QLogic SHMEM Users Guide
Information furnished in this manual is believed to be accurate and reliable. However, QLogic Corporation assumes no responsibility for its use, nor for any infringements of patents or other rights of third parties which may result from its use. QLogic Corporation reserves the right to change product specifications at any time without notice. Applications described in this document for any of these products are for illustrative purposes only. QLogic Corporation makes no representation nor warranty that such applications are suitable for the specified use without further testing or modification. QLogic Corporation assumes no responsibility for any errors that may appear in this document.

No part of this document may be copied nor reproduced by any means, nor translated nor transmitted to any magnetic medium without the express written consent of QLogic Corporation. In accordance with the terms of their valid QLogic agreements, customers are permitted to make electronic and paper copies of this document for their own exclusive use.

### Document Revision History

<table>
<thead>
<tr>
<th>Rev. B, 10/8/2009</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Changes</strong></td>
</tr>
<tr>
<td>Added reference to the QMH7342 InfiniBand adapter.</td>
</tr>
</tbody>
</table>
# Table of Contents

1 **Introduction**
   - How this Guide is Organized ........................................ 1-1
   - Overview ......................................................... 1-1
   - Interoperability .................................................. 1-2
   - Conventions Used in this Guide .................................. 1-2
   - Documentation ..................................................... 1-2
   - Contact Information ............................................... 1-3

2 **Installation**
   - Requirements ....................................................... 2-1
   - Installation Tasks ................................................. 2-1
   - SHMEM RPMs ......................................................... 2-2
     - RPM Contents .................................................... 2-3

3 **Getting Started**
   - Basic SHMEM Program ............................................. 3-1
   - Compiling SHMEM Programs ....................................... 3-2
   - Running SHMEM Programs ........................................ 3-3
     - Using shmemrun ................................................ 3-3
     - Running programs without using shmemrun .................. 3-4

4 **Using QLogic SHMEM with MPI**

5 **Sizing Global Shared Memory**
   - Overview .......................................................... 5-1
   - Environment Variables ............................................ 5-3

A **Application Programming Interface**
   - General Operations .............................................. A-1
   - Symmetric heap .................................................. A-1
   - Contiguous Put Operations ...................................... A-1
   - Non-blocking Put Operations ................................... A-2
   - Strided Put Operations .......................................... A-2
   - Indexed Put Operations ........................................ A-3
   - Put and Non-blocking Ordering, Flushing and Completion .... A-3
   - Contiguous Get Operations ..................................... A-3
Non-blocking Get Operations .................................................. A-4
Strided Get Operations ......................................................... A-4
Indexed Get Operations .......................................................... A-5
Barriers .................................................................................. A-5
Broadcasts ................................................................................. A-5
Concatenation ........................................................................... A-5
Synchronization operations ....................................................... A-5
Atomic operations ....................................................................... A-6
Reductions ................................................................................ A-7
All-to-all (an extension beyond classic SHMEM) ......................... A-8
Locks ...................................................................................... A-8
Events ..................................................................................... A-8
General Operations (for compatibility) ....................................... A-9
Cache Operations (for compatibility) .......................................... A-9
Stack/Pointer Operations (for compatibility) ............................ A-9

B SHMEM Benchmark Programs
1 Introduction

This chapter describes the contents, intended audience, and organization of the QLogic SHMEM Users Guide.

How this Guide is Organized

The QLogic SHMEM Users Guide is organized into these sections:

- **Section 1, Introduction**, contains an overview of Qlogic SHMEM, lists all related documentation, and provides QLogic contact information.
- **Section 2, Installation**, contains SHMEM installation procedures.
- **Section 3, Getting Started**, provides information on compiling and running SHMEM programs.
- **Section 4, Using QLogic SHMEM with MPI**, provides information on using SHMEM with MPI.
- **Section 5, Sizing Global Shared Memory**, provides information on using SHMEM calls to allocate and release memory, as well as a list of SHMEM environment variables.
- **Appendix A, Application Programming Interface**, contains a list of SHMEM API calls.
- **Appendix B, SHMEM Benchmark Programs**, provides a list of included SHMEM micro-benchmark programs.

Overview

QLogic SHMEM is a user-level communications library for one-sided operations. It implements the SHMEM Application Programming Interface (API) and runs on the QLogic TrueScale InfiniBand stack. The SHMEM API provides global distributed shared memory across a network of hosts. Details of the API implementation are included in Appendix A, Application Programming Interface.

SHMEM is quite distinct from local shared memory (often abbreviated as "shm" or even "shmem"). Local shared memory is the sharing of memory by processes on the same host running the same OS system image. SHMEM provides access to global shared memory that is distributed across a cluster. The SHMEM API is completely different from the standard System V Shared Memory API provided by UNIX operating systems.
Interoperability

QLogic SHMEM depends on the Performance Scaled Messaging (PSM) protocol layer, implemented as a user-space library. This library is being submitted to the Open Fabrics Alliance for inclusion in future OFED versions. At this time is only available to run with QLogic TrueScale HCAs.

Conventions Used in this Guide

This guide uses the typographical conventions listed in Table 1-1.

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>command</td>
<td>Fixed-space font is used for literal items such as commands, functions, programs, files and pathnames, and program output.</td>
</tr>
<tr>
<td>variable</td>
<td>Italic fixed-space font is used for variable names in programs and command lines.</td>
</tr>
<tr>
<td>concept</td>
<td>Italic font is used for emphasis and concepts, as well as for documentation names/titles.</td>
</tr>
<tr>
<td>user input</td>
<td>Bold fixed-space font is used for literal items in commands or constructs that you type.</td>
</tr>
<tr>
<td>$</td>
<td>Indicates a command line prompt.</td>
</tr>
<tr>
<td>#</td>
<td>Indicates a command line prompt as a root user.</td>
</tr>
<tr>
<td>[]</td>
<td>Brackets enclose optional elements of a command or program construct.</td>
</tr>
<tr>
<td>...</td>
<td>Ellipses indicate that a preceding element can be repeated.</td>
</tr>
<tr>
<td>&gt;</td>
<td>A right caret identifies the cascading path of menu commands used in a procedure.</td>
</tr>
<tr>
<td>QLogic OFED 1.4.x</td>
<td>The current version number of the software included in this documentation.</td>
</tr>
<tr>
<td>NOTE:</td>
<td>Indicates important information.</td>
</tr>
</tbody>
</table>

Documentation

The product documentation includes:

- The *QLogic SHMEM Users Guide*
- Release Notes
Contact Information

| **Support Headquarters** | QLogic Corporation  
4601 Dean Lakes Blvd  
Shakopee, MN 55379  
USA |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>QLogic Web Site</td>
<td><a href="http://www.qlogic.com">www.qlogic.com</a></td>
</tr>
<tr>
<td>Technical Support Web Site</td>
<td><a href="http://support.qlogic.com">support.qlogic.com</a></td>
</tr>
<tr>
<td>Technical Support Email</td>
<td><a href="mailto:support@qlogic.com">support@qlogic.com</a></td>
</tr>
<tr>
<td>Technical Training Email</td>
<td><a href="mailto:tech.training@qlogic.com">tech.training@qlogic.com</a></td>
</tr>
</tbody>
</table>

Additional contact information is available from the Contact Support area of the Technical Support Web Site.
2 Installation

This section provides procedures for installing SHMEM.

Requirements

Before installing SHMEM, the following is required:

- QLogic InfiniBand Fabric Suite or QLogic OFED+ Host software (version 1.4.2 or greater) must be installed.
- Every node must have a QLogic TrueScale adapter card. The current models are:
  - QLE7340
  - QLE7342
  - QMH7342
- One (or more) Message Passing Interface (MPI) implementations are required and Performance Scaled Messaging (PSM) support must be enabled within the MPI:
  - The preferred MPI implementations for this release are QLogic MPI, Open MPI and HP-MPI.

Installation Tasks

1. Go to the QLogic web site (www.qlogic.com).
2. Click on the Downloads tab.
3. Click on the InfiniBand Adapters link.
4. Click on the appropriate InfiniBand adapter model in the middle frame, then click on the required operating system in the right frame. Click Go.
6. Login as user root.
7. Copy the tar file to the /root directory.
8. Type cd /root and press ENTER.
9. Type tar xvfz QLogic-SHMEM-VERSION-DATE-DISTRO.tgz and press ENTER. Then type cd QLogic-SHMEM-VERSION-DATE-DISTRO and press ENTER.
This unpacks the tar file to the QLogic-SHMEM-VERSION-DATE-DISTRO directory. Go (cd) to that directory and enter ls * to see the RPM files in the next level of sub-directories: Benchmarks, Devel and SHMEM.

10. Install the RPM files on all nodes in the cluster to be used for SHMEM with the command `rpm -Uvh */qlogic-shmem-*.rpm`. The -U flag to rpm ensures that any previous QLogic SHMEM RPMs on the host are removed before upgrading to the selected RPMs.

**NOTE:**

It is recommended to install all 3 RPMs on all nodes in the cluster using SHMEM.

**SHMEM RPMs**

By default QLogic SHMEM is installed with a prefix of /usr/shmem/qlogic into this directory structure:

- /usr/shmem/qlogic
- /usr/shmem/qlogic/bin
- /usr/shmem/qlogic/bin/qlogic
- /usr/shmem/qlogic/bin/openmpi
- /usr/shmem/qlogic/bin/hpmpi
- /usr/shmem/qlogic/lib64
- /usr/shmem/qlogic/lib64/qlogic
- /usr/shmem/qlogic/lib64/openmpi
- /usr/shmem/qlogic/lib64/hpmpi
- /usr/shmem/qlogic/include

It is recommended that /usr/shmem/qlogic/bin is added onto the $PATH. If it is not on $PATH, then the user needs to give full path names to find the shmemrun and shmemcc wrapper scripts. If desired, the entire SHMEM install can be relocated to another directory using the `rpm --prefix` option.

**NOTE:**

There are subdirectories inside of bin for each MPI that is supported. These contain SHMEM benchmark programs that are linked directly against the MPI libraries as well as the SHMEM libraries.

QLogic SHMEM contains the following RPM files:
Installation

Installation of `qlogic-shmem` on all cluster nodes is mandatory for running SHMEM programs. The RPM `qlogic-shmem-devel` is required for building SHMEM programs and only needs to be installed on nodes used for compiling (e.g., a front-end or head node). The RPM `qlogic-shmem-benchmarks` is an optional RPM providing some SHMEM benchmark programs. It must be installed on any nodes running the benchmarks.

RPM Contents

**qlogic-shmem**

The QLogic SHMEM RPM for the for the TrueScale interconnect. It has the following contents:

- `/usr/shmem/qlogic/bin/shmemrun`
- `/usr/shmem/qlogic/lib64/hpmpi/libqlogic_shmem.so.1`
- `/usr/shmem/qlogic/lib64/hpmpi/libqlogic_shmem.so.1.0`
- `/usr/shmem/qlogic/lib64/openmpi/libqlogic_shmem.so.1`
- `/usr/shmem/qlogic/lib64/openmpi/libqlogic_shmem.so.1.0`
- `/usr/shmem/qlogic/lib64/qlogic/libqlogic_shmem.so.1`
- `/usr/shmem/qlogic/lib64/qlogic/libqlogic_shmem.so.1.0`

These are the required files to run SHMEM programs. The file `shmemrun` is used to launch SHMEM programs and is basically a wrapper around `mpirun`. SHMEM programs are linked against the SHMEM library. This is required to run a SHMEM program.

**NOTE:**

There is a different SHMEM library for each version of MPI supported.

The `shmemrun` scripts chooses the correct SHMEM library automatically at program run time.
qlogic-shmem-devel
These are the developer files for QLogic SHMEM. It has the following contents:
- /usr/shmem/qlogic/bin/shmemcc
- /usr/shmem/qlogic/include/shmem.h
- /usr/shmem/qlogic/lib64/hpmpi/libqlogic_shmem.so
- /usr/shmem/qlogic/lib64/openmpi/libqlogic_shmem.so
- /usr/shmem/qlogic/lib64/qlogic/libqlogic_shmem.so

These are the files required to compile SHMEM programs. The file shmemcc is a wrapper around the C compiler. It sets up include paths, linkage paths, libraries and other options as appropriate. The RPM also provides symbolic links to library files as is conventional in RPM development packages.

qlogic-shmem-benchmarks
These are benchmark programs for QLogic SHMEM. It has the following contents:
- /usr/shmem/qlogic/bin/hpmpi/shmem-rand
- /usr/shmem/qlogic/bin/openmpi/shmem-rand
- /usr/shmem/qlogic/bin/qlogic/shmem-rand
- /usr/shmem/qlogic/bin/shmem-get-bibw
- /usr/shmem/qlogic/bin/shmem-get-bw
- /usr/shmem/qlogic/bin/shmem-get-latency
- /usr/shmem/qlogic/bin/shmem-put-bibw
- /usr/shmem/qlogic/bin/shmem-put-bw
- /usr/shmem/qlogic/bin/shmem-put-latency

This RPM contains pre-compiled SHMEM benchmark programs. These programs are described in Appendix B.
This section provides information on SHMEM programs.

**Basic SHMEM Program**

Following is an example of a basic SHMEM program:

```
% cat shmem_world.c
#include <shmem.h>
#include <stdio.h>

int main ()
{
    shmem_init();
    printf("Hello from PE %d out of %d\n", my_pe(), num_pes());
}
```

**NOTE:**

These instructions assume a standard SHMEM installation and that `/usr/shmem/qlogic/bin` has been added to the `$PATH`.

The `%` character above is used to indicate the shell prompt and is followed by a command. The program can be compiled and linked using the `shmemcc` wrapper script:

```
% shmemcc shmem_world.c -o shmem_world
```

It can be run using the `shmemrun` wrapper script:

```
% shmemrun -m hosts -np 2 ./shmem_world
Hello from PE 1 out of 2
Hello from PE 0 out of 2
```

This assumes a `hosts` file, containing the host names on which the program is run. The `-np` option is used to specify the number of processing elements (PEs) to be run (e.g., 2).
Compiling SHMEM Programs

The script `shmemcc` is a wrapper for the compilation of SHMEM C programs. The main purpose of the script is to call the C compiler with additional options to specify the SHMEM include directory, the SHMEM library directory, and to appropriately link in the SHMEM library. The `shmemcc` script automatically determines the correct directories by finding them relative to its own location. The standard directory layout of the QLogic SHMEM software is assumed.

The default C compiler is gcc, and this can be overridden by specifying a compiler with the `$SHMEM_CC` environment variable.

If the option `-show` is added to `shmemcc`, it displays the command line that would be used to invoke the C compiler, but the C compiler will not be invoked. All other arguments to `shmemcc` are passed through to the C compiler without modification.

The C compiler can be used directly without using `shmemcc`. In that case the user must do the following:

Compilations need the option:

```bash
-I $SHMEM_DIR/include
```

Linkages need the options:

```bash
-Wl,--export-dynamic,--allow-shlib-undefined
-L $SHMEM_DIR/lib64/qlogic
-lqlogic_shmem
```

where `$SHMEM_DIR` denotes the top-level directory of the SHMEM installation, typically `/usr/shmem/qlogic`. The `-L` option specifically picks out the version of the SHMEM libraries for use with QLogic MPI. However, this choice does not constrain the SHMEM binary in any way and it can be run over any of the supported MPIs.

**NOTE:**

If the SHMEM RPM is installed with `--prefix=usr` then the `-I` option is not necessary since the header files are in system default locations. All of the linkage options are still required.
The rationale for the `-Wl,--export-dynamic,-allow-shlib-undefined` option is to prevent other library and symbol dependencies in the SHMEM library from percolating up into the application binaries. These symbols include those from the underlying MPI implementation. There is no need to couple the application binary to a particular MPI, and these symbols will be correctly resolved at run-time. The advantage of this approach is that SHMEM application binaries will be portable across different implementations of the QLogic SHMEM library, including portability over different underlying MPIs.

Running SHMEM Programs

Using shmemrun

The script `shmemrun` is a wrapper script for running SHMEM programs using `mpirun`. The main purpose of the script is to call `mpirun` with additional options to specify the SHMEM library directory so that its dynamic libraries can be resolved. The script detects which `mpirun` is being used and remaps some common `mpirun` options to present a convenient and consistent interface to SHMEM users. Additionally, it enables PSM support in the underlying `mpirun` if required, and auto-propagates PSM, IPATH and SHMEM environment variables to the MPI processes. The script `shmemrun` automatically determines the correct directories by finding them relative to its own location. The standard directory layout of the QLogic SHMEM software is assumed.

By default `mpirun` is picked up from the path and is assumed to be called "mpirun". Alternatively, the pathname of `mpirun` can be specified with the `$SHMEM_MPIRUN` environment variable.

If `-show` is one of the options with `shmemrun`, it displays the command line used to invoke `mpirun`, but will not invoke it. Options that specify the number of processes and the hosts file are mapped by `shmemrun` to options that are accepted by the underlying `mpirun`. Additionally, the contents of the host file can be parsed and regenerated if necessary and options to propagate environment variables are provided. The rationale for this script is to allow you to use the familiar options from the `mpirun` chosen and they will automatically be remapped as required for the actual `mpirun`. This makes it possible to write scripts that call `shmemrun` without exposing these details of the underlying `mpirun` command.

If `shmemrun` finds the special option "--" while processing the option list, that option is deleted and subsequent options and command line arguments are passed through without any modification. This can be useful to prevent `shmemrun` from modifying options of the program that are being run.
Running programs without using shmemrun

If you do not wish to use this wrapper script, then you must arrange for the SHMEM libraries to be found at run time using $LD_LIBRARY_PATH or an equivalent mechanism, and ensure that PSM support is enabled in your MPI implementation. The libraries can be found at:

$SHMEM_DIR/lib64/$MPI

where $SHMEM_DIR denotes the top-level directory of the SHMEM installation, typically /usr/shmem/qlogic, and $MPI is your choice of MPI (one of qlogic, hpmpi or openmpi).

Additionally, the PSM receive thread must be disabled using:

export PSM_RCVTHREAD=0

The receive thread causes multiple threads of activity to be in the SHMEM library and this is not allowed since the SHMEM library is not designed to be thread-safe.
Using QLogic SHMEM with MPI

QLogic SHMEM requires the QLogic PSM layer to provide the network transport function and this runs exclusively on QLogic TrueScale InfiniBand HCAs. It also requires a compatible MPI implementation (also running over PSM) to provide program start up and other miscellaneous services. The one-sided operations in QLogic SHMEM are not layered on top of MPI, however, and go directly to PSM to give low-latency, high-performance access to the TrueScale HCA architecture.

Typical SHMEM programs are written using calls to the SHMEM API and do not use MPI calls. In this case the program binary generated by `shmemcc` contains references to the SHMEM dynamic library and no references at all to all MPI libraries. These binaries are portable across all MPI implementations supported by QLogic SHMEM. This is true of the `get/put` micro-benchmarks provided in the `qlogic-shmem-benchmarks` RPM.

The desired MPI can be selected at run time simply by placing the desired `mpirun` on `$PATH`, or by using the `$SHMEM_MPIRUN` environment variable.

Alternatively, it is possible to write hybrid SHMEM/MPI programs that use features from both the SHMEM and MPI libraries. These programs must call `shmem_init()` to initialize the SHMEM library state. They may also use `MPI_Init()` and `MPI_Finalize()` if needed. Note that there will be a direct one-to-one correspondence between the SHMEM and `MPI_COMM_WORLD` rank assignments:

- `shmem_my_pe()` will match `MPI_Comm_rank()` on `MPI_COMM_WORLD`
- `shmem_n_pes()` will match `MPI_Comm_size()` on `MPI_COMM_WORLD`

Hybrid SHMEM/MPI programs must be linked against SHMEM libraries and the correct MPI libraries. It is recommended that the implementation of the MPI wrapper script(s) (`mpicc`) is used for compilation and that additional options are specified to find the SHMEM include and library files. The generated binary has references to both libraries and is specific to that MPI implementation. It is recommended that `shmemrun` is used to run the program. The user must ensure that the correct `mpirun` is picked up from `$PATH` or using the `$SHMEM_MPIRUN` environment variable.
5 Sizing Global Shared Memory

Overview

SHMEM provides `shmalloc`, `shrealloc` and `shfree` calls to allocate and release memory using a symmetric heap. These functions are called collectively across the processing elements (PEs) so that the memory is managed symmetrically across them. The extent of the symmetric heap determines the amount of global shared memory per PE that is available to the application.

This is an important resource and this section discusses the mechanisms available to size it. Applications can access this memory in various ways and this maps into quite different access mechanisms:

- Accessing global shared memory on my PE: this is achieved by direct loads and stores to the memory.
- Accessing global shared memory on a PE on the same host: this is achieved by mapping the global shared memory using the local shared memory mechanisms (e.g., System V shared memory) operating system and then accessing the memory by direct loads and stores. This means that each PE on a host needs to map the global shared memory of each other PE on that host. Note that these accesses do not use the adapter and interconnect.
- Accessing global shared memory on a PE on a different host: this is achieved by sending put, get and atomic requests across the interconnect.

**NOTE:**

There is a connection between the sizing of the global shared memory and local shared memory because of the mechanism used for accessing global shared memory in a PE that happens to be on the same host.
The QLogic SHMEM library pre-allocates room in the virtual address space according to $SHMEM_SHMALLOC_MAX_SIZE (default of 4GB). It then populates this with enough pages to cover $SHMEM_SHMALLOC_INIT_SIZE (default 16MB) bytes of memory. The global shared memory segment can then grow dynamically from its initial size up to its maximum size. If an allocation attempts to exceed the max-size allocations are no longer guaranteed to succeed, and will fail if there is no room in the virtual memory space of the process following the global shared memory segment. Upon failure the call to shmalloc or shrealloc returns NULL. The only down-side of using a large max-size is occupancy of virtual address space (48 bits for 64-bit processes is very plentiful), and set-up of page table entries by the OS. A reasonable limit is 4GB per process. One side-effect of this approach is that SHMEM programs consume a large amount of virtual memory when viewed with the "top" program. This is due to the large maximum size setting. The RES field of top indicates the actual amount of memory that is resident in memory (i.e., in actual use).

If a SHMEM application program runs out of global shared memory, increase the value of $SHMEM_SHMALLOC_MAX_SIZE. The value of $SHMEM_SHMALLOC_INIT_SIZE can also be changed to pre-allocate more memory up front rather than dynamically.

It is possible for SHMEM to fail at start-up or while allocating global shared memory due to limits placed by the operating system on the amount of *local* shared memory that SHMEM can use. Since SHMEM programs can use very large amounts of memory this can exceed typical OS configurations. As long as there is sufficient physical memory for the program, the following steps can be used to solve local shared memory allocation problems:

- **Check for low ulimits on memory:**
  
  `ulimit -l : max locked memory (important for PSM not SHMEM)`
  `ulimit -v : max virtual memory`

- **Check the contents of these sysctl variables:**
  
  `sysctl kernel.shmmax (maximum size of a single shm allocation in bytes)`
  `sysctl kernel.shmall (maximum size of all shm allocations in *pages*)`
  `sysctl kernel.shmmni (maximum number of shm segments)`

- **Check the size of /dev/shm:**
  
  `df /dev/shm`

- **Check for stale files in /dev/shm:**
  
  `ls /dev/shm`

If any of these checks indicate a problem, please ask the cluster administrator to increase the limit.
Environment Variables

The following environment variables are currently provided by the SHMEM run time library:

$SHMEM_SHMALLOC_INIT_SIZE (default 16MB):
the initial size of the global shared memory segment.

$SHMEM_SHMALLOC_MAX_SIZE (default 4GB):
the maximum size of the global shared memory segment.

$SHMEM_SHMALLOC_CHECK (default on):
0 to disable and 1 to enable shared memory consistency checks. These are good checks for correctness but degrade the performance of shmalloc() and shfree(). These routines are usually not important for benchmark performance, so for now the checks are turned on to catch bugs early.

$SHMEM_IDENTIFY:
if set each SHMEM process will print out the SHMEM identity string and the path to the SHMEM library file.

$SHMEM_GET_REQ_LIMIT (64):
The maximum number of outstanding short get requests for this end-point for the short get protocol (0 means unlimited). Each short get request can be up to 2KB.

$SHMEM_GET_LONG_REQ_LIMIT (16):
The maximum number of outstanding get requests for this end-point for the long get protocol (0 means unlimited).

$SHMEM_PUT_FRAG_LIMIT (4096):
The maximum number of outstanding put fragments for this end-point for the short put protocol (0 means unlimited). Each short put fragment can be up to 2KB.

$SHMEM_PUT_LONG_FRAG_LIMIT (128):
The maximum number of outstanding put fragment requests for this end-point for the long get protocol (0 means unlimited).

$SHMEM_GET_LONG_SIZE (default = 8KB for non-blocking gets, 32KB for blocking gets):
gets of this size and larger use the SHMEM long get message protocol. Note that the parameter only allows the size to be changed in unison for both non-blocking and blocking gets.
**$SHMEM_PUT_LONG_SIZE** (default = 8KB for non-blocking puts, 16KB for blocking puts):

puts of this size and larger use the SHMEM long put message protocol. Note that the parameter only allows the size to be changed in unison for both non-blocking and blocking puts.

**$SHMEM_PUT_REPLY_COMBINING_COUNT** (default 8):

The number of consecutive put replies on a flow to combine together into a single reply.
The command *shmemrun* automatically propagates *SHMEM* environment variables from its own environment to all the SHMEM processes. This means that the environment variables can be simply setup in the front-end shell used to invoke *shmemrun*. The command *shmemrun* also has its own environment variables:

**$SHMEM_MPIRUN** (default is *mpirun* from the PATH):

Specifies where to find *mpirun*.

**$SHMEMRUN_VERBOSE**:

Enables verbose output for *shmemrun*.

**$SHMEMRUN_SLEEP**:

Specifies a sleep time (in seconds) after the job completes. This variable is intended for testing use.

**$SHMEMRUN_TIMEOUT**:

Specifies a time-out value (in seconds). When the timeout value is reached, the *mpirun* is killed. This variable is intended for testing use.

**NOTE:**

The set of supported environment variables and their defaults may vary from release to release.
A Application Programming Interface

This appendix lists the provided SHMEM API calls and details any restrictions.

General Operations

shmem_init
start_pes
my_pe
_my_pe
shmem_my_pe
num_pes
__num_pes
shmem_n_pes

Symmetric heap

shmalloc
shfree
shrealloc

Contiguous Put Operations

shmem_short_p
shmem_int_p
shmem_long_p
shmem_float_p
shmem_double_p
shmem_double_put
Non-blocking Put Operations

shmem_double_put_nb
shmem_float_put_nb
shmem_int_put_nb
shmem_long_put_nb
shmem_longdouble_put_nb
shmem_longlong_put_nb
shmem_put_nb
shmem_put32_nb
shmem_put64_nb
shmem_put128_nb
shmem_putmem_nb
shmem_short_put_nb

Strided Put Operations

shmem_double_iput
shmem_float_iput
shmem_int_iput
shmem_iput
shmem_iput32
Indexed Put Operations

shmem_iiput64
shmem_iiput128
shmem_long_iiput
shmem_longdouble_iiput
shmem_longlong_iiput
shmem_short_iiput

Put and Non-blocking Ordering, Flushing and Completion

shmem_fence
shmem_quiet
shmem_wait_nb
shmem_test_nb

Contiguous Get Operations

shmem_short_g
shmem_int_g
shmem_long_g
shmem_float_g
shmem_double_g
shmem_double_get
shmem_float_get
shmem_get
shmem_get32
shmem_get64
shmem_get128
shmem_getmem
shmem_int_get
shmem_long_get
shmem_longdouble_get
shmem_longlong_get
shmem_short_get

Non-blocking Get Operations

shmem_double_get_nb
shmem_float_get_nb
shmem_int_get_nb
shmem_long_get_nb
shmem_longdouble_get_nb
shmem_longlong_get_nb
shmem_short_get_nb
shmem_get_nb
shmem_get32_nb
shmem_get64_nb
shmem_get128_nb
shmem_getmem_nb

Strided Get Operations

shmem_double_iget
shmem_float_iget
shmem_int_iget
shmem_iget
shmem_iget32
shmem_iget64
shmem_iget128
shmem_long_iget
shmem_longdouble_iget
shmem_longlong_iget
shmem_short_iget
Indexed Get Operations

shmem_ixget
shmem_ixget32
shmem_ixget64

Barriers

barrier
shmem_barrier_all
shmem_barrier

Broadcasts

shmem_broadcast
shmem_broadcast32
shmem_broadcast64

Concatenation

shmem_collect
shmem_collect32
shmem_collect64
shmem_fcollect
shmem_fcollect32
shmem_fcollect64

Synchronization operations

shmem_int_wait
shmem_long_wait
shmem_longlong_wait
shmem_short_wait
shmem_wait
shmem_int_wait_until
shmem_long_wait_until
shmem_longlong_wait_until
Atomic operations

shmem_double_swap
shmem_float_swap
shmem_short_swap
shmem_int_swap
shmem_long_swap
shmem_longlong_swap
shmem_swap
shmem_short_cswap
shmem_int_cswap
shmem_long_cswap
shmem_longlong_cswap
shmem_short_mswap
shmem_int_mswap
shmem_long_mswap
shmem_longlong_mswap
shmem_short_inc
shmem_short_add
shmem_short_finc
shmem_int_finc
shmem_long_finc
shmem_longlong_finc
shmem_short_fadd
shmem_int_fadd
shmem_long_fadd
shmem_longlong_fadd
Reductions

- `shmem_int_and_to_all`
- `shmem_long_and_to_all`
- `shmem_longlong_and_to_all`
- `shmem_short_and_to_all`
- `shmem_int_or_to_all`
- `shmem_long_or_to_all`
- `shmem_longlong_or_to_all`
- `shmem_short_or_to_all`
- `shmem_int_xor_to_all`
- `shmem_long_xor_to_all`
- `shmem_longlong_xor_to_all`
- `shmem_short_xor_to_all`
- `shmem_double_min_to_all`
- `shmem_float_min_to_all`
- `shmem_int_min_to_all`
- `shmem_long_min_to_all`
- `shmem_longdouble_min_to_all`
- `shmem_longlong_min_to_all`
- `shmem_short_min_to_all`
- `shmem_double_max_to_all`
- `shmem_float_max_to_all`
- `shmem_int_max_to_all`
- `shmem_long_max_to_all`
- `shmem_longdouble_max_to_all`
- `shmem_longlong_max_to_all`
- `shmem_short_max_to_all`
- `shmem_complexd_sum_to_all`
  - complex collectives are not implemented
- `shmem_complexf_sum_to_all`
  - complex collectives are not implemented
A–Application Programming Interface
All-to-all (an extension beyond classic SHMEM)

shmem_double_sum_to_all
shmem_float_sum_to_all
shmem_int_sum_to_all
shmem_long_sum_to_all
shmem_longdouble_sum_to_all
shmem_longlong_sum_to_all
shmem_short_sum_to_all
shmem_complexd_prod_to_all

- complex collectives are not implemented
shmem_complexf_prod_to_all

- complex collectives are not implemented
shmem_double_prod_to_all
shmem_float_prod_to_all
shmem_int_prod_to_all
shmem_long_prod_to_all
shmem_longdouble_prod_to_all
shmem_longlong_prod_to_all
shmem_short_prod_to_all

All-to-all (an extension beyond classic SHMEM)
shmem_alltoall
shmem_alltoall32
shmem_alltoall64

Locks
shmem_set_lock
shmem_clear_lock
shmem_test_lock

Events
clear_event
set_event
ait_event

test_event

**General Operations (for compatibility)**

globalexit
- allows any process to abort the job

**Cache Operations (for compatibility)**

shmem_clear_cache_inv
- implemented as a no-op

shmem_set_cache_inv
- implemented as a no-op

shmem_set_cache_line_inv
- implemented as a no-op

shmem_udcflush
- implemented as a no-op

shmem_udcflush_line
- implemented as a no-op

**Stack/Pointer Operations (for compatibility)**

shmem_stack
- implemented as a no-op

shmem_ptr
- returns the address if the PE is my PE, otherwise NULL
SHMEM Benchmark Programs

The following SHMEM micro-benchmark programs are included:

- `shmem-get-latency`: measures get latency
- `shmem-get-bw`: measures streaming get bandwidth (uni-directional)
- `shmem-get-bibw`: measures streaming get bandwidth (bi-directional)
- `shmem-put-latency`: measures put latency
- `shmem-put-bw`: measures streaming put bandwidth (uni-directional)
- `shmem-put-bibw`: measures streaming put bandwidth (bi-directional)

The programs can be used to measure roundtrip get latency, one way put latency, get and put bandwidth, as well as get and put message rates.

The benchmarks must be run with an even number of processes. They are typically run on exactly two hosts with the processes equally-divided between them. The processes are split up into pairs, with one from each pair on either host and each pair is loaded with the desired traffic pattern. The benchmark automatically determines the correct mapping, regardless of the actual rank order of the processes and their mapping to the two hosts. Alternatively, if the `-f` option is specified the benchmark is forced to use the rank order when arranging the communication pattern. In this mode and with `np` ranks, each rank `i` in `[0, np/2)` is paired with rank `(np / 2) + i`. For example, this mode can be used to test SHMEM performance within a single node.

The micro-benchmarks have the following command line options:

- QLogic SHMEM `shmem-put-bw` micro-benchmark
  
  Usage: `shmem-put-bw [options]`


Table B-1. QLogic SHMEM shmem.Put-bw micro-benchmark options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a INT</td>
<td>a log2 of desired alignment for buffers (default = 12)</td>
</tr>
<tr>
<td>-b INT</td>
<td>batch size, number of concurrent operations (default = 64)</td>
</tr>
<tr>
<td>-f</td>
<td>force order for bifurcation of PEs based on rank order</td>
</tr>
<tr>
<td>-h</td>
<td>displays the help page</td>
</tr>
<tr>
<td>-l INT</td>
<td>set minimum message size (default = 2)</td>
</tr>
<tr>
<td>-m INT</td>
<td>sets the maximum message size (default = 4194304)</td>
</tr>
</tbody>
</table>

One additional SHMEM micro-benchmark program is also included to measure get and put performance with randomized PE selection and randomized target memory locations:

- shmem-rand: randomized put/get benchmark

This is actually a hybrid SHMEM/MPI code, so a binary is provided per supported MPI implementation. It has the following command line options:

- QLogic SHMEM random access benchmark

Usage: shmem-rand [options] [list of message sizes]. Message sizes are specified in bytes (default = 8)

Table B-2. QLogic SHMEM random access benchmark options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a</td>
<td>use automatic (NULL) handles for NB ops (default explicit handles)</td>
</tr>
<tr>
<td>-b</td>
<td>use a barrier every window</td>
</tr>
<tr>
<td>-c INTEGER</td>
<td>specify loop count (see also -t)</td>
</tr>
<tr>
<td>-f</td>
<td>fixed window size (default is scaled)</td>
</tr>
<tr>
<td>-h</td>
<td>displays the help page</td>
</tr>
</tbody>
</table>
Table B-2. QLogic SHMEM random access benchmark options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-l</td>
<td>enable communication to local ranks</td>
</tr>
<tr>
<td>-m INT</td>
<td>memory size in MB (default = 8MB): or in KB with a K suffix</td>
</tr>
<tr>
<td>-n</td>
<td>use non-pipelined mode for NB ops (default pipelined)</td>
</tr>
<tr>
<td>-o OP</td>
<td>choose OP from get, getnb, put, putnb</td>
</tr>
<tr>
<td>-r</td>
<td>use ring pattern (default is random)</td>
</tr>
<tr>
<td>-s</td>
<td>enable communication to self</td>
</tr>
<tr>
<td>-t FLOAT</td>
<td>if the loop count is not given, run the test for this many seconds (default is 10s)</td>
</tr>
<tr>
<td>-u</td>
<td>run in uni-directional mode</td>
</tr>
<tr>
<td>-v</td>
<td>verbose mode (repeat for more verbose)</td>
</tr>
<tr>
<td>-w INT</td>
<td>set the window size (default = 32)</td>
</tr>
<tr>
<td>-x INT</td>
<td>window size limit (default = 16384)</td>
</tr>
</tbody>
</table>