include the domain name) of the Virtual Center server. Press **Enter**.

   The system responds:

```
Use secure connection (https) to virtual center? [Y/n]
```

3. Enter **Y** (or press **Enter**) to use SSL to connect to the Virtual Center server.

   The system responds:

```
Virtual center username
```

4. Enter the appropriate Virtual Center VC user name. This should be provided to you by your VMware administrator. Press **Enter**.

   The system responds:

```
Do you want to provide the password for your virtual center user? if you
do the password will be stored in the configuration file encrypted, if
you don't you will be asked for the password every time you launch a
LiquidVM. [Y/n]
```

5. If you enter **Y**, the system responds:

   ```
   Virtual center password (you will not see what you type)
   ```

6. Enter the VC password you want to use to control access to the Virtual Center server. This should be provided to you by you VMware administrator.

   Note: If you have already set up a password for the Virtual Center server and want to use that one, simply press **Enter**.

   The system responds:

   ```
   Connecting to Virtual Center. May take 30 seconds or more...
   ```

   ```
   Looking up datacenters...
   VMware Datacenter
   [numbered list of available datacenters]
   Please select one of the above numbers
   ```

   Note: To connect to the Virtual Center and retrieve configuration information using an unsecure connection, the Virtual Center must be configured to support access to the SDK using HTTP. Otherwise, a connection failure message will be displayed when an unsecure connection is attempted:

   ```
   Failed to connect to VMware. It appears that the webservices stack
   is not running on the specified port.
   ```

   For more information, refer to the Virtual Center documentation.

7. Enter the number that corresponds to the name of your VMware datacenter (ESX datacenter name). Press **Enter**.

   The system responds:

   ```
   Looking up compute resources (hosts) in datacenter [datacenter name]...
   Default VMware Compute Resource (ESX Host or Cluster)
   [numbered list of available resources]
   Please select one of the above numbers
   ```

8. Enter the number that corresponds to the name of your ESX compute resource. Press **Enter**.

   The system responds:

   ```
   Looking up resource pools in [ESX host name]...
   VMware Resource Pool (or type any for default resource pool)
   ```

```
[numbered list of available resource pools]
Please select one of the above numbers [default: <root>]
```

9. Enter the name of the ESX resource pool, if specified.

   The system responds:

   ```
   Looking up VMware Networks available to [ESX host name]...
   VMware Network (or type any to use any available)
   [numbered list of available virtual networks]
   Please select one of the above numbers [default: <any>]
   ```

10. Enter the number that corresponds to the VMware network to use.

    The system responds:

    ```
    ISO-image datastore
    [numbered list of available datastores]
    Please select one of the above numbers [default: storage1]
    ```

11. Enter the ISO image datastore. Press **Enter**.

    The system responds:

    ```
    Now you should provide the path on storage1 where to find the wloc.iso
    An example of a path is wloc10/wloc10.iso
    ISO-image path:
    ```

12. Enter the path to the ISO image file and press **Enter**.

    The system responds:

    ```
    Checking path...
    ```

    The system responds:

    ```
    Datastore for new VMs
    [numbered list of available datastores]
    Please select one of the above numbers [default: storage1]
    ```

13. Enter the datastore name where the WLOC Vmware configuration files should be stored. Press **Enter**.

    The system responds with this confirmation message:

    ```
    The LiquidVM configuration has now completed successfully.
    Configuration data has been stored in the ‘bea.lvm.info’ file
    ```

When you have successfully configured LiquidVM, the LiquidVM Configuration Wizard creates a file named `bea.lvm.info` in your home directory on your system (e.g., `c:\Document and Settings\`*`username`*), which contains all of the information you provided while running the wizard. LiquidVM reads this file when it launches to determine the location of critical files. For more information on the `bea.lvm.info` file, see "Understanding the bea.lvm.info File" on page 2-9.

Understanding the bea.lvm.info File

After you have run the LiquidVM Configuration Wizard and connected to the Virtual Center server, the Configuration Wizard creates a file named `bea.lvm.info` and stores it in your home directory (for Windows users, that's `\\Documents and Settings\`*`yourHome`*; for example `C:\Documents and Settings\jtsmith`). This file contains all of the configuration information you entered while running the wizard. Listing 2-1 shows an example of a `bea.lvm.info` file.

Listing 2-1 Sample bea.lvm.info File

```
#BareMetal ESX-launcher configuration information
#Thu Jan 03 21:08:15 EST 2008
vmwareUsername=user1
vmwareDiskPath=[storage1] wloc10/wloc10.iso
vmwareVcHost=vmwarevc.bea.com
vmwareKeystore=C\:\\Documents and
Settings\\jtsmith\\bm_vmwarevc.bea.com.keystore
vmwarePassword=d90f1423925849c78e6dd9100d162f3f
vmwareVmDatastore=storage2
vmwareComputeResource=esx.bea.com
vmwareKeystorePassword=07300ca783a0a3e92f8fc6121e2d14aa
LiquidVM.config.version=5
vmwareDatacenter=TESTCENTER
```

The LiquidVM launcher reads the `bea.lvm.info` file at startup to obtain your LiquidVM configuration specifics. The LiquidVM launcher looks for `bea.lvm.info` in the location specified by the `LVM_INFO` environment variable (your home directory by default). This file contains information about VirtualCenter and default information about the ESX Server on which to start new LiquidVM instances. Typically, none of this information in this file is specific to the machine on which you ran the Configuration Wizard so you can copy it between different launching machines. However, since the launcher machine searches for this file in your home directory, if you move it to another location (or rename it), you need to set the `LVM_INFO` environment variable in the start script to point to the new location of the file.

Using the WLOC ISO Builder

This section discusses how to use the WLOC ISO Builder tool bundled with the LVM Tools.

WLOC ISO Builder Tool

The WLOC ISO Builder component allows you to create an ISO-9660 file from a template ISO-9660 file, typically a WLS-VE or LVM ISO file. The tool also allows you to add data files (e.g., application files) along with the content of the source template file in the resulting ISO-9660 file. There are two scenarios for using the ISO Builder tool:

- WLS environments – Use the WLS-VE ISO file as a template and use the ISO Builder to add your application files or any WLS patches.
- Non-WLS environments – Use the LiquidVM ISO file as a template and use the ISO Builder to add other application server and service files. The LVM ISO file (`lvm-1.2.iso`) is installed in the *`BEA_HOME/WLOC_HOME/`*`lvm-1.2` directory as a part of the optional LVM Tools installation.

Note: LVM Tools are not supported on Solaris platforms.

How the WLOC ISO Builder Works

The WLOC ISO Builder tool collects the following input:

- The ISO-9660 template file location (required).
- The file system locations of the additional source files and folders, and their locations in the destination image file structure.

The tool copies the content of the template file to the destination ISO image file, and includes user-selected data files in their specified locations in the destination ISO file structure.

Using the WLOC ISO Builder to Create an ISO Image

Follow these steps to create an ISO image from an existing ISO template.

Note: The ISO Builder works only in GUI mode on Windows and Linux platforms.

1. Invoke the WLOC ISO Builder wizard described in Table 3-1.

Table 3-1 Invoking the WLOC ISO Builder Wizard

Platform	Command
Windows	From the command-line: 1. Open a Command Prompt window and navigate to *`BEA_HOME`*`\`*`WLOC_HOME`*`\lvm-1.2\bin\` 2. Enter the following command: `isobuilder.cmd` Or From the Start Menu, select **WebLogic Operations Control 1.0 > WLOC ISO Builder**
Linux	1. Set the `DISPLAY` environment variable 2. Open a command shell and navigate to *`BEA_HOME`*`/`*`WLOC_HOME`*`/lvm-1.2/bin` 3. Enter the following command: `isobuilder.sh` **Note:** Certain Linux environments may need x server to run the ISO Builder.

Note: In these command-lines, *`BEA_HOME`* represents the BEA Home directory in which you installed WLOC. For more information about the BEA Home directory, see "The BEA Home Directory" in the *WLOC Installation Guide*.

The ISO Builder Wizard starts.

2. On the **ISO Template Selection** window, accept the default ISO file to use as a template or use the **Browse** button to specify a different file, and then click **Next**.

3. On the **ISO Template Additions** window, you can add data files using the buttons described in the following table:
4. Complete the fields as described in Table 3-2.

Table 3-2 ISO Configuration Options

Button	Description
New Folder	Opens a dialog that allows you to create a new folder in the location selected in the tree table. You can modify the name of any folder in the tree (except the root "/" folder) by using the Rename button, as described in this table.
Add Data	Opens a standard file selection dialog, which allows you to select multiple files or folders from the file system. The selected files and folders will be added to the folder selected in the ISO image file tree.
Remove	Removes the selected file or folder from ISO image file after a confirmation.
Rename	Opens a dialog that allows you to rename the selected file or folder from ISO image file.

5. The file tree shows the selected files and their location in the final ISO image file. When you are finished selecting files, click **Next**.
6. On the **ISO Image Selection** window, use the **Browse** button to specify a name and location for new ISO image.

   Before clicking **Create**, verify that the folder(s) in the destination path already exist. (The ISO Builder tool does not create the destination folder if it does not exist.)

   Note: When using the **Browse** button, if do not specify the `.iso` file type in the file name, then the `.iso` file type is automatically applied to the file name, as indicated in the **Select ISO image destination file** field. If you do not use the **Browse** button, then the file name is saved without the `.iso` file type (e.g., entering *myfile* will be saved as *`myfile`*, not as *`myfile.iso`*).
7. The Image Creation window, shows the progress of the ISO image file creation. When the process is completed, click **Done**.

CHAPTER 4

Using the LiquidVM Launcher Commands

This section discusses how to use the LiquidVM Launcher arguments.

- LiquidVM Launcher Overview
- Starting and Creating a LiquidVM Instance
- LiquidVM Launcher Commands
- Stopping a LiquidVM Instance

LiquidVM Launcher Overview

The LiquidVM launcher is used to create and/or start LiquidVM instances. The launcher is a Java program that is run from an ordinary OS. The program connects to VMware Virtual Infrastructure over Web Services (either **`https`** or `http` depending on how you have configured it), and asks VMware to create, start or, stop LiquidVM instances.

Starting and Creating a LiquidVM Instance

Use the following general format to create and start a LiquidVM instance:

```
  java -jar lvm-esxlauncher-1.1.jar ...liquidvm-arguments... -- ...java
arguments...
```

The LiquidVM launcher arguments are typically on the `key=value` form, as follows:

```
  java -jar lvm-esxlauncher-1.1.jar  name=MyVM ip=172.23.82.105 -- ...java
arguments...
```

LiquidVM Launcher Commands

This section describes the available command-line arguments when using the LiquidVM Launcher.

General Commands

Table 4-1 describes the general commands for using the LiquidVM Launcher.

Table 4-1 General LiquidVM Launcher Commands

Key	Default Value	Description
`name`	`LiquidVM-<user>`	The name the virtual machine will have (as displayed by VMware).
`configFile`		A path to a file on the local machine that can include additional LiquidVM arguments on this `key=value` format. The `configfile` key can be used to shorten the command line and to organize configuration options that are common among many different LiquidVM instances in the same location.
`cpus`	1	The number of CPUs this instance should have. (VMware supports 1, 2, or 4 instance VMs.)

Table 4-1 General LiquidVM Launcher Commands

`memory`		How much virtual physical memory the LiquidVM instance should have. This should typically be greater than the maximum Java heap size (`-Xmx`).
`logFile`	`<`*`vm-name`*`>.lvm.out`	The path inside the LiquidVM instance where the log file will be stored.
`startMode=passive`	`active`	Causes the LiquidVM to start all its built-in services (in particular SSH) but not start the main class. This allows you to copy files using SSH to and from the server before the main application is started. The main class is started by logging in using SSH and executing the `start` command.
`copyAtBoot`		Usage: `copyAtBoot='src dst'` Copies the `src` directory recursively to the destination directory at startup before the Java application itself has started. `src` and `dst` could also be files. If you want to copy multiple different directories you can separate them using a semicolon. For example, `copyAtBoot='src1 dst1;src2 dst2'`. When this feature is used, `src` is typically a directory mapped to an NFS-share and used to initialize or update the LiquidVM instance using external files.
`copyAtShutdown`		Usage: see `copyAtBoot`. Copies the `src` directory to the destination directory after the Java application has shutdown, but before LiquidVM itself shuts down. The destination directory is typically an NFS-share so that an external agent can process the end-result of some operation.

Disk Commands

Table 4-2 describes the disk commands for using the LiquidVM Launcher.

Table 4-2 General LiquidVM Launcher Commands

Key	Description
`diskSize`	The size of the local disk (if no disk is specified at VM-creation time no local disk is added)
`cwd`	The working directory the LiquidVM instance will stand in
`mount`	Can be used to mount an NFS-share somewhere in the LiquidVM directory tree
`mountFile`	A path to a file on the launching machine that contains a number of mounts

Networking Commands

Table 4-3 describes the networking commands for using the LiquidVM Launcher.

Table 4-3 Networking LiquidVM Launcher Commands

Key	Default Value	Description
`ip`	If not set, `dhcp` is attempted	The networking IP address the virtual machine will use.
`netMask`	If not set, `dhcp` is attempted	The networking `netmask` the virtual machine will use.
`gateway`	If not set, `dhcp` is attempted	The networking gateway the virtual machine will use.
`dns`	If not set, `dhcp` is attempted	The networking domain name server this LiquidVM instance will use.
`networkDomainName`		The networking domain name this LiquidVM instance will use.

SSH Commands

Table 4-4 describes the SSH commands for using the LiquidVM Launcher.

Table 4-4 SSH LiquidVM Launcher Commands

Key	Default Value	Description
`ssh=on`	`off`	Turns the built-in SSH-server on to enable file transfers to and from the LiquidVM instance.
`sshPublicKey`		Specifies the launcher local path to the SSH public key that will be put in the LiquidVM instance's `authorized_keys` file upon VM creation. Presenting the corresponding private key at login will result in successful authentication. Keep in mind that you still have to log in as the user `liquidvm`.
`waitForSSH=true`	`false`	Makes the launcher wait for the SSH-server to be up and running before the launcher returns. This makes it possible to write scripts that first call the launcher, and, upon return from the launcher, can copy files to the LiquidVM instance, since the return from the launcher guarantees that the `ssh-server` is running. The default is false, and the launcher will most likely return before the `ssh-server` has been fully started.
`sshUnsafePassword`		Specifies a clear-text, unsafe password to be used to log in on the LiquidVM instance over SSH. This option should be avoided as it is unsafe. As soon as someone has put a public key in the `authorized_keys` file, or set a SSH password explicitly, this option is ignored.

Logging Commands

Table 4-5 describes the logging commands for using the LiquidVM Launcher.

Table 4-5 Logging LiquidVM Launcher Commands

Key	Default Value	Description
`logReceiver`	`no remote log receiver`	Should be a hostname or IP address of a remote `syslog` receiver. This is a RFC3164-compliant remote `syslog` receiver that LiquidVM will send `syslog` messages to.

VMware Virtual Infrastructure-specific Commands

Table 4-6 describes the VMware Virtual Infrastructure-specific commands for using the LiquidVM Launcher.

Table 4-6 VMware Virtual Infrastructure-specific LiquidVM Launcher Commands

Key	Default Value	Description
`vmwareDatacenter`		The datacenter to use.
`vmwareComputeResource`		The cluster or host in which to put the VM (also see the `vmwareResourcePool`).
`vmwareResourcePool`	`none`	The pool within a compute resource in which to put the VM
`vmwareVmFolder`	The datacenter	The logical folder in which to put a newly created VM. The user needs to have Inventory rights on this folder.
`vmwareVmDatastore`	The same datastore as the ISO	The datastore in which to put the VM.
`vmwareNetwork`		The VMware network to connect the VMs NIC to.

Stopping a LiquidVM Instance

The LiquidVM launcher can be used to stop a LiquidVM instance. The first argument is `stop`, followed by the `name=<`*`name-of-the-vm-you-want-to-stop`*`>`, as follows:

```
java -jar lvm-esxlauncher-1.1.jar stop name=MyVM
```

Native Code Library Support in LiquidVM

This section discusses how to use native code libraries in LiquidVM.

Overview

As a best practice, BEA recommends that you deploy pure Java applications on LiquidVM, since this guarantees the maximum benefit of LiquidVM. However, this does not mean that LiquidVM cannot run native code; in fact, the JVM itself and the OS kernel below it are both native code. Native libraries can be built for LiquidVM and loaded the same way they are loaded for Java in any environment.

Moreover, it is possible to use your shared libraries that are built for Linux on LiquidVM if your library only uses the functions in Linux that LiquidVM also provides. LiquidVM's native interface is nix-like, but it only implements the functions the JVM needs from the OS because BEA does not provide a fully POSIX-compliant function interface to LiquidVM. LiquidVM's OS-kernel is developed entirely by BEA and is not a Linux or BSD-derivative, even though some Linux and BSD binaries can be loaded.

Caution: BEA provides no guarantees and no support for third-party native code that is trying to use functions that are not on the list of officially available functions.

Native Functions That Will Work with LiquidVM

The following list is an overview of the kind of native functions LiquidVM does provide. For a complete list of provided functions, see the Supported Native Code Functions in LiquidVM.

- The JVM, JNI, and JVMTI-interface. LiquidVM implements the full JVM, JNI, and JVMTI-specifications as the standards require.
- Malloc, calloc and free to allocate and free memory of standard C.
- Normal string manipulations functions of standard C (strchr, strcmp, strcpy, memcpy, etc.)
- Standard mathematical functions of standard C.
- A set of standard functions to do I/O that are used by the JVM and the surrounding native libraries.

Native Functions That Will Not Work with LiquidVM

The following list is an overview of the native functions LiquidVM does not provide. For a complete list of provided native functions, see the Supported Native Code Functions in LiquidVM.

- Native code that tries to directly interact with another OS-specific process in the same OS. If this kind of interaction is important to your application, you should not consider LiquidVM.
  - For example, if the reason you use native code is to access an Excel spreadsheet, LiquidVM will not be able to help you, LiquidVM is a single-process environment.
  - Another example is communication with MQSeries Transaction Manager, which normally is another process on the same machine. In the LiquidVM case, you can use a MQSeries Transactional Client, which is Java-based to make this work anyhow.
- Native code that tries to create another process using exec/fork or an equivalent. LiquidVM is a single-process environment and does not support the creation of additional processes.

Building a Shared Library for LiquidVM

LiquidVM can load 32-bit, x86 shared libraries that follows the ELF-standard. This is the linker format typically used by Linux and BSD OSes. This means that you can compile your shared library for LiquidVM on a Linux system exactly as you would compile a normal library for this OS.

In fact, in many cases an existing shared library for Linux will work for LiquidVM too, as long as the called functions exist in LiquidVM as well.

Loading a Shared Library Inside LiquidVM

Use the following guidelines when loading a shared library inside LiquidVM:

- Make the shared library available from LiquidVM (e.g., put it on the LiquidVM local disk).
- By default LiquidVM rejects third-party shared libraries; therefore, you will have to tell LiquidVM not to reject your library. The simplest way to do this is to pass in the `lnkLibsAllowAll=true` argument to the LiquidVM kernel.
- Load the shared library the same way you would do on an ordinary OS (e.g., `System.loadLibrary(…)`).

What Happens If a Shared Library Calls Non-existent Functions?

The shared library will load as normal, but as soon as you try to call a function that does not exist in LiquidVM, the VM will display an error message and a stack-trace of the offending call, and will then shut down.

Supported Native Code Functions in LiquidVM

Table A-1 lists the native code functions that are supported in LiquidVM.

Table A-1 Supported Native Code Functions

Supported Native Code Functions		
`JNI_GetDefaultJavaVMInitArgs`	`jni->CallCharMethodV`	`jni->CallNonvirtualShortMethodV`
`JNI_CreateJavaVM`	`jni->CallCharMethodA`	`jni->CallNonvirtualShortMethodA`
`JNI_GetCreatedJavaVMs`	`jni->CallShortMethod`	`jni->CallNonvirtualIntMethod`
`JNI_OnLoad`	`jni->CallShortMethodV`	`jni->CallNonvirtualIntMethodV`
`JNI_OnUnload`	`jni->CallShortMethodA`	`jni->CallNonvirtualIntMethodA`
`jni->GetVersion`	`jni->CallIntMethod`	`jni->CallNonvirtualLongMethod`
`jni->DefineClass`	`jni->CallIntMethodV`	`jni->CallNonvirtualLongMethodV`
`jni->FindClass`	`jni->CallIntMethodA`	`jni->CallNonvirtualLongMethodA`
`jni->FromReflectedMethod`	`jni->CallLongMethod`	`jni->CallNonvirtualFloatMethod`
`jni->FromReflectedField`	`jni->CallLongMethodV`	`jni->CallNonvirtualFloatMethodV`
`jni->ToReflectedMethod`	`jni->CallLongMethodA`	`jni->CallNonvirtualFloatMethodA`
`jni->GetSuperclass`	`jni->CallFloatMethod`	`jni->CallNonvirtualDoubleMethod`
`jni->IsAssignableFrom`	`jni->CallFloatMethodV`	`jni->CallNonvirtualDoubleMethodV`
`jni->ToReflectedField`	`jni->CallFloatMethodA`	`jni->CallNonvirtualDoubleMethodA`
`jni->Throw`	`jni->CallDoubleMethod`	`jni->CallNonvirtualVoidMethod`
`jni->ThrowNew`	`jni->CallDoubleMethodV`	`jni->CallNonvirtualVoidMethodV`
`jni->ExceptionOccurred`	`jni->CallDoubleMethodA`	`jni->CallNonvirtualVoidMethodA`
`jni->ExceptionDescribe`	`jni->CallIntMethod`	`jni->GetFieldID`
`jni->ExceptionClear`	`jni->CallIntMethodV`	`jni->GetObjectField`
`jni->FatalError`	`jni->CallIntMethodA`	`jni->GetBooleanField`
`jni->PushLocalFrame`	`jni->CallLongMethod`	`jni->GetByteField`
`jni->PopLocalFrame`	`jni->CallLongMethodV`	`jni->GetCharField`
`jni->NewGlobalRef`	`jni->CallLongMethodA`	`jni->GetShortField`
`jni->DeleteGlobalRef`	`jni->CallFloatMethod`	`jni->GetIntField`
`jni->DeleteLocalRef`	`jni->CallFloatMethodV`	`jni->GetLongField`
`jni->IsSameObject`	`jni->CallFloatMethodA`	`jni->GetFloatField`
`jni->NewLocalRef`	`jni->CallDoubleMethod`	`jni->GetDoubleField`
`jni->EnsureLocalCapacity`	`jni->CallDoubleMethodV`	`jni->SetObjectField`
`jni->AllocObject`	`jni->CallDoubleMethodA`	`jni->SetBooleanField`
`jni->NewObject`	`jni->CallVoidMethod`	`jni->SetByteField`
`jni->NewObjectV`	`jni->CallVoidMethodV`	`jni->SetCharField`
`jni->NewObjectA`	`jni->CallVoidMethodA`	`jni->SetShortField`
`jni->GetObjectClass`	`jni->CallNonvirtualObjectMethod`	`jni->SetIntField`
`jni->IsInstanceOf`	`jni->CallNonvirtualObjectMethodV`	`jni->SetLongField`
`jni->GetMethodID`	`jni->CallNonvirtualObjectMethodA`	`jni->SetFloatField`
`jni->CallObjectMethod`	`jni->CallNonvirtualBooleanMethod`	`jni->SetDoubleField`
`jni->CallObjectMethodV`	`jni->CallNonvirtualBooleanM..V`	`jni->GetStaticMethodID`
`jni->CallObjectMethodA`	`jni->CallNonvirtualBooleanM.A`	`jni->CallStaticObjectMethod`
`jni->CallBooleanMethod`	`jni->CallNonvirtualByteMethod`	`jni->CallStaticObjectMethodV`
`jni->CallBooleanMethodV`	`jni->CallNonvirtualByteMethodV`	`jni->CallStaticObjectMethodA`
`jni->CallBooleanMethodA`	`jni->CallNonvirtualByteMethodA`	`jni->CallStaticBooleanMethod`
`jni->CallByteMethod`	`jni->CallNonvirtualCharMethod`	`jni->CallStaticBooleanMethodV`
`jni->CallByteMethodV`	`jni->CallNonvirtualCharMethodV`	`jni->CallStaticBooleanMethodA`
`jni->CallByteMethodA`	`jni->CallNonvirtualCharMethodA`	`jni->CallStaticByteMethod`
`jni->CallCharMethod`	`jni->CallNonvirtualShortMethod`	`jni->CallStaticByteMethodV`

Table A-1 Supported Native Code Functions

Supported Native Code Functions		
jni->CallStaticByteMethodA	jni->NewStringUTF	jni->SetLongArrayRegion
jni->CallStaticCharMethod	jni->GetStringUTFLength	jni->SetFloatArrayRegion
jni->CallStaticCharMethodV	jni->GetStringUTFChars	jni->SetDoubleArrayRegion
jni->CallStaticCharMethodA	jni->ReleaseStringUTFChars	jni->RegisterNatives
jni->CallStaticShortMethod	jni->GetArrayLength	jni->UnregisterNatives
jni->CallStaticShortMethodV	jni->NewObjectArray	jni->MonitorEnter
jni->CallStaticShortMethodA	jni->GetObjectArrayElement	jni->MonitorExit
jni->CallStaticIntMethod	jni->SetObjectArrayElement	jni->GetJavaVM
jni->CallStaticIntMethodV	jni->NewBooleanArray	jni->GetStringRegion
jni->CallStaticIntMethodA	jni->NewByteArray	jni->GetStringUTFRegion
jni->CallStaticLongMethod	jni->NewCharArray	jni->GetPrim…ArrayCritical
jni->CallStaticLongMethodV	jni->NewShortArray	jni->ReleasePrim...ArrayCritical
jni->CallStaticLongMethodA	jni->NewIntArray	jni->GetStringCritical
jni->CallStaticFloatMethod	jni->NewLongArray	jni->ReleaseStringCritical
jni->CallStaticFloatMethodV	jni->NewFloatArray	jni->NewWeakGlobalRef
jni->CallStaticFloatMethodA	jni->NewDoubleArray	jni->DeleteWeakGlobalRef
jni->CallStaticDoubleMethod	jni->GetBooleanArrayElements	jni->ExceptionCheck
jni->CallStaticDoubleMethodV	jni->GetByteArrayElements	jni->NewDirectByteBuffer
jni->CallStaticDoubleMethodA	jni->GetCharArrayElements	jni->GetDirectBufferAddress
jni->CallStaticVoidMethod	jni->GetShortArrayElements	jni->GetDirectBufferCapacity
jni->CallStaticVoidMethodV	jni->GetIntArrayElements	Agent_OnLoad
jni->CallStaticVoidMethodA	jni->GetLongArrayElements	Agent_OnUnload
jni->GetStaticFieldID	jni->GetFloatArrayElements	jvmti->SetEventNotificationMode
jni->GetStaticObjectField	jni->GetDoubleArrayElements	jvmti->GetAllThreads
jni->GetStaticBooleanField	jni->ReleaseBooleanArrayElements	jvmti->SuspendThread
jni->GetStaticByteField	jni->ReleaseByteArrayElements	jvmti->ResumeThread
jni->GetStaticCharField	jni->ReleaseCharArrayElements	jvmti->StopThread
jni->GetStaticShortField	jni->ReleaseShortArrayElements	jvmti->InterruptThread
jni->GetStaticIntField	jni->ReleaseIntArrayElements	jvmti->GetThreadInfo
jni->GetStaticLongField	jni->ReleaseLongArrayElements	jvmti->GetOwnedMonitorInfo
jni->GetStaticFloatField	jni->ReleaseFloatArrayElements	jvmti->GetCurrentCont..Monitor
jni->GetStaticDoubleField	jni->ReleaseDoubleArrayElements	jvmti->RunAgentThread
jni->SetStaticObjectField	jni->GetBooleanArrayRegion	jvmti->GetTopThreadGroups
jni->SetStaticBooleanField	jni->GetByteArrayRegion	jvmti->GetThreadGroupInfo
jni->SetStaticByteField	jni->GetCharArrayRegion	jvmti->GetThreadGroupChildren
jni->SetStaticCharField	jni->GetShortArrayRegion	jvmti->GetFrameCount
jni->SetStaticShortField	jni->GetIntArrayRegion	jvmti->GetThreadState
jni->SetStaticIntField	jni->GetLongArrayRegion	jvmti->GetFrameLocation
jni->SetStaticLongField	jni->GetFloatArrayRegion	jvmti->NotifyFramePop
jni->SetStaticFloatField	jni->GetDoubleArrayRegion	jvmti->GetLocalObject
jni->SetStaticDoubleField	jni->SetBooleanArrayRegion	jvmti->GetLocalInt
jni->NewString	jni->SetByteArrayRegion	jvmti->GetLocalLong
jni->GetStringLength	jni->SetCharArrayRegion	jvmti->GetLocalFloat
jni->GetStringChars	jni->SetShortArrayRegion	jvmti->GetLocalDouble
jni->ReleaseStringChars	jni->SetIntArrayRegion	jvmti->SetLocalObject

Table A-1 Supported Native Code Functions

Supported Native Code Functions		
jvmti->SetLocalInt	jvmti->GetMethodLocation	jvmti->GetPhase
jvmti->SetLocalLong	jvmti->GetLocalVariableTable	jvmti->GetCur…ThreadCpuTimerInfo
jvmti->SetLocalFloat	jvmti->GetBytecodes	jvmti->GetCurrentThreadCpuTime
jvmti->SetLocalDouble	jvmti->IsMethodNative	jvmti->GetThreadCpuTimerInfo
jvmti->CreateRawMonitor	jvmti->IsMethodSynthetic	jvmti->GetThreadCpuTime
jvmti->DestroyRawMonitor	jvmti->GetLoadedClasses	jvmti->GetTimerInfo
jvmti->RawMonitorEnter	jvmti->GetClassLoaderClasses	jvmti->GetTime
jvmti->RawMonitorExit	jvmti->PopFrame	jvmti->GetPotentialCapabilities
jvmti->RawMonitorWait	jvmti->RedefineClasses	jvmti->AddCapabilities
jvmti->RawMonitorNotify	jvmti->GetVersionNumber	jvmti->RelinquishCapabilities
jvmti->RawMonitorNotifyAll	jvmti->GetCapabilities	jvmti->GetAvailableProcessors
jvmti->SetBreakpoint	jvmti->GetSourceDebugExtension	jvmti->GetEnv…LocalStorage
jvmti->ClearBreakpoint	jvmti->IsMethodObsolete	jvmti->SetEnv..LocalStorage
jvmti->SetFieldAccessWatch	jvmti->SuspendThreadList	jvmti->AddToBootstrapClass…
jvmti->ClearFieldAccessWatch	jvmti->ResumeThreadList	jvmti->SetVerboseFlag
jvmti->SetFieldModificationWatch	jvmti->GetAllStackTraces	jvmti->GetObjectSize
jvmti->ClearFieldMod…Watch	jvmti->GetThreadListStackTraces	accept
jvmti->Allocate	jvmti->GetThreadLocalStorage	access
jvmti->Deallocate	jvmti->SetThreadLocalStorage	atan
jvmti->GetClassSignature	jvmti->GetStackTrace	atan2
jvmti->GetClassStatus	jvmti->GetTag	basename
jvmti->GetSourceFileName	jvmti->SetTag	bind
jvmti->GetClassModifiers	jvmti->ForceGarbageCollection	bsearch
jvmti->GetClassMethods	jvmti->IterateOverObjectsReach..	calloc
jvmti->GetClassFields	jvmti->IterateOverReachable...	ceil
jvmti->GetImplementedInterfaces	jvmti->IterateOverHeap	close
jvmti->IsInterface	jvmti->IterateOverInst...	closedir
jvmti->IsArrayClass	jvmti->GetObjectsWithTags	connect
jvmti->GetClassLoader	jvmti->SetJNIFunctionTable	exit
jvmti->GetObjectHashCode	jvmti->GetJNIFunctionTable	exp
jvmti->GetObjectMonitorUsage	jvmti->SetEventCallbacks	floor
jvmti->GetFieldName	jvmti->GenerateEvents	getsockopt
jvmti->GetFieldDeclaringClass	jvmti->GetExtensionFunctions	gettimeofday
jvmti->GetFieldModifiers	jvmti->GetExtensionEvents	isinf
jvmti->IsFieldSynthetic	jvmti->SetExtensionEventCallback	isnan
jvmti->GetMethodName	jvmti->DisposeEnvironment	listen
jvmti->GetMethodDeclaringClass	jvmti->GetErrorName	log
jvmti->GetMethodModifiers	jvmti->GetJLocationFormat	lseek
jvmti->GetMaxLocals	jvmti->GetSystemProperties	malloc
jvmti->GetArgumentsSize	jvmti->GetSystemProperty	memchr
jvmti->GetLineNumberTable	jvmti->SetSystemProperty	memcmp

Table A-1 Supported Native Code Functions

Supported Native Code Functions	
memcpy	sleep
memmove	snprintf
memset	socket
nanosleep	sprintf
open	sqrt
opendir	stat
pow	stderr
pread	stdout
printf	strcasecmp
pthread_cond_broadcast	strcat
pthread_cond_destroy	strchr
pthread_cond_init	strcmp
pthread_cond_signal	strcpy
pthread_cond_timedwait	strdup
pthread_cond_wait	strlen
pthread_mutex_destroy	strncasecmp
pthread_mutex_init	strncat
pthread_mutex_lock	strncmp
pthread_mutex_unlock	strncpy
putc	strnlen
putchar	strpbrk
puts	strrchr
pwrite	strstr
qsort	strtol
read	strtoll
readdir	strtoul
realloc	strtoull
realpath	tan
recv	tolower
recvfrom	toupper
recvmsg	vfprint
scalbn	vsnprintf
sem_destroy	vsprintf
sem_init	write
sem_post	
sem_wait	
semdmsg	
send	
sendto	
setsockopt	
sin	