HP NonStop XML Parser User Guide

HP Part Number: 731047-001
Published: August 2013
Edition: J06.16 and subsequent J-series RVUs and H06.27 and subsequent H-series RVUs.

Legal Notice

Contents

About this document

This guide contains NonStop specific installation instructions for NonStop XML Parser (T0970). It contains a brief overview of the open source products, International Components for Unicode (ICU) and Xerces-C++ libraries, used for building the NonStop XML parser. It also provides instructions on how to build and execute the samples provided with the product.

Supported Release Version Updates (RVUs)

This manual supports J06.16 and all subsequent J-series RVUs, and H06.27 and all subsequent H-series RVUs, until otherwise indicated by its replacement publications.

Intended audience

This manual is intended for users developing XML applications on NonStop.

Document organization

This manual is organized as follows:

"Introduction" (page 7)	This chapter provides a brief overview about the supported open source products.
"Installation" (page 8)	This chapter provides the procedure to install NonStop XML Parser on the NonStop system.
"Sample applications" (page 13)	This chapter provides information about how to build and run the samples packaged with NonStop XML Parser.
"Sample `defs.mk` file" (page 17)	This appendix provides an example of the definition file used for building the samples.

Notation conventions

Bold Type

Bold type within text indicates terms defined in the Glossary. For example:

abstract class

Computer Type

`Computer type` letters within text indicate keywords, reserved words, command names, class names, and method names; enter these items exactly as shown. For example:

`myfile.jar`

Italic Computer Type

`Italic computer type` letters in syntax descriptions or text indicate variable items that you supply. For example:

`pathname`

[ ] Brackets

Brackets enclose optional syntax items. For example:

`jdb [options]`

A group of items enclosed in brackets is a list from which you can choose one item or none. Items are separated by vertical lines. For example:

```
where [threadID|all]
```

{ } Braces

A group of items enclosed in braces is a list from which you must choose one item. For example:

```
-c identity {true|false}
```

| Vertical Line

A vertical line separates alternatives in a list that is enclosed in brackets or braces. For example:

```
where [threadID|all]
```

... Ellipsis

An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
print {objectID|objectName} ...
```

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

```
dump objectID ...
```

Punctuation

Parentheses, commas, equal signs, and other symbols not previously described must be entered as shown. For example:

```
-D propertyName=newValue
```

Item Spacing

Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or comma. If there is no space between two items, spaces are not permitted. In the following example, spaces are not permitted before or after the period:

```
subvolume-name.filename
```

Line Spacing

If the syntax of a command is too long to fit on a single line, each line that is to be continued on the next line ends with a back slash (\ ) and each continuation line begins with a greater-than symbol (>). For example:

```
/usr/bin/c89 -c -g -I /usr/tandem/java/include \
> -I /usr/tandem/java/include/oss -I . \
> -Wextensions -D_XOPEN_SOURCE_EXTENDED=1 jnative01.jar
```

Related information

For more information about ICU and Xerces-C++ libraries, see:

- http://site.icu-project.org/
- http://xerces.apache.org/xerces-c/

Publishing history

Part Number	Product Version	Publication Date
731047-001	1.0	August 2013

HP encourages your comments

HP encourages your comments concerning this document. We are committed to providing documentation that meets your needs. Send any errors found, suggestions for improvement, or compliments to **docsfeedback@hp.com**.

Include the document title, part number, and any comment, error found, or suggestion for improvement you have concerning this document.

1 Introduction

The NonStop XML parser (T0970H01) is based on the Apache Xerces-C++ open source parser, version 3.1.1, and IBM's open source International Components for Unicode (ICU), version 50.1.2.

This product is available along with the existing two XML parsers listed below, which are available as two different T-numbers, each supporting a particular floating-point type.

- T0535: XML Parser with TANDEM float
- T0563: XML Parser with IEEE float

The new XML Parser T0970 provides both IEEE and TANDEM floating-point types in the same T-number.

The following table provides a comparison of the earlier XML Parsers and the current XML Parser:

Product T-number	Product Name	Xerces-C++ version	ICU version	Floating-point support
T0535	XML Parser	2.4.0	2.6.1	TANDEM
T0563	XML Parser	2.4.0	2.6.1	IEEE
T0970	XML Parser	3.1.1	50.1.2	IEEE, TANDEM

NOTE: This manual is applicable only for the new XML Parser T0970.

International Components for Unicode

ICU is a widely used set of C/C++ libraries providing unicode and globalization support for software applications. It is portable and provides the same result to applications on all platforms.

ICU is released under a non-restrictive open source license that is suitable for use with both commercial software and with other open source or free software.

For more information about ICU, see http://site.icu-project.org/.

For licensing information of ICU, see http://source.icu-project.org/repos/icu/icu/trunk/license.html.

Xerces-C++

Xerces-C++ is a validating XML parser written in a portable subset of C++. Xerces-C++ makes it easy for applications to read and write XML data. It provides a shared library, which is used to parse, generate, manipulate, and validate XML documents using the DOM, SAX, and SAX2 APIs. Xerces-C++ conforms to the XML 1.0 recommendation and many associated standards.

The parser provides high performance, modularity, and scalability.

For more information about Xerces-C++, see http://xerces.apache.org/xerces-c/.

Xerces-C++ is available under the Apache Software License, Version 2.0.

2 Installation

This chapter describes the procedure to install and verify the NonStop XML Parser on NonStop systems.

Pre-requisites

Before getting started, ensure that you have the following software installed on the NonStop system:

- Open System Services (OSS) environment running a H06.27 or J06.16 RVU or later
- c89 C++ compiler
- OSS Core Utilities (T1202)
- C++ runtime library version 2 and version 3

Installing the NonStop XML Parser

Installing the NonStop XML Parser on a NonStop system involves:

- "Extracting the PAX files" (page 8)
- "Verifying the extracted files" (page 10)

Extracting the PAX files

You can extract the PAX files to one of the following directories:

- The OSS directory using DSM/SCM
- A user-specified installation directory using DSM/SCM and PINSTALL
- The OSS directory using the COPYOSS command

Extracting the PAX files to the OSS directory using DSM/SCM

Perform the following steps to extract the PAX files to the standard OSS directory (`/usr/tandem/xml/t0970h01`) using the Distributed Systems Management/Software Configuration Manager (DSM/SCM):

1. Obtain the following product files from the disk (distribution subvolume (DSV) locations) or tape:

PAX files	Contents of the PAX
`ICUCPAX`	ICU samples, license information, ICU data, and ICU header files
`ICUIPAX`	ICU libraries and tools built with IEEE float and C++ libraries version2 and version3
`ICUTPAX`	ICU libraries and tools built with Tandem float and C++ libraries version2 and version3
`XERCPAX`	Xerces-C++ samples, license information, Xerces-C++ documentation, and Xerces-C++ header files
`XERIPAX`	Xerces-C++ libraries built with IEEE float and C++ libraries version2 and version3
`XERTPAX`	Xerces-C++ libraries built with Tandem float and C++ libraries version2 and version3

2. In the DSM/SCM planner interface, select the **Manage OSS Files** option for the target configuration.

NOTE: If you do not select the **Manage OSS Files** option in the DSM/SCM planner interface, DSM/SCM places the PAX files in the Guardian subvolume $*ISV*.ZOSSUTL (where, *ISV* is the installation volume). You must then use the `COPYOSS` command to extract and place the contents of the PAX files to the OSS file system.

3. Copy the files to the NonStop system.
4. Run the `Build` request and the `Apply` request on the configuration revision.
5. Run `ZPHIRNM` to rename the product files.

For more information about using DSM/SCM, see the *DSM/SCM User's Guide*.

Extracting the PAX files to a user-specified installation directory using DSM/SCM and PINSTALL

1. Obtain the product files from the disk (DSV locations) or tape.
2. In the DSM/SCM planner interface, clear the **Manage OSS Files** option for the target configuration, if selected.
3. Copy the files to the NonStop system.
4. Run the `Build` request and the `Apply` request on the configuration revision.
5. Run `ZPHIRNM` to rename the product files.
6. Use `PINSTALL` to extract the PAX files to the OSS file system as follows:
    1. Log on to the NonStop system as super user.

       ```
       TACL> LOGON SUPER.SUPER
       ```

    2. Go to the Guardian subvolume $*ISV*.ZOSSUTL.

       ```
       TACL> VOLUME $ISV.ZOSSUTL
       ```

       where, *ISV* is the installation volume.

    3. Extract the PAX files using the PINSTALL utility.

       ```
       TACL> PINSTALL -s:/usr/tandem/xml/t0970h01:<install-dir>: -rvf
       <PAX files>
       ```

       where,

       `<install-dir>`: the user-specified installation directory,

       `<PAX files>`: all PAX files specified one by one, separated by space.

The PINSTALL utility extracts the product files from the PAX files and places them in the user-specified directory.

For more information about using DSM/SCM, see the *DSM/SCM User's Guide*.

For more information about using PINSTALL, see the *Open System Services Management and Operations Guide*.

Extracting the PAX files to the OSS directory using the `COPYOSS` command

1. Log on to the NonStop system as super user.

   ```
   TACL> LOGON SUPER.SUPER
   ```

2. Go to the Guardian subvolume $*ISV*.ZOSSUTL.

   ```
   TACL> VOLUME $ISV.ZOSSUTL
   ```

   where, *ISV* is the installation volume.

3. Extract the PAX files using the TACL macro COPYOSS.

   ```
   TACL> RUN COPYOSS <PAX files>
   ```

   where,

   <PAX files>: all PAX files specified one by one, separated by space.

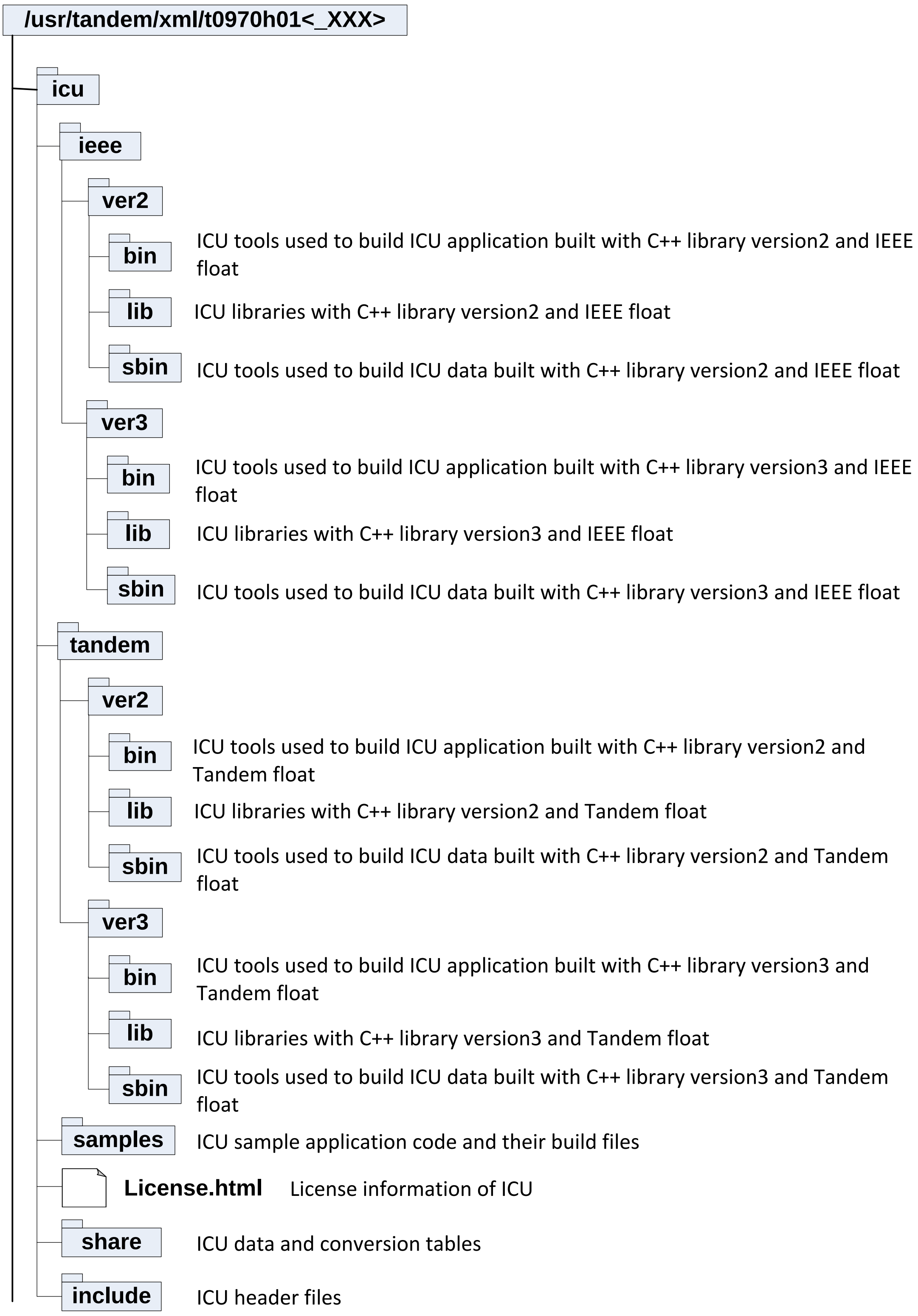
The `COPYOSS` command extracts the product files from the PAX files and places them in the `/usr/tandem/xml/t0970h01` OSS directory.

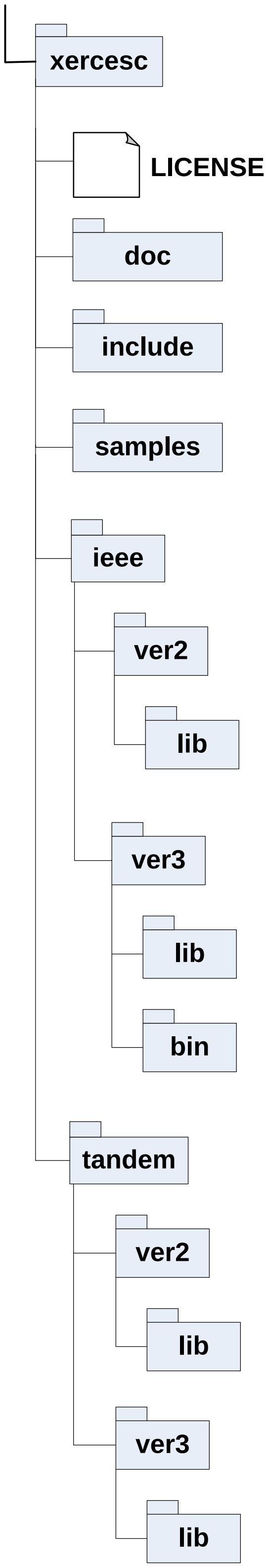
For more information about using the `COPYOSS` command, see the *Open System Services Management and Operations Guide*.

Verifying the extracted files

Verify that all the files and directories are extracted in the *<NonStop_XML_Parser_Installation_Directory>*. This is the directory where XML Parser is installed, for example, `/usr/tandem/xml/t0970h01`. Figure 1 (page 11) illustrates the directory structure.

Figure 1 Directory structure


/usr/tandem/xml/t0970h01<_XXX>
icu
ieee
ver2
bin
ICU tools used to build ICU application built with C++ library version2 and IEEE float
lib
ICU libraries with C++ library version2 and IEEE float
sbin
ICU tools used to build ICU data built with C++ library version2 and IEEE float
ver3
bin
ICU tools used to build ICU application built with C++ library version3 and IEEE float
lib
ICU libraries with C++ library version3 and IEEE float
sbin
ICU tools used to build ICU data built with C++ library version3 and IEEE float
tandem
ver2
bin
ICU tools used to build ICU application built with C++ library version2 and Tandem float
lib
ICU libraries with C++ library version2 and Tandem float
sbin
ICU tools used to build ICU data built with C++ library version2 and Tandem float
ver3
bin
ICU tools used to build ICU application built with C++ library version3 and Tandem float
lib
ICU libraries with C++ library version3 and Tandem float
sbin
ICU tools used to build ICU data built with C++ library version3 and Tandem float
samples
ICU sample application code and their build files
License.html
License information of ICU
share
ICU data and conversion tables
include
ICU header files



License information of XERCESC

Xerces-C++ open source documentation

Header files to build applications using Xerces-C++ libraries

Xerces-C++ sample application code and their build files

Xerces-C++ libraries with C++ library version2 and IEEE float

Xerces-C++ libraries with C++ library version3 and IEEE float

Xerces C++ samples built with C++ library version3 and IEEE float

Xerces-C++ libraries with C++ library version2 and Tandem float

Xerces-C++ libraries with C++ library version3 and Tandem float

3 Sample applications

The NonStop XML Parser is packaged with sample applications. You can use the sample applications to get started with the NonStop XML Parser. These sample applications demonstrate the important features of the parser.

You can build all the samples simultaneously or build each sample individually. However, you must set some key environment variables before building the samples.

This section includes information about:

- building and running the ICU samples
- building and running the Xerces-C++ samples

NOTE: Before building the samples, copy the `samples` directory and sub-directories to an OSS location where you have read, write, and execute permissions.

Building the ICU samples

The NonStop XML Parser is packaged with the following ICU samples. These samples are available in the `<ICUROOT>/samples` directory, where ICUROOT is the `<NonStop_XML_Parser_Installation_Directory>/icu` directory.

Sample	Function	Sample exe filename
break	Demonstrates how to use BreakIterators in C and C++	break
cal	Prints out a calendar	icucal
case	Demonstrates how to do Unicode case conversion in C and C++	case
csdet	Demonstrates using ICU's CharSet Detection API	csdet
date	Prints out the current date, localized	icudate
datefmt	Demonstrates the use of the date formatting API	datefmt
msgfmt	Demonstrates the use of the message format	msgfmt
numfmt	Demonstrates the use of the number format	numfmt
props	Demonstrates the use of Unicode properties	props
strsrch	Demonstrates how to search for patterns in Unicode text using the usearch interface	strsrch
translit	Demonstrates the use of ICU transliteration	translit
uciter8	Demonstrates how to read 8-bit Unicode text	uciter8
ucnv	Demonstrates the use of ICU codepage conversion	ucnv
udata	Demonstrates the use of ICU low level data routines	writer, reader
ufortune	Demonstrates packaging and use of resources in an application	ufortune
ugrep	Demonstrates ICU regular expressions	ugrep
uresb	Demonstrates building and loading resource bundles	uresb
ustring	Demonstrates ICU string manipulation functions	ustring

Perform the following steps before building the samples:

- Copy the `samples` directory and sub-directories to an OSS location where you have read, write, and execute permissions. This location is referred to as *<icu_user_location>*`/samples`.
- Set the ICUROOT and ICU_DATA environment variables by entering the following commands:

```
OSS> export ICUROOT=/usr/tandem/xml/T0970H01/icu
```

```
OSS> export ICU_DATA=$ICUROOT/share/icu/50.1.2/
```

You can build the samples with different combinations of libraries that are available. You can compile the samples by setting the values of the `FLOAT` and `VERSION` variables in the *<icu_user_location>*`/samples/defs.mk` file. For a sample `defs.mk` file, see appendix “Sample `defs.mk` file” (page 17).

The following table lists the valid values for these variables. You can define a combination of these values for building the samples. Default values are set in the `defs.mk` file. If you do not change these values, the default values are used.

Variable	Valid values
FLOAT	ieee, tandem. Default is `ieee`.
VERSION	2, 3. Default is `3`.

NOTE:

- The value of each variable in the `defs.mk` file is case sensitive.
- There is no support for ICU on Guardian platform.

After setting the required values, execute the following command to build all samples from the *<icu_user_location>*`/samples` directory:

```
OSS> make all
```

To build a particular sample, use the following command:

```
OSS> make <sample_name>-sample
```

ICU configuration helper script

The ICU configuration helper script (`icu-config` script) is available in the `bin` folder of each ICU variant. It simplifies the task of building and linking the object files against ICU as compared to manually configuring user makefiles or equivalent. As `icu-config` is an executable script, it locates the ICU libraries and headers by using the system PATH variable. Using `icu-config` is convenient for trivial, single-file programs using ICU.

`icu-config` can be used with or without a makefile. If it is used without a makefile, the following command is sufficient for building a single-file C++ program against ICU (For example, icu/source/samples/ufortune/ufortune.cpp):

```
`icu-config --cxx --cxxflags --cppflags --ldflags` -o ufortune
ufortune.cpp
```

Mostly, `icu-config` is called from within a makefile, and used to set up variables.

For more information on `icu-config` tool, see `icu-config --help`.

Running the ICU samples

When you build the samples, the executable files are created in the *<icu_user_location>*`/samples/`*<sample-dir>* directory, where *<sample-dir>* is the respective directory of each sample.

You can execute the samples from OSS command line directly as follows:

```
OSS> cd <icu_user_location>/samples/<sample-dir>
OSS> ./<sample exe file>
```

Building the Xerces-C++ samples

The NonStop XML Parser is packaged with the following Xerces-C++ samples. These samples are available in the *<XERCESCROOT>*/samples directory, where XERCESCROOT is the *<NonStop_XML_Parser_Installation_Directory>*/xercesc directory.

Sample	Function
CreateDOMDocument	Creates a DOM tree in memory from scratch.
DOMCount	Counts the elements in an XML file.
DOMPrint	Parses an XML file and prints it.
EnumVal	Displays how to enumerate the markup declarations in a DTD validator.
MemParse	Parses XML in a memory buffer and prints the number of elements and attributes.
PParse	Demonstrates progressive parsing.
PSVIWriter	Parses the specified XML file, and exposes the PSVI and Schema Component Model information.
Redirect	Redirects the input stream for external entities.
SAX2Count	Parses an XML file and prints out a count of the number of elements and characters in the file.
SAX2Print	Parses an XML file and prints it.
SAXCount	Counts the elements, attributes, spaces, and characters of a given XML file.
SAXPrint	Parses an XML file and prints it.
SCMPrint	Parses the specified XSD file, then shows how to access the Schema Content Model information.
SEnumVal	Displays how to enumerate the markup declarations in a Schema Grammar.
StdInParse	Demonstrates streaming XML data from the standard input.
Xinclude	Converts an input XML file into an expanded output XML file.

Perform the following steps before building the samples:

- Copy the samples directory and sub-directories to an OSS location where you have read, write, and execute permissions. This location is referred to as *<xercesc_user_location>*/samples.
- Set the XERCESCROOT environment variable by entering the following command:

  ```
  OSS> export XERCESCROOT=/usr/tandem/xml/T0970H01/xercesc
  ```

  If you want to build these samples with ICU support, then set the ICUROOT and ICU_DATA environment variables by entering the following commands:

  ```
  OSS> export ICUROOT=/usr/tandem/xml/T0970H01/icu
  OSS> export ICU_DATA=$ICUROOT/share/icu/50.1.2/
  ```

You can build the samples with different combinations of libraries that are available. You can compile the samples by setting the values of the FLOAT, VERSION, ICU_SUPPORT, and PLATFORM variables in the *<xercesc_user_location>*/samples/defs.mk file. For a sample defs.mk file, see appendix "Sample defs.mk file" (page 17).

The following table lists the valid values for these variables. You can define a combination of these values for building the samples. Default values are set in the `defs.mk` file. If you do not change these values, the default values are used.

Variable	Valid values
FLOAT	ieee, tandem. Default is `ieee`.
VERSION	2, 3. Default is `3`.
ICU_SUPPORT	yes, no. Default is `no`.
PLATFORM	oss, guardian. Default is `oss`. For `guardian`, ICU_SUPPORT variable must be set to `no`.

NOTE:

- The value of each variable in the `defs.mk` file is case sensitive.
- There is no support for ICU on guardian platform.

After setting the required values, execute the following command to build all samples from the *<xercesc_user_location>*`/samples` directory:

```
OSS> make
```

To build a particular sample, use the following command:

```
OSS> make <sample_name>
```

NOTE: When building an executable on Guardian, you must not return a non-zero value from the main() function as it results in abend. For more information, see the *Guardian Procedure Calls Reference* manual and the *CRE Programmer's Guide*.

Running the Xerces-C++ samples

When you build the samples, the executable files are created in the *<xercesc_user_location>*`/samples` directory. If you have built OSS executable samples, you can execute them from the OSS command line directly as follows:

```
OSS> ./<sample exe file>
```

If you have built Guardian executable samples, then perform the following steps:

1. Copy the executable to the Guardian location in binary mode.

   ```
   OSS> cp <sample exe file> /G/<volname>/<subvolname>/<filename>
   ```

   NOTE: Guardian does not accept more than 8 characters as file name.

   If there are other data files such as xml and dtd, you must copy them as well.

2. Go to the Guardian prompt, change the file code to TNS/E executable.

   ```
   TACL> fup alter <filename>, code 800
   ```

3. Execute the sample.

   ```
   TACL> run <filename>
   ```

A Sample defs.mk file

This appendix provides an example of the definition files used for building the samples.

For ICU:

A sample defs.mk file is located in the *<icu_user_location>*/samples/ directory and is shown here:

```
FLOAT=ieee
VERSION=3

CXX=c++
CC=cc
MAKE=make

ifeq ($(FLOAT),ieee)
FLOATING_POINT=IEEE
endif

ifeq ($(FLOAT),tandem)
FLOATING_POINT=Tandem
endif

ifeq ($(VERSION),2)
VER2DEF=-D_USER_CRTL_VERSION2
endif

CPP_FLAGS= -Wsystype=oss -Wtarget=tns/e -I$(ICUROOT)/include $(VER2DEF)
 -Woptimize=1 -W$(FLOATING_POINT)_float -g -Wversion$(VERSION) -Winline
 -Ww -Wrefalign=8 -Wenv=common -Wfieldalign=auto

C_FLAGS= -Wsystype=oss -Wtarget=tns/e -I$(ICUROOT)/include $(VER2DEF)
 -Woptimize=1 -W$(FLOATING_POINT)_float -g -Winline -Ww -Wrefalign=8
 -Wenv=common -Wfieldalign=auto -Wallow_cplusplus_comments

LINK_FLAGS= -L$(ICUROOT)/$(FLOAT)/ver$(VERSION)/lib -licui18n -licuuc
 -licudata -licuio -licui18n -licuuc -lput -lm -Wsystype=oss -Wtarget=tns/e
 -Wcall_shared -Weld=-bLocalized -Weld="-unres_symbols Error"
 -Weld=-Noverbose -Winspect -Whighpin=on -Whighrequesters=on -Wcplusplus
 -Wversion$(VERSION)
```

For Xerces-C++

A sample defs.mk file is located in the *<xercesc_user_location>*/samples/ directory and is shown here:

```
FLOAT=ieee
VERSION=3
ICU_SUPPORT=no
PLATFORM=oss

CXX=c89
CC=c89
MAKE=make

ifeq ($(FLOAT),ieee)
FLOATING_POINT=IEEE
endif

ifeq ($(FLOAT),tandem)
FLOATING_POINT=Tandem
endif

ifeq ($(VERSION),2)
VER2DEF=-D_USER_CRTL_VERSION2
```

```
endif

ifeq ($(ICU_SUPPORT),no)
XLIB=xerces-c
else
XLIB=icuxerces-c -L$(ICUROOT)/$(FLOAT)/ver$(VERSION)/lib -licui18n -licuuc
 -licudata -licuio -licui18n -licuuc -lm -lput -lZRLDDLL
endif

ifeq ($(PLATFORM),guardian)
PUTILS=GuardianTandemPlatformUtils.o
PGUARD=-D_GUARDIAN_SYSTYPE=1
else
PUTILS=OssTandemPlatformUtils.o
endif

CPP_FLAGS= -Wsystype=$(PLATFORM) -Wtarget=tns/e -I$(XERCESCROOT)/include
 -DHAVE_CONFIG_H $(VER2DEF) $(PGUARD) -Woptimize=1 -W$(FLOATING_POINT)_float
 -g -Wversion$(VERSION) -Winline -Ww -Wrefalign=8 -Wenv=common -Wfieldalign=auto

LINK_FLAGS= $(XERCESCROOT)/$(FLOAT)/ver$(VERSION)/lib/$(PUTILS)
 -Weld=-allow_duplicate_procs -Wsystype=$(PLATFORM)
 -L$(XERCESCROOT)/$(FLOAT)/ver$(VERSION)/lib -l$(XLIB)
 -Wtarget=tns/e -Weld=-bLocalized -Weld="-unres_symbols error"
 -Weld=-Noverbose   -Winspect -Whighpin=on -Whighrequesters=on -Wcplusplus
 -Wversion$(VERSION)
```

Index