Envoy XIPC

Version 3.4.0

Platform Notes - UNIX

Envoy Technologies Inc.

555 Route 1 South
Iselin, NJ 08830

http://www.envoytech.com

Printed in United States of America

TABLE OF CONTENTS

1. PREFACE

1.1 Purpose

The primary objective of these Platform Notes is to provide the information necessary for working with XIPC on a variety of UNIX Operating Systems. Wherever appropriate, platform-specific information is detailed for the supported UNIX platforms; these include:

- AIX 4.x+
- Solaris 2.5+
- Tru64 UNIX
- LINUX
- HPUX 10.x
- HPUX 11.x
- SCO OpenServer
- SCO UnixWare

The document is divided into three sections: a preface that outlines the purpose of the Platform Notes; detailed step-by-step instructions to be followed when installing XIPC on UNIX platforms; and information necessary for developing applications with the current release of XIPC on the various UNIX platform.

1.2 Audience

These instructions are intended as a guide for the person who will be installing the XIPC product on a UNIX platform. Portions of the installation process require `root` privileges. The installation may additionally require that certain kernel parameters be reconfigured. It is therefore required that individuals performing the installation have the necessary authorization and experience, or should otherwise contact the system administrator for assistance.

These Platform Notes also describe the platform-specific details of actually working with the XIPC tool set. This section is thus intended for the experienced UNIX software developer. A significant degree of familiarity with 'C' language program development concepts on the applicable platform is assumed.

1.3 Contents

The balance of these Platform notes consists of the following sections:

- *Installation:*
  - Selecting the machine to install on.
  - Reading XIPC from the provided media onto the target machine.
  - Configuring the kernel for using XIPC.
  - Configuring the network databases for using XIPC in a network environment.
- *Using XIPC on UNIX Platforms:*
  - Calculating the UNIX operating system resources required by XIPC.
  - XIPC daemon programs: their functions and how to use them.
  - Notes on compiling, linking and other information necessary for developing software applications with XIPC on UNIX platforms.

2. INSTALLATION

2.1 Selecting A Target Machine

Rules for selecting a target machine for the installation revolve around the issues of accessibility and convenience. The product can be installed on any of the following UNIX platforms covered by the developer license: AIX 4.x+, Tru64 UNIX, HPUX 10.x, HPUX 11.x, LINUX, SCO OpenServer, SCO UnixWare and Solaris 2.5+. Where possible, a readily accessible workstation or server should be used as this will simplify the sharing of the multi-user XIPC tool set.

An important consideration in this regard is that XIPC installation usually requires that the operating system for the host machine be reconfigured before XIPC can be used. Selecting a machine for which re-configuration is easy and unconstrained will expedite the installation process.

Installation of the XIPC *Network* product additionally requires that the selected machine be outfitted with the appropriate network connectivity.

2.2 Reading XIPC Onto The Target Machine

The XIPC release media includes all the software, sample programs and other related files necessary for working with the product. Reading XIPC from the provided media onto the target machine should be performed in the following steps:

- ❑ Create a directory for XIPC (such as **/user/xipc**):

```
mkdir /user/xipc
```

- ❑ Make this directory the current directory:

```
cd /user/xipc
```

- ❑ Insert the release diskette/tape/CD into the computer.
- ❑ Set the XIPCDEV environment variable so that it identifies the device being used to read the XIPC media:

  Example (Bourne or Korn Shell):

```
XIPCDEV=/dev/rmt/0
export XIPCDEV
```

  Example (C-Shell):

```
setenv XIPCDEV /dev/rmt/0
```

- ❑ Enter the command:

```
tar xvf $XIPCDEV xipc-install
```

- ❑ Enter the command:

```
./xipc-install
```

As a result of this process, the XIPC directory structure will be created and XIPC software will be installed.

The directories and files to be installed on the supported UNIX platforms are described below. (Those whose *descriptions* are in italic type are not present on all of the platforms.) The table which follows shows the directory distribution for all supported UNIX platforms.

lib/	a directory containing the XIPC libraries.
lib/shared/	*a directory containing shared libraries.*
lib/threads/	*a directory containing thread-safe libraries.*
bin/	a directory containing XIPC commands and shell scripts.
pdb/	*a directory containing the platform database files used for data translation.*
include/	a directory containing XIPC header files.
samples/	a directory containing XIPC sample programs.
log/	a directory for XIPC log files.
xipc.env	a sample environment file for XIPC.

Directories and Files	AIX 4.x+	Tru64 UNIX	HPUX 10.x	HPUX 11.x	LINUX	SCO	Solaris 2.5+
lib/	√	√	√	√	√	√	√
lib/shared	√	√	√	√	√	X	√
lib/threads	√	√	√	√	√	X	√
bin/	√	√	√	√	√	√	√
Include/	√	√	√	√	√	√	√
samples/	√	√	√	√	√	√	√
log/	√	√	√	√	√	√	√
xipc.env	√	√	√	√	√	√	√

2.3 Configuring the Platform Kernel

With the exception of the AIX platforms, which have no kernel configuration requirements, platform kernel configuration is addressed as follows:

XIPC uses native kernel IPC facilities. The native IPC facilities must therefore be present and configured correctly in order for XIPC to work properly. Certain kernel parameter values are likely to require adjustment. Formulae for calculating the exact level of native IPC resource requirements are included later in this document. Ultimately, the kernel parameter values should be set to meet the overall needs of XIPC and other activity occurring on the machine.

In order to support a reasonable number of users simultaneously, it is necessary to configure the UNIX platform kernel with a sufficient number of semaphore identifiers (e.g., SEMMNI) and semaphores in the system (e.g., SEMMNS). A minimum of 60 is recommended as a starting point for each. For example, on the Solaris platform, in `/etc/system`, set:

```
semsys:seminfo_semmni=60.
semsys:seminfo_semmns=60.
```

For the Solaris platform:

The Solaris kernel is by default configured to allow a user to attach to a maximum of six (6) shared memory segments at one time. To use an instance that needs more than six shared memory segments or to use more than one instance, it is necessary to configure the kernel appropriately. For example:

```
shmsys:shminfo_shmseg=60.
```

2.4 Configuring the Platform Environment

2.4.1 XIPCROOT

`XIPCROOT` is an environment variable which must be set to the path of the platform directory in which XIPC was installed before invoking `xipcstart` or any other API call.

2.4.2 PATH

It is advisable to add the path of the XIPC bin directory to the `PATH` environment variable of users that will be working with the product.

2.5 Configuring the Network Environment

Installing *XIPC/Network* requires that the network be notified of XIPC's intention to use certain TCP/IP services.

This is accomplished by adding three entries within the `/etc/services` file**.** The port numbers in these entries should be selected so that they relate to unused port numbers.

Example:

```
xipcetc       4000/udp
xipcetc       4000/tcp
xipcserv      4001/tcp
```

These entries *must* be added on *each* of the platforms using XIPC in a network environment. *XIPC/Network* will not work properly on platforms where these entries have not been added. If some form of network directory service facility is being used for finding network services, the above indicated changes to `/etc/services` are not necessary. The network directory services must nonetheless be updated.

2.6 Configuring the XIPC Daemon Programs

As will be described below, XIPC employs daemon programs for certain aspects of its processing. Three of these programs *must* run with an effective uid of `root`. They are `xipclad`, `xipciad` and `xipcidld`.

One approach is to have `root` own the program files and to limit their execution rights to owner (i.e., `root`). Alternatively, setting the `setuid` bit of the `root`-owned files will permit the daemons to be started by non-`root` users as well.

In the following example, the user has logged on as `root`:

```
# cd $XIPCROOT/bin
# chown root xipclad xipciad xipcidld
# chmod 4111 xipclad xipciad xipcidld
```

3. USING XIPC ON UNIX PLATFORMS

The XIPC paradigm is not specific to any particular operating system environment. From the programmer's perspective the model presented by XIPC's API is almost entirely portable across environments. The underlying issues of how XIPC relates to a particular platform and how it utilizes the native operating system resources are, however, important for understanding how to *best* use the product on that platform.

3.1 Calculating Resource Requirements

3.1.1 UNIX NATIVE IPC REQUIREMENTS

3.1.1.1 Basic Resource Requirements

A platform that supports *any* XIPC activity uses the following native IPC resource:

- ❑ Two message queues.

3.1.1.2 XIPC Instance Requirements

3.1.1.2.1 General Instance Requirements

Each XIPC instance uses the following additional native IPC resources from the platform upon which it is started:

- ❑ One semaphore identifier (with one semaphore) for each user logging into the instance.

3.1.1.2.2 SemSys Specific Requirements

The SemSys subsystem of an XIPC instance uses the following native IPC resources from the platform upon which it is started:

- ❑ Two semaphore identifiers with one semaphore each.
- ❑ One additional sempahore identifier if the critical section algorithm is `SEMAPHORE`. (See Section 3.4 on Advanced Instance Configuration.)
- ❑ One shared-memory segment for control information whose size is determined by the configuration parameters and is displayed by `xipcstart` in the TOTAL line for SemSys.

3.1.1.2.3 QueSys Specific Requirements

The QueSys subsystem of an XIPC instance uses the following native IPC resources from the platform upon which it is started:

- ❑ Two semaphore identifiers with one semaphore each.
- ❑ One additional sempahore identifier if the critical section algorithm is `SEMAPHORE`. (See Section 3.4 on Advanced Instance Configuration.)
- ❑ One shared-memory segment for control information, whose size is determined by the configuration parameters and is displayed by xipcstart in the TOTAL line for QueSys.

- One shared-memory segment for the message text pool, whose size is reported by xipcstart in the TextPool line for QueSys.

3.1.1.2.4 MemSys Specific Requirements

The MemSys subsystem of an XIPC instance uses the following native IPC resources from the platform upon which it is started:

- Two semaphore identifiers with one semaphore each.
- One additional sempahore identifier if the critical section algorithm is `SEMAPHORE`. (See Section 3.4 on Advanced Instance Configuration.)
- One shared-memory segment for control information, whose size is determined by the configuration parameters and is displayed by xipcstart in the TOTAL line for MemSys.
- One shared-memory segment for the memory segment pool, whose size is reported by xipcstart in the MemoryPool line for MemSys.

3.1.1.2.5 MomSys Specific Requirements

MomSys subsystem requirements are equivalent to QueSys. In addition, each XIPC instance uses the following additional native IPC resources from the platform upon which it is started:

- One semaphore identifier (with one semaphore) for each user logging into the instance.

3.1.2 REQUIREMENTS FOR XIPC NETWORK

The following network resources are used by the XIPC *Network* environment:

- One transport end point (socket) is used for each process logged into a network instance.
- An additional two to seven end points are used by the platform's xipciad daemon, depending on the amount of network XIPC asynchronous activity involving the platform.
- MomSys requires two sockets per instance-link. In addition, if the xipcicd catalog server daemon is being employed - usually the case for MomSys - then an additional socket is utilized.

3.2 The XIPC Daemons

The following is a list of daemons that are installed:

Daemon Program	Function
`xipcisd`	XIPC's TCP/IP server
`xipclad`	XIPC's Local asynchronous server
`xipcicd`	XIPC's TCP/IP catalog server
`xipciad`	XIPC's TCP/IP asynchronous server
`xipcidld`	XIPC's Idle user detection mechanism

The `xipcisd` daemon is the TCP/IP Server program that handles remote `XipcLogin()` and `XipcList()` requests. `xipcisd` reports any errors in the `$XIPCROOT/log/xipcisd.log` log file.

The `xipclad` daemon is used by any platform employing XIPC asynchronous functionality. `xipclad` reports any errors in the `$XIPCROOT/log/xipclad.log` log file.

The xipciad daemon is used by: any platform employing XIPC asynchronous functionality in a network environment; XipcPing(); XipcAbort(); and MomSys, in a local or network instance. xipciad reports any errors in the $XIPCROOT/log/xipciad.log log file.

The xipcicd daemon is used by MomSys in order to access the XIPC catalog. xipcicd reports any errors in the $XIPCROOT/log/xipcicd.log log file.

The xipcidld daemon is used whenever an XIPC instance needs to be monitored against idle users. xipcidld reports any errors on the file $XIPCROOT/log/xipcidld.log log file. Refer to the XIPC User Guide Appendix for the Technical Note on the Idle User Detection Mechanism for a complete description of this service.

3.2.1 STARTING AND STOPPING XIPC DAEMONS

The XIPC daemons are started and stopped whenever the xipcinit and xipcterm commands are used to initialize and terminate the platform for XIPC activity. The xipcisd, xipclad, xipcicd, xipciad and xipcidld daemons are started by default. This list can be overridden by editing the $XIPCROOT/xipc.env file. Refer to the XIPC User Guide and the XIPC Reference Manual for further details.

3.3 Application Development With XIPC on UNIX Platforms

Divergent methods of program development, specific to each Operating System environment, affect how XIPC is used in that environment, most notably in the areas of Compiling, Linking and Trap Handling. These topics are examined in this section.

3.3.1 PREDEFINED DATATYPES

Much of XIPC's documentation refer to predefined datatypes such as XINT, CHAR, etc. The mapping between these types and the underlying "C" language datatypes is machine-dependent. For the UNIX platforms, type XINT is defined as a 32-bit signed integer and CHAR is defined as char.

It is recommended that programs making XIPC function calls use these definitions for declaring parameter variables that are passed to the XIPC functions. This will ensure portability across different hardware platforms.

The definitions are in the file mmcos.h and are included automatically by any program that includes xipc.h.

3.3.2 COMPILING

XIPC currently comes with a "C" language binding. While other languages may also be used to invoke XIPC, that may require preparation of function prototypes and data type definitions for that language.

When compiling a "C" program using XIPC, the header files directory should be made known to the compiler by specifying it in the -I compiler option.

Example:

```
cc -c foo.c -I/$(XIPCROOT)/include ...
```

3.3.3 LINKING

The XIPC API library comes in three flavors, each of which address a specific class of application. The libraries are:

- The XIPC Stand-Alone Library
- The XIPC Network Library
- The XIPC Combined Library

Refer to the "Using XIPC Libraries" section of the XIPC User Guide for a detailed discussion of when each library is appropriate.

This section presents technical instructions for using the XIPC libraries to develop applications on the AIX 3.2.x platform.

3.3.3.1 XIPC Libraries

The following libraries are included in the `lib` directory:

libsxipc.a The XIPC Stand-Alone Library

If you have installed the network version of the tool set, the following libraries are also included:

libnxipc.a The XIPC Network Library

libxipc.a The XIPC Combined Library

Programs that link with the Network or Combined API Library need to additionally be linked with the following protocol support library:

libnxipctcp.a The XIPC TCP/IP protocol support library

A set of shared libraries is included in the `lib/shared` directory.

3.3.3.2 Linking

Examples of platform-specific compile/link options are provided below for the XIPC Stand-Alone, Network and Combined libraries:

For the AIX Platform:

Stand-Alone: **`cc -o foo foo.o -L/$(XIPCROOT)/lib -lsxipc`**

Network: **`cc -o foo foo.o -L/$(XIPCROOT)/lib -lnxipc -lnxipctcp -lbsd -ldl`**

Combined: **`cc -o foo foo.o -L/$(XIPCROOT)/lib -lxipc -lnxipctcp -lbsd -ldl`**

For the HPUX 10.x and HPUX 11.x Platforms:

Stand-Alone: **`cc -o foo foo.o -L/$(XIPCROOT)/lib -lsxipc -lV3`**

Network: **`cc -o foo foo.o -L/$(XIPCROOT)/lib -lnxipc -lnxipctcp -lV3`**

Combined: **`cc -o foo foo.o -L/$(XIPCROOT)/lib -lxipc -lnxipctcp -lV3`**

For the SCO Platform:

Stand-Alone: **`cc -o foo foo.o -L/$(XIPCROOT)/lib -lsxipc -lsocket -lx`**

Network: **`cc -o foo foo.o -L/$(XIPCROOT)/lib -lnxipc -lnxipctcp -lsocket -lx`**

Combined: **`cc -o foo foo.o -L/$(XIPCROOT)/lib -lxipc -lnxipctcp -lsocket -lx`**

For the Solaris Platform:

Stand-Alone: **cc -o foo foo.o –L/$(XIPCROOT)/lib –lsxipc –lsocket -lnsl**

Network: **cc -o foo foo.o -L/$(XIPCROOT)/lib -lnxipc -lnxipctcp –lsocket -lnsl**

Combined: **cc -o foo foo.o -L/$(XIPCROOT)/lib -lxipc -lnxipctcp –lsocket -lnsl**

For the Tru64 UNIX and LINUX Platforms:

Stand-Alone: **cc -o foo foo.o –L/$(XIPCROOT)/lib –lsxipc**

Network: **cc -o foo foo.o -L/$(XIPCROOT)/lib -lnxipc -lnxipctcp**

Combined: **cc -o foo foo.o -L/$(XIPCROOT)/lib -lxipc -lnxipctcp**

3.3.4 SIGNALS

XIPC uses `SIGUSR1` in the course of its asynchronous operations. Programs that issue XIPC asynchronous operations cannot use this signal for other purposes. I/O descriptors can be used as an alternative to signals for asynchronous XIPC operations. For details, refer to the XIPC User Guide Appendix for the Technical Note, Using I/O Descriptors for Asynchronous Operations.

XIPC catches and ignores the `SIGPIPE` signal; however, XIPC's handling of `SIGPIPE` can be overridden by a user application that has a signal header for this purpose.

3.3.5 TRAP FUNCTIONS

Programs built with the XIPC tool set have the full capability of handling signal interrupts. This is described in the Section 6.7 on Trap Handling in the XIPC User Guide. Reading that description is a prerequisite for understanding this Section. This section provides the specific details of working with UNIX signals.

3.3.5.1 Test Macro for UNIX Platforms (SVR 4)

The calling sequence for the trap function test macro `XIPC_TRAP_FUNCTION_TEST()` is as follows:

```
XIPC_TRAP_FUNCTION_TEST(
                         TrapName,    /* Trap function name */
                         SigNum       /* Signal number      */
                        );
```

Example:

```
/*
 * A simple startup program that creates a queue and a semaphore
 * for usage by other programs. The program traps SIGINT and
 * SIGTERM signals. Note: no error checking is performed.
 */

#include <xipc.h>

main()
{

    VOID TrapFunction();

    /*
     * Trap the SIGINT and SIGTERM signals to execute the
     * "TrapFunction" function.
     */

    sigset(SIGINT, TrapFunction);
    sigset(SIGTERM, TrapFunction);

    /*
     * Login into the "DownLoad" network instance
     * and create necessary XIPC objects.
     */

    XipcLogin("@DownLoad", "InitTask");

    QueCreate("DownQueue", QUE_NOLIMIT, 32767);

    SemCreate("DownLoadDone", SEM_CLEAR);

      ...
      ...

    XipcLogout();

} /* main */



VOID
TrapFunction (SigNum)
INT SigNum;

{

    /*
     * Check that it is safe to run trap handler function.
     */

    XIPC_TRAP_FUNCTION_TEST(TrapFunction, SigNum);

    /*
     * Run the trap function.
     */

    printf("Received signal %d\n", SigNum);
```

```
    /*
     * The rest of the trap handling logic goes here.
     */

    return;

} /* TrapFunction */
```

3.3.5.2 Test Macro for UNIX Platforms (BSD)

The calling sequence for the trap function test macro `XIPC_TRAP_FUNCTION_TEST()` is as follows:

```
XIPC_TRAP_FUNCTION_TEST(
                        TrapName,    /* Trap function name */
                        SigNum       /* Signal number      */
                       );
```

Example:

```
/*
 * A simple startup program that creates a queue and a semaphore
 * for usage by other programs. The program traps SIGINT and
 * SIGTERM signals. Note: no error checking is performed.
 */

#include <xipc.h>

main()
{

    VOID TrapFunction();

    /*
     * Trap the SIGINT and SIGTERM signals to execute the
     * "TrapFunction" function.
     */

    nsig.sv_handler = TrapFunction;
    nsig.sv_mask = 0;
    nsig.sv_flags = 0;
    sigvec(SIGINT, &nsig, (struct sigvec *)NULL);
    sigvec(SIGTERM, &nsig, (struct sigvec *)NULL);

    /*
     * Login into the "DownLoad" network instance
     * and create necessary XIPC objects.
     */

    XipcLogin("@DownLoad", "InitTask");

    QueCreate("DownQueue", QUE_NOLIMIT, 32767);

    SemCreate("DownLoadDone", SEM_CLEAR);

      ...
      ...

    XipcLogout();
```

```
} /* main */


VOID
TrapFunction (SigNum)
INT SigNum;

{

    /*
     * Check that it is safe to run trap handler function.
     */

    XIPC_TRAP_FUNCTION_TEST(TrapFunction, SigNum);

    /*
     * Run the trap function.
     */

    printf("Received signal %d\n", SigNum);

    /*
     * The rest of the trap handling logic goes here.
     */

    return;

} /* TrapFunction */
```

3.3.6 *SAMPLE PROGRAMS*

A number of sample programs and makefiles are included with the XIPC product. They are installed under the $XIPCROOT/`samples` directory.

3.4 XIPC Advanced Instance Configuration (".cfg" File)

For a complete description of the configuration file, please refer to the XIPC User Guide and XIPC Reference Manual which provide all general information. If pertinent, operating system-specific information on certain file options is provided below.

3.4.1 CONFIGURING XIPC FOR MULTIPLE-CPU (SMP) SYSTEMS

Information on configuration for multiple-CPU (SMP) systems is found in Section 6.1.1 of the XIPC User Guide. Exceptions are described below.

AIX 4.x, HPUX and Solaris Platforms:

XIPC is able to detect whether the underlying hardware is an SMP or not. If it detects more than one processor active, then the `CSEC_ALGORITHM` parameter is set to a default value of `Semaphore`. If it detects that only one processor is active, then the `CSEC_ALGORITHM` is set to the default value of `GATE`.

Tru64 UNIX and LINUX Platforms:

The `Semaphore` value is used by default because `CSEC_ALGORITHM`'s `GATE` option is not supported.

3.4.2 CONFIGURING TO USE A SINGLE SHARED MEMORY SEGMENT

Information on configuration for a single shared memory segment is found in Section 6.1.2.1 of the XIPC User Guide. Exceptions are described below.

AIX Platforms:

The AIX platforms limit a process to attaching to ten (10) shared memory segments. Therefore the `SINGLE` option should be selected for the `SHARED_MEM` parameter in the `[XIPC]` section of the configuration file.

3.4.3 MEMORY-MAPPED FILES

Information on configuration for memory-mapped files is found in Section 6.1.2.2 of the XIPC User Guide.

3.5 Using XIPC Threads

This section contains information on writing XIPC applications that employ UNIX threads and pertains solely to the current release. Subsequent releases are subject to updates. *Be sure to review Section 5.3 on Threads in the "XIPC Programming" chapter of the XIPC User Guide before you begin using XIPC threads.*

The following platforms support XIPC threads:

- AIX 4.x (or higher)
- Tru64UNIX 3.2 (or higher)
- HPUX 10.x (or higher)
- LINUX
- Solaris 2.5 (or higher)

Other UNIX platforms do *not* support XIPC threads.

3.5.1 COMPILING

There are no changes to the compiling information provided earlier in these Platform Notes.

3.5.2 LINKING

For applications using threads, thread-safe libraries have been added. They are located in the `lib/threads` directory.

The reentrant versions of the libraries are:

1. `libxipc_r.a`
2. `libsxipc_r.a`
3. `libnxipc_r.a`
4. `libnxiptcp_r.a`

Applications that use threads with XIPC should link with these libraries. Refer to the sample `makefile` for an example of linking with the threaded libraries.

3.5.3 EXAMPLES

For actual examples, please refer to the `$XIPCROOT/samples/threads` subdirectory.

4. INDEX