CXpa V4.2
Release Notice

B5639-90010

June 1997

Hewlett-Packard Company
Convex Division
3000 Waterview Parkway
P.O. Box 833851
Richardson TX 75083-3851 USA

CXpa V4.2
Release Notice

B5639-90010

CXpa V4.2

Release Notice
June 1, 1997

Introduction

This document describes the V4.2 release of CXpa on Exemplar S-Class and X-Class (S2000 and X2000) technical servers and SPP1200 and SPP1600 Series systems. It describes new features and includes information about known problems and workarounds for this release.

Please refer to this release notice before asking questions or reporting problems with CXpa. This release notice contains:

- Contents of this distribution
- Software requirements
- Documentation
- Product licensing
- Directories and files
- Limitations
- Known problems
- What's new in this release?
- CXpa and `mpa`
- Instrumenting and profiling on different architectures (the `-tm` option)
- Installation notes
- Getting assistance and reporting problems

This release notice is also available through the CXpa online help system. To view this document online:

- In the Executable Manager or Analysis Control window, select Release Notice from the Help menu.
- In line mode, from the CXpa prompt, type `help release`.

Contents

The distribution package for CXpa V4.2 contains:

- CXpa V4.2 release notice
- distribution media for the software (including online help)

The specific contents of the software distribution are listed below.

Part No.	Description
B5639-10001	CXpa V4.2 software (including this online release notice and online help).
B5639-90002	*CXpa Reference: Exemplar S-Class and X-Class Servers*, Third Edition.
B5639-10010	*CXpa V4.2 Release Notice.*

Software requirements

CXpa V4.2 is supported on Exemplar S2000 and X2000 technical servers and SPP1200 and SPP1600 Series systems; it requires installation of the following supporting products:

Product	**Version**
Exemplar Fortran	V1.2.1 or newer
Exemplar C	V1.2.1 or newer
Exemplar C++	11.0.1 or newer
SPP-UX	V5.2 or newer. This includes the Exemplar assembler, linker (`ld`), and system libraries.

Compatibility with earlier releases:

Executables

Your application must be linked with the Exemplar linker to use CXpa.

The error message

```
WARN  D7: Executable not properly
linked. Relink with HP/Convex linker.
```

occurs when you attempt to profile an executable compiled or linked with a prior version of the compilers or a stripped executable. If this message is displayed, recompile and relink your application with the version of the Exemplar compiler and linker listed above, using the `+pa` option.

PDF files

PDF (performance data) files must be created with CXpa V4.2. PDF files created with versions of CXpa prior to V4.2 are incompatible with CXpa 4.2.

Documentation

CXpa 4.2 is documented in the following locations:

- *CXpa V4.2 Release Notice*
- *CXpa Reference: Exemplar S-Class and X-Class Servers,* HP Part No. B5639-90002 (hardcopy)
- cxpa(1) and cxoi(1) man pages
- CXpa online help system

The most complete and up-to-date information for using CXpa is available online through the CXpa help system.

Product Licensing

CXpa is a licensed product. If you do not acquire a license when running CXpa, contact the system administrator at your site for assistance. CXpa is licensed using the FlexLM licensing system. Refer to the *FLEXlm End User Manual* for licensing information.

Directories and files

The base directory for CXpa (/opt/cxpa) contains the following subdirectories:

- **bin**—contains the `cxpa` executable and the `cxoi` (object file instrumentor) utility.
- **lib**—contains the CXpa runtime profiling routines (cxpamon.o).
- **share**—contains all man pages and online help files.
- **newconfig**—contains the following subdirectories and files:
  - RelNotes/cxpa.4.2—ASCII text version of this release notice.
  - X11/app-defaults/Cxpa—Sample X11 app-defaults file for CXpa.

Limitations

The following limitations apply to CXpa V4.2:

- CXpa is incompatible with optimization levels greater than `+O3` (`+O4` and `+Oall`).
- If a process forks, but does not perform an `exec()`, CXpa only profiles the parent process. To profile with CXpa in the fork/exec scenario, use CXpa's pre-instrumented executables feature.

  Refer to the "Using pre-instrumented executables" online help topic for instructions on how to pre-instrument an executable.

- Times and metrics for threads that are involved in asymmetric execution are not propagated upward into their enclosing regions. Asymmetric parallelism occurs when each thread executes a different, independent instruction stream. Asymmetric threads are analogous to the UNIX `fork()` system call construct. For example, asymmetric threads can be spawned using the `cps_thread_create()` function and terminated using the `cps_thread_exit()` function.

  Inclusive metrics (metrics and time spent in called routines or inner loops) for enclosing regions do not take into account the time and metrics attributed to these asymmetric threads. As a result, inclusive metrics, from the point of view of the main program or enclosing regions, are inaccurate. Exclusive timings and metrics for applications using asymmetric parallelism are correct and are not affected by this problem.

  For example:

  - `main()` calls `sub1()`, which then calls `sub2()` in parallel, using asymmetric threads
  - `sub2()` is not called from main() or sub1() in addition to the forking off of the threads that work asymmetrically on `sub2()`

  In this case, inclusive metrics for `main()` and `sub1()` do not include any of the time spent in `sub2()`. The parallel concurrency factor (CPU time/wall clock time) reported by CXpa for this example is less than what is really achieved, because the CPU time spent in `sub2()` is not included, and so on.

- 3D profile graphs containing a large number of data points take a long time to render.

- If you are using a virtual window manager, such as `tvtwm`, CXpa may position xterms it creates in unexpected locations (that is, not in the current virtual root window).

  To work around the problem, you can disable auto-positioning using the `-nap` (no auto-position) option of the `cxpa` command when you start CXpa or set the X application resource `Cxpa*autoPosition:` to `False` (the default setting is `True`).

- PDFs to be merged must be generated from the same executable and must contain identical region and metric selections.

Known problems

This section describes known problems and workarounds for CXpa V4.2:

- CXpa V4.2 does not support profiling of loops or parallel loops.

- CXpa does not report the number of loop iterations in a chunk (chunk size). A "chunk" is a packet of loop iterations assigned to execute on a particular thread in a parallel loop region.
- A pre-instrumented executable cannot be fully uninstrumented.

  To work around the problem, save the instrumentation to a copy of the original executable (using the command `save executable` *<filename>* or the Save Executable As option) or relink the application after profiling is completed.
- In line mode, if the number of characters you type on the command line exceeds the screen width, the line does not wrap properly. Instead, every character you type beyond the screen width produces another copy of the first line.

What's new in this release?

This section briefly summarizes changes in functionality between CXpa V4.0 and CXpa 4.2. For more information, refer to the CXpa online help.

New features in CXpa and functionality changes since V4.0 are discussed in the following sections:

- support for profiling applications on Exemplar S-Class (S2000) and X-Class (X2000) technical servers
- new compiling instructions

Support for profiling on Exemplar S-Class and Exemplar X-Class systems

This release of CXpa supports collection and analysis of the following metrics on Exemplar S-Class and Exemplar X-Class systems:

- Wall clock time
- CPU time
- Execution counts
- Dynamic call graph
- Locally and remotely resolved cache miss counts and latency

During analysis, CXpa also reports performance data derived from the above metrics, such as CPU/Wall clock time.

CXpa no longer supports profiling of basic block regions.

Compiling applications for profiling with CXpa

To compile applications for routine-level profiling with CXpa using Exemplar compilers, add the +pa option to the compilation/link line.

The Exemplar C, Fortran 77, and C++ compilers are based on the Hewlett-Packard compilers. They have been modified to support the Exemplar programming model, allowing for creation, profiling, and debugging of thread-parallel applications on Exemplar S2000 systems.

For example:

```
% /opt/fortran/bin/f77 +pa myprog.f sub1.f sub2.f
```

If you are compiling and linking separately, you must include the +pa option when you link the application to ensure that the CXpa runtime monitoring routines are linked in. For example:

```
% /opt/fortran/bin/f77 +pa -c main.f sub1.f sub2.f
% /opt fortran/bin/f77 +pa main.o sub1.o sub2.o
```

The following compiler options are incompatible with the +pa and +pal options:

- -p and -G—These options generate instrumentation for other profilers.
- Optimization levels greater than +O3 (+O4 and +Oall)
- -s—This option causes the output of the linker to be stripped of symbol table information. The strip(1) command also removes this information.

Refer to the f77(1), c89(1), and CC(1) man pages installed on the Exemplar system for more information about the Exemplar compilers.

Major bug fixes or enhancements since CXpa V3.5

Works correctly with pthreads— previously, you would get unpredictable results from any application that used pthreads.

Works better with f90— +pa section now supported by the compiler. The +pal section will be included in a future release.

C++ name filtering— can now select the function prototype and class name in output.

Version control for CXpa routines and compiler generated information— cxpa will now complain if the PDF or runtime (cxpamon.o) is not the same as the version used to run the application or if an incompatible compiler was used to generate the application.

Pausing your application inside a parallel region now reports accurate times—before you could see high times until the application left the parallel region.

CXpa and mpa

If you are using the `mpa` utility on an Exemplar S2000, Exemplar X2000, SPP1200 Series, or SPP1600 Series system to modify process attributes (for example, data size or parallel process attributes such as maximum and minimum number of threads) and want to profile with CXpa, the preferred method is to pre-instrument the executable for profiling with CXpa, then run the executable with `mpa`.

For example:

```
% /opt/cxpa/bin/cxpa -nw a.out

        CONVEX Performance Analyzer

Type 'help' for help.
Reading executable a.out...
Selecting profile a.out.pdf...
(CXpa) select routine all
(CXpa) collect wall_clock cpu
(CXpa) save executable as a.out.inst
(CXpa) instrumenting, enter <CTRL+C> to abort.
...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
(CXpa) instrumentation finished.
(CXpa) quit
% /bin/mpa <mpa_options> a.out.inst
% /opt/cxpa/bin/cxpa a.out.inst.15237.pdf
```

The commands in the above example demonstrate how to profile executables run with the `mpa` utility:

- The command `/opt/cxpa/bin/cxpa -nw a.out`, runs CXpa from the shell in line mode (`-nw`) and specifies the executable file `a.out` for profiling.
- The command `select routine all` selects all routines in the program for profiling.
- The command `collect cpu wall_clock` enables collection of CPU and wall clock time.
- The command `save executable a.out.inst` writes the instrumentation specified by the `collect` and `select` commands to a copy of the executable file name which is named `a.out.inst`.
- The `quit` command exits CXpa.

- The command `/bin/mpa` *<mpa_options>* `a.out.inst` runs the instrumented executable file `a.out.inst` from the shell using `mpa` to modify the program's process attributes with the specified options. As the program runs, profiling data is collected in the file a.out.inst.*<pid>*.pdf (the program's UNIX process ID, *pid*, is inserted into the name of the PDF file).
- The last command runs CXpa in X window mode from the shell and specifies the PDF file a.out.inst.15237.pdf for analysis.

Refer to the `mpa` man page for information about using the `mpa` utility and its options.

Refer to the "Using pre-instrumented executables" topic in the CXpa online help for detailed information about pre-instrumenting executables.

Instrumentng and profiling on different architectures (the -tm option)

If you are using CXpa to pre-instrument an executable on an architecture that is different from the architecture the executable will be run on to generate profiling data, you must specify the `-tm` *<architecture>* option when you start CXpa from the shell command line. For example, if you are pre-instrumenting an executable on an SPP1600, but running the executable on an S2000, start CXpa with a command similar to the following:

```
% /opt/cxpa/bin/cxpa -tm s2000 a.out
```

This ensures that the correct timing routines are called to collect hardware event metrics for the target system and that the appropriate hardware event metric selections are available.

Valid values for *<architecture>* are as follows:

`spp1200`—Exemplar SPP1200 Series
`spp1600`—Exemplar SPP1600 Series
`s2000`—Exemplar S-Class (S2000) technical servers
`x2000`—Exemplar X-Class (x2000) technical servers

Installation notes

Activity logging

CXpa logs selected product operations through the `syslog` facility. It logs messages to the destination(s) associated with the user subsystem at the `debug` priority level (`user.debug`). Refer to the syslog, syslogd, and the syslog.conf man pages for more information.

The `cxpa` executable is shipped with activity logging enabled. CXpa activity logging can be permanently disabled in the executable file using the `stampEnableFlog` utility found in the directory /opt/cxpa/newconfig/etc, as shown below.

```
% /opt/cxpa/newconfig/etc/stampEnableFlog 0
```

Specifying a value of 1 with the `stampEnableFlog` utility enables activity logging; specifying a value of 0 disables activity logging. You must have the correct permissions to modify the executable file.

Getting assistance

If you have questions about CXpa, contact the Hewlett-Packard Company Convex Technical Assistance Center (TAC). To contact the TAC, use one of the following phone numbers:

- Within the continental U.S., call 1(800)952-0379.
- From Canada, call 1(800)345-2384
- All other locations, contact the nearest sales office.

The TAC recommends using the `contact` utility to report a hardware, software, or documentation problem. The `contact` utility is an interactive program that helps the TAC track reports and route them to the personnel most qualified to fix a problem.

After you invoke `contact`, your are prompted for information about the problem. When you finish your report, `contact` mails it to the TAC electronically. The TAC notifies you within 48 hours that your report has been received.

If possible, when submitting problem reports for CXpa, include:

- Version of Exemplar C, C++, or Fortran compiler used
- Version of SPP-UX installed on your system
- CXpa version number and a small test case, if available

To display the version of the Exemplar compilers, linker, or CXpa, use the `what` command. For example:

```
% /usr/bin/what /opt/fortran/bin/f77
/opt/fortran/bin/f77:
    SPP-UX f77 970509 (164218)  B5600-10002 Exemplar V1.2.1
     Internal_Unsupported_Version libc.a_ID@@/main/r10sac//1
     /us/core/libs/libc/archive_pal/libc.a_ID
     Oct 24 1996 23:13:02
```

In the above example, the version number of the Exemplar Fortran 77 compiler is V1.2.1.

To display the version of the SPP-UX operating system installed on your system, use the `sysinfo` command. For example:

```
% sysinfo
SPP-UX_mk      5.2
SPP-UX_server  5.2
SPP-UX_ail     5.2
```

Using `contact` requires:

- UNIX-to-UNIX Communication Protocol (UUCP) connection to the TAC.
- Full path name of the program or utility in question.
- Version number of the program or utility in question.

Refer to the contact(1) man page for complete details.